

# Python and Everyday Computing

*Note: This is an MVP course for learning and teaching Python.*

## Course Description

This introductory course is designed for students who do not have any prior knowledge of text-based programming. It explains the basic programming concepts and the fundamentals of Python programming language. It includes numerical mathematics component that offers opportunities to help transfer students' mathematical experience into the understanding of computing. This course also addresses the following key themes in contextualised scenarios:

- Probability and Statistics
- Mathematics and Digital Application
- Financial Mathematics
- Popular Culture and Creativity

This course combines both essential concepts of computing and the application of mathematical reasoning for problem-solving. The transferable skills across computing, mathematics, design and technology, and other subject areas are highlighted in the course.

Lesson	Area of Enquiry
01. <i>Arithmetic with Python</i>	(Python Introduction)
02. <i>Text and Numbers in Python</i>	
03. <i>Triangle Area Calculator</i>	<b>Mathematics and Digital Application</b>
04. <i>Body Mass Index Calculator</i>	
05. <i>Buy Low, Sell High</i>	<b>Financial Mathematics</b>
06. <i>Repay Loans</i>	
07. <i>Toss a Coin</i>	<b>Probability and Statistics</b>
08. <i>Guess the Number</i>	
09-10. <i>Python Quizzes</i>	(Physical Computing)
11. <i>Data Protection and Passwords</i>	
12. <i>Normal Distribution</i>	
13. <i>Data Storage</i>	
14. <i>Remix Culture</i>	

## Course Goals

On completing the course, students will:

- acquire and master fundamental knowledge and skills that are required to use Python programming language
- employ computational thinking and modelling skills to analyse real life problems, create computational artefacts, and develop step-by-step solutions to the problems
- transfer their acquired knowledge and skills to computing modelling and abstraction within real life scenarios

## Target Audience

**Age groups:** 11~14 years old (12~13 years old)

- UK: Key Stage 3
- US: Grade 6–8

## Level of Difficulty

Introductory, no prior programming background is required

## Effort

45 minutes per lesson, 14 lessons in total for one semester/term

## Lesson 1    Arithmetic with Python

**Category:** Python

**Level:** Introductory

**Time Frame:** 45 minutes

**Core Subject Area:** Computing

**Supplementary Subject Area:** Mathematics

**Ages:** 11~14 years old

**Year Groups:** Key Stage 3 (UK) / Grades

6–8 (US)

### Overview

This lesson will introduce the textual programming language – Python. Python looks more human-readable than other textual programming languages such as Java and C++. Students will gain an initial conceptual understanding of Python and try to write their first Python code to do calculations and solve problems.

### Key Focus

- Features of Python programming language
- Layout of the mBlock Python editor

### Intended Learning Outcomes

*By the end of this lesson, students will be able to:*

- Recognise Python programming language, its IDE and code editor
- Write and execute Python programs to solve mathematical problems



## Content Standards

(UK)

### **National Curriculum in England – Computing Programmes of Study: Key Stage 3**

- Use two or more programming languages, at least one of which is textual, to solve a variety of computational problems

### **National Curriculum in England – Mathematics Programmes of Study: Key Stage 3**

- Select and use appropriate calculation strategies to solve increasingly complex problems





## Preparation

### For the teacher:

- A laptop or desktop with mBlock Python editor installed  
(Available from <https://python.mblock.cc/>)
- Lesson plan
- Worksheet

### For students:

- Laptops or desktops with mBlock Python editor installed  
(Available from <https://python.mblock.cc/>)
- Worksheet

## Procedures

### Section 1 *What is Python?* (8 minutes)

**Step 1.1** Explain to the class that they will learn a textual programming language named **Python** in this course.

- **Say:** This is the icon of Python. Don't you think it looks like two snakes twisting together?

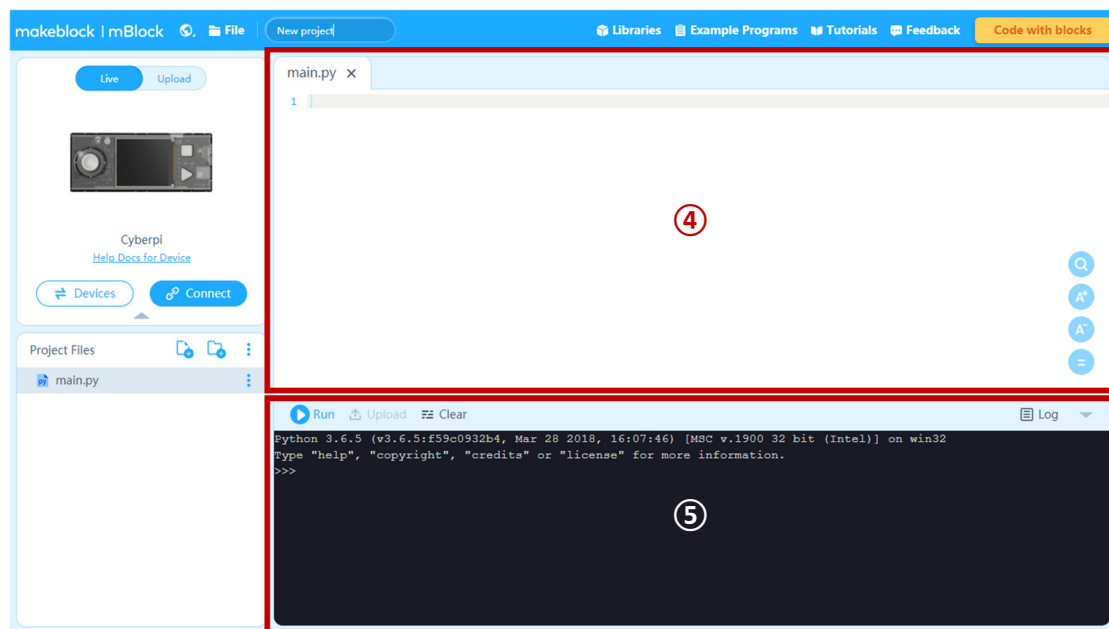
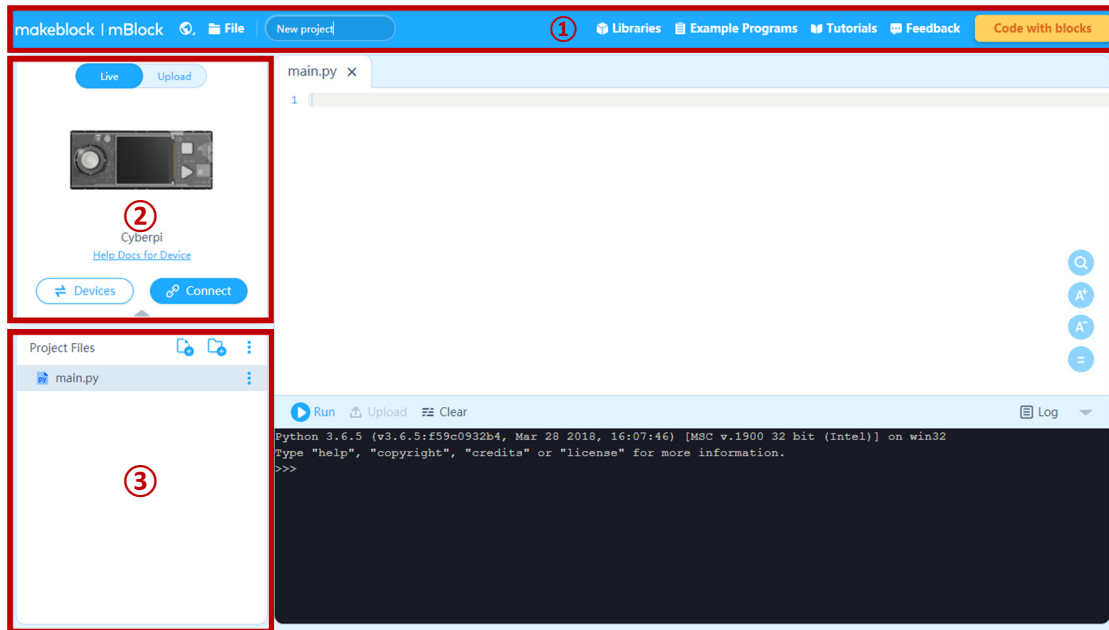


- Python is an interpreted, high-level and general-purpose programming language. It was first released in 1991, created by Guido van Rossum and was designed to be easily readable. It helps programmers write clear and logical code.

**Step 1.2** Show some example Python code to the class. Have students read and try to understand the code. Demonstrate the human-readable feature of Python code.

**Step 1.3** Introduce the layout of the mBlock Python editor. It has five main areas:

- |                            |                |
|----------------------------|----------------|
| ① Navigation Bar           |                |
| ② Devices Management Area  | ④ Editing Area |
| ③ Projects Management Area | ⑤ Console      |



- Instruct students to click on **Tutorials** and select **Quick Start Guide** in the menu. Have them walk through the layout of the mBlock Python editor.

## Section 2 First Python Code (12 minutes)

**Step 2.1** Instruct students to enter the mBlock Python editor and write the code to introduce their names or ask questions.

- **Say:** Let's have a try and write our first code in Python. Write in the Python editor what your name is – for example, 'My name is Harry'.
- Instruct students to use the **print()** function and write the code. Ask them to pay attention to the use of double quotes (") or single quotes (').

- Show the example code:

```
print("My name is Harry")
```

**Say:** Click **Run** and see what happens.

**Step 2.2** Ask students to write another code to ask a question – for example, 'What's your name?'

- Show the example code:

```
print("What's your name?")
```

**Say:** Click **Run** and see what happens.

- Students may notice that in the console two sentences appear:

```
My name is Harry
```

```
What's your name?
```

Remind students that they could click **Clear** to delete the previous content.

- **Ask:** What would be the imaginary Python's name? Give a name to your Python and write a third line of code – for example, 'I'm ...'.

### Example 1-1

```
1. print("My name is Harry")
2. print("What's your name?")
3. print("I'm Tom Riddle")
```

**Section 3 Python Calculator (20 minutes)**

**Step 3.1** Instruct students to explore how to use Python to do some calculations.

- **Say:** Let's use Python as a calculator and do some simple calculations.
- Introduce the common arithmetic operators in Python:

**Table 1-1 Arithmetic Operators**

Example	Symbol	Name	Python Representation
$5 + 6 = 11$	+	Addition	+
$11 - 5 = 6$	-	Subtraction	-
$5 \times 2 = 10$	$\times$	Multiplication	*
$10 \div 5 = 2$	$\div$	Division	/
$5^2 = 25$	$x^n$	Exponentiation	**
$11 \bmod 5 = 1$		Modulo	%

**Step 3.2** Have students complete the tasks as follows:

- **Task 1:** Translate the above listed expressions into Python code. Use Python to calculate the results.

**Tip:** Use the '`print()`' function to solve this task.

**Example 1-2**

```
1. print(5 + 6)
2. print(11 - 5)
3. print(5 * 2)
4. print(10 / 5)
5. print(5 ** 2)
6. print(11 % 5)
```

- **Task 2:** Calculate the results of the expressions below:

$$181 + 125 + 669 = ?$$

$$2160 - 439 - 57 = ?$$

$$79 \times 21 \times 3 \times 108 = ?$$

$$40257 \div 1917 = ?$$

$$156^4 = ?$$

$$40257 \text{ modulo } 41 = ?$$

**Example 1-3**

```
1. print(181 + 125 + 669)
2. print(2160 - 439 - 57)
3. print(79 * 21 * 3 * 108)
4. print(40257 / 1917)
5. print(156 ** 4)
6. print(40257 % 41)
```

- **Task 3:** Suppose your current computing skill level is 100. If you spend 1 hour per day in practising Python, your skill level will increase by 1%. For example, on Day 1, the increase is  $100 \times 0.01 = 1$ , so your new skill level would be  $100 \times 1.01 = 101$ . On Day 2, you also practise Python for 1 hour so you improve another 1%, which means it would be  $101 \times 1.01 = 102.01$ . Create the Python code to calculate the level of your computing skills if you keep doing this for 7 days. What will happen after 180 days (and 365 days)?

**Example 1-4**

```
1. print(100 * 1.01**7)
2. print(100 * 1.01**180)
3. print(100 * 1.01**365)
```

**Note:** Explain the concept of compound interest and invite students to use Python to solve a compound interest problem, for example:

Suppose you deposit £100 for 2 years in a bank account. The deposit interest rate is 0.01% and the interest is compounded monthly.

How much will you get after 1 month, 6 months, 1 year, and 2 years?

Use Python to calculate your balance and see how much you can earn.



## Section 4 Recap (5 minutes)

### Step 4.1 Summarize this lesson.

- Review the layout of the mBlock Python editor. Reiterate that students can write code in the **Editing Area** and when they run a program, the result will be shown in **Console**.
- Review the use of the `'print()'` function. Highlight the difference between the `'print("My name is Harry")'` and the `'print(5+6)'`.

If the `'print()'` function handles the text, it should contain a pair of double or single quotes; if the function processes the numerical data, there is no need to add the quotes.

**Reflection**

What I Know	What I Wonder	What I Learned
<ul style="list-style-type: none"><li>• Send text messages</li><li>• Use a calculator</li></ul>	<ul style="list-style-type: none"><li>• What is Python</li></ul>	<ul style="list-style-type: none"><li>• mBlock Python editor interface</li><li>• Python scripts</li><li>• The print() function</li><li>• Arithmetic operators</li></ul>

**Extension**

Have students research compound interest products of a bank (or other financial institution). Ask students to calculate the interest through Python and see how compound interest works.



## Lesson 2 Text and Numbers in Python

**Category:** Python

**Level:** Introductory

**Time Frame:** 45 minutes

**Core Subject Area:** Computing

**Supplementary Subject Area:** Mathematics

**Ages:** 11~14 years old

**Year Groups:** Key Stage 3 (UK) / Grades 6–8

(US)

### Overview

This lesson will introduce some of the fundamental data types in Python used to store and manage data, including the string, the integer, and the floating-point number. Students will explore how the mBlock Python editor reads the input data and converts the data into different data types. Students will also learn how to use the **'type()'** function to identify the data type of the input information and data.

### Key Focus

- Data types (strings, integers, floating-point numbers) and their conversion
- User input handling

### Intended Learning Outcomes

*By the end of this lesson, students will be able to:*

- Recognise the common data types in Python by using the **'type()'** function
- Write and execute Python programs to gather input data and convert them into appropriate data types



## Content Standards

### (UK)

#### **National Curriculum in England – Computing Programmes of Study: Key Stage 3**

- Understand several key algorithms that reflect computational thinking
- Use two or more programming languages, at least one of which is textual, to solve a variety of computational problems

### (US)

#### **CSTA K-12 Computer Science Standards: Grades 6~8**

- **2-AP-11:** Create clearly named variables that represent different data types and perform operations on their values.



## Preparation

### For the teacher:

- A laptop or desktop with mBlock Python editor installed  
(Available from <https://python.mblock.cc/>)
- Lesson plan
- Worksheet

### For students:

- Laptops or desktops with mBlock Python editor installed  
(Available from <https://python.mblock.cc/>)
- Worksheet

## Example Program

```
1. print("What's your name?")
2. print(type("What's your name?"))
3. # String
4.
5. print(2160 - 439 - 57)
6. print(type(2160 - 439 - 57))
7. # Integer
8.
9. print(40257 / 1917)
10. print(type(40257 / 1917))
11. # Floating Point Number
12.
13. input("How old are you? Your answer: ")
14. print(type(input("How old are you? Your answer: ")))
15. # The input() function returns a string
16.
17. int(input("How old are you? Your answer: "))
18. # Convert a string to an integer
19.
20. float(input("How tall are you? Your answer:(metre) "))
21. # Convert a string to a floating point number
```

**Note:** Familiarise yourself with the key points that you are going to teach in this lesson:

- Three Python data types: the string [**str()**], the integer [**int()**], and the floating-point number [**float()**]
- The '**type()**' function and '**input()**' function



## Pre-assessment

Have students fill out the **'What I Know'** column of the **K-W-L chart** before the class.

What I Know	What I Wonder	What I Learned
<ul style="list-style-type: none"><li>• mBlock Python editor interface</li><li>• Python scripts</li><li>• The <b>'print()'</b> function</li><li>• Arithmetic operators</li></ul>		

## Procedures

### Section 1 Revision (5 minutes)

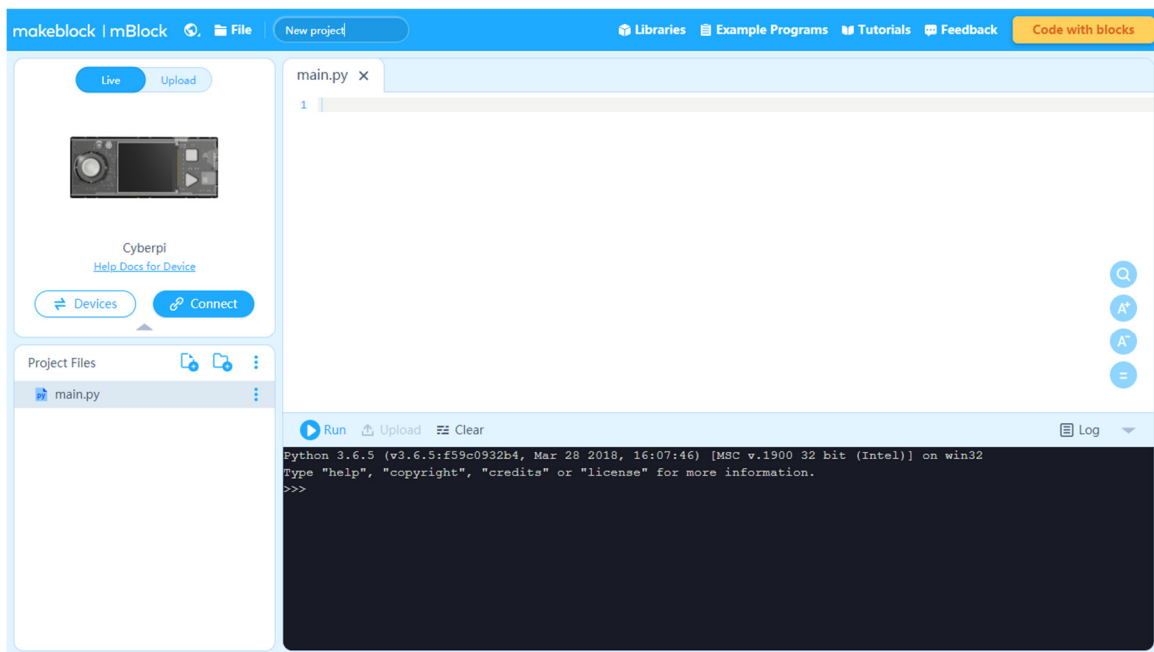
**Step 1.1** Ask students to reflect upon what they have learned about the mBlock Python editor and the Python programming language.

- **Ask:** What do you recall from the last lesson? Please think about the following questions:

Look at the interface, where can I write and edit Python code?

How can I open a Python project file?

What is this 'black area' called? What is it used for?



- Instruct students to use the **'print()'** function and write the code in one line (or several lines).

Ask them to write down what they think Python can do – for example, 'Python can add, subtract, multiply, and divide numbers' or 'Python can read my sentences'.

```
print("Python can add, subtract, multiply, and divide numbers")
```

- Briefly summarize the previous lesson.

### Section 2 Python Data Types (15 minutes)

**Step 2.1** Instruct students to compare the following two lines of code and figure out the difference between them:

```
print("What's your name?")
```

```
print(79 * 21 * 3 * 108)
```

- **Say:** In the previous lesson, we used the **'print()'** function to write sentences and do calculations. This function can handle both text and numbers.
  - **Ask:** Look at the two example codes. What is the difference between them?
  - **Explain:** If the **'print()'** function handles text, it should contain a pair of double or single quotes; if it processes numerical data, there is no need to add the quotes.
- **Step 2.2** Introduce the **'type()'** function and instruct students to identify the data type by using the **'type()'** function.
    - **Say:** Let's further explore the question by using a new function – the **'type()'** function.
    - Instruct students to rewrite the following example code by adding the **'type()'** function inside the **'print()'** statement (rewrite the code with both the **'print()'** statement and **'type()'** statement after the line of example code):

```
print("What's your name?")
```

```
print(2160 - 439 - 57)
```

```
print(40257 / 1917)
```

```
print(40257 % 41)
```

#### Example 2-1

```
1. print(type("What's your name?"))
2. print(type(2160 - 439 - 57))
3. print(type(40257 / 1917))
4. print(type(40257 % 41))
```

#### Example 2-2

```

1. print("What's your name?")
2. print(type("What's your name?"))
3.
4. print(2160 - 439 - 57)
5. print(type(2160 - 439 - 57))
6.
7. print(40257 / 1917)
8. print(type(40257 / 1917))
9.
10. print(40257 % 41)
11. print(type(40257 % 41))

```

- **Explain:** The `'type()'` function can identify the data type of a line of code. For example, the text data 'What's your name?' is a string.

The numerical data returned by the `'print(2160 - 439 - 57)'` or `'print(40257 % 41)'` are integers. The numeric data displayed by the `'print(40257 / 1917)'`, however, is a floating-point number.

In Python (Python 3), division of two integers will produce a floating-point number. Although 40257 divided by 1917 equals 21 which is an integer, in Python it returns the value '21.0'.

- Instruct students to convert the floating-point number into an integer by modifying the example code `'print(40257 / 1917)'`.

### Example 2-3

```

1. print(int(40257 / 1917))
2. print(type(int(40257 / 1917)))

```

**Step 2.3** Have students complete the task individually:

- **Task 1:** Calculate the expressions below and determine the data types of the results:

`print(100000 + 365.12)`       $100000 + 365.12 = ?$

`print(1989.12 - 917.8)`       $1989.12 - 917.8 = ?$

`print(36.6 * 50.1)`       $36.6 \times 50.1 = ?$

`print(100.25 * 40)`       $100.25 \times 40 = ?$



```
print(3.3 ** 10)
```

 $3.3^{10} = ?$ 

```
print(2501 % 2.5)
```

 $2501 \bmod 2.5 = ?$ 

#### Example 2-4

```
1. print(100000 + 365.12)
2. print(type(100000 + 365.12))
3.
4. print(1989.12 - 917.8)
5. print(type(1989.12 - 917.8))
6.
7. print(36.6 * 50.1)
8. print(type(36.6 * 50.1))
9.
10. print(100.25 * 40)
11. print(type(100.25 * 40))
12.
13. print(3.3 ** 10)
14. print(type(3.3 ** 10))
15.
16. print(2501 % 2.5)
17. print(type(2501 % 2.5))
```

- Instruct students to summarize the rules:

Data Type	Operation	Data Type	Data Type of Result
Integer	+ –	Integer	<input type="checkbox"/> Integer <input type="checkbox"/> Floating-point number
Integer		Floating-point	<input type="checkbox"/> Integer <input type="checkbox"/> Floating-point number
Floating-point		Floating-point	<input type="checkbox"/> Integer <input type="checkbox"/> Floating-point number
Integer	*	Integer	<input type="checkbox"/> Integer <input type="checkbox"/> Floating-point number
Integer		Floating-point	<input type="checkbox"/> Integer <input type="checkbox"/> Floating-point number
Floating-point		Floating-point	<input type="checkbox"/> Integer <input type="checkbox"/> Floating-point number
Integer	/	Integer	<input type="checkbox"/> Integer <input type="checkbox"/> Floating-point number
Integer		Floating-point	<input type="checkbox"/> Integer <input type="checkbox"/> Floating-point number
Floating-point		Floating-point	<input type="checkbox"/> Integer <input type="checkbox"/> Floating-point number
Integer	**	N/A	<input type="checkbox"/> Integer <input type="checkbox"/> Floating-point number
Floating-point			<input type="checkbox"/> Integer <input type="checkbox"/> Floating-point number
Integer	%	Integer	<input type="checkbox"/> Integer <input type="checkbox"/> Floating-point number
Integer		Floating-point	<input type="checkbox"/> Integer <input type="checkbox"/> Floating-point number

Tick the box to record the data type of the result produced by the Python Calculator.

Summarize and write down the rules.

### Section 3 Data Types Conversion (20 minutes)

**Step 3.1** Distribute the example program file and instruct students to complete the task below:

- **Task 2:**

```
1. print(type(input("What's your name? Your answer: ")))  
2. print(type(input("How old are you? Your answer: ")))  
3. print(type(input("How tall are you? Your answer:(metre) ")))
```

Give students a few minutes to read the script.

**Ask:** There is a new function. Can you find it? Guess what this function is used for and how it works in the program.

**Step 3.2** Introduce the use of the **'input()'** function and have students predict its effect.

- **Explain:** The new function is **'input()'**. It can read the information a user input in a line and store the information for processing.
- **Ask:** Before you run the program, guess the data type of the results from the three statements.

Think about the data type of the input and answer to the three questions.

**Note:** The answer for 'What's your name?' could be a string; the answer for 'How old are you?' could be an integer; the answer for 'How tall are you?' could be a floating-point number.

How will the **'input()'** function handle the input information? Will the **'input()'** function convert the data type of the input data?

Write down your predictions and then test your assumption.

**Step 3.3** Instruct students to investigate the data type of the output returned by the 'input()' function.

Script	Prediction	Test Result
<code>print(type(input("What's your name? Your answer: ")))</code>		
<code>print(type(input("How old are you? Your answer: ")))</code>		
<code>print(type(input("How tall are you? Your answer:(metre) ")))</code>		

- **Explain:** The 'input()' function will convert the input data into a string.
- **Ask:** How can you convert them into the data type you expected?

#### Example 2-5

```
1. print(type(input("What's your name? Your answer: ")))
2. print(type(int(input("How old are you? Your answer: "))))
3. print(type(float(input("How tall are you? Your answer:(metre) "))))
```

**Step 3.4** Distribute the example program file and instruct students to complete the task below:

- **Task 3:** Convert the data types of input information by adding the relevant syntax:

```
1. print(type(input("Your school's postcode: ")))
2. print(type(input("Your school's name: ")))
3. print(type(input("Street/Block number nearby: ")))
4. print(type(input("Your body temperature:(°C) ")))
5. print(type(input("Today's temperature:(°C) ")))
6. print(type(input("Your lucky number: ")))
```

- For example:

```
print(type(input("1 Pound Sterling = ? Euro: ")))
```

The input data (e.g., '1.20') should be converted into a floating-point number:

```
print(type(float(input("1 Pound Sterling = ? Euro: "))))
```

- Instruct students to add the 'int()' or 'float()' into the original code and convert the data type of input information.

**Step 3.5** Based on Task 3, organise students to have an open-ended discussion.

- **Ask:** Suppose that if the input data is a numerical value (e.g. the postcode, the street or block number, or a lucky number of someone), in what situation should we convert it into an integer



(or a floating-point number), or should we just leave it as a string?

**Note:** For example, a postcode or street number can be a string, as we do not use it to do calculations. The daily temperature should be converted into the floating-point number if we want to calculate the average temperature by dividing the sum of measured temperatures by the number of measurements.

- Ask students to express their opinions and explain the reasons.

**Section 4    Recap (5 minutes)****Step 4.1** Summarize this lesson.

- Review the three data types learned in this lesson: the string, the integer, and the floating-point number. Review the **'type()'** function that can identify the data type.
- Review the conversion method and syntaxes: **int()**, **float()**, [and **str()**].
- Review the use of the **'input()'** function that can read and store the input information from the user.
- Have students fill out the **K-W-L chart**.

What I Know	What I Wonder	What I Learned
<ul style="list-style-type: none"><li>● Python interface</li><li>● Python scripts</li><li>● The <b>'print()'</b> function</li><li>● Arithmetic operators</li></ul>	<ul style="list-style-type: none"><li>● What is a string?</li><li>● What is an integer?</li><li>● What is a floating-point number?</li><li>● How to handle user input?</li></ul>	<ul style="list-style-type: none"><li>● Python data types</li><li>● Data type conversion</li><li>● The <b>'type()'</b> function</li><li>● The <b>'input()'</b> function</li></ul>

## Lesson 3 Triangle Area Calculator

**Category:** Python

**Level:** Introductory

**Time Frame:** 45 minutes

**Core Subject Area:** Computing

**Supplementary Subject Area:** Mathematics

**Ages:** 11~14 years old

**Year Groups:** Key Stage 3 (UK) / Grades

6–8 (US)

### Overview

In this lesson, students will explore how to calculate the area of a triangle in Python by using the Heron's formula. Students need to utilize the Python knowledge they have learned in the previous learning activities to handle this geometric problem. This lesson will also introduce variables in Python, as well as the good practice of writing comments while writing Python code.

### Key Focus

- Use of variables
- Python mathematical operations

### Intended Learning Outcomes

*By the end of this lesson, students will be able to:*

- Explain that a variable is a named location used to store and manage data
- Operate variables and appropriate arithmetic operators to process data and write formulae
- Write and execute Python programs to calculate the perimeter and area of a triangle by using appropriate formulae in geometry



## Content Standards

(UK)

### National Curriculum in England – Computing Programmes of Study: Key Stage 3

- Design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems
- Use logical reasoning to compare the utility of alternative algorithms for the same problem
- Use two or more programming languages, at least one of which is textual, to solve a variety of computational problems

### National Curriculum in England – Mathematics Programmes of Study: Key Stage 3

- Develop their mathematical knowledge, in part through solving problems and evaluating the outcomes, including multi-step problems
- Use a calculator and other technologies to calculate results accurately and then interpret them appropriately
- Derive and apply formulae to calculate and solve problems involving: perimeter and area of triangles, parallelograms, trapezia, volume of cuboids (including cubes) and other prisms (including cylinders)
- Derive and illustrate properties of triangles, quadrilaterals, circles, and other plane figures [for example, equal lengths and angles] using appropriate language and technologies





(US)

**CSTA K-12 Computer Science Standards: Grades 6~8**

- **2-AP-11:** Create clearly named variables that represent different data types and perform operations on their values.
- **2-AP-16:** Incorporate existing code, media, and libraries into original programs, and give attribution.

**Common Core State Standards Mathematics Standards: Grades 6~8**

- **CCSS.MATH.CONTENT.6.G.A.1:** Find the area of right triangles, other triangles, special quadrilaterals, and polygons by composing into rectangles or decomposing into triangles and other shapes; apply these techniques in the context of solving real-world and mathematical problems.
- **CCSS.MATH.CONTENT.7.G.B.6:** Solve real-world and mathematical problems involving area, volume and surface area of two- and three-dimensional objects composed of triangles, quadrilaterals, polygons, cubes, and right prisms.



## Preparation

### For the teacher:

- A laptop or desktop with mBlock Python editor installed  
(Available from <https://python.mblock.cc/>)
- Lesson plan
- Worksheet

### For students:

- Laptops or desktops with mBlock Python editor installed  
(Available from <https://python.mblock.cc/>)
- Worksheet

## Example Program

```
1. print("Triangle Area Calculator")
2.
3. a = float(input("Value of side a:(centimetre) "))
4. b = float(input("Value of base b:(centimetre) "))
5. c = float(input("Value of side c:(centimetre) "))
6. # Enter the values of the sides a, b, and c
7.
8. s = (a + b + c) / 2
9. # Calculate the semi-perimeter
10.
11. area = (s * (s-a) * (s-b) * (s-c)) ** 0.5
12. # Calculate the area - Heron's Formula
13.
14. print("The area of this triangle is", area, "square centimetres.")
```

**Note:** Familiarise yourself with the key points that you are going to teach in this lesson:

- How to create a variable in Python – e.g. `x = 1`; `x = input("What's your name? ")`
- Arithmetic operators in Python
- The Heron's formula:  $A = \sqrt{s(s-a)(s-b)(s-c)}$
- Concatenate strings and numeric data in the `'print()'` function



## Task for Enquiry – Geometry

*How to find the area of a triangle if only three sides is given?*

- If we know the base and height of a triangle, we can calculate the area of the triangle by using this formula:

$$Area = \frac{base \times height}{2}$$

This formula is a simple way we have learned in mathematics.

- However, what if we don't know the height of a triangle, but rather all its three sides, can we still calculate the area?



## Pre-assessment

Have students fill out the **'What I Know'** column of the **K-W-L chart** before the class.

What I Know	What I Wonder	What I Learned
<ul style="list-style-type: none"><li>• Python data types</li><li>• Data type conversion</li><li>• The <b>'type()'</b> function</li><li>• The <b>'input()'</b> function</li></ul>		

## Procedures

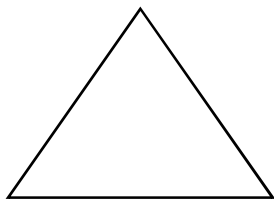
### Section 1 Introduction: *How to calculate the area of a triangle?* (5 minutes)

**Step 1.1** Have students recall how to calculate the area of a triangle.

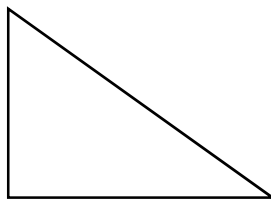
- **Ask:** What factors do you need to calculate the area of a triangle?

**Note:** The base and height of the triangle.

- Show the three types of triangles:



Acute Triangle



Right Triangle

Obtuse Triangle

**Ask:** How do you calculate the area of a triangle?

**Tip:**  $Area(A) = \frac{base \times height}{2}$

- **Ask:** What if you only know the length of the three sides? Can you calculate the area?

**Step 1.2** Briefly explain Heron's formula:

**Semi-perimeter:**  $S = \frac{a+b+c}{2}$ ,

**Area:**  $A = \sqrt{s(s-a)(s-b)(s-c)}$ .

- **Say:** There is another method to calculate the triangle area by using the given three sides. The method was developed by an ancient Greek mathematician Hero of Alexandria (10 AD – 70 AD).
- **Explain:** You can use Heron's formula to find the area of a triangle using the length of three sides. First, you need to calculate the half of the perimeter of the triangle (represented by the 's').

Then, use this formula:  $A = \sqrt{s(s-a)(s-b)(s-c)}$  to calculate the area.

**Section 2    Use (15 minutes)**

**Step 2.1**        Distribute the example program file to the class. Have students read the example program and run it.

- **Say:** Let's use Python to calculate the area of a triangle with Heron's formula.

This is a Triangle Area Calculator program. First, have a close look at it. Then, enter the value of the sides of the triangles in the program to do calculations.

- Have students read the script:

```
1. print("Triangle Area Calculator")
2.
3. a = float(input("Value of side a:(centimetre) "))
4. b = float(input("Value of base b:(centimetre) "))
5. c = float(input("Value of side c:(centimetre) "))
6.
7. s = (a + b + c) / 2
8.
9. area = (s * (s-a) * (s-b) * (s-c)) ** 0.5
10.
11. print("The area of this triangle is", area, "square centimetres.")
```

- Instruct students to use the Triangle Area Calculator to calculate the area of the triangle given in the worksheet.
- Instruct students to think about the questions below:
  - Why is the **'float()'** function used together with the **'input()'** function?
  - How does the program remember the values of three sides of a triangle?
  - How to calculate the semi-perimeter in Python?
  - How to calculate the area in Python?
  - Look at the final line of the **'print()'** statement. Do you notice anything different?

**Step 2.2** Have students discuss the above questions and report what they have found while using the Triangle Area Calculator.

- *'Why is the **'float()'** function used together with the **'input()'** function?'*

**Note:** This question is about the data type conversion learned in the previous lesson. The **'input()'** function converts the input data into a string. To calculate the sum of the three sides, the input data should be converted into numerical data.

- *'How does the program remember the values of three sides of a triangle?'*

**Note:** This question is about the use of variables, which will be explained later. At this stage, students only need to identify the expression **'a/b/c/s/area = ...'** in association with their algebra knowledge.

- *'How to calculate the semi-perimeter in Python?'*

**Note:** Students need to point out the seventh line of code.

- *'How to calculate the area in Python?'*

**Note:** Students need to point out the ninth line of code. To extract the square root in Python, one of the methods is to use the exponent, i.e. square root = number \*\* 0.5.

- *'Look at the final line of the **'print()'** statement. Do you notice anything different?'*

**Note:** The **'print()'** statement here contains both the string and the floating-point number printed in the same line. This example extends students' understanding of the **'print()'** function.

**Step 2.3** Explain the use of variables.

- **Say:** In the example program, we use **'a'**, **'b'**, **'c'**, **'s'**, and **'area'** to represent the values of the three sides, the semi-perimeter, and the area of the triangle. These symbols are variables.

- **Explain:** A variable is a named location used to store data. **'a'**, **'b'**, **'c'**, **'s'**, and **'area'** are the names of the variables.

The value of a variable can be changed. For example:



```
x = 1
```

```
x = x + 1
```

The variable 'x' equals 2. You can assign a new value to the variable, and when you do this, the old value will be replaced.

**Step 2.4** Instruct students to annotate the example program.

- **Say:** We can comment on the program to describe each step of Heron's formula. Comments are notes that explain what each part of the program is doing so that people can understand the logic behind the program. Comments will not be run in Python as they are not part of the code.
- **Say:** In Python, a comment starts with a hashtag (#). Comments should be concise.
- Instruct students to write comments inside the example program.

**Example:**

```
1. print("Triangle Area Calculator")
2.
3. a = float(input("Value of side a:(centimetre) "))
4. b = float(input("Value of base b:(centimetre) "))
5. c = float(input("Value of side c:(centimetre) "))
6. # Enter the values of the sides a, b, and c
7.
8. s = (a + b + c) / 2
9. # Calculate the semi-perimeter
10.
11. area = (s * (s-a) * (s-b) * (s-c)) ** 0.5
12. # Calculate the area - Heron's Formula
13.
14. print("The area of this triangle is", area, "square centimetres.")
```

### Section 3 Modify (10 minutes)

**Step 3.1** Have students complete the following tasks:

- **Task 1:** Create another Triangle Area Calculator that calculates the area of a triangle with the base and height.

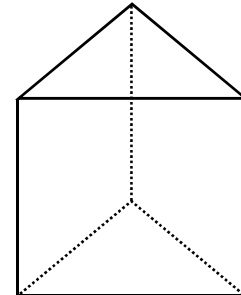
#### Example – Task 1

```
1. print("Triangle Area Calculator")
2.
3. b = float(input("Value of the length of the base:(centimetre) "))
4. h = float(input("Value of the height:(centimetre) "))
5.
6. area = b * h * 0.5
7.
8. print("The area of this triangle is", area, "square centimetres.")
```

- **Task 2:** Based on the Triangle Area Calculator, create a Triangular Prism Volume Calculator.

**Formula:**  $Volume = area\ of\ base\ (triangle) \times Height$

**Note:** You can modify the Triangle Area Calculator program either created in Task 1 or the given example by the teacher in the previous section.



#### Example – Task 2

```
1. print("Triangular Prism Volume Calculator")
2.
3. b = float(input("Value of the length of the base:(centimetre) "))
4. th = float(input("Value of the triangle height:(centimetre) "))
5. ph = float(input("Value of the prism height:(centimetre) "))
6.
7. area = b * th * 0.5
8.
9. volume = area * ph
10.
11. print("The volume is", volume, "cubic centimetres.")
```

- **Task 3:** Create an application to calculate the capacity of a milk carton. Remember, you need

to use and convert the unit of measurement for liquids.

**Note:** The milk carton is in the shape of a rectangular prism.

### Example – Task 3

```
1. print("Measure and calculate the volume of your milk carton:")
2. # Calculate the volume of a rectangular prism
3.
4. l = float(input("Length:(centimetre) "))
5. w = float(input("Width:(centimetre) "))
6. h = float(input("Height:(centimetre) "))
7.
8. volume = l * w * h
9.
10. mL = volume
11. # Convert cubic centimetres to millilitres
12.
13. print("My milk carton can contain about", mL, "mL of milk.")
```

**Section 4    Create (15 minutes)**

**Step 4.1**      Have students create a calculator application in Python that can calculate the perimeter and area of a learned plane shape as well as the volume of a solid shape, such as a prism or prism-like figure.



## Section 5 Recap (5 minutes)

**Step 5.1** Summarize this lesson.

- Review the meaning and use of the variable.
- Review the rules of writing comments in the mBlock Python editor.
- Review the process of creating a calculator application in Python.
- Have students fill out the **K-W-L chart**.

What I Know	What I Wonder	What I Learned
<ul style="list-style-type: none"><li>● Python data types</li><li>● Data type conversion</li><li>● The <b>'type()'</b> function</li><li>● The <b>'input()'</b> function</li></ul>	<ul style="list-style-type: none"><li>● What is a variable?</li><li>● How to do calculations in Python?</li></ul>	<ul style="list-style-type: none"><li>● Variables</li><li>● Arithmetic operations in Python</li></ul>

## Lesson 4    Body Mass Index Calculator

**Category:** Python

**Level:** Introductory

**Time Frame:** 45 minutes

**Core Subject Area:** Computing

**Supplementary Subject Area:** Mathematics

**Ages:** 11~14 years old

**Year Groups:** Key Stage 3 (UK) / Grades 6–8

(US)

### Overview

This lesson will introduce the **if** statement in Python through a BMI Calculator application. The **if** statement is used to assess information and choose one course of action or another for decision making. This construct processes conditional execution of a statement or a group of statements according to the value of an expression. Students will learn about conditional algorithms and develop a BMI Calculator to assess their weight status.

### Key Focus

- Selection constructs (if statements)
- The off-side rule for writing Python code (indentation)

### Intended Learning Outcomes

*By the end of this lesson, students will be able to:*

- Evaluate and explain computational abstraction through a BMI Calculator application
- Explain the concept of selection and recognise the conditional program structure in the given Python scripts
- Write and execute conditional algorithms to calculate the BMI and assess weight status



## Content Standards

(UK)

### **National Curriculum in England – Computing Programmes of Study: Key Stage 3**

- Design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems
- Use two or more programming languages, at least one of which is textual, to solve a variety of computational problems

### **National Curriculum in England – Mathematics Programmes of Study: Key Stage 3**

- Develop their mathematical knowledge, in part through solving problems and evaluating the outcomes, including multi-step problems
- Use the symbols  $=$ ,  $\neq$ ,  $<$ ,  $>$ ,  $\leq$ ,  $\geq$
- Use approximation through rounding to estimate answers and calculate possible resulting errors expressed using inequality notation  $a < x \leq b$
- Use a calculator and other technologies to calculate results accurately and then interpret them appropriately
- Understand and use the concepts and vocabulary of expressions, equations, inequalities, terms and factors

(US)

**CSTA K-12 Computer Science Standards: Grades 6~8**

- **2-AP-11:** Create clearly named variables that represent different data types and perform operations on their values.
- **2-AP-12:** Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.

**Common Core State Standards Mathematics Standards: Grades 6~8**

- **CCSS.MATH.CONTENT.6.EE.A.2:** Write, read, and evaluate expressions in which letters stand for numbers.
- **CCSS.MATH.CONTENT.6.EE.B.6:** Use variables to represent numbers and write expressions when solving a real-world or mathematical problem; understand that a variable can represent an unknown number, or, depending on the purpose at hand, any number in a specified set.
- **CCSS.MATH.CONTENT.6.EE.B.8:** Write an inequality of the form  $x > c$  or  $x < c$  to represent a constraint or condition in a real-world or mathematical problem. Recognize that inequalities of the form  $x > c$  or  $x < c$  have infinitely many solutions; represent solutions of such inequalities on number line diagrams.





## Preparation

### For the teacher:

- A laptop or desktop with mBlock Python editor installed  
(Available from <https://python.mblock.cc/>)
- Lesson plan
- Worksheet

### For students:

- Laptops or desktops with mBlock Python editor installed  
(Available from <https://python.mblock.cc/>)
- Worksheet

## Example Program

```
1. print("BMI Calculator")
2.
3. height = float(input("Height:(metre) "))
4. weight = float(input("Weight:(kilogram) "))
5. # Enter the values of the height and weight
6.
7. BMI = weight / (height**2)
8. print(BMI)
9. # Calculate the BMI value
10.
11. # Report the weight status:
12. if BMI >= 30.0:
13.     print("Obese")
14. if BMI >= 25.0 and BMI <= 29.9:
15.     print("Overweight")
16. if BMI >= 18.5 and BMI <= 24.9:
17.     print("Healthy Weight")
18. if BMI < 18.5:
19.     print("Underweight")
```

**Note:** Familiarise yourself with the key points that you are going to teach in this lesson:

- How to convert and calculate the input data
- The **if** statements and indentation
- Comparison operators in Python
- The concept and standards of the BMI



## Task for Enquiry – Measurement and Data; Health Management

*What is BMI? What is my BMI? What does my BMI mean?*

- The Body Mass Index (BMI) is a measure that uses your height and weight to work out if your weight is healthy. To calculate the BMI, you can divide your weight in kilograms by the square of your height in meters.
- A normal BMI is between 18.5 and 25; a person with a BMI between 25 and 30 is considered overweight, and a person with a BMI over 30 is considered obese. A person is considered underweight if their BMI is less than 18.5.
- Do you know your BMI? How about creating an application to calculate your BMI through Python programming?



## Pre-assessment

Have students fill out the 'What I Know' column of the **K-W-L chart** before the class.

What I Know	What I Wonder	What I Learned
<ul style="list-style-type: none"><li>• Variables</li><li>• Arithmetic operations</li></ul>		

## Procedures

### Section 1 Introduction: *What is BMI?* (3 minutes)

#### Step 1.1 Explain the concept of **BMI (Body Mass Index)**.

- **Say:** The **Body Mass Index** is a measure that uses your height and weight to work out if your weight is healthy. To calculate the BMI, you can divide your weight in kilograms by the square of your height in meters.
- Demonstrate how to calculate the BMI:  
Suppose a person's height is 1.72m and his weight is 60kg, his BMI would be  
 $60 \div 1.72^2 \approx 20.3$ .
- **Ask:** What does it mean to have a BMI of 20.3? Let's have a look at this table:

Table 4-1 BMI Healthy Weight (Adults)

BMI	Level	Meaning
Below 18.5	Underweight	Being underweight may lead to a weakened immune system and feeling tired.
18.5 – 24.9	Healthy Weight	You have a healthy weight for your height.
25.0 – 29.9	Overweight	You are heavier than is healthy for someone of your height. Try to lose weight by keeping a balanced diet and physical activity.
30.0 and Above	Obese	

Source: UK NHS, US CDC

**Explain:** The BMI of 20.3 indicates that this person has a healthy weight.

- **Ask:** How can we calculate the BMI more effectively?

#### Step 1.2 Explain to the class that in this lesson they will create a BMI calculator in Python, which can automatically calculate and indicate the BMI.

**Section 2 Predict (5 minutes)**

**Step 2.1** Distribute the example program file to the class. Have students read the code and discuss what they think it will do.

```
1. print("BMI Calculator")
2.
3. height = float(input("Height:(metre) "))
4. weight = float(input("Weight:(kilogram) "))
5.
6. BMI = weight / (height**2)
7. print(BMI)
8.
9. if BMI >= 30.0:
10.     print("Obese")
11.
12. if BMI >= 25.0 and BMI <= 29.9:
13.     print("Overweight")
14.
15. if BMI >= 18.5 and BMI <= 24.9:
16.     print("Healthy Weight")
17.
18. if BMI < 18.5:
19.     print("Underweight")
```

- Instruct students to write down their predictions and annotations in the code on the worksheet.

**Section 3 Run (2 minutes)**

**Step 3.1** Ask students to run the example program and check it against their predictions.

- Remind them that they could either use their own height and weight or the following data:

**Exercise: Calculate the BMI**

Name	Height (m)	Weight (kg)	BMI	Weight Status
(Your Name)				
Ai Fukuhara	1.55	48		
Angelina Jolie	1.69	54		
Arnold Schwarzenegger	1.88	113		
Benedict Cumberbatch	1.83	79		
Jackie Chan	1.70	65		
Kobe Bryant	1.98	96		
Małgorzata Dydek	2.18	101		
Maya Plisetskaya	1.77	59		
Sarah Hyland	1.57	45		
Serena Williams	1.75	70		
Tom Cruise	1.70	77		
Yao Ming	2.29	141		
Zoe	1.70	72		

- Instruct students to think about the questions below while running the program:
  - Did the program work in accordance with your predictions?
  - What is the data type?
  - How can you divide the BMI into ranges?

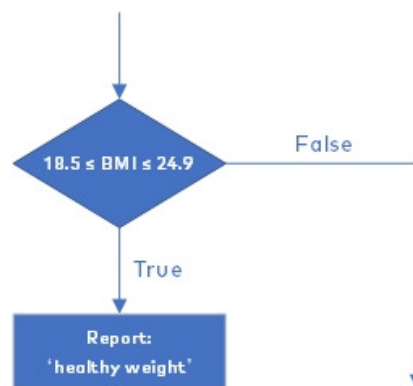
**Section 4 Investigate (15 minutes)****Step 4.1** Explain the use of if statements.

- **Say:** The BMI Calculator program uses selection statements (**if** statements) to divide the BMI into four ranges.

In Python, a selection statement (or selection construct) is used to test a condition and make a decision. To decide weight status represented by the BMI of 20.3 (or the height and weight data), the program works like this:

- *Ask the height and weight;*
- *Calculate the BMI by using the equation;*
- **IF** the result (20.3) is equal to or greater than 18.5 **AND** equal to or less than 24.9, **THEN** the program reports 'healthy weight'.
- **Ask:** Can you draw the flow of the BMI Calculator program?

Instruct students to draw the flowchart on their worksheet. Show an example of the selection construct flow for reference:

**Step 4.2** Briefly review the functions below:`float()``input()`



**Step 4.3** Have students work in pairs to complete the following tasks:

- **Task 1:** Comment the code on the worksheet. Briefly explain the function.
- **Task 2:** Investigate the conditions of the statements. Why does the program use the '≥' and '≤' operators to define the lowest and/or highest values of a range?

```
if BMI >= 30.0:
```

```
if BMI >= 25.0 and BMI <= 29.9:
```

```
if BMI >= 18.5 and BMI <= 24.9:
```

- **Task 3:** Investigate the indentation. Remove the leading whitespace before the `print("obese")/print("overweight")/print("healthy weight")/print("underweight")`, run the program, and see what happens. Does the program still work?
- **Task 4:** Calculate Zoe's BMI and output her weight status. Could the program decide on the weight status? Why or why not?

**Solution for Task 4:**

Input Zoe's height and weight: 1.70(m) and 72(kg).

The result would output like this:

```
BMI Calculator
```

```
height (meters): 1.70
```

```
weight (kilograms): 72
```

```
24.913494809688583
```

The program cannot decide whether the BMI of '24.913494809688583' should belong to 'healthy weight' or 'overweight' because the number does not fall into either range.

The `round()` function can solve this problem by rounding off a number to the specified number of digits:

```
round(parameter 1, parameter 2)
```

The 'parameter 1' is the number (or variable) to be rounded, while the 'parameter 2' represents the number up to which the given number is rounded.

Example of using the `round()` function:

```
BMI = 24.913494809688583
```

```
BMI = round(BMI,1)
```

```
print(BMI)
```

Therefore, to solve the problem in Task 4, we can modify the code in the fourth line like this:

```
BMI = round(weight / (height**2),1);
```

or add another line like this:

```
BMI = weight / (height**2)
```

```
BMI = round(BMI,1).
```

**Section 5    Modify and Make (20 minutes)**

**Step 5.1** Instruct students to modify the example program to create a BMI Calculator for children and teenagers.

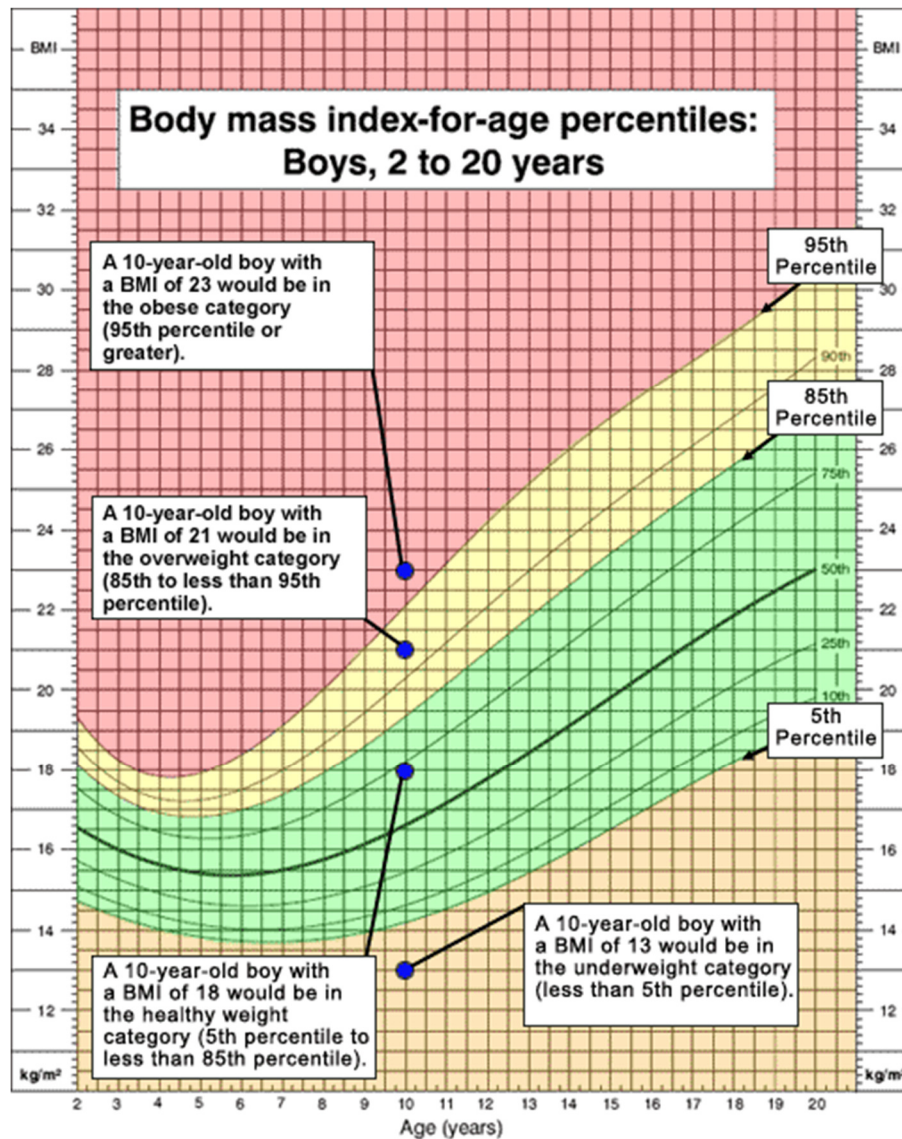
- **Say:** The example program of the BMI Calculator is for adults. Let's make a new one for teenagers.
- Base on the year group of the class, select an age group, for example, 13 years old. Explain to the class that they can create a BMI calculator application specifically for their age group.
- Instruct students to refer to the article *About Child & Teen BMI* available on:

[https://www.cdc.gov/healthyweight/assessing/bmi/childrens\\_bmi/about\\_childrens\\_bmi.html](https://www.cdc.gov/healthyweight/assessing/bmi/childrens_bmi/about_childrens_bmi.html)

**Table 4-2    BMI-for-age Weight Status**

Weight Status	Percentile Range
Underweight	Less than the 5 <sup>th</sup> percentile
Healthy Weight	5 <sup>th</sup> percentile to less than the 85 <sup>th</sup> percentile
Overweight	85 <sup>th</sup> to less than the 95 <sup>th</sup> percentile
Obese	Equal to or greater than the 95 <sup>th</sup> percentile

**Source:** US CDC



The above table and chart show how to rescale the BMI for teenagers.

**Note:**

If you are a boy, refer to this chart available on:

<https://www.cdc.gov/growthcharts/data/set1clinical/cj41l023.pdf>;

If you are a girl, refer to this chart available on:

<https://www.cdc.gov/growthcharts/data/set1clinical/cj41l024.pdf>.

- Take the 13-year-old boy as an example to demonstrate how to rescale the ranges of his BMI weight status categories:

- Find the '13 years old' from the axis 'Age (years)';
- Read the maximum and/or minimum values of each category, for example:

Table 4-3 BMI-for-13-year-age Weight Status

BMI	Weight Status	Percentile Range
Below 16.0 (BMI < 16.0)	Underweight	Less than the 5 <sup>th</sup> percentile
16.0 – 22.0 (16.0 ≤ BMI < 22.0)	Healthy Weight	5 <sup>th</sup> percentile to less than the 85 <sup>th</sup> percentile
22.0 – 25.0 (22.0 ≤ BMI ≤ 25.0)	Overweight	85 <sup>th</sup> to less than the 95 <sup>th</sup> percentile
25.0 and Above (BMI > 25.0)	Obese	Equal to or greater than the 95 <sup>th</sup> percentile

- Based on the above ranges, modify and test the BMI Calculator program.
- **Extension:** Encourage advanced learners to think about how to add another selection construct that can divide the BMI calculation into two paths taking gender into account.

**Step 5.2** Summarize the use of selection construct and reflect upon the BMI Calculator project.

**Section 6    Recap**

- Have students fill out the **K-W-L chart** after the class.

What I Know	What I Wonder	What I Learned
<ul style="list-style-type: none"><li>• Variables</li><li>• Arithmetic operations</li></ul>	<ul style="list-style-type: none"><li>• What is selection?</li><li>• The <b>if</b> statements</li></ul>	<ul style="list-style-type: none"><li>• The <b>if</b> statements</li><li>• Indentation</li><li>• Comparison operators</li><li>• Logical operators</li></ul>

**Extension**

**Activity 1:** Have students research the BMI and health management literature. Learn about how to advise on losing and gaining weight. Add some health advice responses in the BMI Calculator project. Informing the user on being aware of his/her weight status and some simple suggestions on health management.

**Read more:**

*What are the health benefits of losing weight?* (UK NHS, 2018)

<https://www.nhs.uk/common-health-questions/lifestyle/what-are-the-health-benefits-of-losing-weight>

**Activity 2:** Do the Black, Asian, and other minority ethnic groups have different standards of BMI health weight? Investigate this issue and figure out how to make the BMI Calculator more inclusive.



## **Appendix**

**The BMI Calculator created by the UK's NHS:**

<https://www.nhs.uk/live-well/healthy-weight/bmi-calculator>

**The BMI Calculator created by the US's CDC:**

**For children and teenagers:**

<https://www.cdc.gov/healthyweight/bmi/calculator.html>

**For adults:**

[https://www.cdc.gov/healthyweight/assessing/bmi/adult\\_BMI/english\\_bmi\\_calculator/bmi\\_calculator.html](https://www.cdc.gov/healthyweight/assessing/bmi/adult_BMI/english_bmi_calculator/bmi_calculator.html)



## Lesson 5 Buy Low, Sell High

**Category:** Python

**Level:** Introductory

**Time Frame:** 45 minutes

**Core Subject Area:** Computing

**Supplementary Subject Area:** Mathematics

**Ages:** 11~14 years old

**Year Groups:** Key Stage 3 (UK) / Grades 6–8

(US)

### Overview

This lesson will introduce the **if...else** and **if...elif...else** statements in Python for multi-way decision making. Both statements can execute the code inside the body of the **if/elif** when certain conditions are met. Code inside the body of **else** will be executed if neither of the condition is met. To explore conditional constructs, students will develop a computational model to simulate the commodity trading and practice the 'buy low, sell high' strategy.

### Key Focus

- Selection constructs (**if...else** and **if...elif... else** statements)
- The off-side rule for writing Python code (indentation)

### Intended Learning Outcomes

*By the end of this lesson, students will be able to:*

- Identify the circumstances to use multi-way conditional expressions in association with real-life scenarios
- Explain the logic of how conditional algorithms detect whether a condition is true and only run a code under certain circumstances
- Write and execute conditional algorithms to demonstrate the 'buy low, sell high' strategy in commodity trading





## Content Standards

(UK)

### National Curriculum in England – Computing Programmes of Study: Key Stage 3

- Design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems
- Use two or more programming languages, at least one of which is textual, to solve a variety of computational problems

### National Curriculum in England – Mathematics Programmes of Study: Key Stage 3

- Develop their mathematical knowledge, in part through solving problems and evaluating the outcomes, including multi-step problems
- Develop their use of formal mathematical knowledge to interpret and solve problems, including in financial mathematics



(US)

**CSTA K-12 Computer Science Standards: Grades 6~8**

- **2-AP-11:** Create clearly named variables that represent different data types and perform operations on their values.
- **2-AP-12:** Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.

**Common Core State Standards Mathematics Standards: Grades 6~8**

- **CCSS.MATH.CONTENT.6.EE.B.6:** Use variables to represent numbers and write expressions when solving a real-world or mathematical problem; understand that a variable can represent an unknown number, or, depending on the purpose at hand, any number in a specified set.
- **CCSS.MATH.CONTENT.6.EE.B.8:** Write an inequality of the form  $x > c$  or  $x < c$  to represent a constraint or condition in a real-world or mathematical problem. Recognize that inequalities of the form  $x > c$  or  $x < c$  have infinitely many solutions; represent solutions of such inequalities on number line diagrams.



## Preparation

### For the teacher:

- A laptop or desktop with mBlock Python editor installed  
(Available from <https://python.mblock.cc/>)
- Lesson plan
- Worksheet

### For students:

- Laptops or desktops with mBlock Python editor installed  
(Available from <https://python.mblock.cc/>)
- Worksheets

## Example Program

```
1. import random
2. # Import the random module
3.
4. price_a = random.randint(100, 200)
5. # Integer from 100 to 200, endpoints included
6. print("The traded price now is", price_a)
7.
8. order = int(input("Your posted price: "))
9.
10. price_b = random.randint(100, 200)
11. # Integer from 100 to 200, endpoints included
12. print("The traded price now is", price_b)
13.
14. # Decide buy it or sell it:
15. if order > price_b:
16.     print("Current price is lower than your posted price. Buy it.")
17. elif order < price_b:
18.     print("Current price is higher than your posted price. Sell it.")
19. else:
20.     print("No transaction.")
```

**Note:** Familiarise yourself with the key points that you are going to teach in this lesson::

- Import the random module in Python by using **'import random'**
- Use the random function to simulate the price change in the commodity market
- The **if...elif...else** statement and indentation
- The difference between the **'if...if...if'** form and the **'if...elif...else'** form



## Task for Enquiry – Financial Mathematics

*What is meant by 'buy low, sell high'?*

- One of the methods to make a profit is to 'buy low, sell high' in the market. You buy something when its price is low and sell it when the price is high. Suppose you pay £300 for a commodity and sell it later for £500, you earn £200 from this transaction.
- The profits come from price inflation. Commodity prices fluctuate from time to time due to the change in demand in the market.
- However, the 'buy low, sell high' method is only a simplified model. In the real-world scenarios, it is hard to do so because most of the people cannot predict the change in prices precisely. The 'buy low, sell high' model introduced in this lesson is merely used to teach the conditional construct. This simplified model cannot be assumed as a scientific model in finance and economics.



## Pre-assessment

Have students fill out the **'What I Know'** column of the **K-W-L chart** before the class.

What I Know	What I Wonder	What I Learned
<ul style="list-style-type: none"><li>• The <b>if</b> statements</li><li>• Indentation</li><li>• Comparison operators</li><li>• Logical operators</li></ul>		



## Procedures

### Section 1 Introduction: Commodity trading (3 minutes)

**Step 1.1** Introduce a simplified model of 'buy low, sell high'.

- **Say:** Today, we are all traders. We will explore a simple but powerful rule of commodity trading – 'buy low, sell high' – to see how it works in the commodity market. We will also create a simple commodity market simulator in Python to 'buy low, sell high'.
- Briefly explain the 'buy low, sell high' strategy.

**Say:** Market prices of commodities fluctuate over time due to the exchange. Some buyers want to purchase commodities from others, while some sellers want to sell commodities at acceptable profits

**Explain:** The simplest way is to 'buy low' and 'sell high'. Buyers decide a price at which they would like to buy a commodity, while sellers propose a price at which they would like to sell a commodity. These decisions affect the current market.

**Note:** The above explanation simplifies the commodity trading operation in the real-world scenario. It is a conceptual model to help understand the logic of the commodity market simulator project used in this lesson.

**Give an example:** Suppose that you purchase a commodity at the price of £100. The price of this commodity rises 20% two weeks later, and you sell it at this price (i.e. £120) in the commodity market. You then earn £20 in this transaction. The price of this commodity drops 20% two weeks after you sell it. You buy it at the price of £96 ( $£120 - £120 \times 20\%$ ).

## Section 2 Predict (5 minutes)

**Step 2.1** Distribute the example program file to the class. Have students read the code and discuss what they think it will do.

- Instruct students to write down their predictions and annotate the code on the worksheet.

```
1. import random
2.
3. price_a = random.randint(100, 200)
4. print("The traded price now is", price_a)
5.
6. order = int(input("Your posted price: "))
7.
8. price_b = random.randint(100, 200)
9. print("The traded price now is", price_b)
10.
11. if order > price_b:
12.     print("Current price is lower than your posted price. Buy it.")
13.
14. elif order < price_b:
15.     print("Current price is higher than your posted price. Sell it.")
16.
17. else:
18.     print("No transaction.")
```

- Ask students to think about the questions below:
  - What is the control flow structure? Find out the syntax.
  - What does 'import random' mean?
  - What are the values of the variable 'price\_a' and 'price\_b'?

**Section 3    Run (2 minutes)**

**Step 3.1**      Ask students to run the example program and check against their predictions.

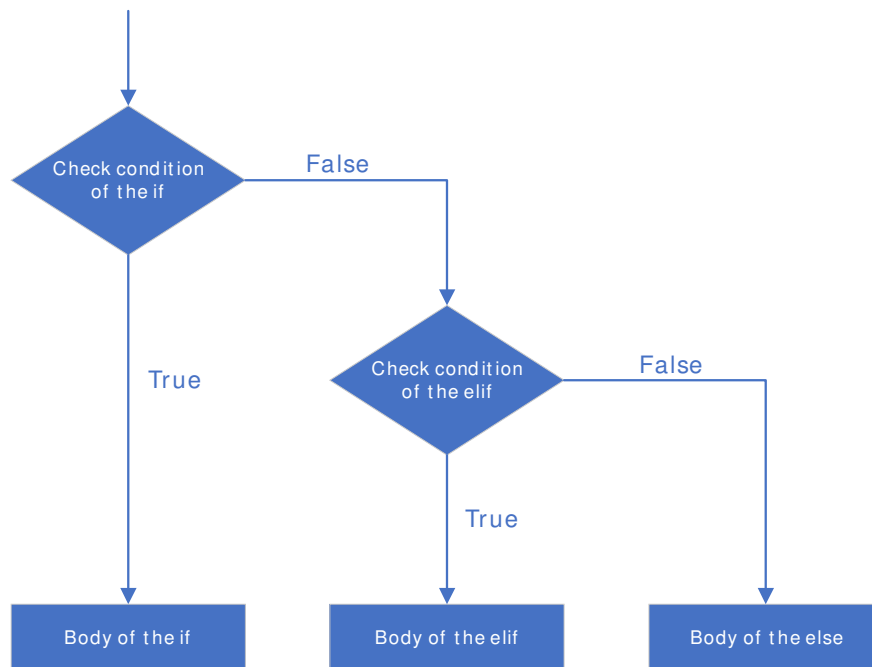
- Remind students that they can input a posted price higher or lower than the first traded price.
- Instruct students to think about the above questions while running the program.

**Section 4 Investigate (10 minutes)**

**Step 4.1** Explain the use of **if...elif...else** statements.

- Review the **if** statements learned in the last lesson first.
- **Say:** Like the BMI Calculator application we developed last time, the Commodity Simulator application also uses a selection construct. However, the syntax is different.
- **Explain the if...elif...else statement:** The **if...elif...else** statement is used for multi-way decision making. The '**elif**' is short for else if. If the condition for the '**if**' is '**False**', the program then checks the condition for the '**elif**'. If both conditions are '**False**', the program executes the indented body of the '**else**'.
- Ask students to explain the selection operation of the Commodity Simulator program and draw the flowchart.

Show an example of the **if...elif...else** control flow for reference:



Instruct students to draw the flowchart on their worksheets and record the transaction results.

**Step 4.2** Explain why and how to generate random numbers.

- **Say:** Each time you run the Commodity Simulator, the traded prices are produced from a set of random numbers. Do you know where these numbers come from?
- **Explain the 'random' module:** They come from a Python file (i.e. the module) named 'random'. It contains a sequence of numbers. We need to add the 'random' module into the mBlock Python editor by using the 'import' statement syntax:

```
import random
```

- **Explain the random.randint() function:** To generate a random number as the traded price and assign it to a variable, we need this function:

```
random.randint(start, end)
```

'randint' is short for 'random integer', which means this function can generate integers from a specified range. For example, the traded price in the example program can be any integer between 100 and 200.

**Step 4.3** Have students work in pairs to complete the tasks as follow:

- **Task 1:** Comment the code on the worksheet. Write notes after the hash mark of the line and briefly explain the function.
- **Task 2:** Investigate the conditions of the statements. How does the program compare the posted price with the traded price?
- **Task 3:** What is the condition of the 'else'. Write down its condition.

**Section 5    Modify (10 minutes)**

**Step 5.1** Have students work in pairs to modify the example program by completing the task below:

- **Task 4:** Use the **if** statements instead to rewrite the example program. Compare the two programs. Is there anything different?

**Example – Task 4**

```
1. import random
2.
3. price_a = random.randint(100, 200)
4. print("The traded price now is", price_a)
5.
6. order = int(input("Your posted price: "))
7.
8. price_b = random.randint(100, 200)
9. print("The traded price now is", price_b)
10.
11. if order > price_b:
12.     print("Current price is lower than your posted price. Buy it.")
13.
14. if order < price_b:
15.     print("Current price is higher than your posted price. Sell it.")
16.
17. if order == price_b:
18.     print("No transaction.")
```

- Instruct students to investigate the difference between using the **if** statement syntax and using the **if...elif...else** statement.

**Note:** Pay attention to the logical procedures when running the scripts. Consider how the computer may evaluate the conditions.

**Section 6    Make (15 minutes)**

**Step 6.1** Summarize the use of selection constructs.

- The selection statement syntax can be:

(1) `if` expression:

`body`

(2) `if` expression:

`body`

`else` expression:

`body`

(3) `if` expression:

`body`

`elif` expression:

`body`

`else:`

`body`

**Step 6.2** Ask students to create a digital artefact that uses a selection construct. The artefact aims to help make the multi-way decisions.

- Have students think about the situations or scenarios where they need to make decisions and the conditions to meet.
- Ask students to draw flowcharts to represent their decision-making scenarios.
- Instruct students to create the artefact and demonstrate how the application help make decisions.



## Section 7 Recap

**Step 7.1** Summarize this lesson.

- Have students fill out the **K-W-L chart** after the class.

What I Know	What I Wonder	What I Learned
<ul style="list-style-type: none"><li>• The <b>if</b> statements</li><li>• Indentation</li><li>• Comparison operators</li><li>• Logical operators</li></ul>	<ul style="list-style-type: none"><li>• The <b>elif</b> statements</li></ul>	<ul style="list-style-type: none"><li>• The <b>elif</b> statements</li><li>• Import modules</li><li>• <b>Import random</b></li><li>• The '<b>random</b>' function</li></ul>



## Lesson 6    Repay Loans

**Category:** Python

**Level:** Introductory

**Time Frame:** 45 minutes

**Core Subject Area:** Computing    **Supplementary Subject Area:** Mathematics

**Ages:** 11~14 years old

**Year Groups:** Key Stage 3 (UK) / Grades 6–8

(US)

### Overview

This lesson will introduce the **for** loop in Python through the Loan Payment Calculator application. The **for** loop is used to repeat an action or sequence a specified number of times. Take the even principal payment as an example, students will explore how to use the **for** loop to help calculate the repayments of a loan in the even principal payment schedule.

### Key Focus

- Iteration constructs (**for** loops)
- The off-side rule for writing Python code (indentation)

### Intended Learning Outcomes

*By the end of this lesson, students will be able to:*

- Explain the mathematical operation of the even principal payment schedule of a loan
- Explain the concept of iteration and recognise the **for** loop construct in Python scripts
- Write and execute repetitive algorithms to calculate the repayment of each term according to the selected repayment schedule



## Content Standards

(UK)

### **National Curriculum in England – Computing Programmes of Study: Key Stage 3**

- Design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems
- Use two or more programming languages, at least one of which is textual, to solve a variety of computational problems

### **National Curriculum in England – Mathematics Programmes of Study: Key Stage 3**

- Develop their mathematical knowledge, in part through solving problems and evaluating the outcomes, including multi-step problems
- Develop their use of formal mathematical knowledge to interpret and solve problems, including in financial mathematics



(US)

**CSTA K-12 Computer Science Standards: Grades 6~8**

- **2-AP-11:** Create clearly named variables that represent different data types and perform operations on their values.
- **2-AP-12:** Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.

**Common Core State Standards Mathematics Standards: Grades 6~8**

- **CCSS.MATH.CONTENT.6.EE.B.6:** Use variables to represent numbers and write expressions when solving a real-world or mathematical problem; understand that a variable can represent an unknown number, or, depending on the purpose at hand, any number in a specified set.
- **CCSS.MATH.CONTENT.7.RP.A.3:** Use proportional relationships to solve multistep ratio and percent problems.



## Preparation

### For the teacher:

- A laptop or desktop with mBlock Python editor installed  
(Available from <https://python.mblock.cc/>)
- Lesson plan
- Worksheet

### For students:

- Laptops or desktops with mBlock Python editor installed  
(Available from <https://python.mblock.cc/>)
- Worksheets

## Example Program

```
1. print("Loan Payment Calculator: Even Principal Payments")
2. # The repayment decreases each month
3.
4. loan = int(input("Loan Amount (GBP £): "))
5. term = int(input("Number of Payments (months): "))
6.
7. print("The annual interest rate is 5.4%")
8. rate = 0.054
9. balance = loan
10.
11. for i in range(1, term + 1):
12.     # Iterate over a sequence of integers
13.     # Return an integer starting from 1, the increment between each integer is 1
14.     # The range ends at the number of 'term'
15.
16.     principal_payment = round(loan / term, 2)
17.     interest_payment = round(balance * (rate/12), 2)
18.
19.     print("Repayment", i, ":", principal_payment + interest_payment)
20.     # Calculate the monthly payment
21.
22.     balance = loan - principal_payment*i
23.     # Reduce the unpaid balance
24.     print("Unpaid Balance:", balance)
```

**Note:** Familiarise yourself with the key points that you are going to teach in this lesson:

- The **for** loop
- Use of variables
- The **'range()'** function



## Task for Enquiry – Financial Mathematics

*What is a loan? How do I know how much I should repay the loan?*

- A loan is a sum of money lent to a person or an organisation by a qualified banking institution. For example, you might need a student loan to pay for your university/college tuition fees and living costs if you are dealing with financial difficulty and need financial assistance.
- When you get a loan, however, you need to pay money (with interest) back to the banking institution. In other words, your repayment is more than your borrowings. You can choose a repayment method that suits your financial situation.



## Pre-assessment

Have students fill out the **'What I Know'** column of the K-W-L chart before the class.

What I Know	What I Wonder	What I Learned
<ul style="list-style-type: none"><li>• The <b>elif</b> statements</li><li>• Indentation</li><li>• The <b>input()</b> function</li><li>• Data type conversion</li></ul>		

## Procedures

### Section 1 Introduction: Even Principal Payment Schedule (3 minutes)

**Step 1.1** Explain the even principal payment loan.

- **Say:** Suppose that if you apply for a loan to pay for your tuition fee, the bank will ask you to select a repayment method to pay for your loan. One of the methods is the even principal payment schedule.
- **Explain the principal payment:** An even principal payment is made on a loan that reduces the amount due. A principal payment means a payment made on a loan. If you apply for a £12,000 loan and will pay back the money after 6 months, the payment amount keeps the same every term – i.e., £2,000 for each month.
- **Explain the interest payment:** You are not free to use the loan. You will be charged interest on your loan by the bank, and therefore, except for the principal payment, you also need to pay off the interest payment every term. In the even principal payment schedule, the interest payment is calculated based on the unpaid amount. If the monthly interest rate of your £12,000 loan is 10%, the repayment schedule would work like this:

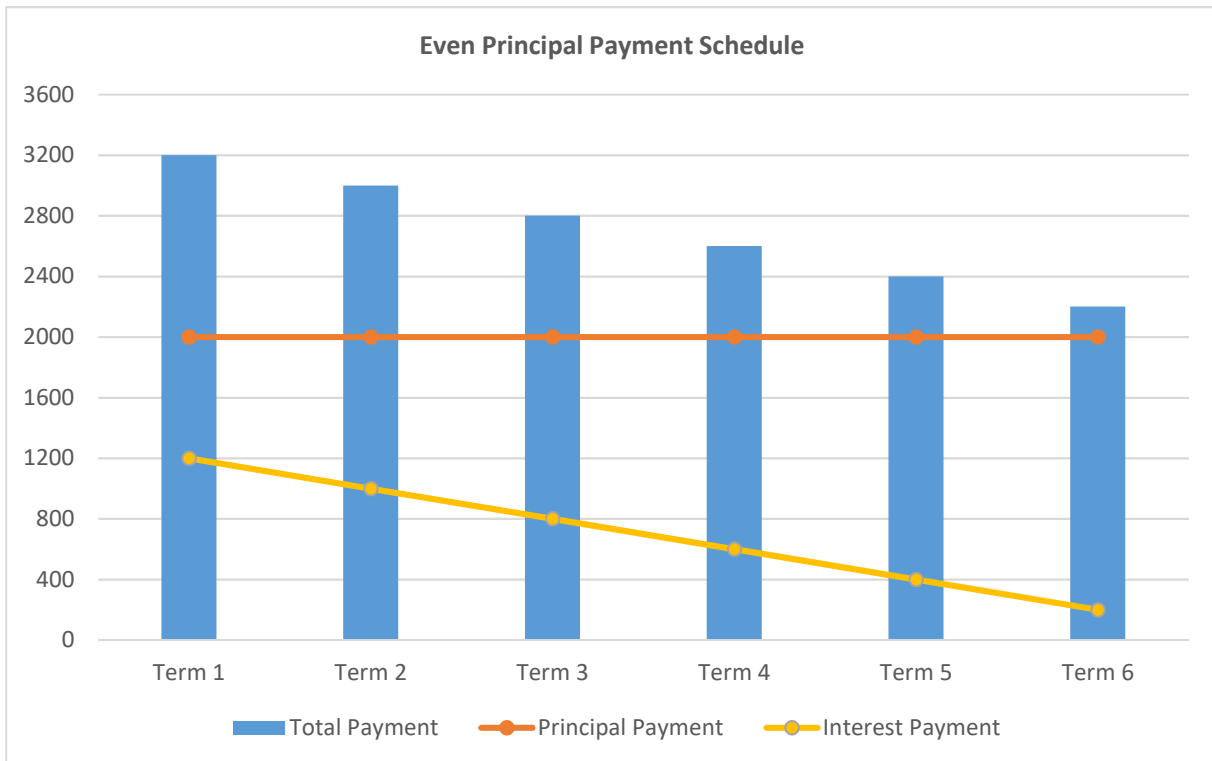
Month	Unpaid Amount	Principal Payment	Interest Payment	Total Payment
0	£12,000	/	/	/
1	£10,000	£2,000	£1,200	£3,200
2	£8,000	£2,000	£1,000	£3,000
3	£6,000	£2,000	£800	£2,800
4	£4,000	£2,000	£600	£2,600
5	£2,000	£2,000	£400	£2,400
6	£0	£2,000	£200	£2,200

- Show the above loan repayment schedule to students and ask students to figure out and summarize the features of the even principal payment schedule.



- Ask students to draw a graph to demonstrate the trend and change of the principal payment, the interest payment, and the total payment every month.

**Example:**



**Step** 1.2 Summarize the features of the even principal payment schedule.

- **Say:** The interest payment decreases over time, which means you will pay less money back as the months go by.

## Section 2 Predict (5 minutes)

**Step 2.1** Distribute the example program file to the class. Have students read the code and discuss what they think it will do.

- Instruct students to write down their predictions and annotate the code on the worksheet.

```
1. print("Loan Payment Calculator: Even Principal Payments")
2.
3. loan = int(input("Loan Amount (GBP £): "))
4. term = int(input("Number of Payments (months): "))
5.
6. print("The annual interest rate is 5.4%")
7. rate = 0.054
8. balance = loan
9.
10. for i in range(1, term + 1):
11.
12.     principal_payment = round(loan / term, 2)
13.     interest_payment = round(balance * (rate/12), 2)
14.     print("Repayment", i, ":", principal_payment + interest_payment)
15.
16.     balance = loan - principal_payment*i
17.     print("Unpaid Balance:", balance)
```

- Describe the background information about this program.

**Say:** This is a loan payment calculator application. It can calculate loan repayments based on the even principal payment schedule. Suppose that you plan to apply for a student loan and the interest rate is 5.4% according to the regulation (see also [www.gov.uk/repaying-your-student-loan/what-you-pay](http://www.gov.uk/repaying-your-student-loan/what-you-pay)).

- Ask students to think about the questions below:
  - What is the control flow structure? Find out the syntax.
  - What does '`range(1, term+1)`' mean?
  - How to calculate the unpaid balance of the loan after each repayment?

## Section 3 Run (2 minutes)

**Step 3.1** Ask students to run the example program and check against their predictions.

- Ask students to choose a programme from the table below and use the programme's tuition fee as the input data.

University or College	Programme	Total Tuition Fee*
University of Cambridge	<i>Mathematics with Physics</i>	£27,750
Cardiff University	<i>Accounting and Finance</i>	£27,000
University of Glasgow	<i>English Literature</i>	£7,280
Norland College	<i>Early Years Development and Learning</i>	£44,970
Ulster University	<i>Irish with Drama</i>	£17,580

\* The total tuition fee is calculated by multiplying the fee for the first year by 3 (or 4).

- Instruct students to think about the above questions while running the program.

## Section 4 Investigate (10 minutes)

**Step 4.1** Explain the use of the **for** loop and the '**range()**' function.

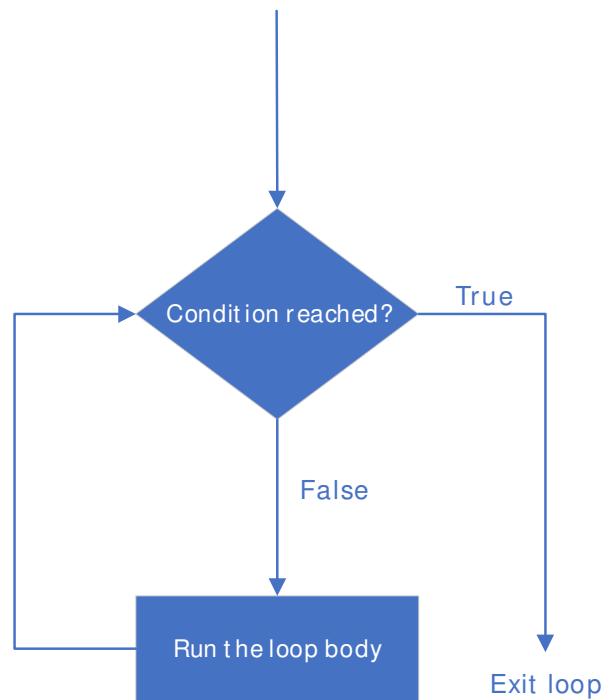
- **Explain:** To simulate the repayment, we can use the **for** loop to construct the program. The **for** loop is a count-controlled loop that iterates specified number of times. In the example program, the **for** loop iterates the terms of repayment of the loan.  
The number of times of iteration can be viewed as the condition of the loop, and the loop exits when the loop body was executed for the specified number of times.
- **Say:** As the number of repayment terms is an essential factor to decide how many times the **for** loop should execute and to reduce the unpaid amount of money, the program uses a function to count the times.
- **Explain the range() function:** The '**range()**' function is used to generate a sequence of numbers that counts the terms of repayment. The start point is 1. The end point is the value of the '**term+1**' because the **for** loop runs for the set amount of times.

`range(start, end)`

In the '**for i in range(1, term+1)**', the '**i**' represents a number of the sequence generated by the **range()** function.

- Remind students that they can compare the '**range()**' function with the '**random()**' function.
- Ask students to explain how iteration in the Loan Payment Calculator works, and draw a flowchart.

Show an example of the **for** loop control flow for reference:



- Remind students that the **for** loop and the '**range()**' function are often used together.

**Step 4.2** Briefly review the learned content as follows:

- Create the variables and assign the values (cf. Lesson 4).
- Print the strings and numeric data in the same line (cf. Lesson 3 and Lesson 5).

**Step 4.3** Have students work in pairs to complete the tasks as follow:

- **Task 1:** Read the python script and summarize the mathematical expression. Write down on the worksheet.
- **Task 2:** Based on the result returned by the Loan Payment Calculator, create a graph to present the loan repayments.

**Section 5    Modify (10 minutes)**

**Step 5.1**      Instruct students working in pairs to modify the example program by completing the tasks below:

- **Task 3:** Make loan repayments annually and modify the program. The interest rate could be the same.
- **Task 4:** The interest rate can be different across different repayment terms. Add the selection construct into the program you have modified in the third task. Refer to the rules below:

Term (Years)	Interest Rate
3	5.4%
5	5.6%
8	6.1%
10	5.8%

Instruct students to complete this task by following the requirements below:

- The borrower will be asked to select a schedule: 3-year, 5-year, 8-year, or 10-year.
- After the borrower makes a choice, the program will inform the interest rate according to the schedule.

**Example – Task 4**

```
1. loan = int(input("Loan Amount (GBP £): "))
2.
3. term = int(input("Split into 3/5/8/10 terms: "))
4. # Select a repayment schedule
5. if term == 3:
6.     print("The annual interest rate is 5.4%")
7.     rate = 0.054
8. if term == 5:
9.     print("The annual interest rate is 5.6%")
10.    rate = 0.056
11. if term == 8:
12.    print("The annual interest rate is 6.1%")
13.    rate = 0.061
14. if term == 10:
15.    print("The annual interest rate is 5.8%")
16.    rate = 0.058
17.
18. balance = loan
19.
20. for i in range(1, term + 1):
21.     # Iterate over a sequence of integers
22.     # Return an integer starting from 1, the increment between each integer is 1
23.     # The range ends at the number of 'term'
24.
25.     principal_payment = round(loan / term, 2)
26.     interest_payment = round(balance * rate, 2)
27.     print("Repayment", i, ":", principal_payment + interest_payment)
28.     # Calculate the annual payment
29.
30.     balance = loan - principal_payment*i
31.     # Reduce the unpaid balance
32.     print("Unpaid Balance:", balance)
```

**Section 6    Make (15 minutes)**

**Step 6.1** Ask students to create another Loan Repayment Calculator application for even total payments.

- **Say:** There is another loan repayment method called the Even Total Payment schedule. In an even total payment loan, the total payment amount is the same every term. The loan repayment schedule looks like this:

**Loan Amount:** £12,000

**Interest Rate:** 10%

**Loan Terms:** 6

Month	Unpaid Amount	Principal Payment	Interest Payment	Total Payment
0	£12,000	/	/	/
1				£2,058.74
2				£2,058.74
3				£2,058.74
4				£2,058.74
5				£2,058.74
6				£2,058.74





## Section 7 Recap

**Step 7.1** Summarize this lesson.

- Have students fill out the **K-W-L chart** after the class.

What I Know	What I Wonder	What I Learned
<ul style="list-style-type: none"><li>• The <b>elif</b> statements</li><li>• Indentation</li><li>• The <b>'input()'</b> function</li><li>• Data type conversion</li></ul>	<ul style="list-style-type: none"><li>• What is iteration?</li><li>• The <b>for</b> loop</li></ul>	<ul style="list-style-type: none"><li>• The <b>for</b> loop</li><li>• The <b>'range()'</b> function</li><li>• Print strings and integers (or floating-point numbers) in a line</li></ul>

## Lesson 7 Toss a Coin

**Category:** Python

**Level:** Introductory

**Time Frame:** 45 minutes

**Core Subject Area:** Computing

**Supplementary Subject Area:** Mathematics

**Ages:** 11~14 years old

**Year Groups:** Key Stage 3 (UK) / Grades 6–8

(US)

### Overview

In this lesson, students will explore the coin tossing probabilities. Students will create a computational model to simulate a coin-tossing experiment to calculate the chance of the coin landing on heads or tails. To model the true randomness of tossing a coin in the virtual environment, students need to create a list of random binary digits, 0 and 1. Students also need to combine the conditional statements and the **for** loop to conduct the random experiment.

### Key Focus

- Conditional statements inside the **for** loop
- Use of random numbers in probability problems

### Intended Learning Outcomes

*By the end of this lesson, students will be able to:*

- Explain the concepts of probability and randomness through real life examples
- Recognise the use of computational models and the logic in probability experiments
- Write and execute algorithms that combine conditional statements and **for** loops to simulate the coin-tossing process



## Content Standards

(UK)

### National Curriculum in England – Computing Programmes of Study: Key Stage 3

- Design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems
- Use two or more programming languages, at least one of which is textual, to solve a variety of computational problems

### National Curriculum in England – Mathematics Programmes of Study: Key Stage 3

- Develop their mathematical knowledge, in part through solving problems and evaluating the outcomes, including multi-step problems
- Record, describe and analyse the frequency of outcomes of simple probability experiments involving randomness, fairness, equally and unequally likely outcomes, using appropriate language and the 0-1 probability scale
- Understand that the probabilities of all possible outcomes sum to 1

(US)

**CSTA K-12 Computer Science Standards: Grades 6~8**

- **2-AP-11:** Create clearly named variables that represent different data types and perform operations on their values.
- **2-AP-12:** Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.

**Common Core State Standards Mathematics Standards: Grades 6~8**

- **CCSS.MATH.CONTENT.7.SP.C.5:** Understand that the probability of a chance event is a number between 0 and 1 that expresses the likelihood of the event occurring. Larger numbers indicate greater likelihood. A probability near 0 indicates an unlikely event, a probability around  $\frac{1}{2}$  indicates an event that is neither unlikely nor likely, and a probability near 1 indicates a likely event.
- **CCSS.MATH.CONTENT.7.SP.C.6:** Approximate the probability of a chance event by collecting data on the chance process that produces it and observing its long-run relative frequency, and predict the approximate relative frequency given the probability. For example, when rolling a number cube 600 times, predict that a 3 or 6 would be rolled roughly 200 times, but probably not exactly 200 times.
- **CCSS.MATH.CONTENT.7.SP.C.7:** Develop a probability model and use it to find probabilities of events. Compare probabilities from a model to observed frequencies; if the agreement is not good, explain possible sources of the discrepancy.



## Preparation

### For the teacher:

- A laptop or desktop with mBlock Python editor installed  
(Available from <https://python.mblock.cc/>)
- Lesson plan
- Worksheet

### For students:

- Laptops or desktops with mBlock Python editor installed  
(Available from <https://python.mblock.cc/>)
- Worksheets

## Example Program

```
1. import random
2.
3. countheads = 0
4. counttails = 0
5. n = int(input("The number of times to throw up a coin: "))
6. # Repeated Trails
7.
8. for i in range(0, n):
9.     result = random.randint(0, 1)
10.    # Generate a list of random binary digits (0 or 1)
11.
12.    if result == 0:
13.        # Heads up
14.        countheads += 1
15.        print(result)
16.
17.    else:
18.        # Tails up
19.        counttails += 1
20.        print(result)
21.
22. print("Result:")
23. # Calculate the total numbers
24. print("Heads up:", countheads)
25. print("Tails up:", counttails)
26.
27. print("Chance:")
28. # Calculate the probability
29. print("Heads : Tails", round(countheads/n, 2), ":", round(counttails/n, 2))
```

**Note:** Familiarise yourself with the key points that you are going to teach in this lesson:

- Import the random module in Python by using 'import random'
- Create the random binary digits 0 and 1
- Use comparison operators to represent the status of 'heads up' and 'tails up'

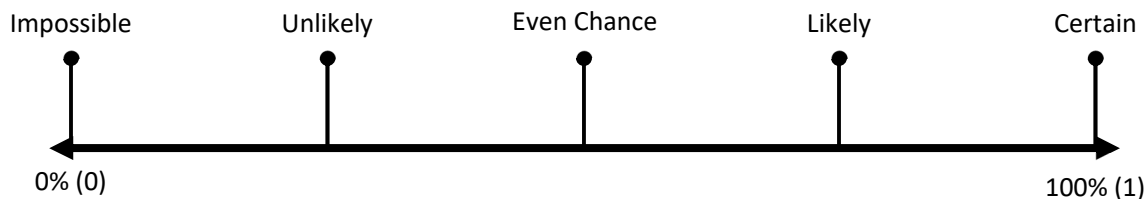


- Combine the **if...else...** statement and the for loop
- Calculate the probability of coin tosses through computational modelling

## Task for Enquiry – Probability

*What is probability?*

- Probability means how likely something is to happen. In everyday life, for example, whether it is going to rain or not is a probability problem; and if the weather forecast reports a 50% chance of rain, it means it is 50% certain that the rain will occur in your city.
- Probability can be shown on a line:



- The probability of a set of outcomes is written as percentages between 0% and 100%, or fractions or decimals between 0 and 1.
- Tossing a coin and rolling dice are two classic approaches to probability problems. For example, when a coin is tossed, there are two possible results:
  - The coin lands on heads; or
  - The coin lands on tails.

If you toss the coin 1 time, the probability to get either a head or a tail is 50%. What if you toss the coin 2 (or 1,000) times, what is the probability to get heads or tails?

### Glossary

- **Event:** A set of outcomes of an experiment that has some properties for investigation.
- **Experiment:** An occurrence with an uncertain outcome
- **Frequency:** How many times an outcome occurs.
- **Outcome:** The result of an experiment.
- **Probability:** The extent to which something is likely to happen. Probabilities are written as fractions, decimals, or percentages with values between 0 and 1.





- **Probability Scale:** The extent that starts at the extreme of 'impossible' and ends at the extreme of 'certain'.
- **Sample Space:** The set of all possible outcomes for the experiment.



## Pre-assessment

Have students fill out the **'What I Know'** column of the **K-W-L chart** before the class.

What I Know	What I Wonder	What I Learned
<ul style="list-style-type: none"><li>• The <b>for</b> loop</li><li>• The <b>'range()'</b> function</li><li>• The <b>if/elif</b> statements</li><li>• Import modules</li><li>• <b>Import random</b></li><li>• The <b>'random'</b> function</li></ul>		



## Procedures

### Section 1 Introduction: Coin Toss Probabilities (5 minutes)

**Step 1.1** Instruct students to calculate the coin toss probabilities by doing a simple experiment.

- Have students work in pairs or groups and give each pair or group a coin.
- **Ask:** What is the probability of a coin toss coming up heads or tails. Use the coin to do an experiment and record the outcomes.
- Instruct students to record the 2, 4, 6, 8, 10, 12 tosses of the coin. Write down the results in the worksheet and then calculate the probabilities.

**Say:** Toss the coin with your thumb. Catch the coin in your palm and place it on top of your other hand. Guess and reveal the result.

Total Flips	Heads	% Heads	Tails	% Tails
2				
4				
6				
8				
10				
12				

**Note:** Students only need to toss the coin 12 times.

- Ask students to share their findings. Have students discuss the results they calculated.

**Step 1.2** Ask students to think about the potential probability of a coin landing on heads (or on tails).

- **Ask:** What is the likely probability of getting the coin heads up (or tails up)?
- **Say:** Some of you may wonder the experiment is insufficient to prove the likely probability of the toss of a coin because we only try to toss the coin limited number of times.



- Ask students to think about how to improve the experiment design?
- **Ask:** Suppose if we want to toss the coin 1,000,000 times to calculate the probability, how can we experiment and do the calculation more effectively and accurately?
- **Say:** Let's see how Python can help us model this coin-tossing experiment.

**Section 2 Predict (3 minutes)**

**Step 2.1** Distribute the example program file to the class. Have students read the code and discuss what they think it will do.

- Instruct students to write down their predictions and annotate the code.

```
1. import random
2.
3. countheads = 0
4. counttails = 0
5.
6. n = int(input("The number of times to throw up a coin: "))
7.
8. for i in range(0, n):
9.     result = random.randint(0, 1)
10.
11.     if result == 0:
12.         countheads += 1
13.         print(result)
14.
15.     else:
16.         counttails += 1
17.         print(result)
18.
19. print("Result:")
20. print("Heads up:", countheads)
21. print("Tails up:", counttails)
22.
23. print("Chance:")
24. print("Heads : Tails", round(countheads/n, 2), ":", round(counttails/n, 2))
```

- Ask students to think about the questions below:
  - How does the program represent the results of heads and tails?
  - How does the program count the number of heads or tails?
  - Explain the use of the **if...else** statement and the **for** loop in this program.

**Section 3    Run (2 minutes)**

**Step 3.1**      Ask students to run the example program and check against their predictions.

- Remind students that they could refer to the table below to use the total flips as input data in the program.

Total Flips	Heads	% Heads	Tails	% Tails
100				
500				
999				
4,040				
6,140				
10,000				
36,000				
80,640				

- Instruct students to think about the questions while running the program.
- Ask students to pay attention to the change of the probability of likely results when increasing the times of tossing.

## Section 4 Investigate (15 minutes)

**Step 4.1** Review the use of the 'random' module.

- **Say:** In *Buy Low, Sell High* (Lesson 5), we learned how to generate a list of random integers. The example program of this lesson also uses the random module.
- **Ask:** What is the difference in the use of the 'random()' function between the Commodity Trading Simulator (Lesson 5) and the Coin-tossing Simulator?
- Have students discuss the above question.
- **Explain:** This program only needs two digits, 0 and 1, to represent the result of heads and the result of tails, and therefore, the 'random' function only produces either of the two integers: 0 or 1.

```
result = random.randint(0,1)
```

**Step 4.2** Explain the use of the operators.

- **Explain the comparison operator '==':** The program evaluates whether the digit number is 0 or 1 to identify the result of the toss. It needs to use comparison operators to compare a random number with the number 0 or 1 that represents the result of heads or tails. The value on which an operator operates is called an operand. In the example program, the statement 'result == 0' means both operands are equal, i.e., the random number equal to 0.
- **Explain the assignment operator '+=':** We have already known how to assign a value to a variable by using the equals sign ('='). If we want to add 1 to the variable, we can write the code like this:

```
x = x + 1
```

This assignment operation can also be written in this way:

```
x += 1
```

The example program uses this way to count the numbers of heads and tails.

**Step 4.3** Ask students to explain the code below:

The for loop with the 'range()' functions: `for i in range(0,n)`



The condition of the **'else'**

**Step 4.4** Have students draw the flowchart of the example program.



**Section 5    Modify and Make (20 minutes)**

**Step 5.1** Instruct students working in pairs to modify the example program by completing the task below:

- **Task 1:** Add a line of code to calculate the probabilities of getting heads and tails.

**Example:**

```
print("Heads:Tails", round(countheads/n, 2), ":", round(counttails/n, 2))
```

- **Task 2:** Change the number of the toss into a random number generated from the range between 4040 and 80640.

**Example:**

```
n = random.randint(4040, 80640)
```

```
print("The number of times to throw up a coin:", n)
```

- **Task 3:** Bet on the toss of the coin. Modify the example program to ask the user to input their guess before the coin-tossing experiment. Report the probability result and the betting result.

**Note:** Remind students that they need to use conditional constructs to modify the example program and add new lines of code.

**Example – Task 3**

```
1. import random
2.
3. guess = input("Heads or tails? Make your choice: ")
4. # Enter the term 'heads' or 'tails', non-capitals
5.
6. countheads = 0
7. counttails = 0
8.
9. n = random.randint(4040, 80640)
10. print("The number of times to throw up a coin:", n)
11.
12. for i in range(0, n):
13.     result = random.randint(0, 1)
14.     if result == 0:
15.         countheads += 1
16.         print(result)
17.     else:
18.         counttails += 1
19.         print(result)
20.
21. print("Heads:Tails", round(countheads/n,3), ":", round(counttails/n, 3))
22.
23. print("Result:")
24. if countheads > counttails:
25.     if guess == "heads":
26.         print("Heads, you win!")
27.     else:
28.         print("Heads, you lose.")
29.
30. elif countheads < counttails:
31.     if guess == "tails":
32.         print("Tails, you win!")
33.     else:
34.         print("Tails, you lose.")
35.
36. else:
37.     print("Tie.")
```



- **Task 4:** Based on the example program, modify the functions. Suppose the program is to model the dice-rolling process, how should you edit the code?

Record the probability to get each number of the dice.

**Note:** You can define the range for the rolling times based on your experiment design. A larger size of the population would be sufficient.

**Example – Task 4**

```
1. import random
2.
3. dice_1 = 0
4. dice_2 = 0
5. dice_3 = 0
6. dice_4 = 0
7. dice_5 = 0
8. dice_6 = 0
9.
10. n = random.randint(12000, 60000)
11. print("The number of times to roll a dice:", n)
12.
13. for i in range(0, n):
14.     result = random.randint(1, 6)
15.     if result == 1:
16.         dice_1 += 1
17.     if result == 2:
18.         dice_2 += 1
19.     if result == 3:
20.         dice_3 += 1
21.     if result == 4:
22.         dice_4 += 1
23.     if result == 5:
24.         dice_5 += 1
25.     if result == 6:
26.         dice_6 += 1
27.
28. print("Result:")
29. print("Number 1:", round(dice_1/n, 2))
30. print("Number 2:", round(dice_2/n, 2))
31. print("Number 3:", round(dice_3/n, 2))
32. print("Number 4:", round(dice_4/n, 2))
33. print("Number 5:", round(dice_5/n, 2))
34. print("Number 6:", round(dice_6/n, 2))
```

## Section 6 Recap (5 minutes)

**Step 6.1** Instruct students to compare and sum up the numbers of probabilities in the coin-tossing experiment and dice rolling experiment. Summarize the findings through these two probability experiments.

- **Summarize:** The chance of a coin landing on heads or tails (or a dice rolling each number) is the same.
- **Summarize:** The probabilities of all possible outcomes sum to 1.

**Step 6.2** Briefly review the learned syntax in this lesson. Emphasise the use of the conditional statements nested inside the **for** loop.

- Have students fill out the **K-W-L chart**.

What I Know	What I Wonder	What I Learned
<ul style="list-style-type: none"> <li>● The <b>for</b> loop</li> <li>● The <b>'range()'</b> function</li> <li>● The <b>if/elif</b> statements</li> <li>● Import modules</li> <li>● <b>Import random</b></li> <li>● The <b>'random'</b> function</li> </ul>	<ul style="list-style-type: none"> <li>● How to import a module?</li> <li>● <b>if...elif...else</b> nested inside the <b>for</b> loop</li> </ul>	<ul style="list-style-type: none"> <li>● <b>if...elif...else</b> nested inside the <b>for</b> loop</li> <li>● Assignment operators</li> </ul>

## Lesson 8    Guess the Number

**Category:** Python

**Level:** Introductory

**Time Frame:** 45 minutes

**Core Subject Area:** Computing

**Supplementary Subject Area:** Mathematics

**Ages:** 11~14 years old

**Year Groups:** Key Stage 3 (UK) / Grades 6–8

(US)

### Overview

This lesson will introduce the **while** loop in Python through another classic probability experiment. Students will explore how to calculate the probability of guessing a selected number from a list of random integers by using the **while** loop. To maximise the probability of guessing the correct number, students need to address the uniform distribution problems and use the computational model to decrease the probability distribution.

### Key Focus

- Iteration constructs (**while** loops)
- Conditional statements inside the **while** loop

### Intended Learning Outcomes

*By the end of this lesson, students will be able to:*

- Recognise the use of computational models and the logic in probability experiments
- Write and execute repetitive algorithms to simulate the number guessing probability situation



## Content Standards

(UK)

### National Curriculum in England – Computing Programmes of Study: Key Stage 3

- Design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems
- Use two or more programming languages, at least one of which is textual, to solve a variety of computational problems

### National Curriculum in England – Mathematics Programmes of Study: Key Stage 3

- Develop their mathematical knowledge, in part through solving problems and evaluating the outcomes, including multi-step problems
- Record, describe and analyse the frequency of outcomes of simple probability experiments involving randomness, fairness, equally and unequally likely outcomes, using appropriate language and the 0-1 probability scale
- Understand that the probabilities of all possible outcomes sum to 1

(US)

**CSTA K-12 Computer Science Standards: Grades 6~8**

- **2-AP-11:** Create clearly named variables that represent different data types and perform operations on their values.
- **2-AP-12:** Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.

**Common Core State Standards Mathematics Standards: Grades 6~8**

- **CCSS.MATH.CONTENT.7.SP.C.5:** Understand that the probability of a chance event is a number between 0 and 1 that expresses the likelihood of the event occurring. Larger numbers indicate greater likelihood. A probability near 0 indicates an unlikely event, a probability around  $1/2$  indicates an event that is neither unlikely nor likely, and a probability near 1 indicates a likely event.
- **CCSS.MATH.CONTENT.7.SP.C.6:** Approximate the probability of a chance event by collecting data on the chance process that produces it and observing its long-run relative frequency, and predict the approximate relative frequency given the probability. For example, when rolling a number cube 600 times, predict that a 3 or 6 would be rolled roughly 200 times, but probably not exactly 200 times.
- **CCSS.MATH.CONTENT.7.SP.C.7:** Develop a probability model and use it to find probabilities of events. Compare probabilities from a model to observed frequencies; if the agreement is not good, explain possible sources of the discrepancy.





## Preparation

### For the teacher:

- A laptop or desktop with mBlock Python editor installed  
(Available from <https://python.mblock.cc/>)
- Lesson plan
- Worksheet

### For students:

- Laptops or desktops with mBlock Python editor installed  
(Available from <https://python.mblock.cc/>)
- Worksheets

## Example Program

```
1. import random
2.
3. num = random.randint(1, 100)
4. # Select a random number
5.
6. t = 7
7. # The attempts
8.
9. print("Guess a number between 1 and 100. You have 7 attempts.")
10. print("Let's start.")
11.
12. while True:
13.     guess = int(input("Guess the number: "))
14.     if guess == num:
15.         print("BINGO!")
16.         break
17.
18.     elif guess > num:
19.         # Failed attempt
20.         t -= 1
21.         print("High! You have", t, "attempts left.")
22.
23.     elif guess < num:
24.         # Failed attempt
25.         t -= 1
26.         print("Low! You have", t, "attempts left.")
27.
28.     if t == 0:
29.         # No attempt left
30.         print("The correct number is", num)
31.         print("Game Over!")
32.         break
```



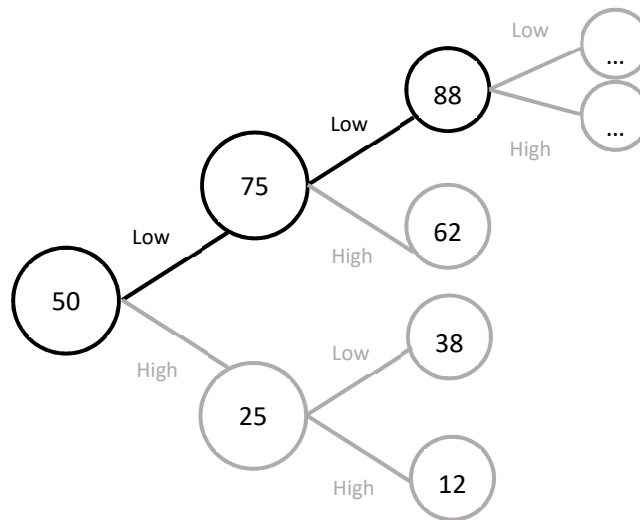
**Note:** *Familiarise yourself with the key points that you are going to teach in this lesson:*

- Use of binary search approach in solving probability problems
- Import the random module in Python by using '**import random**'
- The **while** loops (forever loops)
- Combine the **if...else** statement and the **while** loop

## Task for Enquiry – Probability

*What is binary search used for?*

- Binary search, or half-interval search, is an efficient method to find a target number within a list of numbers. The use of this method depends upon the probability of each number in the list being searched.
- A binary search can start from the middle element and check whether the middle element is lower or higher than the target number. This method then narrows the interval to the lower or upper half and selects another number in the middle of the remained half.
- We can use the probability tree to explain how binary search works. For example, you are asked to guess a selected number from the range between 1 and 100:





## Pre-assessment

Have students fill out the **'What I Know'** column of the **K-W-L chart** before the class.

What I Know	What I Wonder	What I Learned
<ul style="list-style-type: none"><li>• <b>if...elif...else</b> nested inside the <b>for</b> loop</li><li>• <b>Import random</b></li><li>• The <b>'random'</b> function</li><li>• Assignment operators</li></ul>		

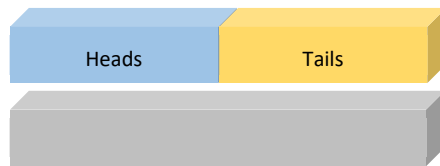
## Procedures

### Section 1 Introduction: *How to Use Probabilities?* (5 minutes)

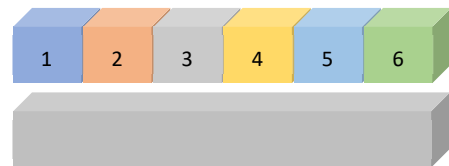
**Step 1.1** Review the last lesson. Instruct students to think about the concept of probability and its application in solving mathematical problems.

- **Say:** In the last lesson, we have summarized two rules of probability:
  - In a probability experiment, each event or possible outcome has the same chance of occurring.
  - The probabilities of all possible outcomes sum to 1.
- Ask students to use a bar model to illustrate the above two rules.

**Example:**



**Coin tossing probabilities**



**Dice rolling probabilities**

**Step 1.2** Ask students to think about the questions below.

- **Say:** Suppose you are playing a guessing game with your computer. The computer will pick up an integer randomly from the range between 1 and 100. You have 7 chances to find the computer's number. If your guess is too high or too low, the computer will give you a hint.
- **Ask:** How can you figure out the computer's number as fast as possible? What can you learn from the probability experiments we did last time to handle the guessing game?
- Have students discuss the possible solutions.
- **Say:** Try and test your methods in the example program later.

## Section 2 Predict and Run (8 minutes)

**Step 2.1** Distribute the example program file to the class. Have students read the code and discuss what they think it will do.

- **Say:** This is the script of the guessing game. Read through the script to figure out how the game mechanics works.

```
1. import random
2.
3. num = random.randint(1, 100)
4.
5. t = 7
6.
7. print("Guess a number between 1 and 100. You have 7 attempts.")
8. print("Let's start.")
9.
10. while True:
11.     guess = int(input("Guess the number: "))
12.     if guess == num:
13.         print("BINGO!")
14.         break
15.
16.     elif guess > num:
17.         t -= 1
18.         print("High! You have", t, "attempts left.")
19.
20.     elif guess < num:
21.         t -= 1
22.         print("Low! You have", t, "attempts left.")
23.
24.     if t == 0:
25.         print("The correct number is", num)
26.         print("Game Over!")
27.         break
```

- **Ask:** Now you know how this game works, so can you find a method to effectively find the number picked by the computer?



**Step 2.2** Ask students to run the example program and test their guessing methods.

- Remind students that they can try and test their methods when playing the guessing game.

Record the numbers and the number of attempts that they used to find the correct number in the worksheet.

Round	1 <sup>st</sup> Try	2 <sup>nd</sup> Try	3 <sup>rd</sup> Try	4 <sup>th</sup> Try	5 <sup>th</sup> Try	6 <sup>th</sup> Try	7 <sup>th</sup> Try	Number
<i>Example</i>	7	85	60	40	20	11	9	9



### Section 3 Investigate (17 minutes)

**Step 3.1** Ask students to report and share their methods and results.

- **Ask:** How many attempts did it take you to find the correct number? What are your methods?  
Discuss with your classmates.
- **Ask:** How to use probability to find the correct number?
- **Say:** Let's look at the script and see how we make good use of the computer's hint and probability theory.

**Step 3.2** Instruct students to analyse the example program in the first place.

- Ask students to explain the use of the random module in the guessing game.
- Ask students to explain the use of conditional constructs.
- Ask students to explain how to reduce the number of attempts.
- Ask students to interpret the meaning of **'while True'**.

**Step 3.3** Explain the **while** loop.

- **Say:** There is a new type of control flow structure used in this program. As you may notice, it executes the loop body of 'Guess the number' to handle the user input. It is an iteration.
- **Explain the concept of the while loop:** The **while** loop iterates over the loop body of instruction when the condition is verified. As with conditional statements and for loops, you need to indent the loop body when you're writing while loops.
- **Explain the 'while True' statement:** In the example program, the condition of the **while** loop is written as **'while True'**, which means the loop body will be repeatedly executed forever.

**Step 3.4** Introduce the 'True' and other Python keywords.

- **Explain the Python keyword:** Please note that the 'True' used in the statement is a reserved word in Python. The **'True'** with 'T' capitalised has its specific meaning because the 'true' and the 'True' are different in Python.

**Say:** There are three reserved words with their first letters capitalised in Python: **'True'**, **'False'**, and **'None'**.

**Introduce the list of Python keywords:** In the previous learning, we have already used some of these reserved words, including **'import'**, **'if'**, **'elif'**, **'else'**, **'for'**, and **'while'** – all of them are in lower case.

**Say:** There are 33 reserved words in Python:

and	as	assert	break	class	continue
def	del	elif	else	except	finally
for	from	False	global	if	import
in	is	lambda	nonlocal	not	None
or	pass	raise	return	try	True
while	with	yield			

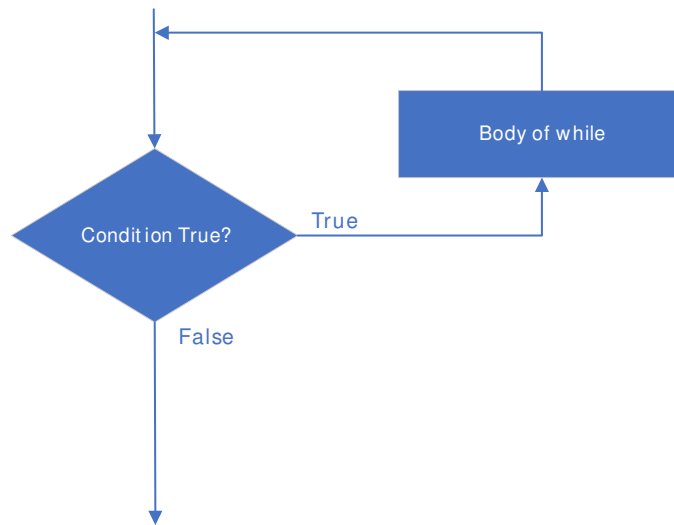
**Explain:** The above reserved words have specific meanings and functions. Remember, you cannot use these words to name variables.

**Step 3.5** Introduce the use of the **'break'** statement.

- **Say:** The **'while True'** loop is a forever loop. However, in the guessing game, we only have 7 attempts and when there is no attempt left, the loop should be stopped.
- **Explain:** To terminate the iteration without setting any specific conditions, we can use the **'break'** statement. The **'break'** statement inside the **while** loop can stop the execution of the loop body when we find the computer's number, or no attempt is left.

**Step 3.6** Ask students to draw a flowchart of the example program.

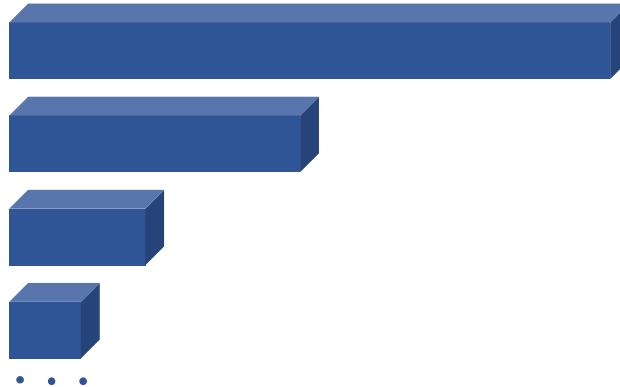
- Show an example of the **while** loop control flow for reference:



## Section 4 Binary Search (10 minutes)

**Step 4.1** Explain the effective method to find the computer's number.

- **Say:** To find the correct number as soon as possible, we can use the binary search method.
- Show the bar model to demonstrate the binary search method.



- **Explain:** You can start by guessing 50, the middle of the range. Suppose if 50 is too high, you can then eliminate all the numbers from 50 to 100. The second guess can be 25; suppose if 25 is still too high, eliminate the numbers from 25 to 50. The third guess can be 12; if 12 is too high, eliminate the numbers from 12 to 25. Then you can guess 7; if 7 is too low, eliminate the numbers from 1 to 7. Eventually, you can find the computer's number.
- **Say:** This method is called 'binary search'. Find the middle of the range first. If it is too high or low, eliminate half the numbers.
- Ask students to use the probability theory to explain why the binary search approach is an effective way to find the computer's number.

**Step 4.2** Instruct students to think about how to apply binary search.

- **Ask:** Do you know why we only have 7 attempts to guess the number from 1 to 100?  
**Note:** Remind students that they can use the bar model to figure out this problem. (Tip:  $2^7 = 128 > 100$ )
- **Ask:** Suppose you are now the designer of the guessing game. If the user needs to find a number from 1 to 1,000, how many attempts can the user have? Modify the example program and explain your reasons.



[Tip:  $2^{10} = 1,024 > 1,000$ ]

**Section 5    Recap (5 minutes)**

**Step 5.1** Summarize the use of binary search.

**Step 5.2** Summarize the use of the **while** loop (forever loop) and reflect upon the algorithm design of the guessing number game.

- Have students fill out the **K-W-L chart**.

What I Know	What I Wonder	What I Learned
<ul style="list-style-type: none"><li>● <b>Import random</b></li><li>● Nested <b>if...elif...else</b> inside the <b>for</b> loop</li><li>● The '<b>random</b>' function</li><li>● Assignment operators</li></ul>	<ul style="list-style-type: none"><li>● What is a <b>while</b> loop</li><li>● What is the difference between <b>for</b> loops and <b>while</b> loops</li></ul>	<ul style="list-style-type: none"><li>● The <b>while</b> loops</li><li>● Boolean values</li></ul>

## Lesson 9~10 Python Quizzes

**Category:** Python

**Level:** Introductory

**Time Frame:** 90 minutes

**Core Subject Area:** Computing

**Ages:** 11~14 years old

**Year Groups:** Key Stage 3 (UK) / Grades

6–8 (US)

### Overview

This lesson will introduce the features of the CyberPi device. Students will explore some commonly used input and output devices of CyberPi, including the buttons, the joystick, the display screen, and the LEDs. Students will also investigate how to utilise these physical components to create a simple keypad. Students will use the keypad to answer quizzes on Python or other topics.

### Key Focus

- Physical components of CyberPi
- The **'cyberpi'** module and how to import it in the mBlock Python editor

### Intended Learning Outcomes

*By the end of this lesson, students will be able to:*

- Identify some common physical components of CyberPi and their corresponding scripts, including the display screen, the LEDs, the buttons, and the joystick
- Explain the use of the **'cyberpi'** module in association with previous programming experience, recognise and execute the functions from the **'import'** module



## Content Standards

### (UK)

#### National Curriculum in England – Computing Programmes of Study: Key Stage 3

- Design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems
- Use two or more programming languages, at least one of which is textual, to solve a variety of computational problems
- Understand how instructions are stored and executed within a computer system; understand how data of various types (including text, sounds and pictures) can be represented and manipulated digitally, in the form of binary digits

### (US)

#### CSTA K-12 Computer Science Standards: Grades 6~8

- **2-CS-02:** Design projects that combine hardware and software components to collect and exchange data.
- **2-DA-07:** Represent data using multiple encoding schemes.
- **2-AP-11:** Create clearly named variables that represent different data types and perform operations on their values.
- **2-AP-12:** Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
- **2-AP-16:** Incorporate existing code, media, and libraries into original programs, and give attribution.





## Preparation

### For the teacher:

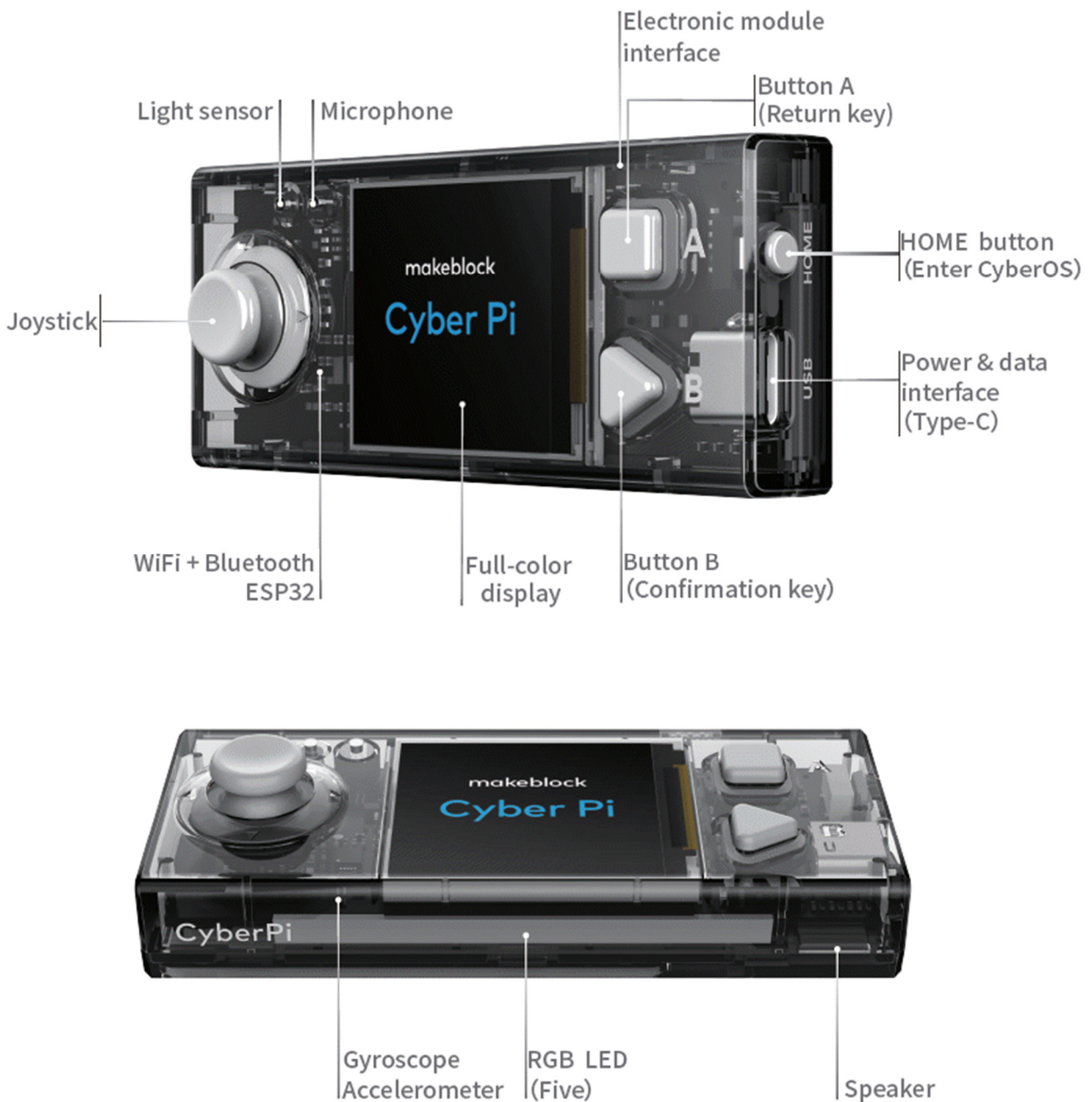
- A laptop or desktop with mBlock Python editor installed
- A CyberPi device
- A Type-C cable
- Lesson plan
- Worksheet

### For students:

- Laptops or desktops with mBlock Python editor installed
- CyberPi devices
- Type-C cables
- Worksheets



## Features of CyberPi



## Example Program

```
1. import cyberpi
2. # Import the cyberpi module to call functions
3.
4. cyberpi.display.clear()
5. # Clear the screen
6. cyberpi.led.off()
7. # Light off
8.
9. cyberpi.console.println("A - True")
10. cyberpi.console.println("B - False")
11. # Print in a new line
12.
13. cyberpi.led.on(255, 255, 255)
14. # Produce white light
15. cyberpi.console.println("Python is a compiled language.")
16. cyberpi.console.println("Your Answer:")
17.
18. while True:
19.     if cyberpi.controller.is_press("a"):
20.         # If the Button A is pressed
21.         cyberpi.led.on(255, 0, 0)
22.         cyberpi.console.println("Incorrect.")
23.         cyberpi.console.println("Correct Answer: False")
24.         break
25.         # Stop all the scripts
26.
27.     if cyberpi.controller.is_press("b"):
28.         # If the Button B is pressed
29.         cyberpi.led.on(0, 255, 0)
30.         cyberpi.console.println("Correct!")
31.         break
```



## Pre-assessment

Have students fill out the **'What I Know'** column of the **K-W-L chart** before the class.

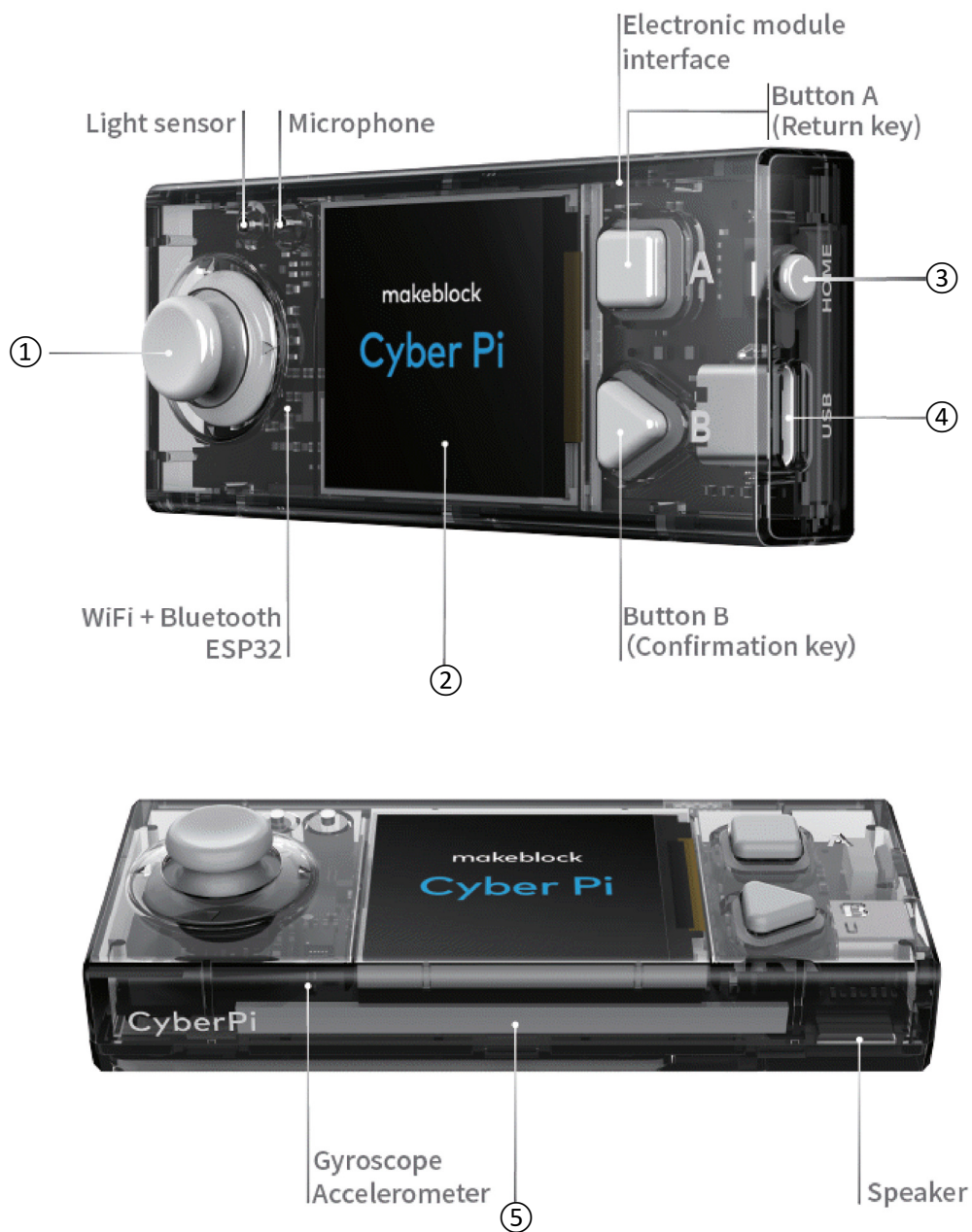
What I Know	What I Wonder	What I Learned
<ul style="list-style-type: none"><li>● Import <b>random</b></li><li>● Loops</li><li>● Conditionals</li><li>● The <b>print()</b> function</li><li>● Python data types</li></ul>		

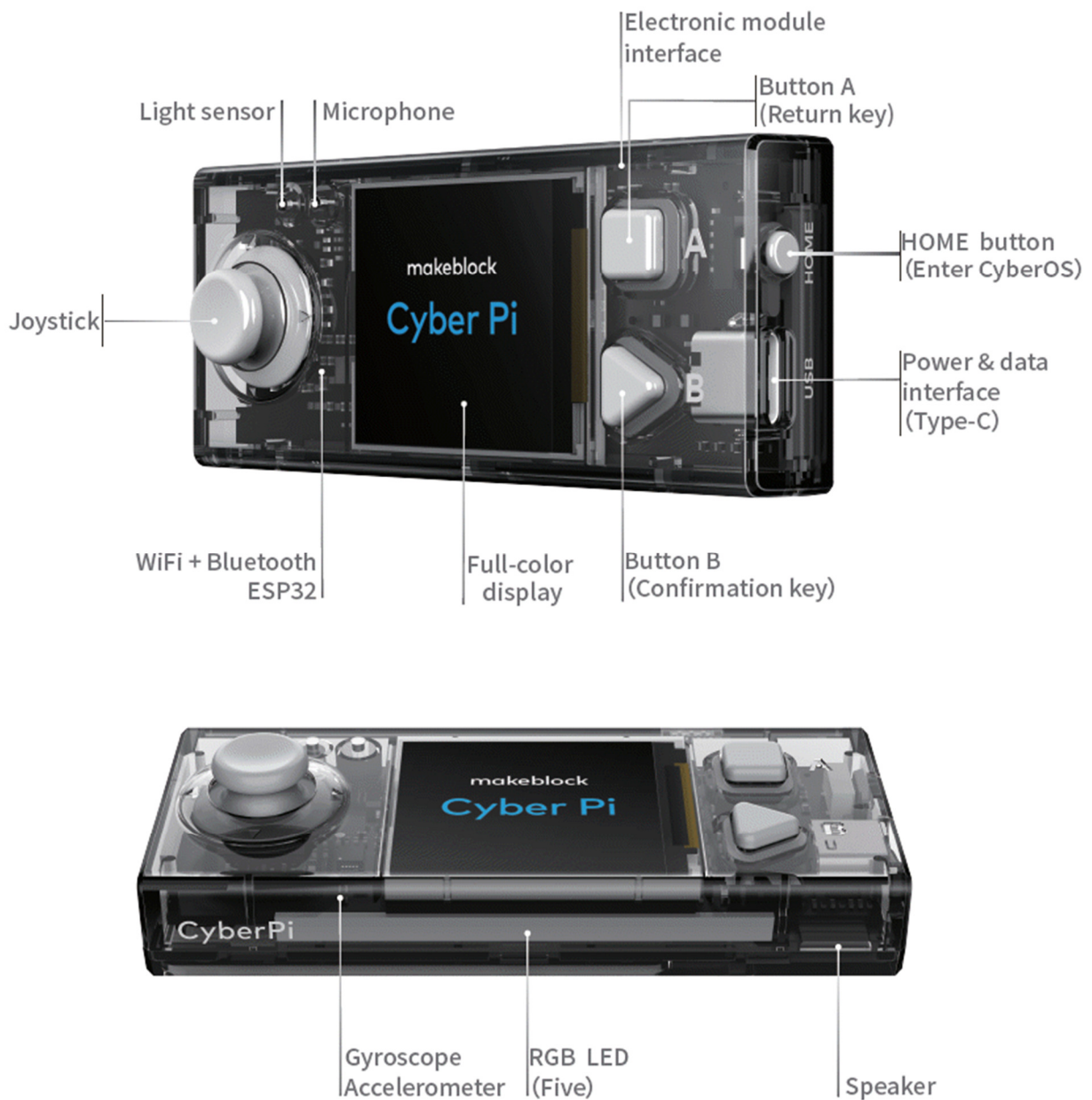
## Procedures

### Section 1 Introduction: CyberPi and API (30 minutes)

**Step 1.1** Display and introduce the CyberPi device.

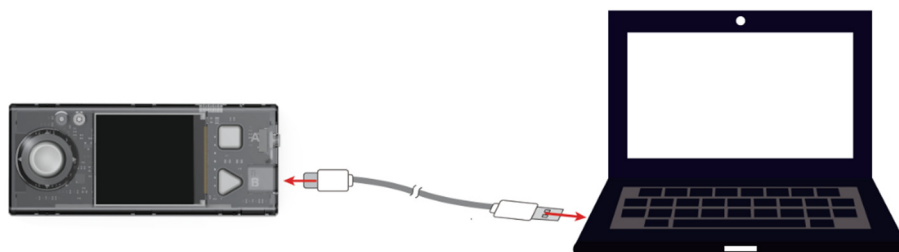
- Ask students to observe their devices. Have them identify the physical components of CyberPi and fill in the blank on the worksheet.





**Step 1.2** Instruct students to connect CyberPi to the laptop or desktop.

- Demonstrate how to use the USB Type-C cable to connect CyberPi to the laptop or desktop.



- Instruct students to open **mLink2** and access the mBlock Python editor.



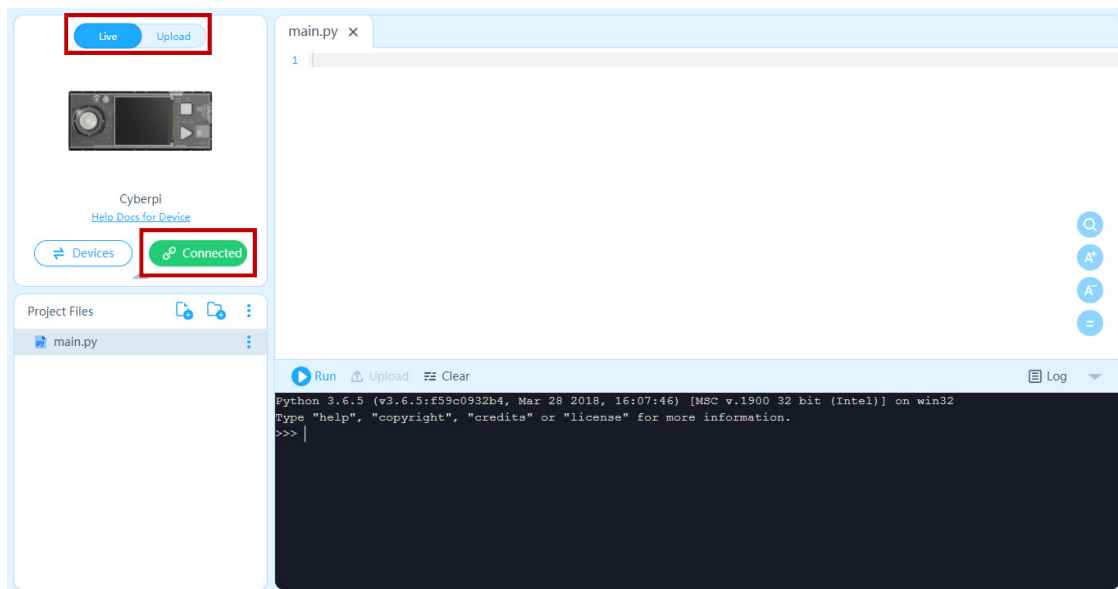
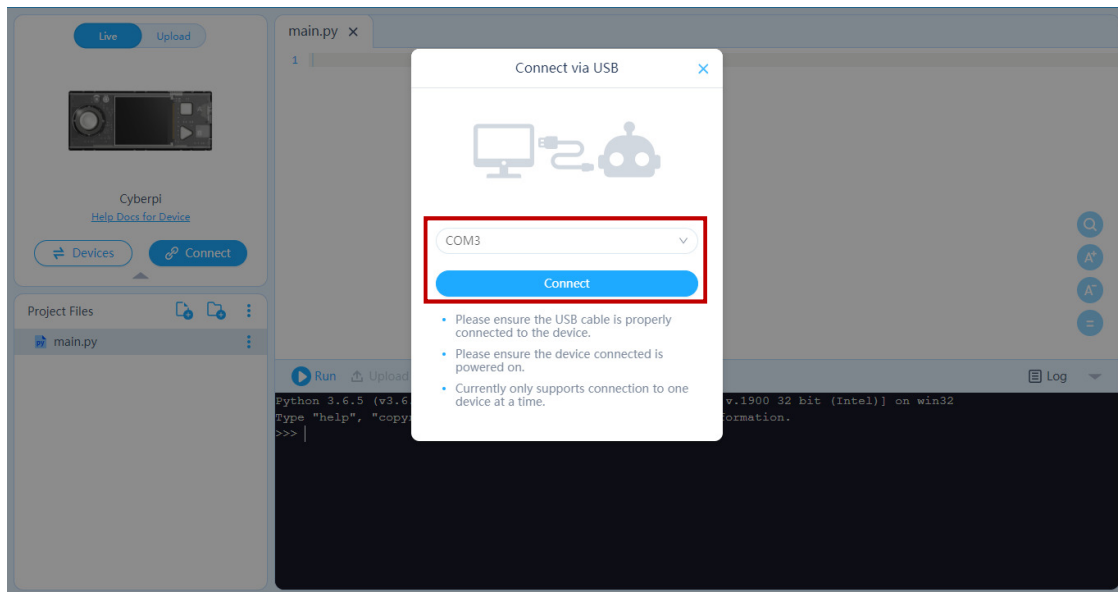
- Instruct students to click **Connect** and select the correct serial port.

The screenshot displays the mBlock Python editor interface. The top navigation bar includes links for 'makeblock', 'mBlock', 'File', 'New project', 'Libraries', 'Example Programs', 'Tutorials', 'Feedback', and a 'Code with blocks' button. The main workspace is divided into several sections:

- Left Panel:** Contains a 'Cyberpi' device image, a 'Help Docs for Device' link, and buttons for 'Devices' and 'Connect'. The 'Connect' button is highlighted with a red rectangle.
- Top Right:** A 'Project Files' section showing a file explorer with files like 'main.py', 'main.doc', and 'main.png'. A message says 'Click on a file in Project Files to open it.'
- Bottom Left:** A 'Project Files' list showing 'main.py'.
- Bottom Right:** A terminal window showing the Python 3.6.5 shell prompt and the command 'Type "help", "copyright", "credits" or "license" for more information.'

A message in the center of the workspace states: 'mLink 2 not started, so you can't run Python programs or connect to devices. Please install and start mLink 2, then try again.' with a 'Download' button.





- Remind students that they should choose **Live** mode, which allows them to code in real time.
- Summarize the connection method and other operation.

### Step 1.3 Introduce the concept of API.

- **Say:** How can we program CyberPi in the mBlock Python editor? If we want to display text on the screen, what functions do we need to use? Could we display text on the screen just with `'print()'` (e.g. `print("Hello, CyberPi!")`)?
- **Explain:** To program CyberPi in the Python editor, we need to understand how to use the





application programming interface of CyberPi. The application programming interface (API) of CyberPi is a computing interface which defines interactions between CyberPi and the Python editor. It allows us to write Python code in the editor to program CyberPi. As this API is not built in Python, we need to write the statement 'import cyberpi' as the first line:

```
import cyberpi
```

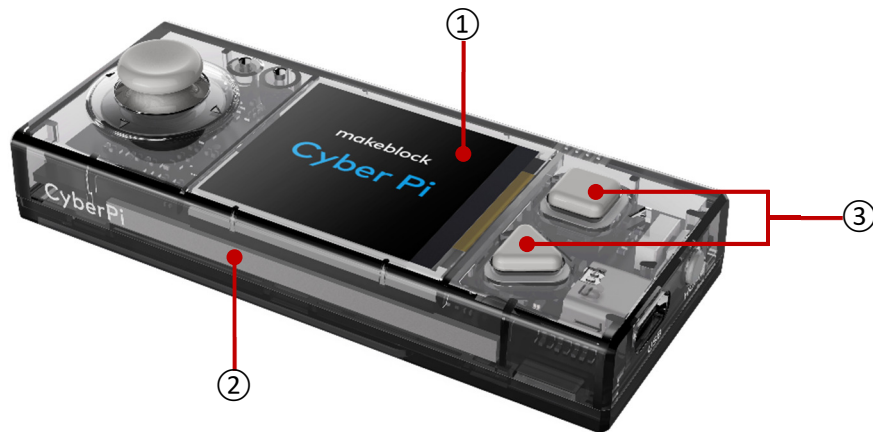
## Section 2 Predict and Run (15 minutes)

**Step 2.1** Distribute the example program file to the class. Have students read the code and discuss what the code does before running it.

- **Say:** The example program is a simple quiz application. Use the button 'A' or 'B' to answer the true or false question displayed on the screen.

```
1. import cyberpi
2.
3. cyberpi.display.clear()
4. cyberpi.led.off()
5.
6. cyberpi.console.println("A - True")
7. cyberpi.console.println("B - False")
8.
9. cyberpi.led.on(255, 255, 255)
10. cyberpi.console.println("Python is a compiled language.")
11. cyberpi.console.println("Your Answer:")
12.
13. while True:
14.
15.     if cyberpi.controller.is_press("a"):
16.         cyberpi.led.on(255, 0, 0)
17.         cyberpi.console.println("Incorrect.")
18.         cyberpi.console.println("Correct Answer: False")
19.         break
20.
21.     if cyberpi.controller.is_press("b"):
22.         cyberpi.led.on(0, 255, 0)
23.         cyberpi.console.println("Correct!")
24.         break
```

- Instruct students to write down their predictions and annotate the code.
- Also, provide some hints to point out the physical components CyberPi has and how these physical components are programmed.



- ① Screen – `cyberpi.display`, `cyberpi.console`
- ② LED Strip – `cyberpi.led`
- ③ Buttons A and B – `cyberpi.controller.is_press()`
- Ask students to think about the questions below:
  - Which module is imported in this program? Explain why we should import this module.
  - How to print text on the screen?
  - How to light up/off the LEDs?
  - How to set the LEDs' color? How to represent LEDs' colors in Python?
  - How to enter the answer?
  - How to evaluate the answer?



- Have students fill out the **'What I Wonder'** column of the **K-W-L chart** after running the example program. Students could write down some main points in the third column if they have some ideas about what they will learn in this lesson.

What I Know	What I Wonder	What I Learned
<ul style="list-style-type: none"><li>• Import modules</li><li>• Loops</li><li>• Conditionals</li><li>• The <b>print()</b> function</li><li>• Python data types</li></ul>	<ul style="list-style-type: none"><li>• How does CyberPi work?</li><li>• How to program CyberPi by using Python?</li></ul>	

### Section 3 Investigate (20 minutes)

**Step 3.1** Explain the use of the ‘**cyberpi**’ module as the prerequisite for programming CyberPi in Python.

- Have students recall when they need to use the ‘**import**’ module. Ask them to explain why they need to use this syntax.

(Tip: `import random`)

- **Give an example:** When we want to model a probability problem in Python, we need to import the ‘**random**’ module to generate a set of numbers. When the ‘**random**’ module is added, we can use functions and expressions such as ‘**random.randint()**’ to create the sequence of integers.
- **Explain:** We need to import the ‘**cyberpi**’ module; otherwise, we cannot program CyberPi in Python.

**Step 3.2** Explain how to display text on CyberPi’s screen.

- **Say:** We cannot write the code – for example, ‘**print(“Hello, CyberPi!”)**’ - in the Python editor to display the text on CyberPi’s screen. We need to call a function that allows us to program the display screen of CyberPi.
- **Emphasise the use of API:** To program the screen or other physical components, we need to know the Application Programming Interface (API) of these components. An API enables data transmission between the Python editor and CyberPi. The API of a physical component specifies executable code on request. For instance, if we want to display ‘Hello, CyberPi!’ on CyberPi’s screen, we need to know the API of the screen.

- Point out the API code samples of CyberPi’s screen to the students:

```
cyberpi.console.print()
```

```
cyberpi.console.println()
```

- Demonstrate how to display the text ‘Hello, CyberPi!’ on the screen. Instruct students to run the code shown below:

```
import cyberpi
```

```
cyberpi.console.print("Hello, CyberPi!")
```

- Explain the difference between the '`cyberpi.console.print()`' and '`cyberpi.console.println()`'. Explain that the latter makes the text go to the next line automatically while printing out the new content.

- Instruct students to run and compare the two examples that follow:

#### Example 1-1 (a)

```
import cyberpi
```

```
cyberpi.console.print("Hello, CyberPi!")
```

```
cyberpi.console.print("I can program you in Python.")
```

#### Example 1-1 (b)

```
import cyberpi
```

```
cyberpi.console.println("Hello, CyberPi!")
```

```
cyberpi.console.println("I can program you in Python.")
```

- Explain how to create a new line of output with this function:  

```
cyberpi.console.println("")
```
- Remind students that if they want to clear the screen, they can use:  

```
cyberpi.display.clear()
```
- Remind students that they should pay attention to the letter case while reading and writing code.
- Instruct students to annotate the presented API code samples in the example program.

**Example 1-2**

```

1. import cyberpi
2. # Import the 'cyberpi' module before programming
3.
4. cyberpi.console.println("A - True")
5. # Print 'A - True' and then move to the next line
6.
7. cyberpi.console.println("B - False")
8. # Print 'B - False' and then move to the next line
9.
10. cyberpi.console.println("")
11. # Enter a line break

```

**Step 3.3** Explain how to program the LEDs.

- Introduce the API code samples of the LEDs used in the example program:

```
cyberpi.led.on()
```

```
cyberpi.led.off()
```

- Introduce other API code samples of the LEDs for reference.

```
cyberpi.led.on("green", id="all")
```

```
cyberpi.led.on("red", id=3)
```

```
cyberpi.led.show("orange yellow cyan blue purple")
```

```
cyberpi.led.play(name="firefly")
```

```
cyberpi.led.off(id="3")
```

- Instruct students to annotate the presented API code samples in the example program.

**Step 3.4** Explain how to program the two buttons.

- **Say:** By pressing Button A or Button B, we choose the answer true or false.
- Introduce the API code samples of the buttons used in the example program:

```
cyberpi.controller.is_press("a")
```

```
cyberpi.controller.is_press("b")
```

- Introduce API code samples that relate to the joystick input:

```
cyberpi.controller.is_press("up")  
  
cyberpi.controller.is_press("down")  
  
cyberpi.controller.is_press("right")  
  
cyberpi.controller.is_press("left")  
  
cyberpi.controller.is_press("middle")
```

**Step 3.5** Have students discuss the use of control flow structures used in the example program.



## Section 4    Modify and Make (20 minutes)

**Step 4.1**       Summarize the API code samples of the screen, the LEDs, the buttons, and the joystick.

**Step 4.2** Have students work individually and complete the following tasks:

- **Task 1:** Modify the quiz and the answers in the example program.

- Suggest some true-or-false questions for consideration:

*In Python, 200.0 is an integer. (False)*

*'True' and 'true' are the same in Python. (False)*

*The word 'break' can be used to name a variable. (False)*

*To program CyberPi, you should import 'cyberpi'. (True)*

*The 'input()' returns a string. (True)*

...

- Students can also design quizzes on other topics, for example:

*Glasgow is the capital city of Scotland. (False)*

*Aberdeen is called the 'Oil Capital of Europe'. (True)*

*The European Central Bank has their headquarter in Amsterdam. (False)*

*Belarus is a member country of the EU. (False)*

*Jane Austin wrote the fiction Emma. (True)*

*H.C. Andersen was a Swedish author. (False)*

*The official languages of the UN include Arabic. (True)*

*The Mona Lisa by Raphael is on display in Louvre. (False)*

*Auguste Rodin created the sculpture The Thinker. (True)*

...

- **Task 2:** Modify the lighting effects.
  - Remind students that they can design the lighting effects or use the default effects.
  - The default lighting effects include:

*Firefly:* `cyberpi.led.play(name="firefly")`

*Rainbow:* `cyberpi.led.play(name="rainbow")`

*Spoondrift:* `cyberpi.led.play(name="spoondrift")`

*Meteor Shower:* `cyberpi.led.play(name="meteor_blue");`

`cyberpi.led.play(name="meteor_green")`

*Flash:* `cyberpi.led.play(name="flash_orange"); cyberpi.led.play(name="flash_red")`

- **Task 3:** Use the joystick instead of the buttons to enter the answer.
  - Instruct students to use the joystick's 'up' and 'down' input to decide the 'True' and 'False' options.

#### Example – Task 3

```
1. import cyberpi
2.
3. cyberpi.display.clear()
4. cyberpi.led.off()
5.
6. cyberpi.led.play(name="rainbow")
7.
8. cyberpi.console.println("Python is a compiled language.")
9. cyberpi.console.println("")
10. cyberpi.console.println("Your Answer:")
11.
12. while True:
13.     if cyberpi.controller.is_press("up"):
14.         # If the Button A is pressed
15.         cyberpi.led.on("red", id="all")
16.         cyberpi.console.println("Incorrect.")
17.         cyberpi.console.println("Correct Answer: False")
18.         break
19.     if cyberpi.controller.is_press("down"):
20.         cyberpi.led.on("green", id="all")
21.         cyberpi.console.println("Correct!")
22.         break
```

- **Differentiation:** Encourage advanced learners to consider a program for multiple-choice questions that provide five options. Use the joystick as an input device to enter the possible input responses, including 'Choose A', 'Choose B', 'Choose C', "Choose D", and "None".

#### Example – Task 3 (Advanced)

```
1. import cyberpi
2.
3. x = 0
4. # Initialise the counter
5. cyberpi.led.off()
6. cyberpi.display.clear()
7. cyberpi.led.on(255, 255, 255)
8. # Initialise the device
9.
10. while x < 14:
11.     # Terminate the while loop after completing 14 quizzes
12.     if cyberpi.controller.is_press("up"):
13.         # Choose Option A
14.         cyberpi.console.print("A ")
15.         cyberpi.led.on("cyan", id="all")
16.         x += 1
17.         # Update counter
18.     if cyberpi.controller.is_press("left"):
19.         # Choose Option B
20.         cyberpi.console.print("B ")
21.         cyberpi.led.on("yellow", id="all")
22.         x += 1
23.         # Update counter
24.     if cyberpi.controller.is_press("down"):
25.         # Choose Option C
26.         cyberpi.console.print("C ")
27.         cyberpi.led.on("purple", id="all")
28.         x += 1
29.         # Update counter
30.     if cyberpi.controller.is_press("right"):
31.         # Choose Option D
32.         cyberpi.console.print("D ")
33.         cyberpi.led.on("orange", id="all")
34.         x += 1
35.         # Update counter
36.     if cyberpi.controller.is_press("middle"):
37.         # None
38.         cyberpi.console.print(" ")
39.         # Leave it empty
40.         cyberpi.led.on("black", id="all")
41.         x += 1
42.         # Update counter
```

**Section 5    Recap (5 minutes)**

**Step 5.1** Summarize the key points learned in this lesson.

- The prerequisite for programming CyberPi in the Python editor.
- CyberPi's physical components.
- Have students fill out the '**What I Learned**' column of the **K-W-L chart**.

What I Know	What I Wonder	What I Learned
<ul style="list-style-type: none"><li>• Import modules</li><li>• Loops</li><li>• Conditionals</li><li>• The <b>print()</b> function</li><li>• Python data types</li></ul>	<ul style="list-style-type: none"><li>• How does CyberPi work?</li><li>• How to program CyberPi by using Python?</li></ul>	<ul style="list-style-type: none"><li>• Concept of API</li><li>• Import <b>cyberpi</b></li><li>• Display screen of CyberPi</li><li>• LED strip of CyberPi</li><li>• Buttons of CyberPi</li><li>• Joystick of CyberPi</li></ul>

## Lesson 11 Data Protection and Passwords

**Category:** Python

**Level:** Introductory

**Time Frame:** 45 minutes

**Core Subject Area:** Computing

**Ages:** 11~14 years old

**Year Groups:** Key Stage 3 (UK) / Grades

6–8 (US)

### Overview

In this lesson, students will explore the topic of data security and create a digital artefact to demonstrate how a password-protected security device can protect personal data. Students need to describe commonly used data security measures and when to use them. Based on real-life scenarios, students should explain the importance of data security measures. Students will also explore how to use physical security devices to verify and confirm information.

### Key Focus

- How to display text on the display screen of CyberPi
- How to nest conditionals and loops

### Intended Learning Outcomes

*By the end of this lesson, students will be able to:*

- Explain common security measures to protect personal data and information with real-life examples
- Explain how to create a password and verify the password through a password-protected security device
- Write and execute algorithms to simulate how a password-protected security device protects personal data



## Content Standards

(UK)

### National Curriculum in England – Computing Programmes of Study: Key Stage 3

- Design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems
- Understand several key algorithms that reflect computational thinking; use logical reasoning to compare the utility of alternative algorithms for the same problem
- Use two or more programming languages, at least one of which is textual, to solve a variety of computational problems
- Understand how instructions are stored and executed within a computer system; understand how data of various types (including text, sounds and pictures) can be represented and manipulated digitally, in the form of binary digits
- Understand a range of ways to use technology safely, respectfully, responsibly and securely, including protecting their online identity and privacy



(US)

**CSTA K-12 Computer Science Standards: Grades 6~8**

- **2-CS-02:** Design projects that combine hardware and software components to collect and exchange data.
- **2-NI-05:** Explain how physical and digital security measures protect electronic information.
- **2-DA-07:** Represent data using multiple encoding schemes.
- **2-AP-11:** Create clearly named variables that represent different data types and perform operations on their values.
- **2-AP-12:** Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
- **2-AP-16:** Incorporate existing code, media, and libraries into original programs, and give attribution.
- **2-IC-20:** Compare trade-offs associated with computing technologies that affect people's everyday activities and career options.





## Preparation

### For the teacher:

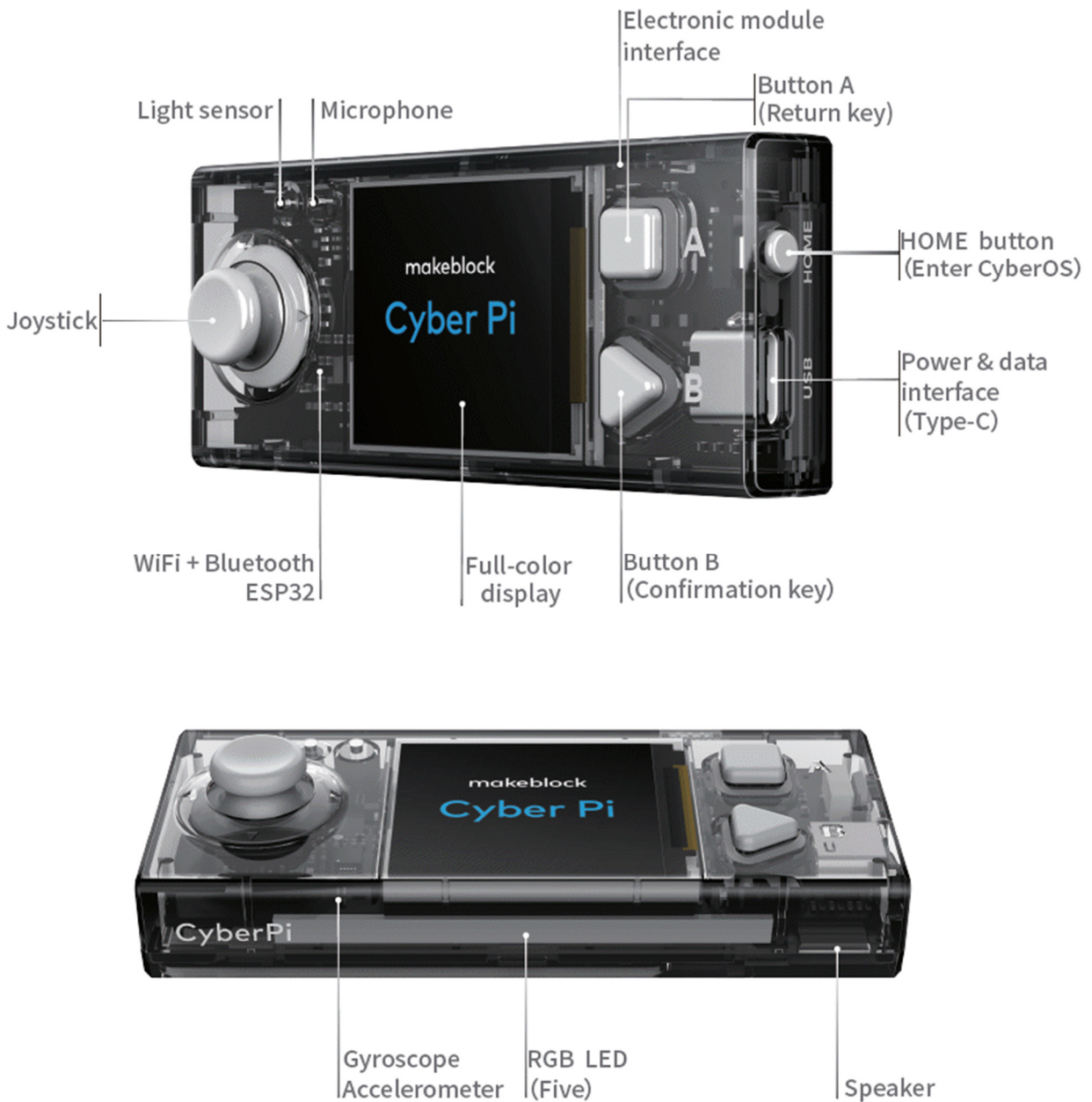
- A laptop or desktop with mBlock Python code editor installed  
(Available from <https://python.mblock.cc/>)
- A CyberPi device
- A Type-C cable
- Lesson plan
- Worksheet

### For students:

- Laptops or desktops with mBlock Python code editor installed  
(Available from <https://python.mblock.cc/>)
- CyberPi devices
- Type-C cables
- Worksheets



## Features of CyberPi



## Example Program

```
1. import cyberpi
2.
3. cyberpi.display.clear()
4. t = 0
5. while True:
6.     print("Create a password")
7.     pin_1 = input("Type password: ")
8.     pin_2 = input("Type password again: ")
9.     if pin_2 == pin_1:
10.        print("Success!")
11.        break
12.    else:
13.        print("Passwords don't match. Try again.")
14.
15. print("Sign in your account")
16. cyberpi.console.println("Sign in")
17. while t < 3:
18.    # Have 3 attempts
19.    # If t < 3 is true, iterate the loop body
20.    pin = input("Password: ")
21.    cyberpi.console.print("Password: ")
22.    cyberpi.console.println(pin)
23.
24.    # Verify the input password:
25.    if pin == pin_1:
26.        # Passwords match
27.        cyberpi.console.println("Success!")
28.        break
29.    else:
30.        # Passwords don't match
31.        cyberpi.console.println("Incorrect. Try again.")
32.        t += 1
33.        # Reduce the number of attempts
34.        if t == 3:
35.            cyberpi.console.println("Too many failed attempts.")
```



## Pre-assessment

Have students fill out the **'What I Know'** column of the K-W-L chart before the class.

What I Know	What I Wonder	What I Learned
<ul style="list-style-type: none"><li>● Concept of the API</li><li>● Import <b>cyberpi</b></li><li>● Display screen of CyberPi</li><li>● LEDs of CyberPi</li><li>● Buttons of CyberPi</li><li>● Joystick of CyberPi</li></ul>		



## Procedures

### Section 1 Introduction: Importance of Password Security (6 minutes)

**Step 1.1** Discuss the essence of password-protected security measures.

- **Ask:** When do you use passwords (or passcode)?
- Have students share their or their family members' experience of using passwords and explain why they use passwords.
- Summarize students' discussion and list some of the common situations that people use passwords to protect their data and privacy (for example, sign into Google account, unlock the mobile phone, withdraw money, make a payment, etc.).
- **Ask:** What if we do not have passwords (or passcode) in these situations, what may happen? What kinds of problems would you have?
- Have students discuss the safety of passwords (or passcode) and real-life examples.

**Section 2 Predict (6 minutes)**

**Step 2.1** Distribute the example program file to the class. Have students read the code and discuss what the code does before running it.

- **Say:** The example program is to demonstrate how to create and verify a password through a physical password-protected security device.

```
1. import cyberpi
2.
3. cyberpi.display.clear()
4. t = 0
5.
6. while True:
7.     print("Create a password")
8.     pin_1 = input("Type password: ")
9.     pin_2 = input("Type password again: ")
10.    if pin_2 == pin_1:
11.        print("Success!")
12.        break
13.    else:
14.        print("Passwords don't match. Try again.")
15.
16.
17. print("Sign in your account")
18. cyberpi.console.println("Sign in")
19.
20. while t < 3:
21.     pin = input("Password: ")
22.     cyberpi.console.print("Password: ")
23.     cyberpi.console.println(pin)
24.     if pin == pin_1:
25.         cyberpi.console.println("Success!")
26.         break
27.     else:
28.         cyberpi.console.println("Incorrect. Try again.")
29.         t += 1
30.     if t == 3:
31.         cyberpi.console.println("Too many failed attempts.")
```

- Ask students to think about the questions below:
  - How to create a password for a new account?
  - How to verify the password entered by a user?
  - Compare the two '**while**' loops used in the example program. What is the difference between them?
  - Identify the syntax that enables the following function: A user is given 3 attempts to enter the account password. If the user has 3 failed attempts, the user is blocked from entering the password a 4<sup>th</sup> time.

**Section 3    Run (3 minutes)**

**Step 3.1**      Ask students to run the example program and check it against their predictions.

- Ask students to think about the above-mentioned questions while running the program.
- Instruct students to annotate the code on the worksheet based on their observation.
- Have students fill out the **'What I Wonder'** column of the **K-W-L chart** after running the example program.

What I Know	What I Wonder	What I Learned
<ul style="list-style-type: none"><li>● Import cyberpi</li><li>● Display screen of CyberPi</li><li>● LEDs of CyberPi</li><li>● Buttons of CyberPi</li><li>● Joystick of CyberPi</li></ul>	<ul style="list-style-type: none"><li>● How does CyberPi interact with the computer (e.g. send and receive data)?</li><li>● How to display text on CyberPi's screen?</li></ul>	



### Section 3 Investigate (10 minutes)

**Step 3.1** Have students discuss the above questions and invite volunteers to share their findings.

**Step 3.2** Explain how the example program works.

- **Explain:** The user needs to type the password twice and the variables '**pin\_1**' and '**pin\_2**' store the password input. The expression '**if pin\_1 == pin\_2 (is True)**' is to confirm whether the two passwords match each other.
- Have students think about why the user should type the password two times.
- Remind students of the use of the '**break**' function in the '**while**' loop.

```
1. while True:
2.     print("Create a password")
3.     pin_1 = input("Type password: ")
4.     pin_2 = input("Type password again: ")
5.     if pin_2 == pin_1:
6.         print("Success!")
7.         break
8.     else:
9.         print("Passwords don't match. Try again.")
```

- Students should identify the part of code as follows to answer how the account password is verified. Have students explain the functions and logic behind the code in their own words.
- Have students think about the similarity between the function of creating a password and the function of verifying the password.
- Have students compare the two '**while**' loops and figure out the difference.

**Possible Answer:** The '**while True**' is to create an indefinite iteration, which means that on specified times the loop is executed and isn't specified on request. In short, the '**while True**' functions as a forever loop.

The '**while <controlling expression> (is True)**' (e.g. '**while t < 3:**') function has a controlling expression as the specified condition for iteration, the loop is executed if the controlling expression is verified as '**True**'. In the example program, the user has three

attempts to type the password to sign into the account, and therefore, the controlling expression is ' $t < 3$ '.

```
1. while t < 3:
2.     # Have 3 attempts
3.     pin = input("Password: ")
4.     # Type the password
5.     cyberpi.console.print("Password: ")
6.     cyberpi.console.println(pin)
7.     # Display the password on the screen
8.     # Verify the input password:
9.     if pin == pin_1:
10.        # Passwords match
11.        cyberpi.console.println("Success!")
12.        break
13.    else:
14.        # Passwords don't match
15.        cyberpi.console.println("Incorrect. Try again.")
16.        t += 1
17.        # Reduce the number of attempts
```

- Students should identify and explain how to lock the account after 3 unsuccessful sign-in attempts.

```
if t == 3:
    cyberpi.console.println("Too many failed attempts.")
```

**Step 3.2** Summarize the key points of the example program.

**Section 4    Modify (10 minutes)**

**Step 4.1**      Have students work in pairs or individually to complete the following tasks:

- **Task 1:** Add the function that allows the user to create a username when the user signs in.

## Example – Task 1

```
1. import cyberpi
2.
3. cyberpi.display.clear()
4. t = 0
5. while True:
6.     user_id = input("Create a username: ")
7.     # Create a username
8.     print("Create a password")
9.     pin_1 = input("Type password: ")
10.    pin_2 = input("Type password again: ")
11.    if pin_2 == pin_1:
12.        print("Success!")
13.        break
14.    else:
15.        print("Passwords don't match. Try again.")
16.
17. print("Sign in your account")
18. cyberpi.console.println("Sign in")
19. while t < 3:
20.     user = input("Username: ")
21.     # Type the username
22.     cyberpi.console.print("Username: ")
23.     cyberpi.console.println(user)
24.     pin = input("Password: ")
25.     cyberpi.console.print("Password: ")
26.     cyberpi.console.println(pin)
27.     if user == user_id and pin == pin_1:
28.         # Verify the username
29.         cyberpi.console.println("Success!")
30.         break
31.     else:
32.         cyberpi.console.println("Incorrect. Try again.")
33.         t += 1
34.     if t == 3:
35.         cyberpi.console.println("Too many failed attempts.")
```

- **Task 2:** Add some lighting effects as the indicator. Use what you have learned to program

CyberPi.

**API code samples of the LED strip:**

```
cyberpi.led.on(255, 255, 255)
```

```
cyberpi.led.on("green", id = "all")
```

```
cyberpi.led.on("red", id = 3)
```

```
cyberpi.led.show("orange yellow cyan blue purple")
```

```
cyberpi.led.play(name = "firefly")
```

```
cyberpi.led.off(id = "3")
```

- **Task 3:** Modify the second **'while'** loop. Use the LEDs as an indicator to visualize the number of attempts.

(**Note:** Accordingly, the user has 5 login attempts.)

**Example – Task 3**

```
1. import cyberpi
2.
3. cyberpi.display.clear()
4. cyberpi.led.off()
5. t = 0
6.
7. while True:
8.     user_id = input("Create a username: ")
9.     print("Create a password")
10.    pin_1 = input("Type password: ")
11.    pin_2 = input("Type password again: ")
12.    if pin_2 == pin_1:
13.        print("Success!")
14.        break
15.    else:
16.        print("Passwords don't match. Try again.")
17.
18.
19. print("Sign in your account")
20. cyberpi.console.println("Sign in")
21. cyberpi.led.on("white", id="all")
22. while t < 5:
23.     user = input("Username: ")
24.     cyberpi.console.print("Username: ")
25.     cyberpi.console.println(user)
26.     pin = input("Password: ")
27.     cyberpi.console.print("Password: ")
28.     cyberpi.console.println(pin)
29.     if user == user_id and pin == pin_1:
30.         cyberpi.console.println("Success!")
31.         cyberpi.led.play(name="rainbow")
32.         break
33.     else:
34.         cyberpi.console.println("Incorrect. Try again.")
35.         t += 1
36.         cyberpi.led.off(id=t)
37.         if t == 5:
38.             cyberpi.console.println("Too many failed attempts. Locked.")
```



- **Task 4:** Modify the conditional expressions. Verify the following three conditions:

*The input username is incorrect;*

*The input password is incorrect;*

*Both the username and password are incorrect.*

**Example – Task 4**

```
1. import cyberpi
2.
3. cyberpi.display.clear()
4. cyberpi.led.off()
5. t = 0
6. while True:
7.     user_id = input("Create a username: ")
8.     print("Create a password")
9.     pin_1 = input("Type password: ")
10.    pin_2 = input("Type password again: ")
11.    if pin_2 == pin_1:
12.        print("Success!")
13.        break
14.    else:
15.        print("Passwords don't match. Try again.")
16. print("Sign in your account")
17. cyberpi.console.println("Sign in")
18. cyberpi.led.on("white", id="all")
19.
20. while t < 5:
21.     user = input("Username: ")
22.     cyberpi.console.print("Username: ")
23.     cyberpi.console.println(user)
24.     pin = input("Password: ")
25.     cyberpi.console.print("Password: ")
26.     cyberpi.console.println(pin)
27.     if user == user_id and pin == pin_1:
28.         cyberpi.console.println("Success!")
29.         cyberpi.led.play(name="rainbow")
30.         break
31.     elif user != user_id and pin == pin_1:
32.         cyberpi.console.println("Incorrect username. Try again.")
33.         t += 1
34.         cyberpi.led.off(id=t)
35.     elif user == user_id and pin != pin_1:
36.         cyberpi.console.println("Incorrect password. Try again.")
37.         t += 1
38.         cyberpi.led.off(id=t)
39.     elif user != user_id and pin != pin_1:
40.         cyberpi.console.println("Both incorrect. Try again.")
41.         t += 1
42.         cyberpi.led.off(id=t)
43.     if t == 5:
44.         cyberpi.console.println("Too many failed attempts. Locked.")
```



**Section 5    Make (8 minutes)**

**Step 5.1** Ask students to develop a bank card reader based on the structure and logic of the example program.

- **Explain what a card reader is:** A card reader is a security device. When your parents want to make a payment through their bank accounts, they will be asked to use the card reader as a security measure. For example, they need to enter the PIN with the keypad on the card reader and will receive a random verification code or passcode number to confirm the transaction.
- Explain how the card reader project functions and list the requirements of the project:
  - Like the example program, first, create a PIN for the bank account and store it on the computer.
  - When a sender starts a transaction, ask the sender to type the name (or other identification code) of the receiver and the amount of the transit money.
  - Then ask the sender to type the PIN.
  - Generate a random 4-digit verification code and display it on CyberPi's screen. Ask the sender to enter the verification code.
  - Check the PIN and verification code. If both are correct, display the transaction information (including the receiver's name or ID and the transaction amount) on the screen.  
  
However, if either the PIN, the verification code or both are incorrect, ask the sender to type them again. The sender has limited attempts (for example, 3 attempts).
  - Ask the sender to check the transaction information and confirm the transaction by pressing Button B of CyberPi.
- Demonstrate the program below to show how to create a 4-digit verification code and display it on CyberPi's screen.

**Example 3-1**

```
1. import cyberpi, random
2.
3. cyberpi.display.clear()
4.
5. while True:
6.     if cyberpi.controller.is_press("a"):
7.         code = random.randint(1000, 9999)
8.         cyberpi.console.print("Verification code: ")
9.         cyberpi.console.println(code)
```

- Provide the part of code below for reference. Students should consider where to insert the lines of code in the card reader program.

```
1. cyberpi.console.println("Confirm - Press B")
2. while not cyberpi.controller.is_press("b"):
3.     pass
4. print("Success!")
5. cyberpi.display.clear()
6. break
```

- The above script is to confirm the transaction information displayed on the screen. The sender should check the receiver and the transfer amount and then confirm the transaction if everything is correct.
- Have student annotate this part of code either in the Python editor or on the worksheet.

**Step 5.2** Instruct students to modify the example program and create the card reader project.

**Section 6    Recap (2 minutes)**

**Step 6.1** Summarize the key points learnt in this lesson.

- Have students fill out the **'What I Learned'** column of the **K-W-L chart**.

What I Know	What I Wonder	What I Learned
<ul style="list-style-type: none"><li>● Import cyberpi</li><li>● Display screen of CyberPi</li><li>● LED strip of CyberPi</li><li>● Buttons of CyberPi</li><li>● Joystick of CyberPi</li></ul>	<ul style="list-style-type: none"><li>● How does CyberPi interact with the computer (e.g. send and receive data)?</li><li>● How to display text on CyberPi's screen?</li></ul>	<ul style="list-style-type: none"><li>● How to use variables to transmit data between the computer and CyberPi</li><li>● How to display text on the screen</li><li>● Importance of password security</li></ul>

## Lesson 12 Normal Distribution

**Category:** Python

**Level:**

Introductory

**Time Frame:** 45 minutes

**Core Subject Area:** Computing

**Supplementary Subject Area:** Mathematics

**Ages:** 11~14 years old

**Year Groups:** Key Stage 3 (UK) / Grades 6–8

(US)

### Overview

Data can be spread out in various ways due to the variation in the data. The normal distribution is the most important data distribution, because it fits many natural phenomena. In this lesson, students will explore the concept and features of the normal distribution through Python. Take the dice roll probabilities as an example, students will create data charts to represent all the possible outcomes and visualise the distribution of all the results of the sum of the two dice. Students will measure and investigate the spread of the data to gain an understanding of the normal distribution.

### Key Focus

- How to display charts on the CyberPi's screen
- Use of more than one module in Python and how to call relevant functions

### Intended Learning Outcomes

*By the end of this lesson, students will be able to:*

- Identify the features of normal distribution model and explain why the kind of data distribution is a normal distribution
- Write and execute repetitive algorithms to simulate probability experiments and create computational models that can demonstrate the normal distribution phenomena



## Content Standards

(UK)

### National Curriculum in England – Computing Programmes of Study: Key Stage 3

- Design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems
- Understand several key algorithms that reflect computational thinking; use logical reasoning to compare the utility of alternative algorithms for the same problem
- Use two or more programming languages, at least one of which is textual, to solve a variety of computational problems
- Make appropriate use of data structures; design and develop modular programs that use procedures or functions



(US)

**CSTA K-12 Computer Science Standards: Grades 6~8**

- **2-CS-02:** Design projects that combine hardware and software components to collect and exchange data.
- **2-DA-07:** Represent data using multiple encoding schemes.
- **2-DA-08:** Collect data using computational tools and transform the data to make it more useful and reliable.
- **2-DA-09:** Refine computational models based on the data they have generated.
- **2-AP-11:** Create clearly named variables that represent different data types and perform operations on their values.
- **2-AP-12:** Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
- **2-AP-16:** Incorporate existing code, media, and libraries into original programs, and give attribution.



## Preparation

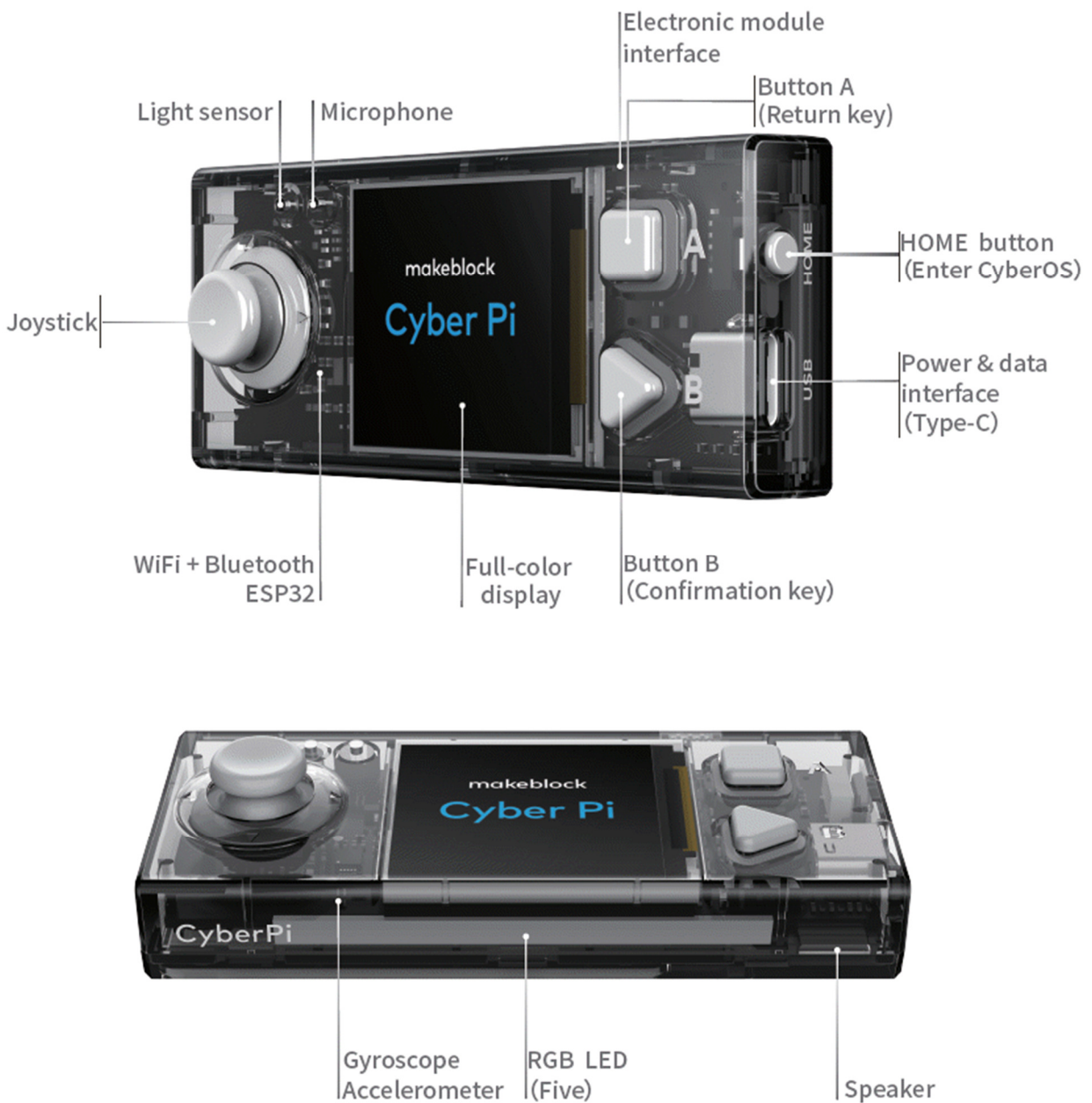
### For the teacher:

- A laptop or desktop with mBlock Python editor installed
- A CyberPi device
- A Type-C cable
- Lesson plan
- Worksheet

### For students:

- Laptops or desktops with mBlock Python editor installed
- CyberPi devices
- Type-C cables
- Worksheets

## Features of CyberPi







## Example Program

```
1. import cyberpi, random
2.
3. cyberpi.display.clear()
4. n = int(input("The number of times to roll 2 dice: "))
5. t = 0
6. sum_list = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
7. # Outcomes of the sum of two dice numbers
8. count_list = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
9. # Count the frequency of the sum in the sample space
10. while t < n:
11.     dice_x = random.randint(1, 6)
12.     dice_y = random.randint(1, 6)
13.     print("(", dice_x, ",", dice_y, ")")
14.     result = dice_x + dice_y
15.     t += 1
16.     sum_index = sum_list.index(result)
17.     # Identify the index of the sum in the list 'sum_list'
18.     count_list[sum_index] += 1
19.     # Add '1' to the value of the corresponding item
20.
21. cyberpi.display.set_brush(128, 0, 0)
22. # Define the colour based on RGB colour model
23. cyberpi.barchart.add(round(count_list[0]/n * 200, 2))
24. # Create a bar
25. cyberpi.display.set_brush(220, 20, 60)
26. cyberpi.barchart.add(round(count_list[1]/n * 200, 2))
27. cyberpi.display.set_brush(255, 0, 0)
28. cyberpi.barchart.add(round(count_list[2]/n * 200, 2))
29. cyberpi.display.set_brush(205, 92, 92)
30. cyberpi.barchart.add(round(count_list[3]/n * 200, 2))
31. cyberpi.display.set_brush(233, 150, 122)
32. cyberpi.barchart.add(round(count_list[4]/n * 200, 2))
33. cyberpi.display.set_brush(255, 69, 0)
34. cyberpi.barchart.add(round(count_list[5]/n * 200, 2))
35. cyberpi.display.set_brush(255, 165, 0)
36. cyberpi.barchart.add(round(count_list[6]/n * 200, 2))
37. cyberpi.display.set_brush(255, 215, 0)
38. cyberpi.barchart.add(round(count_list[7]/n * 200, 2))
39. cyberpi.display.set_brush(240, 230, 140)
40. cyberpi.barchart.add(round(count_list[8]/n * 200, 2))
41. cyberpi.display.set_brush(255, 255, 0)
42. cyberpi.barchart.add(round(count_list[9]/n * 200, 2))
43. cyberpi.display.set_brush(154, 205, 50)
44. cyberpi.barchart.add(round(count_list[10]/n * 200, 2))
```



## Pre-assessment

Have students fill out the **'What I Know'** column of the **K-W-L chart** before the class.

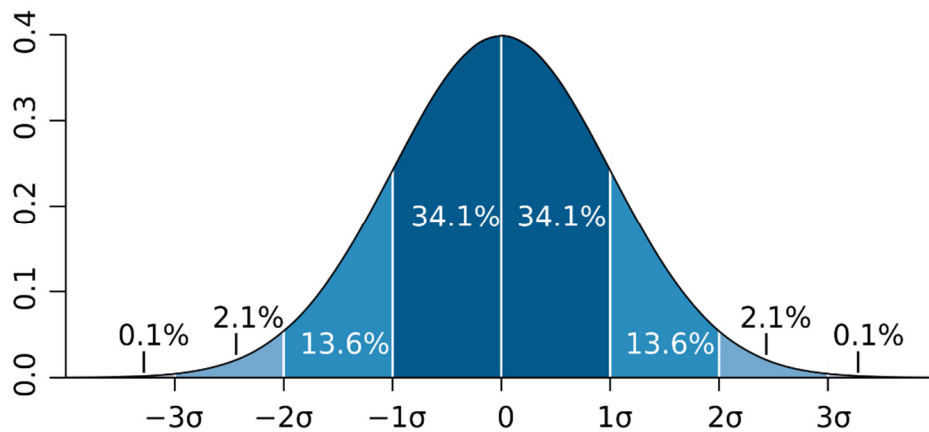
What I Know	What I Wonder	What I Learned
<ul style="list-style-type: none"><li>● Import <code>random</code></li><li>● Import <code>cyberpi</code></li><li>● Display screen of CyberPi</li><li>● <code>'cyberpi.console'</code></li><li>● <code>'cyberpi.display'</code></li></ul>		

## Procedures

### Section 1 Introduction: Normal Distributions (8 minutes)

**Step 1.1** Explain the concept of normal distribution.

- **Explain:** The graph of a standard normal distribution has a symmetrical bell-shaped curve. The mean and median are equal in the normal distribution. Its standard deviation is 1.



(Source: M. W. Toews © Wikimedia Commons)

- **Describe the graph of the normal distribution:** In a normal distribution, most of the continuous data values tend to cluster around the mean, and the further a value is from the mean. Normal distributions are important in statistics because many continuous data in nature displays this bell-shaped curve when compiled and graphed.

**Step 1.2** Use the two dice rolling probability experiment to further explain the normal distribution and its application.

- **Ask:** Suppose if you roll two dice and calculate the sum of the two dice, how many results do you get?
- Ask students to write down and summarize all the possible results of the sum of two dice as well as the frequency of occurrence:

Dice A \ Dice B	1	2	3	4	5	6
1	2	3	4	5	6	7
2	3	4	5	6	7	8
3	4	5	6	7	8	9
4	5	6	7	8	9	10
5	6	7	8	9	10	11
6	7	8	9	10	11	12

Result of Sum	2	3	4	5	6	7	8	9	10	11	12
Frequency	1	2	3	4	5	6	5	4	3	2	1

- **Say:** It seems that this is not efficient enough to demonstrate the spread of the results of the sum. Let's use Python to create a computational model for this probability experiment and graph a chart on CyberPi's screen.



## Section 2 Predict (4 minutes)

**Step 2.1** Distribute the example program file to the class. Have students read the code and discuss what the code can do before running it.

- Instruct students to write down their predictions and annotate the code.
- Ask students to figure out the points below:
  - The modules imported in this program;
  - The variables displayed in the bar chart;
  - Are '**sum\_list**' and '**count\_list**' variables? What are the values of them?
  - The function that sets the colour of the bar chart;
  - The function that creates the bars.

```
1. import cyberpi, random
2.
3. cyberpi.display.clear()
4.
5. n = int(input("The number of times to roll 2 dice: "))
6. t = 0
7. sum_list = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
8. count_list = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
9.
10. while t < n:
11.     dice_x = random.randint(1, 6)
12.     dice_y = random.randint(1, 6)
13.     print("(", dice_x, ",", dice_y, ")")
14.     result = dice_x + dice_y
15.     t += 1
16.     sum_index = sum_list.index(result)
17.     count_list[sum_index] += 1
18.
19. cyberpi.display.set_brush(128, 0, 0)
20. cyberpi.barchart.add(round(count_list[0]/n * 200, 2))
21. cyberpi.display.set_brush(220, 20, 60)
22. cyberpi.barchart.add(round(count_list[1]/n * 200, 2))
23. cyberpi.display.set_brush(255, 0, 0)
24. cyberpi.barchart.add(round(count_list[2]/n * 200, 2))
25. cyberpi.display.set_brush(205, 92, 92)
26. cyberpi.barchart.add(round(count_list[3]/n * 200, 2))
27. cyberpi.display.set_brush(233, 150, 122)
28. cyberpi.barchart.add(round(count_list[4]/n * 200, 2))
29. cyberpi.display.set_brush(255, 69, 0)
30. cyberpi.barchart.add(round(count_list[5]/n * 200, 2))
31. cyberpi.display.set_brush(255, 165, 0)
32. cyberpi.barchart.add(round(count_list[6]/n * 200, 2))
33. cyberpi.display.set_brush(255, 215, 0)
34. cyberpi.barchart.add(round(count_list[7]/n * 200, 2))
35. cyberpi.display.set_brush(240, 230, 140)
36. cyberpi.barchart.add(round(count_list[8]/n * 200, 2))
37. cyberpi.display.set_brush(255, 255, 0)
38. cyberpi.barchart.add(round(count_list[9]/n * 200, 2))
39. cyberpi.display.set_brush(154, 205, 50)
40. cyberpi.barchart.add(round(count_list[10]/n * 200, 2))
```

**Section 3    Run (3 minutes)**

**Step 3.1**      Ask students to run the example program and check against their predictions.

- Instruct students to annotate the code on the worksheet based on their observation.
- Have students fill out the **'What I Wonder'** column of the **K-W-L chart** after running the example program.

What I Know	What I Wonder	What I Learned
<ul style="list-style-type: none"><li>• Import <b>random</b></li><li>• Import <b>cyberpi</b></li><li>• Display screen of CyberPi</li><li>• <b>'cyberpi.console'</b></li><li>• <b>'cyberpi.display'</b></li></ul>	<ul style="list-style-type: none"><li>• What if I need to use more than one module, what should I do in Python?</li><li>• How to create data charts on CyberPi's screen?</li></ul>	



## Section 4 Investigate (15 minutes)

### Step 4.1 Explain the use of lists.

- **Say:** 'sum\_list' and 'count\_list' are lists in Python. A list is an ordered collection of data. For example, the 'sum\_list' contains all the possible results of the sum of the two dice, which you can see inside the square brackets.
- **Explain the list index:** We use the 'index()' method to check the position of the sum of the two dice. The 'index()' method returns the position of the given number in a list.
  - Remind students that the index of a list starts from '0'. For example, in the 'sum\_list', the index of the item '2' is '0' and the index of the item '3' is '1'.

### Step 4.2 Explain how to read and rewrite the value of an element on the list.

- **Explain:** The 'list.index[]' expression can not only return the value of an item on the list according to the given index in the square brackets but also modify the value. In the example program, the 'count\_list[sum\_index] += 1' expression is to modify the value of the corresponding element by adding '1'.
- **Explain the relation between the 'sum\_list' and 'count\_list':** The example program creates two lists: 'sum\_list' represents all the possible results of the sum of the two dice, and 'count\_list' records the frequency of the corresponding sum. The 'count\_list[sum\_index] += 1' expression counts the frequency of the sum and modify the value in the corresponding position.

### Step 4.3 Explain how to graph a bar chart on the screen.

- Ask students to identify the modules imported in Python which then allow them to program CyberPi's screen and call the 'random' functions.

```
import cyberpi, random
```

- Point out the relevant functions for graphing the bar chart:

```
cyberpi.display.clear()
```

```
cyberpi.display.set_brush()
```

```
cyberpi.barchart.add()
```

- **Explain the syntax** `cyberpi.display.clear()`: It is used to clear the content displayed on the screen. When we start a new program or project, we can use this syntax to clear the previous content shown on the screen and initialise the screen.
- **Explain the syntax** `cyberpi.display.set_brush()`: To graph the bar chart, first, we can decide which colour the bars are. The parameter inside the round brackets can be made up of numbers or a string.

If the parameters are integers – e.g. '(255, 255, 255)', these digits or values represent a specified colour in the RGB colour model. The 'RGB' represents the three kinds of additive primary colours: red, green, and blue. The parameter in the round bracket is the RGB colour code – e.g. '(red, green, blue)'.

However, the parameter of this syntax can be a string. Use these keywords to define the colour of the bar:

*'red', 'orange', 'yellow', 'green', 'cyan', 'blue', 'purple', 'white', 'black'*

- Remind students that the colour keywords must be all in lower case.
- **Explain the syntax** `cyberpi.barchart.add()`: Use this syntax to import the data we have to the bar chart.
- Explain the values represented by the bars. The values come from the '**count\_list**' that records the frequency of each sum.

## Section 5 Modify and Make (10 minutes)

**Step 5.1** Have students work individually to create another computational model for the coin toss probabilities.

- Ask students to simulate the experiment of tossing two coins together. Calculate all the possible outcomes in this experiment and visualise the distribution with a bar chart.

### Example 4-1 (a)

```

1. import cyberpi, random
2.
3. cyberpi.display.clear()
4. n = int(input("The number of times to throw up 2 coins: "))
5. event_list = [0, 0, 0]
6.
7. for i in range(0, n):
8.     coin_x = random.randint(0, 1)
9.     coin_y = random.randint(0, 1)
10.    # '0' represents 'head', '1' represents 'tail'
11.    print("(", coin_x, ",", coin_y, ")")
12.    if coin_x == 0 and coin_y == 0:
13.        # 2 heads
14.        event_list[0] += 1
15.    elif coin_x == 1 and coin_y == 1:
16.        # 2 tails
17.        event_list[2] += 1
18.    else:
19.        # 1 head, 1 tail
20.        event_list[1] += 1
21.
22. cyberpi.display.set_brush(255, 0, 0)
23. cyberpi.barchart.add(round(event_list[0]/n * 200, 2))
24. cyberpi.display.set_brush(0, 255, 0)
25. cyberpi.barchart.add(round(event_list[1]/n * 200, 2))
26. cyberpi.display.set_brush(0, 0, 255)
27. cyberpi.barchart.add(round(event_list[2]/n * 200, 2))

```

### Example 4-1 (b)

```

1. import cyberpi, random
2.
3. cyberpi.display.clear()
4. n = int(input("The number of times to throw up 2 coins: "))
5. coin_list = ["head", "tail"]
6. event_list = [0, 0, 0]
7.
8. for i in range(0, n):
9.     coin_x = random.choice(coin_list)
10.    coin_y = random.choice(coin_list)
11.    print("(", coin_x, ",", coin_y, ")")
12.    if coin_x == "head" and coin_y == "head":
13.        # 2 heads
14.        event_list[0] += 1
15.    elif coin_x == "tail" and coin_y == "tail":
16.        # 2 tails
17.        event_list[2] += 1
18.    else:
19.        # 1 head, 1 tail
20.        event_list[1] += 1
21.
22. cyberpi.display.set_brush(255, 0, 0)
23. cyberpi.barchart.add(round(event_list[0]/n * 200, 2))
24. cyberpi.display.set_brush(0, 255, 0)
25. cyberpi.barchart.add(round(event_list[1]/n * 200, 2))
26. cyberpi.display.set_brush(0, 0, 255)
27. cyberpi.barchart.add(round(event_list[2]/n * 200, 2))

```






















**Note:** Create a list to store the 'head' and 'tail' and use the 'random.choice' function to randomly select one of the outcomes.

- Ask students to think about this question: Toss a coin three times and what is the probability of getting three heads, two heads, one head, and no head?

Have students plot a graph on CyberPi's screen to demonstrate all the possible outcomes.

- Have students list all the possible outcomes.

Head, Head, Head	
------------------	--

Head, Head, Tail	  
Head, Tail, Head	  
Head, Tail, Tail	  
Tail, Head, Head	  
Tail, Head, Tail	  
Tail, Tail, Head	  
Tail, Tail, Tail	  

- Have students calculate the probabilities of the events:

$$P(3 \text{ Heads}) = P(HHH) = 1/8$$

$$P(2 \text{ Heads}) = P(HHT) + P(HTH) + P(THH) = 3/8$$

$$P(1 \text{ Head}) = P(HTT) + P(THT) + P(TTH) = 3/8$$

$$P(0 \text{ Head}) = P(TTT) = 1/8$$

- Instruct students to plot the graph on the screen. Program CyberPi to set the bar colour and the numeric values represented by the bars.

**Example 4-2**

```
1. import cyberpi
2.
3. cyberpi.display.clear()
4.
5. # 3 Heads:
6. cyberpi.display.set_brush(255, 65, 0)
7. cyberpi.barchart.add(1/8 * 200)
8.
9. # 2 Heads:
10. cyberpi.display.set_brush(255, 100, 0)
11. cyberpi.barchart.add(3/8 * 200)
12.
13. # 1 Head:
14. cyberpi.display.set_brush(255, 165, 0)
15. cyberpi.barchart.add(3/8 * 200)
16.
17. # 0 Head:
18. cyberpi.display.set_brush(255, 215, 0)
19. cyberpi.barchart.add(1/8 * 200)
```

**Step 5.2** Summarize the methods of creating a bar chart on CyberPi's screen.

- To graph a new data chart, remember to clear the screen by using this syntax:

```
cyberpi.display.clear()
```

- Define the colour of the bar by using this syntax:

```
cyberpi.display.set_brush()
```

The parameter inside the round bracket can be either the RGB colour code (integers) or the name of the colour (e.g., 'red', 'orange', 'yellow', 'green', 'cyan', 'blue', 'purple', 'white', 'black').

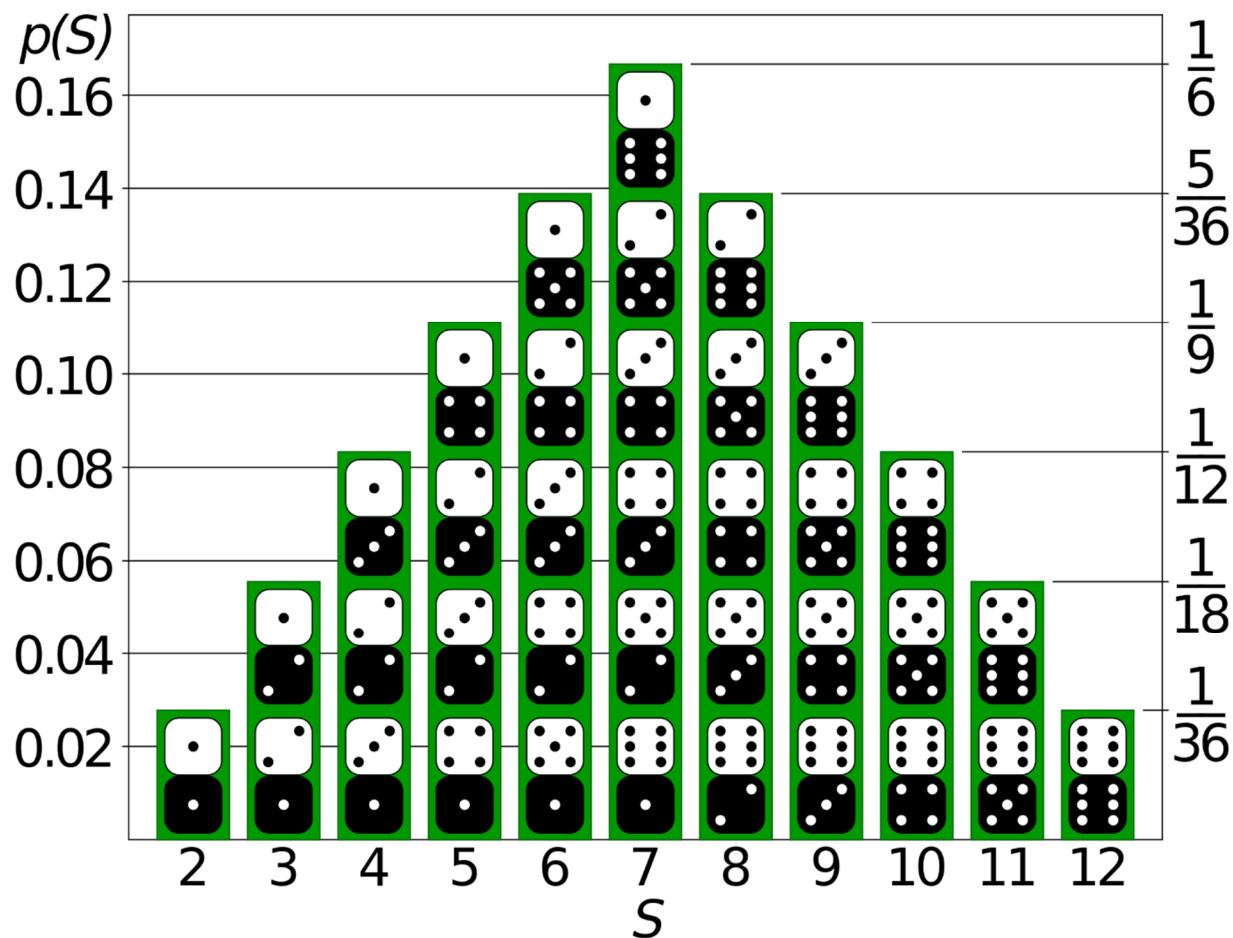
- Add a numeric value to the bar by using this syntax:

```
cyberpi.barchart.add()
```

## Section 6 Recap (5 minutes)

**Step 6.1** Summarize the key points learned in this lesson.

- Review the concept of normal distribution and the examples demonstrated in this lesson.



(Source: StackExchange Mathematics)



- Have students fill out the **'What I Learned'** column of the **K-W-L chart**.

What I Know	What I Wonder	What I Learned
<ul style="list-style-type: none"><li>• Import <b>cyberpi</b>, <b>random</b></li><li>• Display screen of CyberPi</li><li>• <b>'cyberpi.console'</b></li><li>• <b>'cyberpi.display'</b></li></ul>	<ul style="list-style-type: none"><li>• What if I need to use more than one module, what should I do in Python?</li><li>• How to create data charts on CyberPi's screen?</li></ul>	<ul style="list-style-type: none"><li>• How to use lists to store a group of data</li><li>• How to modify data stored in a list</li><li>• How to graph a bar chart on CyberPi's screen</li></ul>



## Lesson 13 Data Storage

**Category:** Python

**Level:** Introductory

**Time Frame:** 45 minutes

**Core Subject Area:** Computing

**Ages:** 11~14 years old

**Year Groups:** Key Stage 3 (UK) / Grades

6–8 (US)

### Overview

In this lesson, students will explore the data storage of a computer – in particular, the CPU and memory usage. A computer's CPU and memory usage fluctuate while the operating system handles different tasks. The utilisation of memory affects the performance of the CPU and hence the performance of the computer. Students will investigate the relationship between the utilisation of memory and the performance of the computer using Python and CyberPi.

### Key Focus

- How to display charts on the display screen of CyberPi
- Use of more than one module in Python and how to call relevant functions
- Use of the '**psutil**' module

### Intended Learning Outcomes

*By the end of this lesson, students will be able to:*

- Describe the relationship between input and output devices, the CPU and memory through simulation
- Write and execute algorithms to monitor the CPU and memory usage and demonstrate the results using data charts



## Content Standards

(UK)

### National Curriculum in England – Computing Programmes of Study: Key Stage 3

- Design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems
- Use two or more programming languages, at least one of which is textual, to solve a variety of computational problems
- Understand the hardware and software components that make up computer systems, and how they communicate with one another and with other systems
- Understand how instructions are stored and executed within a computer system; understand how data of various types (including text, sounds and pictures) can be represented and manipulated digitally, in the form of binary digits



(US)

**CSTA K-12 Computer Science Standards: Grades 6~8**

- **2-CS-01:** Recommend improvements to the design of computing devices, based on an analysis of how users interact with the devices.
- **2-DA-07:** Represent data using multiple encoding schemes.
- **2-DA-08:** Collect data using computational tools and transform the data to make it more useful and reliable.
- **2-DA-09:** Refine computational models based on the data they have generated.
- **2-AP-11:** Create clearly named variables that represent different data types and perform operations on their values.
- **2-AP-12:** Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
- **2-AP-16:** Incorporate existing code, media, and libraries into original programs, and give attribution.
- **2-IC-20:** Compare trade-offs associated with computing technologies that affect people's everyday activities and career options.



## Preparation

### For the teacher:

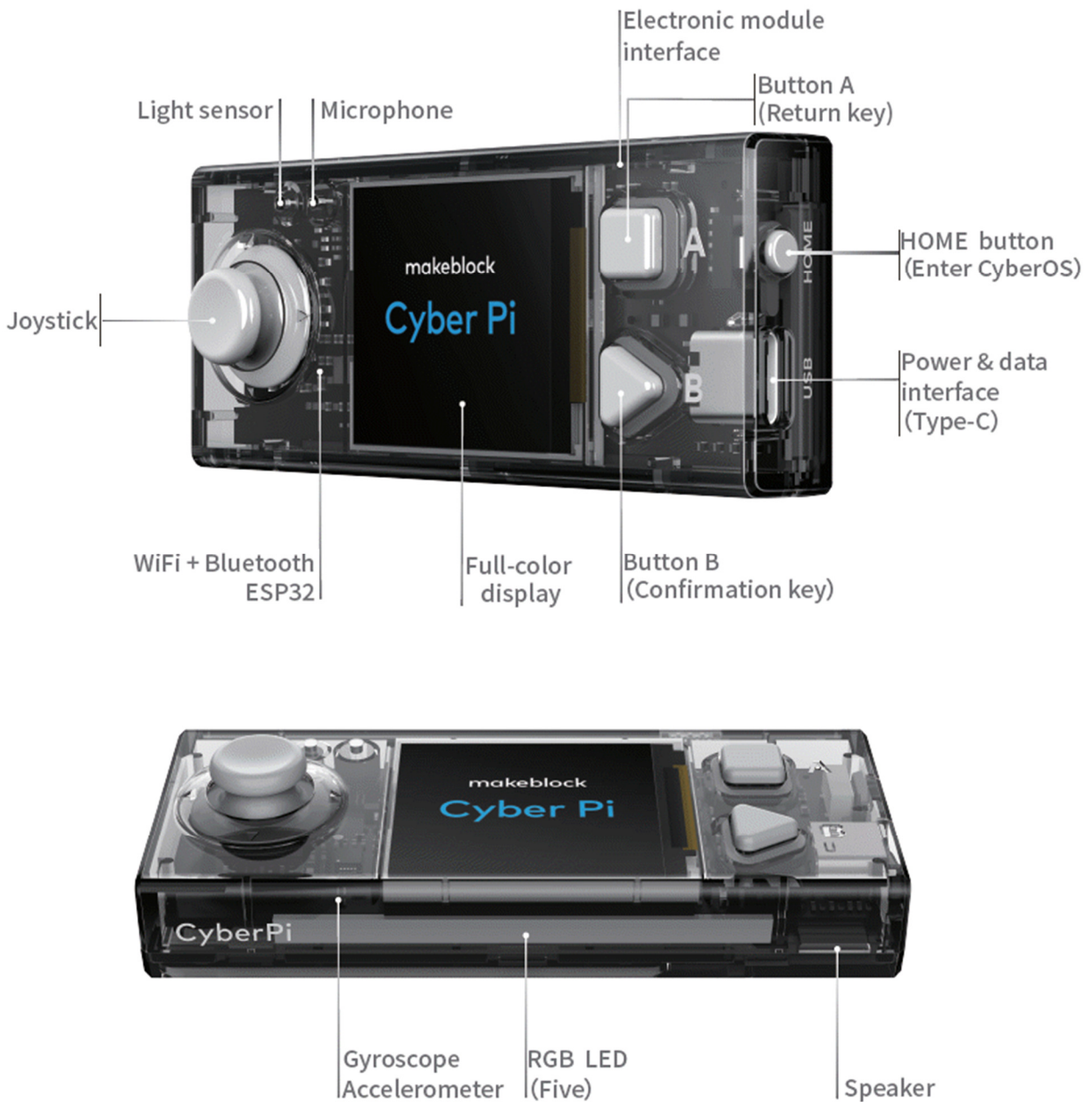
- A laptop or desktop with mBlock Python editor installed
- A CyberPi device
- A Type-C cable
- Lesson plan
- Worksheet

### For students:

- Laptops or desktops with mBlock Python editor installed
- CyberPi devices
- Type-C cables
- Worksheet



## Features of CyberPi

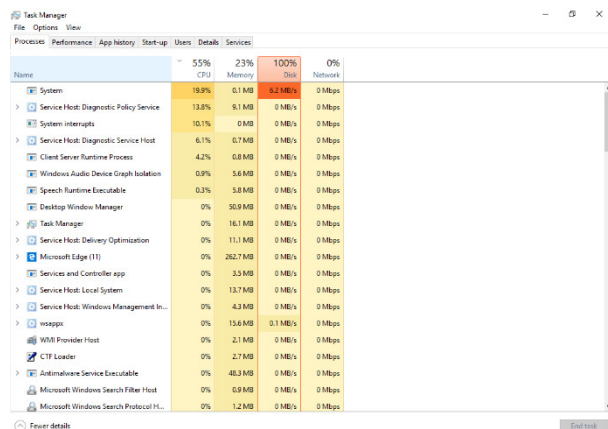


## Example Program

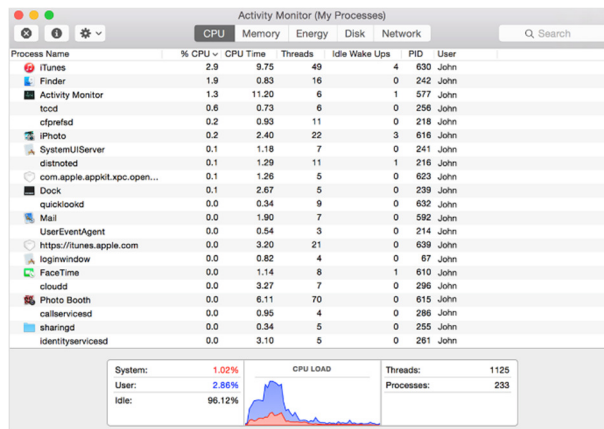
```
1. import cyberpi, psutil
2. # Import both 'cyberpi' and 'psutil' modules
3.
4. cyberpi.chart.clear()
5.
6. while True:
7.     CPU = psutil.cpu_percent()
8.     # Monitor the CPU usage
9.     mem = psutil.virtual_memory()
10.    mem_p = mem.percent
11.    # Monitor the memory usage
12.
13.    # Plot the line chart on the screen:
14.    cyberpi.display.set_brush(0, 0, 255)
15.    cyberpi.linechart.add(int(CPU))
16.    cyberpi.display.set_brush(255, 255, 0)
17.    cyberpi.linechart.add(int(mem_p))
18.    # Report the CPU and memory usage data:
19.    print("CPU:", CPU, "% Memory:", mem_p, "%")
```

## Pre-assessment

Ask students to have a look at the Task Manager (of the Window System) or the Activity Monitor (of the macOS System) on the computer before the class.



Task Manager of the Window System



Activity Monitor of the macOS System

Have students fill out the ‘What I Know’ column of the K-W-L chart before the class.

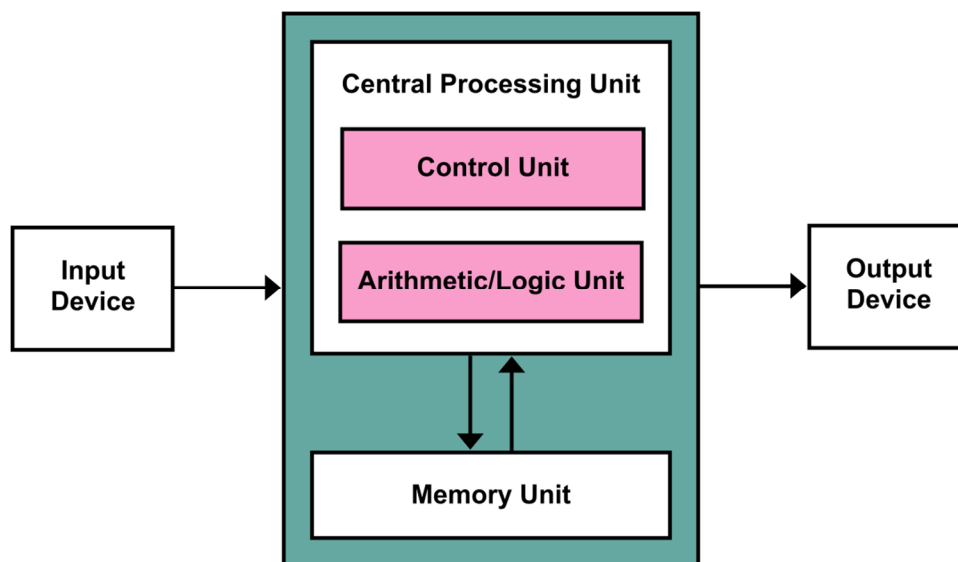
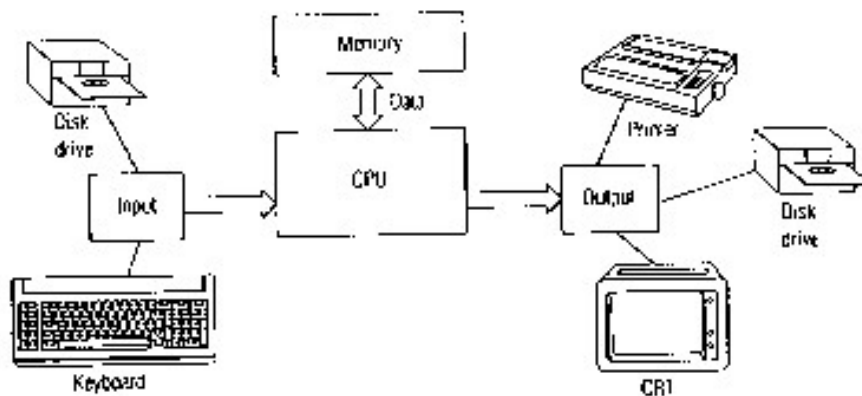
What I Know	What I Wonder	What I Learned
<ul style="list-style-type: none"> <li>● Import <code>cyberpi</code>, <code>random</code></li> <li>● Display screen of CyberPi</li> <li>● <code>'cyberpi.display'</code></li> <li>● <code>'cyberpi.barchart'</code></li> </ul>		

## Procedures

### Section 1 Introduction: CPU and Memory Usage (5 minutes)

**Step 1.1** Explain the role of the CPU and memory.

- If students already have a basic understanding of the computer system, briefly review the components of a computer through demonstration – for example, bring computer components (or microcomputers) to the classroom and ask students to identify the name of different parts.



(Source: Wikipedia, creator: Kapoht, 2013)

However, if this is your students' first time to learn about computer components, use the



above figure (Von Neumann Architecture) to explain each component. Highlight the components as followed:

- **CPU**

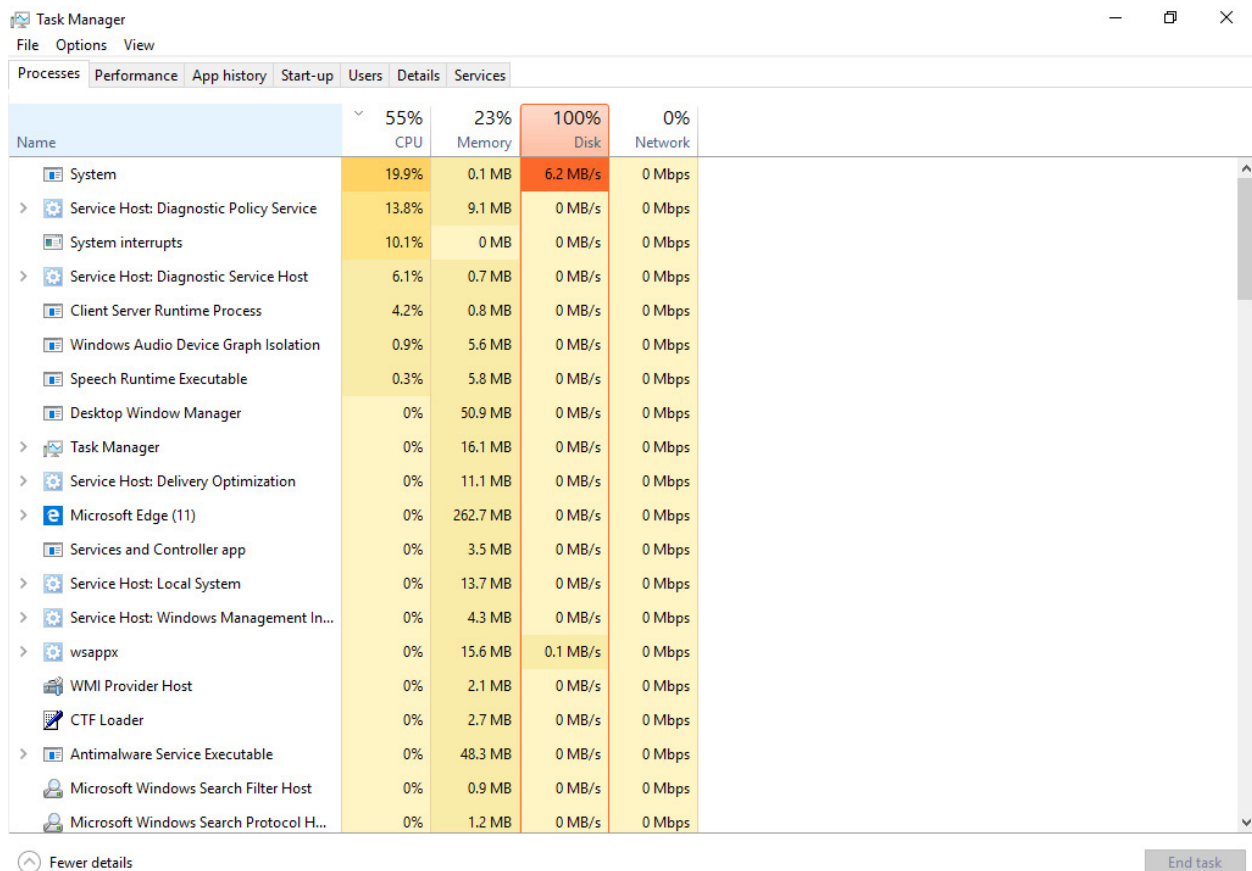
CPU stands for 'Central Processing Unit'. It is the chip that contains all the circuitries for performing arithmetic and logic operations and directing data to and from memory.

- **Memory**

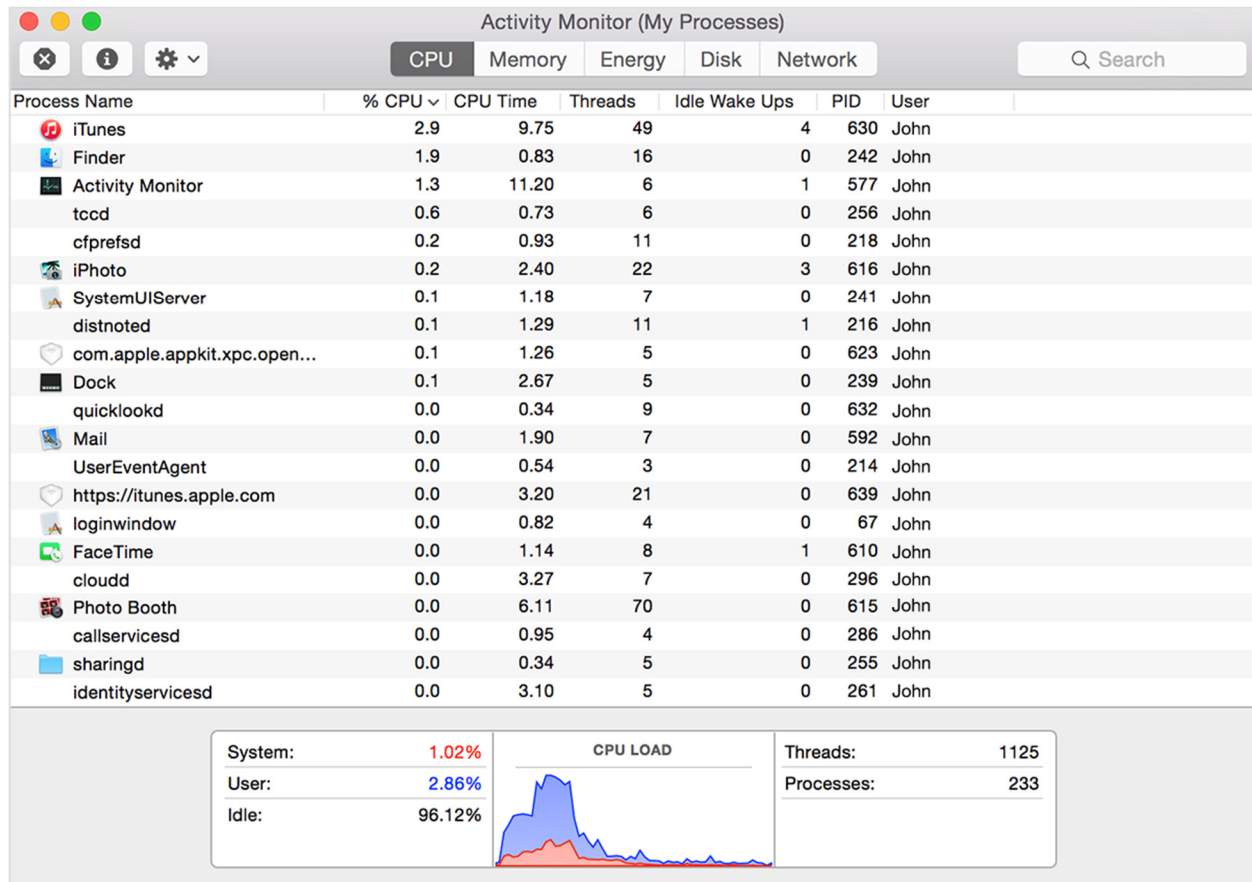
Like a human brain, computer memory is the storage space in the computer system.

Have students gain an initial understanding and impression of computer components. Explain the 'input-process-output' model of information processing in the computer system.

- Instruct students to open the Task Manager (of the Window System) or the activity monitor (of the macOS System) and check the CPU usage and memory.



Task Manager of the Window System



Activity Monitor of the macOS System

## Section 2 Predict (3 minutes)

**Step 2.1** Distribute the example program file to the class. Have students read the code and discuss what the code can do before running it.

- **Say:** Like the Task Manager or Activity Monitor, this program can show you the CPU and memory usage. Look through the scripts and consider how it displays the CPU and memory usage.

```
1. import cyberpi
2. import psutil
3.
4. cyberpi.chart.clear()
5. while True:
6.     CPU = psutil.cpu_percent()
7.     mem = psutil.virtual_memory()
8.     mem_p = mem.percent
9.     cyberpi.display.set_brush(0, 0, 255)
10.    cyberpi.linechart.add(int(CPU))
11.    cyberpi.display.set_brush(255, 255, 0)
12.    cyberpi.linechart.add(int(mem_p))
13.    print("CPU:", CPU, "% Memory:", mem_p, "%")
```

- Instruct students to write down their predictions and annotate the code on the worksheet.
- Ask students to identify the points below:
  - The library for calling functions to monitor the CPU and memory usage
  - The two variables plotted in the line chart;
  - The function that plots the lines;
  - The function that set the color of the line.

**Section 3    Run (2 minutes)**

**Step 3.1**      Ask students to run the example program and check against their predictions.

- Instruct students to write down their predictions and annotate the code.
- Have students fill out the **'What I Wonder'** column of the **K-W-L chart** after running the example program.

What I Know	What I Wonder	What I Learned
<ul style="list-style-type: none"><li>● Import cyberpi, random</li><li>● Display screen of CyberPi</li><li>● 'cyberpi.display'</li><li>● 'cyberpi.barchart'</li></ul>	<ul style="list-style-type: none"><li>● How can I create other types of charts using Python?</li><li>● What is going on inside my computer? How does the CPU usage affect it?</li></ul>	

## Section 4 Investigate (20 minutes)

**Step 4.1** Invite volunteer students to report and share their findings.

- Students should identify the following functions:
  - The library for retrieving information on running processes and system utilisation (such as the CPU and memory usage demonstrated in this example program):

```
import psutil
```

- To retrieve data on the CPU usage:

```
psutil.cpu_percent()
```

**Note:** Have students write and run the code below in the Python editor to experience how this function works:

### Example 5-1

```
1. import psutil, time
2.
3. while True:
4.     CPU = psutil.cpu_percent()
5.     print("CPU Usage:", CPU, "%")
6.     time.sleep(0.2)
```

- To retrieve data on memory usage:

```
psutil.virtual_memory()
```

**Note:** Have students write and run the code below in the Python editor to experience how this function works:

### Example 5-2

```
1. import psutil, time
2.
3. while True:
4.     memory = psutil.virtual_memory()
5.     print(memory)
6.     time.sleep(0.2)
```

**Note:** Ask students to look at the results displayed in the console:

```
svmem(total=8479207424, available=383148032, percent=95.5, used=8096059392, free=383148032)
svmem(total=8479207424, available=384077824, percent=95.5, used=8095129600, free=384077824)
svmem(total=8479207424, available=384045056, percent=95.5, used=8095162368, free=384045056)
svmem(total=8479207424, available=384258048, percent=95.5, used=8094949376, free=384258048)
svmem(total=8479207424, available=380776448, percent=95.5, used=8098430976, free=380776448)
svmem(total=8479207424, available=380792832, percent=95.5, used=8098414592, free=380792832)
svmem(total=8479207424, available=380801024, percent=95.5, used=8098406400, free=380801024)
svmem(total=8479207424, available=381706240, percent=95.5, used=8097501184, free=381706240)
svmem(total=8479207424, available=381878272, percent=95.5, used=8097329152, free=381878272)
svmem(total=8479207424, available=381886464, percent=95.5, used=8097320960, free=381886464)
svmem(total=8479207424, available=381882368, percent=95.5, used=8097325056, free=381882368)
svmem(total=8479207424, available=381890560, percent=95.5, used=8097316864, free=381890560)
svmem(total=8479207424, available=377241600, percent=95.6, used=8101965824, free=377241600)
svmem(total=8479207424, available=343040000, percent=96.0, used=8136167424, free=343040000)
svmem(total=8479207424, available=299368448, percent=96.5, used=8179838976, free=299368448)
svmem(total=8479207424, available=276340736, percent=96.7, used=8202866688, free=276340736)
svmem(total=8479207424, available=262914048, percent=96.9, used=8216293376, free=262914048)
svmem(total=8479207424, available=247328768, percent=97.1, used=8231878656, free=247328768)
svmem(total=8479207424, available=235933696, percent=97.2, used=8243273728, free=235933696)
svmem(total=8479207424, available=294957056, percent=96.5, used=8184250368, free=294957056)
svmem(total=8479207424, available=360026112, percent=95.8, used=8119181312, free=360026112)
svmem(total=8479207424, available=394805248, percent=95.3, used=8084402176, free=394805248)
svmem(total=8479207424, available=394907648, percent=95.3, used=8084299776, free=394907648)
svmem(total=8479207424, available=394776576, percent=95.3, used=8084430848, free=394776576)
```

This expression returns a set of data about the system memory usage, including the total physical memory, the available memory that can be assigned instantly to processes, the memory usage in percentages, and more.

To extract the data that is needed, it is necessary to use the `'memory.percent'` to select the dataset of the memory usage in percentage.

- To plot the line:

```
cyberpi.linechart.add()
```

- To set the color of the line:

```
cyberpi.display.set_brush()
```

**Step 4.2** Have students run some tasks on their computer while executing the program. Ask them to investigate how their operation may affect the CPU and memory usage.

**Step 4.3** Explain how to measure and assess the performance of the CPU of the computer.

- **Say:** We could use the percentage of the CPU usage as an indicator of the CPU performance. However, it is difficult to assess, because a computer might excel at some tasks but not perform so well at other tasks.
- **Explain:** Four factors affect the CPU performance: the number of cores, the clock speed or rate, the cache size, and the type of CPU.

- **Explain the cores:** A core is a processing unit of the CPU. The CPU can contain more than one core. Computers nowadays have 4, 6, 8, and even 10 cores. The more cores a computer has, the more power the computer gains to handle tasks at the same time. Yet it does not mean that doubling the number of cores will double a computer's performance or processing speed.
- Instruct students to use the '**pustil**' functions to check the number of cores their computers have.

**Tip:** Use the syntax: `psutil.cpu_count()`

#### Example 5-3

```
1. import psutil
2.
3. core = psutil.cpu_count()
4. print("Number of Cores:", core)
```

- **Explain the clock speed:** The clock speed or rate indicates how fast the CPU can run. This is measured in megahertz (MHz) or gigahertz (GHz). The clock speed describes the number of tasks or activities the CPU can deal with in a second, that is, the frequency of the CPU performance. A computer normally has a maximum clock speed.
- Instruct students to use the '**pustil**' functions to read the minimum and maximum clock speed of their computers.

**Tip:** Use the syntax: `psutil.cpu_freq()`

**Example 5-4**

```

1. import psutil
2.
3. speed = psutil.cpu_freq()
4. max_speed = speed.max
5. print("Maximum Clock Speed:", max_speed, "Mhz")
6. min_speed = speed.min
7. print("Minimum Clock Speed:", min_speed, "Mhz")

```

- **Explain the cache:** Cache is a small amount of memory which is a part of the CPU. It is used to temporarily hold instructions and data that the CPU is likely to reuse.
- **Briefly mention the two types of the processor:** There are two types of CPU: Complex Instruction Set Computing (CISC) and Reduced Instruction Set Computing (RISC). The latter type of CPU is usually used in smartphones and tablets. We will explore the two types of CPU in the next lesson.
- Have students discuss the features of lower and higher CPU performances based on the information mentioned above. Ask them to summarize and fill out the table below:

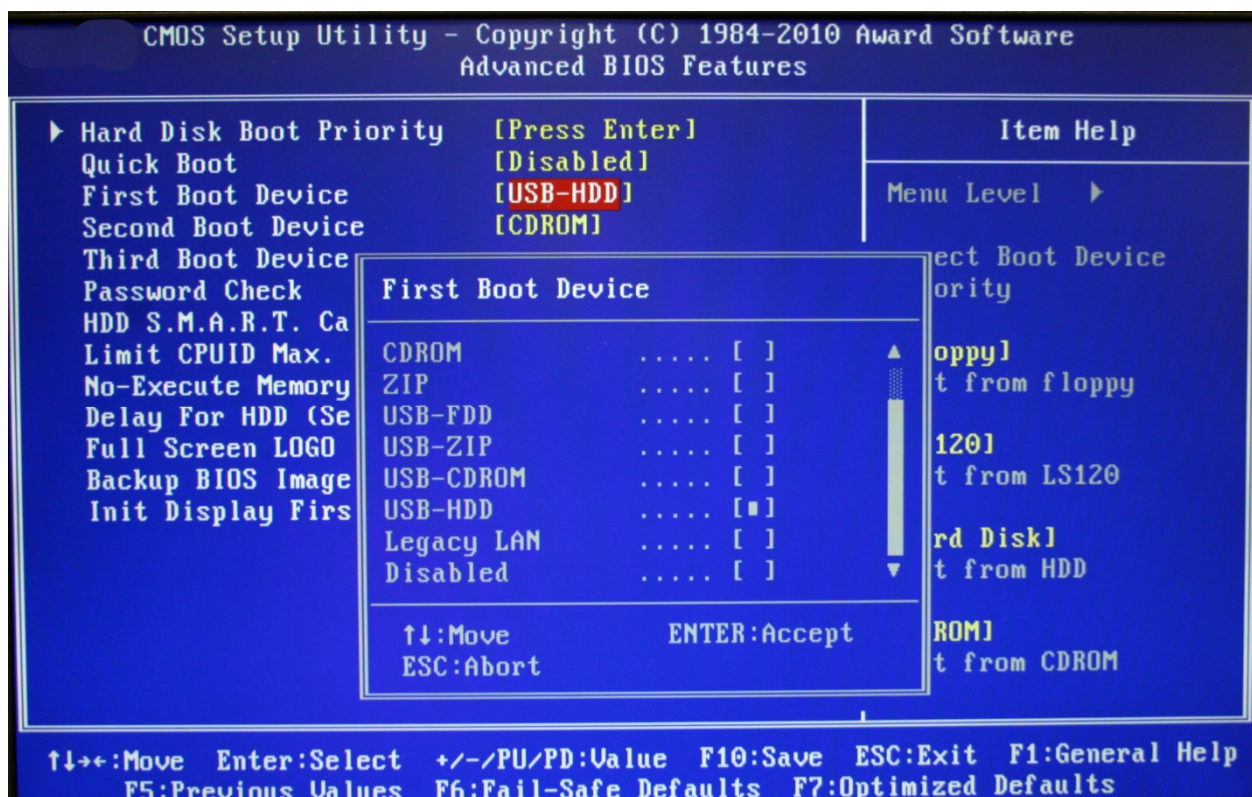
Lower CPU performance	Higher CPU performance

**Step 4.4** Explain the two types of computer memory: Random Access Memory (RAM) and Read Only Memory (ROM).

- **Explain the RAM:** Random Access Memory, or RAM, stores user programs that control what the CPU does including the data used by these programs and the results of operations performed by these programs. RAM is accessible to the user. The memory size of RAM affects computer performance.  
  
RAM is a kind of volatile memory. It means that everything stored in RAM is lost when the computer is switched off, even for an instant.



- Explain the ROM:** Read Only Memory, or ROM, stores the instructions a computer needs to get itself started after the user turns on the power. As its name indicates, ROM cannot be modified by the user, which means the data stored in ROM can only be read by the user. Unlike RAM, things stored in ROM will not be lost when the computer is switched off. The data and instructions are still stored in ROM even when the computer is switched off.
- Say:** The Basic Input Output System is an example of a program that is stored in ROM. The BIOS runs as soon as the power is turned on.



(Source: Wikimedia Commons © Toniperis)

## Section 5    Modify and Make (10 minutes)

**Step 5.1** Have students work individually to create a CPU usage alarm based on the example program.

- Ask students to add control structures and light effects to the example program. Define the thresholds of the alarm and the corresponding alarm indicators. For example, if the CPU usage exceeds a threshold of 70% usage, CyberPi gives a red warning light; if the CPU usage is between 50% and 70%, CyberPi lights up in orange (or amber).
- Encourage students to program other functions. For example, hint at adding an alarm sound by using the syntax `cyberpi.audio.play_tone()`. This function can make CyberPi play the sound of a buzzer. The first parameter in the round bracket refers to the buzzer's frequency range between 20Hz and 5000Hz. Remind students that they should use an appropriate frequency and avoid high-frequency sounds to protect their ears. The second parameter represents the duration of the sound.

### Example 5-5

```
cyberpi.audio.play_tone(1047, 0.3)
```

```
cyberpi.audio.play_tone(262, 0.3)
```

- Remind students that if they want to add a chart title and display it on CyberPi's screen, they can use the syntax:

```
cyberpi.chart.set_name()
```

**Note:** Students may ignore one thing: the content of the chart title should be a strings variable. The values of the two variables `'CPU'` and `'mem_p'` are integers. Students need to debug and convert the data type.

**Example 5-6**

```
1. import cyberpi, psutil
2.
3. cyberpi.chart.clear()
4.
5. while True:
6.     CPU = psutil.cpu_percent()
7.     mem = psutil.virtual_memory()
8.     mem_p = mem.percent
9.
10.    cyberpi.chart.set_name("CPU: " + str(CPU) + "%")
11.
12.    cyberpi.display.set_brush(255, 0, 0)
13.    cyberpi.linechart.add(int(CPU))
14.    cyberpi.display.set_brush(0, 0, 255)
15.    cyberpi.linechart.add(int(mem_p))
16.
17.    if CPU >= 70:
18.        cyberpi.led.on("red")
19.        cyberpi.audio.play_tone(1047, 0.3)
20.
21.    elif 70 > CPU >= 50:
22.        cyberpi.led.on("orange")
23.        cyberpi.audio.play_tone(262, 0.3)
24.
25.    else:
26.        cyberpi.led.on("white")
```

## Section 6 Recap (5 minutes)

**Step 6.1** Summarize features of the CPU performance.

Lower CPU performance	Higher CPU performance
Single-core	Multi-core
Low clock speed	High clock speed
Small or no cache	Large, multi-level cache

- Summarize the key points:
  - A multi-core CPU will have a higher performance than a single-core CPU with the same clock speed.
  - A CPU with a high clock speed will process more instructions per second and will, therefore, have a higher performance than the equivalent CPU with the lower clock speed.
  - A larger cache size suggests a higher CPU performance because the CPU will spend less time accessing RAM so programs will execute faster.
- Have students fill out the 'What I Learned' column of the K-W-L chart.

What I Know	What I Wonder	What I Learned
<ul style="list-style-type: none"> <li>Import <code>cyberpi, random</code></li> <li>Display screen of CyberPi</li> <li><code>'cyberpi.display'</code></li> <li><code>'cyberpi.barchart'</code></li> </ul>	<ul style="list-style-type: none"> <li>How can I create other types of charts using Python?</li> <li>What is going on inside my computer? How does the CPU usage affect it?</li> </ul>	<ul style="list-style-type: none"> <li>Import <code>cyberpi, psutil</code></li> <li><code>'cyberpi.linechart'</code></li> <li>The roles of the CPU and computer memory</li> <li>Factors that affect the CPU performance</li> </ul>

## Lesson 14    Remix Culture

**Category:** Python

**Level:** Introductory

**Time Frame:** 45 minutes

**Core Subject Area:** Computing

**Supplementary Subject Area:** Music

**Ages:** 11~14 years old

**Year Groups:** Key Stage 3 (UK) / Grades

6–8 (US)

### Overview

This lesson will introduce the use of functions in Python. A function is a set of well-defined, organised, and reusable code that can perform a specific task when it is called. Using functions can reduce duplication of code, improve the clarity of code, and decompose complex problems into simpler pieces while creating complex algorithms. In this lesson, students will explore the use of functions by creating functions for musical sounds. Students will use the computer keyboard to combine sound effects and play sounds.

### Key Focus

- Use of functions
- Use of the **'pynput'** module
- Audio features of CyberPi

### Intended Learning Outcomes

*By the end of this lesson, students will be able to:*

- Recognise the syntax and features of a function in Python, including the keyword, the rule of indentation, and the method to call and reuse the function
- Write and execute programs to control and monitor the computer keyboard input by using the **'pynput'** functions
- Create and execute functions to store and modify bars of music notes in Python



## Content Standards

(UK)

### National Curriculum in England – Computing Programmes of Study: Key Stage 3

- Design, use and evaluate computational abstractions that model the state and behaviour of real-world problems and physical systems
- Understand several key algorithms that reflect computational thinking; use logical reasoning to compare the utility of alternative algorithms for the same problem
- Use two or more programming languages, at least one of which is textual, to solve a variety of computational problems
- Make appropriate use of data structures; design and develop modular programs that use procedures or functions
- Understand how instructions are stored and executed within a computer system; understand how data of various types (including text, sounds and pictures) can be represented and manipulated digitally, in the form of binary digits
- Undertake creative projects that involve selecting, using, and combining multiple applications, preferably across a range of devices, to achieve challenging goals, including collecting and analysing data and meeting the needs of known users
- Create, re-use, revise and re-purpose digital artefacts for a given audience, with attention to trustworthiness, design and usability



(US)

**CSTA K-12 Computer Science Standards: Grades 6~8**

- **2-CS-02:** Design projects that combine hardware and software components to collect and exchange data.
- **2-DA-08:** Collect data using computational tools and transform the data to make it more useful and reliable.
- **2-AP-11:** Create clearly named variables that represent different data types and perform operations on their values.
- **2-AP-12:** Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
- **2-AP-13:** Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
- **2-AP-14:** Create procedures with parameters to organize code and make it easier to reuse.
- **2-AP-16:** Incorporate existing code, media, and libraries into original programs, and give attribution.
- **2-IC-20:** Compare tradeoffs associated with computing technologies that affect people's everyday activities and career options.
- **2-IC-21:** Discuss issues of bias and accessibility in the design of existing technologies.
- **2-IC-22:** Collaborate with many contributors through strategies such as crowdsourcing or surveys when creating a computational artifact.



## Preparation

### For the teacher:

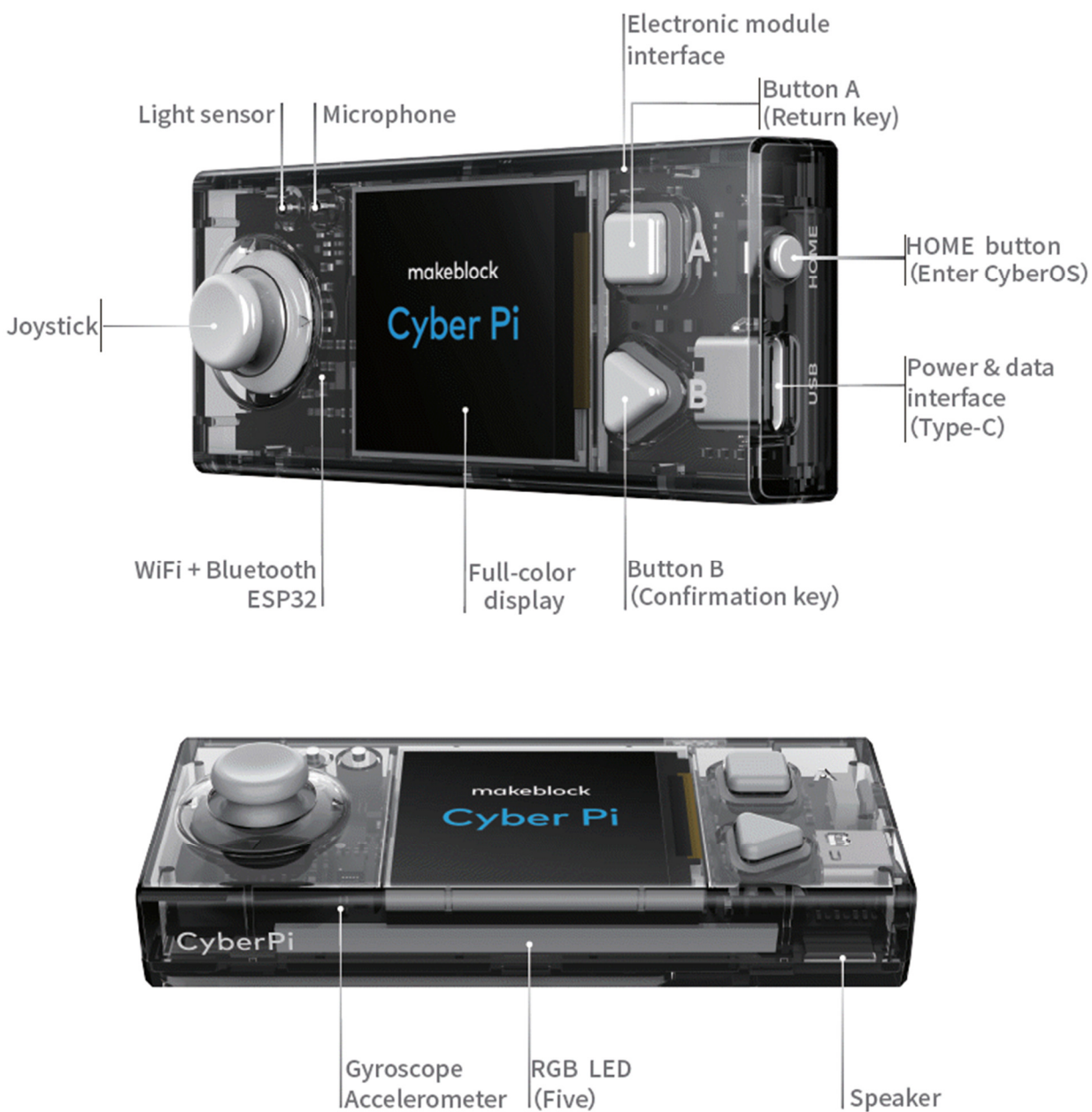
- A laptop or desktop with mBlock Python code editor installed
- A CyberPi device
- A Type-C cable
- Lesson plan
- Worksheet

### For students:

- Laptops or desktops with mBlock Python code editor installed
- CyberPi devices
- Type-C cables
- Worksheet



## Features of CyberPi





## Example Program

```
1. import cyberpi
2. from pynput.keyboard import Key, Listener
3. # Monitor the keyboard input
4. cyberpi.audio.set_vol(50)
5. # Adjust the volume on CyberPi
6. def bar1():
7.     cyberpi.audio.play_music(60, 0.2)
8.     cyberpi.audio.play_music(64, 0.2)
9.     cyberpi.audio.play_music(67, 0.2)
10. def bar2():
11.     cyberpi.audio.play_music(64, 0.2)
12.     cyberpi.audio.play_music(65, 0.2)
13.     cyberpi.audio.play_music(69, 0.2)
14. def bar3():
15.     cyberpi.audio.play_music(64, 0.2)
16.     cyberpi.audio.play_music(67, 0.2)
17.     cyberpi.audio.play_music(71, 0.2)
18. def bar4():
19.     cyberpi.audio.play_music(65, 0.2)
20.     cyberpi.audio.play_music(69, 0.2)
21.     cyberpi.audio.play_music(72, 0.2)
22. def on_press(key):
23.     if key.char == "1":
24.         # A key produces a character value '1'
25.         cyberpi.led.on("red")
26.         bar1()
27.     if key.char == "2":
28.         # A key produces a character value '2'
29.         cyberpi.led.on("orange")
30.         bar2()
31.     if key.char == "3":
32.         # A key produces a character value '3'
33.         cyberpi.led.on("yellow")
34.         bar3()
35.     if key.char == "4":
36.         # A key produces a character value '4'
37.         cyberpi.led.on("green")
38.         bar4()
39. def on_release(key):
40.     cyberpi.led.off()
41.     pass
42. with Listener(on_press=on_press, on_release=on_release) as listener:
43.     # Collect events until released
44.     listener.join()
```



## Pre-assessment

Have students fill out the **'What I Know'** column of the **K-W-L chart** before the class.

What I Know	What I Wonder	What I Learned
<ul style="list-style-type: none"><li>• The <b>'psutil'</b> module</li><li>• Import <b>cyberpi, psutil</b></li></ul>		



## Procedures

### Section 1 Introduction: Remix Culture and Creativity (2 minutes)

#### Step 1.1 Introduce the remix culture.

- **Explain the concept of remix culture:** Remix culture refers to a cultural practice of artists that create and produce creative products by combining or editing existing materials. Sometimes, remix culture is also called read-write culture, which indicates that the cultural artefacts may not be considered as the original work of someone and hence are indicated as 'cultural collective work'.
- **Say:** Digital technologies are suited for adaptation and remixing and facilitate the remix culture creation. In music, for example, we can use music applications to edit and modify a piece and combine different parts of existing songs to create a new piece of music.



## Section 2 Predict (5 minutes)

- **Step 2.1** Distribute the example program file to the class. Have students read the code and discuss what the code does before running it.
  - **Introduce the example program:** This program allows you to remix a song by combining different segments of chords. Before you run the program, read the script, identify the new syntax in the example program, and guess how it re-organizes the track.

```
1. import cyberpi
2. from pynput.keyboard import Key, Listener
3. cyberpi.audio.set_vol(50)
4.
5. def bar1():
6.     cyberpi.audio.play_music(60, 0.2)
7.     cyberpi.audio.play_music(64, 0.2)
8.     cyberpi.audio.play_music(67, 0.2)
9.
10. def bar2():
11.     cyberpi.audio.play_music(64, 0.2)
12.     cyberpi.audio.play_music(65, 0.2)
13.     cyberpi.audio.play_music(69, 0.2)
14.
15. def bar3():
16.     cyberpi.audio.play_music(64, 0.2)
17.     cyberpi.audio.play_music(67, 0.2)
18.     cyberpi.audio.play_music(71, 0.2)
19.
20. def bar4():
21.     cyberpi.audio.play_music(65, 0.2)
22.     cyberpi.audio.play_music(69, 0.2)
23.     cyberpi.audio.play_music(72, 0.2)
24.
25. def on_press(key):
26.     if key.char == "1":
27.         cyberpi.led.on("red")
28.         bar1()
29.     if key.char == "2":
30.         cyberpi.led.on("orange")
31.         bar2()
32.     if key.char == "3":
33.         cyberpi.led.on("yellow")
34.         bar3()
35.     if key.char == "4":
36.         cyberpi.led.on("green")
37.         bar4()
38.
39. def on_release(key):
40.     cyberpi.led.off()
41.     pass
42.
43. with Listener(on_press=on_press, on_release=on_release) as listener:
44.     listener.join()
```



- Ask students to think about the questions below:
  - Identify the new module that can monitor the input from your keyboard.
  - What is meant by the Python keyword 'def'?
  - Why does it separate the set of expressions within each 'def' code block?
  - How to produce interactive sound and light effects?
- Instruct students to write down their predictions and annotate the code on the worksheet.



**Section 3    Run (3 minutes)**

**Step 3.1**      Ask students to run the example program and check it against their predictions.

- Instruct students to write down their predictions and annotate the code.
- Have students fill out the **'What I Wonder'** column of the **K-W-L chart** after running the example program. Ask students to write down the main points of what they will learn in this lesson.

What I Know	What I Wonder	What I Learned
<ul style="list-style-type: none"><li>● The <b>'psutil'</b> module</li><li>● Import <b>cyberpi, psutil</b></li></ul>	<ul style="list-style-type: none"><li>● What is meant by the word <b>'def'</b>?</li><li>● How to play sounds or music through CyberPi?</li></ul>	

## Section 4 Investigate (20 minutes)

### Step 4.1 Introduce the 'pynput' module.

- Instruct students to identify the line of code below and figure out what is different about this statement.

```
from pynput.keyboard import Key, Listener
```

- **Say:** In the example program, the 'pynput' module is added into the Python code editor. The 'pynput' module is a third-party library that controls and monitors input devices.
- **Explain the 'pynput' module:** In the example, we call relevant 'pynput' functions to monitor keyboard input by obtaining the current status of the keyboard. The program can monitor which key is pressed or released. 'pynput' can also monitor mouse input and even control the keyboard and mouse. For example, some functions can make the computer type a word.
- **Explain the syntax:** In the example, the 'from' indicates the source of the library that we want to import. The 'import' indicates the set of functions we need in the library – the 'Key, Listener' means that we want functions that can monitor the keyboard input.
- Instruct students to utilise 'pynput' to control the mouse. Demonstrate the following example program:
  - Send the file to students and ask them to run the program first.

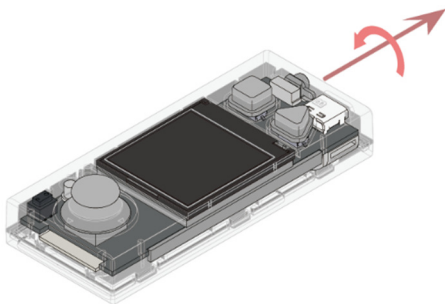
**Example 6-1**

```
1. import cyberpi
2. from pynput.mouse import Button, Controller
3. # Control the mouse
4.
5. mouse = Controller()
6.
7. while True:
8.     if cyberpi.is_tiltleft():
9.         # Tilt CyberPi left
10.        mouse.move(-5, 0)
11.        # Move the mouse cursor relative to current position
12.        print(mouse.position)
13.
14.    if cyberpi.is_tiltright():
15.        # Tilt CyberPi right
16.        mouse.move(5, 0)
17.        print(mouse.position)
18.
19.    if cyberpi.controller.is_press("a"):
20.        # Press CyberPi's Button A
21.        mouse.press(Button.left)
22.        mouse.release(Button.left)
23.        # Press and release the left mouse button
```

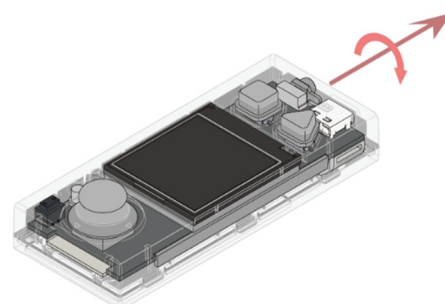
- Then ask students to add new functions that make the mouse cursor move up and down by tilting CyberPi forward and backward.

**Tips:** The syntax for reference:

```
cyberpi.is_tiltforward; cyberpi.is_tiltback
```



CyberPi is tilted forward



CyberPi is tilted backward

**Example 6-2**

```

1. if cyberpi.is_tiltforward():
2.     mouse.move(0, -5)
3.
4. if cyberpi.is_tiltback():
5.     mouse.move(0, 5)

```

- Ask students to add a new function to program the right mouse button to be remotely controlled by CyberPi.

**Example 6-3**

```

1. if cyberpi.controller.is_press("b"):
2.     mouse.press(Button.right)
3.     mouse.press(Button.right)

```

- **Summarize:** The keyword 'Key' refers to the keyboard. The keyword 'Button' refers to the mouse. The keyword 'Listener' indicates the action of monitoring an input device while the keyword 'Controller' indicates the action of controlling an input device.

**Step 4.2** Explain the use of functions.

- Instruct students to identify the lines of code below in the example program:

**Line 5:** `def bar1():`

**Line 10:** `def bar2():`

**Line 15:** `def bar3():`

**Line 20:** `def bar4():`

**Line 25:** `def on_press(key):`

**Line 39:** `def on_release(key):`

- **Say:** The keyword 'def' indicates that the following lines of code are a function.
- **Explain:** A function is a group of related statements that performs a specific task. A function contains a set of well-defined, organized, and reusable code. To create a function, we need to use the keyword 'def' as the function header.

To define the function, we need to use the indentation to indicate that a group of code

belongs to the function. In the example program, for instance, we create the functions **'bar1'**, **'bar2'**, **'bar3'**, and **'bar4'** to store and represent the bars.

- Remind students that the name of a function, like the name of a variable, should be readable and explanatory.
- Instruct students to define a function. Have them pay attention to the following points:
  - Start with the keyword **'def'** and do not forget the colon;
  - Indent the statement, otherwise an error will occur.
- **Explain the statement below:** The word **'key'** in the round brackets is a parameter that indicates the input device.

```
def on_press(key)
```

```
def on_release(key)
```

**Step 4.3** Explain how to play sounds.

- Instruct students to identify the lines of code below:

```
key.char == "1"
```
- **Explain:** This expression is to check whether Key **'1'** is pressed or not; if Key **'1'** is pressed (i.e., the statement is **'True'**) execute the **'bar1'** function.

We can modify the parameters to use other characters such as **'a'**, **'b'**, and **'c'**.



## Section 5    Modify and Make (10 minutes)

**Step 5.1** Instruct students to work individually on modifying the example program by completing the task below:

- **Task 1:** Modify the parameters in the example program. Use letters to replace the numeric parameters.

### Example 6-4

```
key.char == "a"
```

```
key.char == "b"
```

```
key.char == "c"
```

```
key.char == "d"
```

**Note:** Remind students that they should use lowercase letters.

- **Task 2:** Modify the 'bar' functions. Find some pieces of songs from music textbooks and combine different parts of them together to make a new song. Consider how to combine the sound effects and LED lights.

## Example 6-5

```
1. def bar1():
2.     cyberpi.led.on("green", id=1)
3.     cyberpi.audio.play_music(72, 0.4)
4.     cyberpi.audio.play_music(67, 0.4)
5.     cyberpi.led.on("green", id=2)
6.     cyberpi.audio.play_music(64, 0.2)
7.     cyberpi.audio.play_music(64, 0.1)
8.     cyberpi.audio.play_music(65, 0.1)
9.     cyberpi.audio.play_music(67, 0.4)
10.    cyberpi.led.on("green", id=3)
11.    cyberpi.audio.play_music(72, 0.1)
12.    cyberpi.audio.play_music(71, 0.1)
13.    cyberpi.audio.play_music(72, 0.1)
14.    cyberpi.audio.play_music(74, 0.1)
15.    cyberpi.audio.play_music(72, 0.1)
16.    cyberpi.audio.play_music(71, 0.1)
17.    cyberpi.audio.play_music(72, 0.1)
18.    cyberpi.audio.play_music(74, 0.1)
19.    cyberpi.led.on("green", id=4)
20.    cyberpi.audio.play_music(72, 0.1)
21.    cyberpi.audio.play_music(71, 0.1)
22.    cyberpi.audio.play_music(72, 0.1)
23.    cyberpi.audio.play_music(74, 0.1)
24.    cyberpi.audio.play_music(72, 0.4)
25.
26.
27. def on_press(key):
28.     if key.char == "1":
29.         bar1()
```

## Example 6-6

```
1. def bar2():
2.     cyberpi.led.on("orange", id=1)
3.     cyberpi.audio.play_music(71, 0.1)
4.     cyberpi.audio.play_music(69, 0.1)
5.     cyberpi.audio.play_music(68, 0.1)
6.     cyberpi.audio.play_music(69, 0.1)
7.     cyberpi.led.on("orange", id=2)
8.     cyberpi.audio.play_music(72, 0.2)
9.     time.sleep(0.2)
10.    cyberpi.audio.play_music(74, 0.1)
11.    cyberpi.audio.play_music(72, 0.1)
12.    cyberpi.audio.play_music(71, 0.1)
13.    cyberpi.audio.play_music(72, 0.1)
14.    cyberpi.led.on("orange", id=3)
15.    cyberpi.audio.play_music(76, 0.2)
16.    time.sleep(0.2)
17.    cyberpi.audio.play_music(77, 0.1)
18.    cyberpi.audio.play_music(76, 0.1)
19.    cyberpi.audio.play_music(75, 0.1)
20.    cyberpi.audio.play_music(76, 0.1)
21.    cyberpi.led.on("orange", id=4)
22.    cyberpi.audio.play_music(83, 0.1)
23.    cyberpi.audio.play_music(81, 0.1)
24.    cyberpi.audio.play_music(80, 0.1)
25.    cyberpi.audio.play_music(81, 0.1)
26.    cyberpi.audio.play_music(83, 0.1)
27.    cyberpi.audio.play_music(81, 0.1)
28.    cyberpi.audio.play_music(80, 0.1)
29.    cyberpi.audio.play_music(81, 0.1)
30.    cyberpi.led.on("orange", id=5)
31.    cyberpi.audio.play_music(84, 0.2)
32.
33.
34. def on_press(key):
35.     if key.char == "2":
36.         bar2()
```



**Section 6    Recap (5 minutes)**

**Step 6.1** Summarize the key points learned in this lesson:

- Summarize why and how to define functions.
- Summarize how to control and monitor the input device by using the **'pynput'** module.
- Have students fill out the **'What I Learned'** column of the **K-W-L chart**.

What I Know	What I Wonder	What I Learned
<ul style="list-style-type: none"><li>• The <b>'psutil'</b> module</li><li>• Import <b>cyberpi, psutil</b></li></ul>	<ul style="list-style-type: none"><li>• What is meant by the word <b>'def'</b>?</li><li>• How to play sounds or music through CyberPi?</li></ul>	<ul style="list-style-type: none"><li>• The <b>'pynput'</b> module</li><li>• Import <b>cyberpi, pynput</b></li><li>• Functions</li><li>• Audio features of the CyberPi</li></ul>