Lesson 1~2 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 LED strip. 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 LED strip, 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

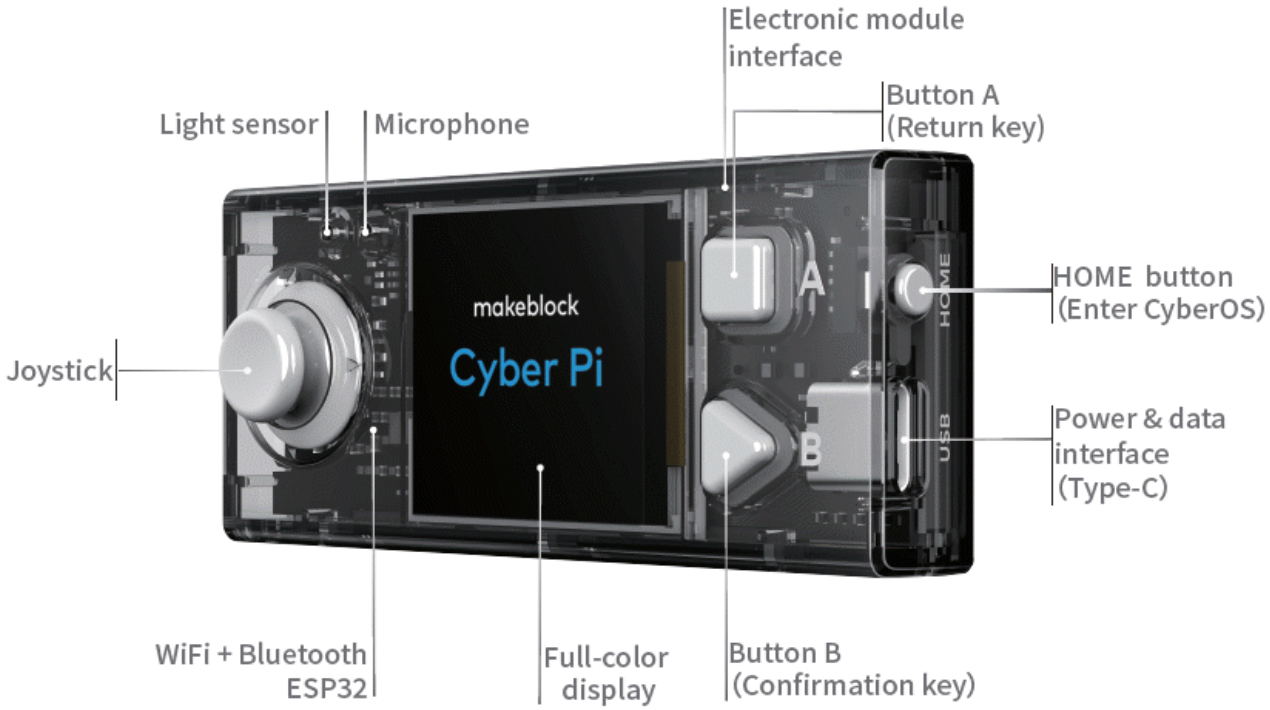
For the teacher:

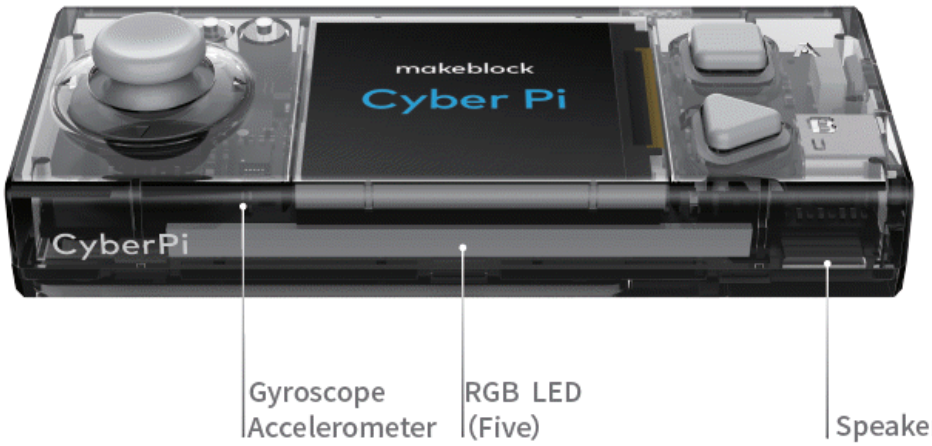
- A laptop or desktop with mBlock Python editor installed
- A CyberPi device
- A Type-C cable
- Lesson plan
- Worksheet

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.

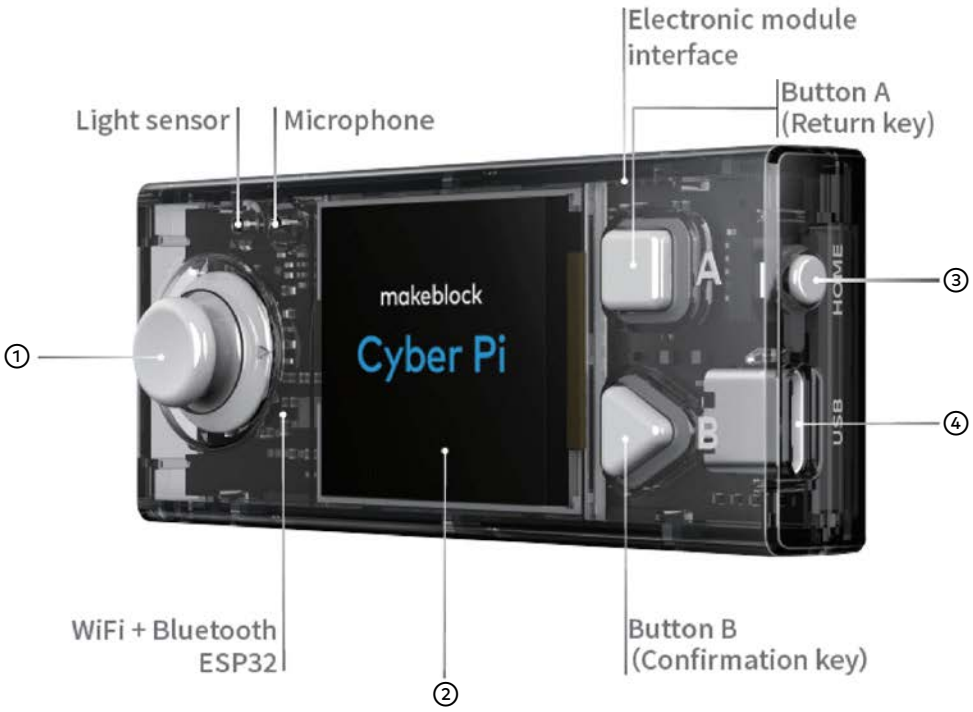
What I Know	What I Wonder	What I Learnt
• Import **random** • Loops • Conditionals • The **print()** function • Python data types		

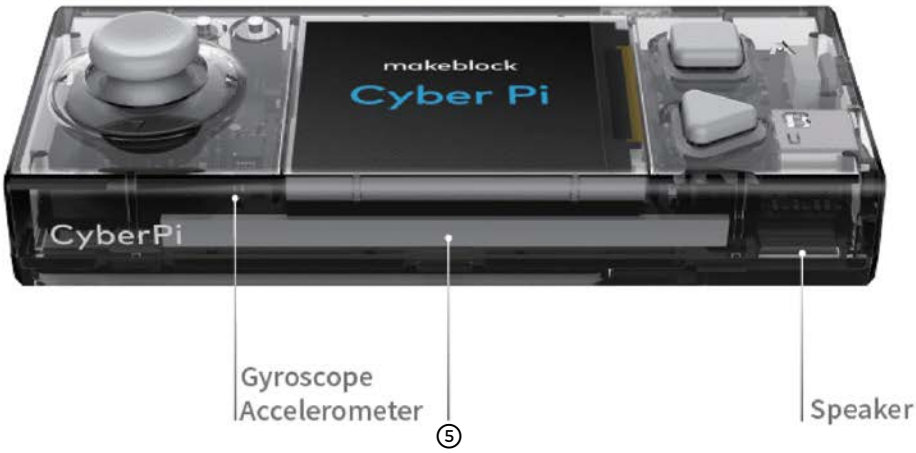
Procedures

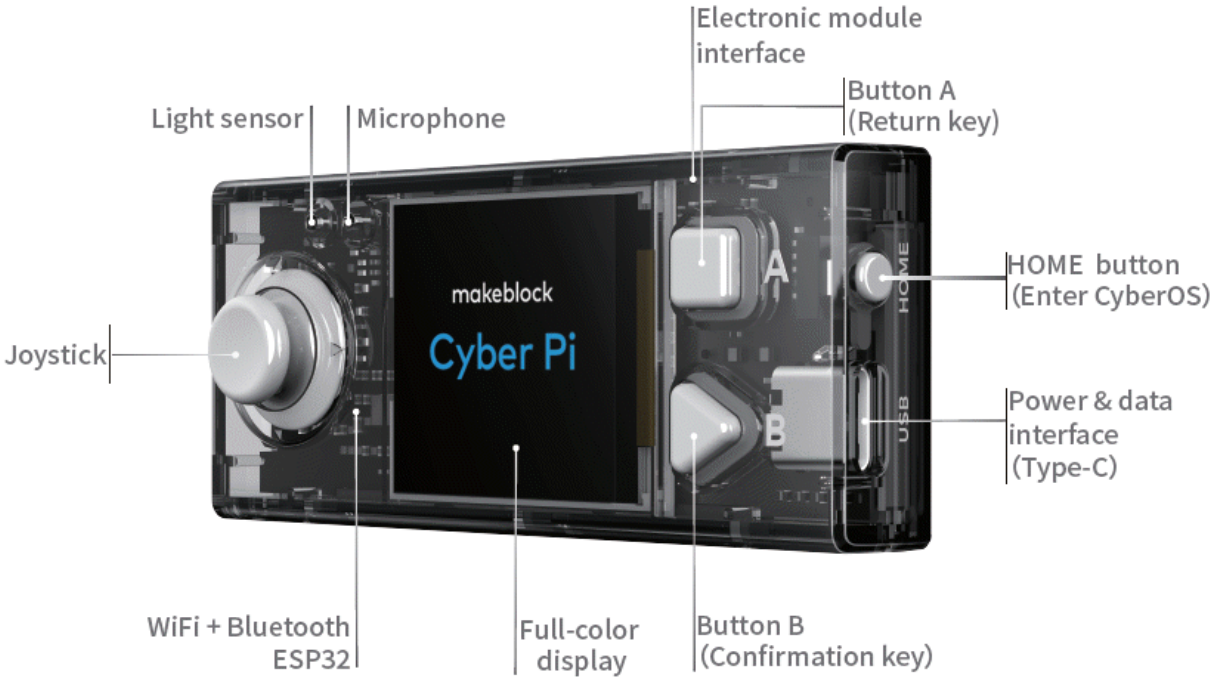
Section 1 Introduction: CyberPi and API (30 minutes)

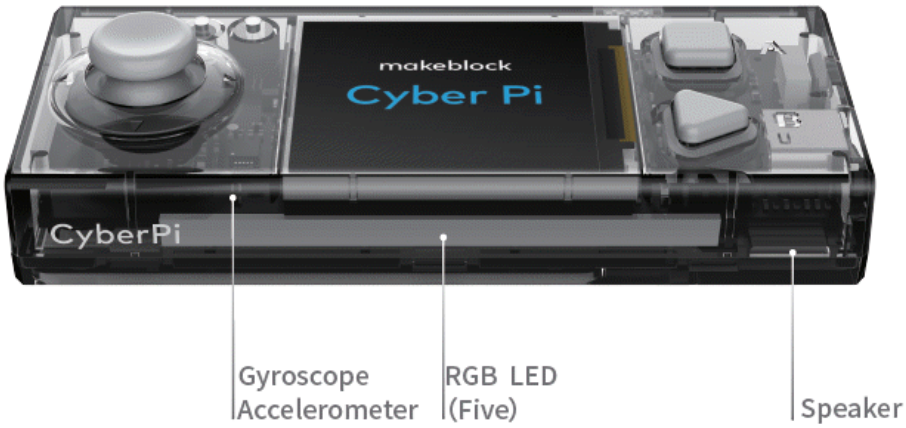
Step 1.1 Display and introduce the CyberPi device.

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



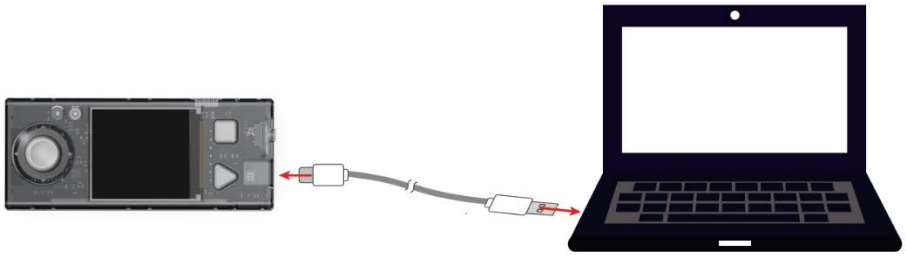






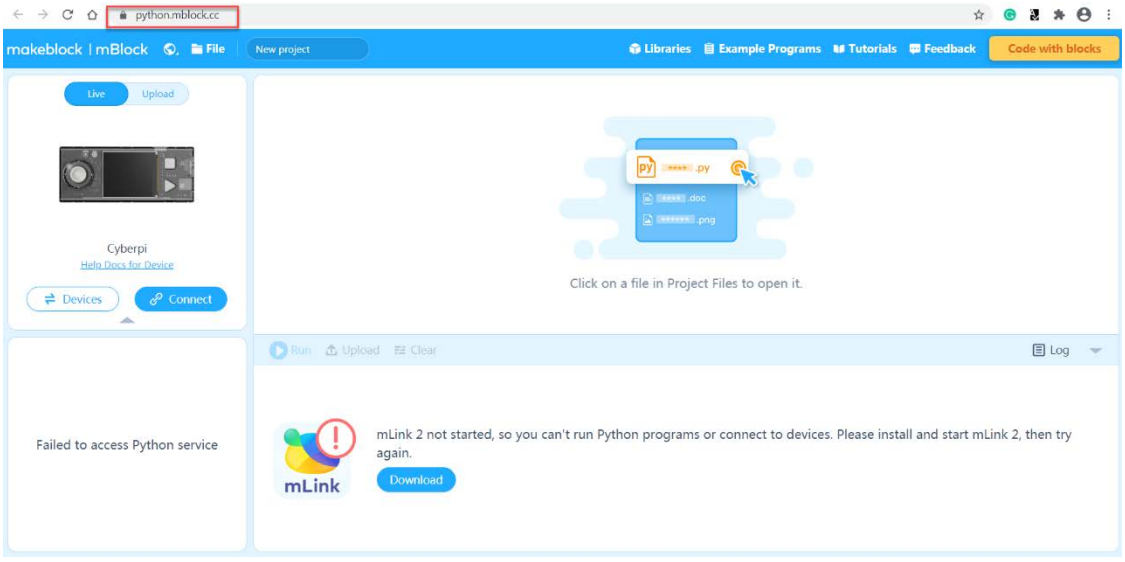
Step 1.2 Instruct students to connect CyberPi to the laptop or desktop.

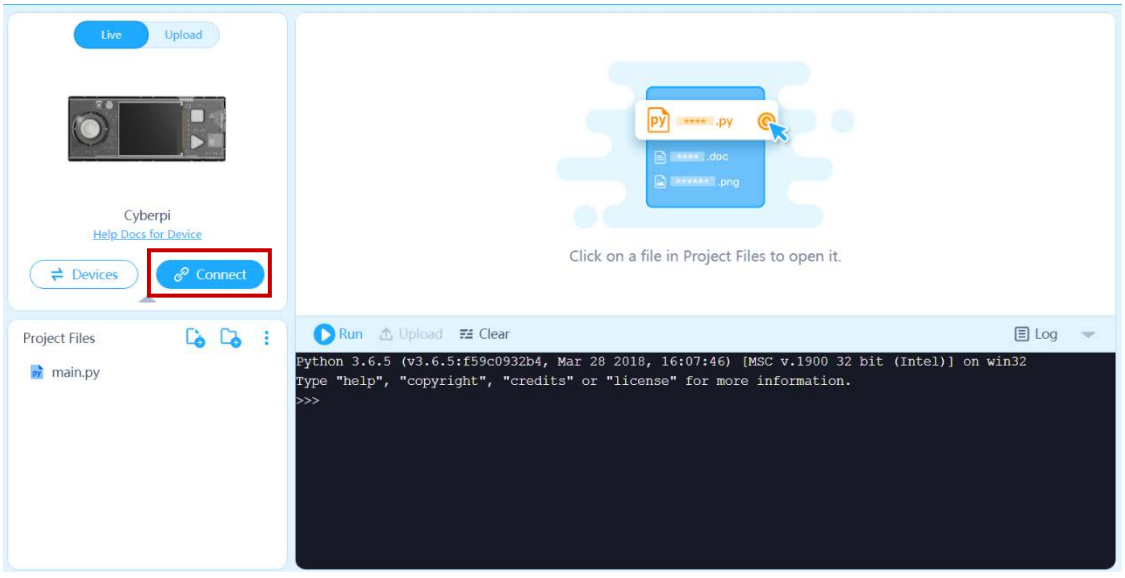
- Demonstrate how to use the USB Type-C cable to connect CyberPi to the laptop or desktop.

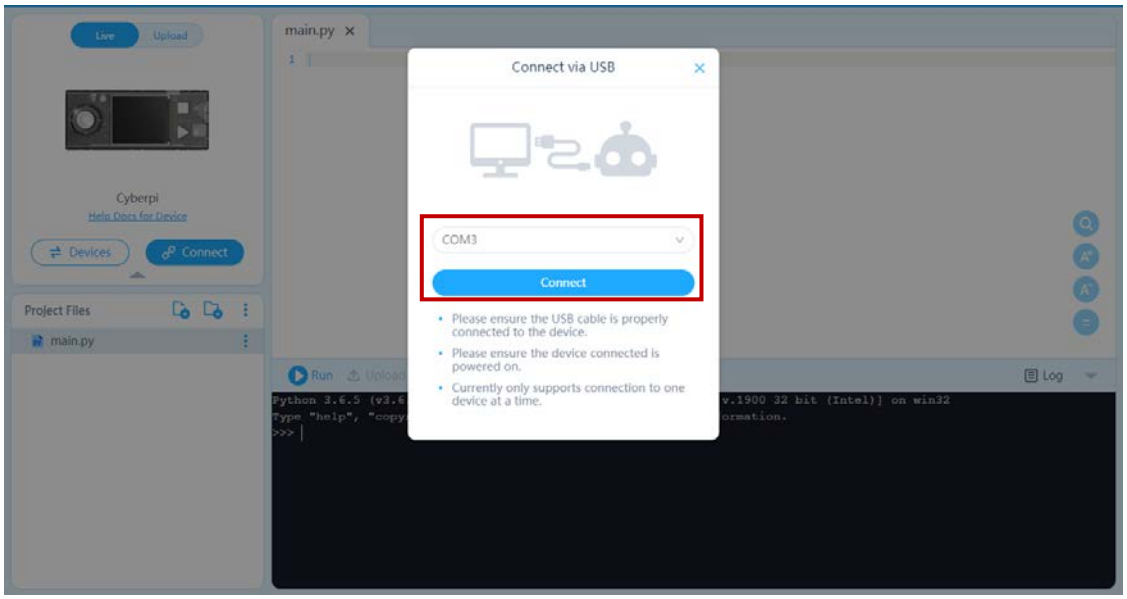


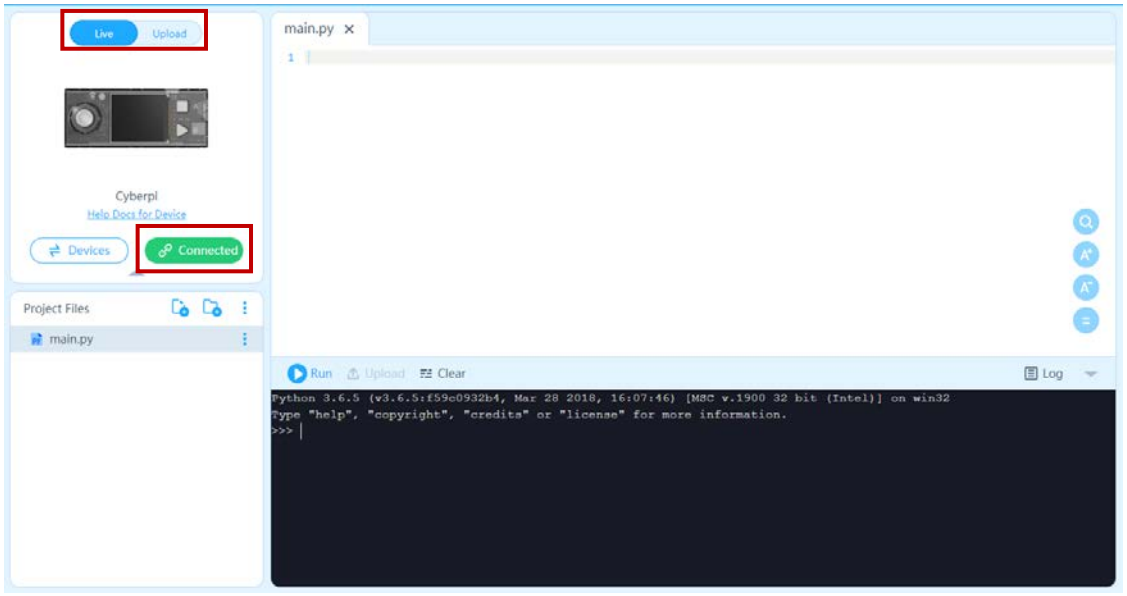
- Instruct students to open **mLink2** and access the mBlock Python editor.

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









- Remind students that they should choose **Live** mode, which allows them to code in real time.
- Summarise 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 know and 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' in the first place:

```
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 can do 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.controllor.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 LED strip?
  - How to set the light colour? How to represent light colours 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 get some ideas about what they will learn in this lesson.

What I Know	What I Wonder	What I Learnt
• Import modules • Loops • Conditionals • The **print()** function • Python data types	• How does CyberPi work? • How to program CyberPi by using Python?	

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'**. 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 then 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. If we want to display 'Hello, CyberPi!' on CyberPi's screen, for instance, we need to know the API of the screen.
- Point out the API code samples of CyberPi's screen:

  `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 as follows:

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 learnt 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 LED strip.

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

  `cyberpi.led.on()`

  `cyberpi.led.off()`

- Introduce other API code samples of the LED strip 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 learnt API code samples in the example program.

Step 3.4 Explain how to program the two buttons.

- **Say:** By pressing the Button A or Button B, we make the true or false choice.
- Introduce the API code samples of the button 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 Summarise the API code samples of the screen, the LED strip, the buttons, and the joystick.

Step 4.2 Have students work individually to complete the tasks as follows:

- **Task 1:** Modify the quiz and the answer in the example program.
  - Suggest some true-or-false questions for consideration:

In Python, 200.0 is an integer. (False)

'True' and 'true' are 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)

...

  - However, 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 is headquartered 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)

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 sources.
  - The default lighting sources 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 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

```
import cyberpi

cyberpi.display.clear()
cyberpi.led.off()

cyberpi.led.play(name="rainbow")

cyberpi.console.println("Python is a compiled language.")
cyberpi.console.println("")
cyberpi.console.println("Your Answer:")

while True:
    if cyberpi.controller.is_press("up"):
        # If the Button A is pressed
        cyberpi.led.on("red", id="all")
        cyberpi.console.println("Incorrect.")
        cyberpi.console.println("Correct Answer: False")
        break
    if cyberpi.controller.is_press("down"):
        cyberpi.led.on("green", id="all")
        cyberpi.console.println("Correct!")
        break
```

  - **Differentiation:** For advanced learners, encourage them to consider what if it is a multiple-choice question that provides 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)

```
import cyberpi

x = 0
# Initialise the counter
cyberpi.led.off()
cyberpi.display.clear()
cyberpi.led.on(255, 255, 255)
# Initialise the device

while x < 14:
    # Terminate the while loop after completing 14 quizzes
    if cyberpi.controller.is_press("up"):
        # Choose Option A
        cyberpi.console.print("A ")
        cyberpi.led.on("cyan", id="all")
        x += 1
        # Update counter
    if cyberpi.controller.is_press("left"):
        # Choose Option B
        cyberpi.console.print("B ")
        cyberpi.led.on("yellow", id="all")
        x += 1
        # Update counter
    if cyberpi.controller.is_press("down"):
        # Choose Option C
        cyberpi.console.print("C ")
        cyberpi.led.on("purple", id="all")
        x += 1
        # Update counter
    if cyberpi.controller.is_press("right"):
        # Choose Option D
        cyberpi.console.print("D ")
        cyberpi.led.on("orange", id="all")
        x += 1
        # Update counter
    if cyberpi.controller.is_press("middle"):
        # None
        cyberpi.console.print("  ")
        # Leave it empty
        cyberpi.led.on("black", id="all")
        x += 1
        # Update counter
```

Section 5 Recap (5 minutes)

Step 5.1 Summarise the key points learnt in this lesson.

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

What I Know	What I Wonder	What I Learnt
• Import modules • Loops • Conditionals • The **print()** function • Python data types	• How does CyberPi work? • How to program CyberPi by using Python?	• Concept of API • Import **cyberpi** • Display screen of CyberPi • LED strip of CyberPi • Buttons of CyberPi • Joystick of CyberPi

Lesson 3 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 data security issue and create a digital artefact to demonstrate how a password-protected security device can protect personal data. Students need to describe the 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 the 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 tradeoffs 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

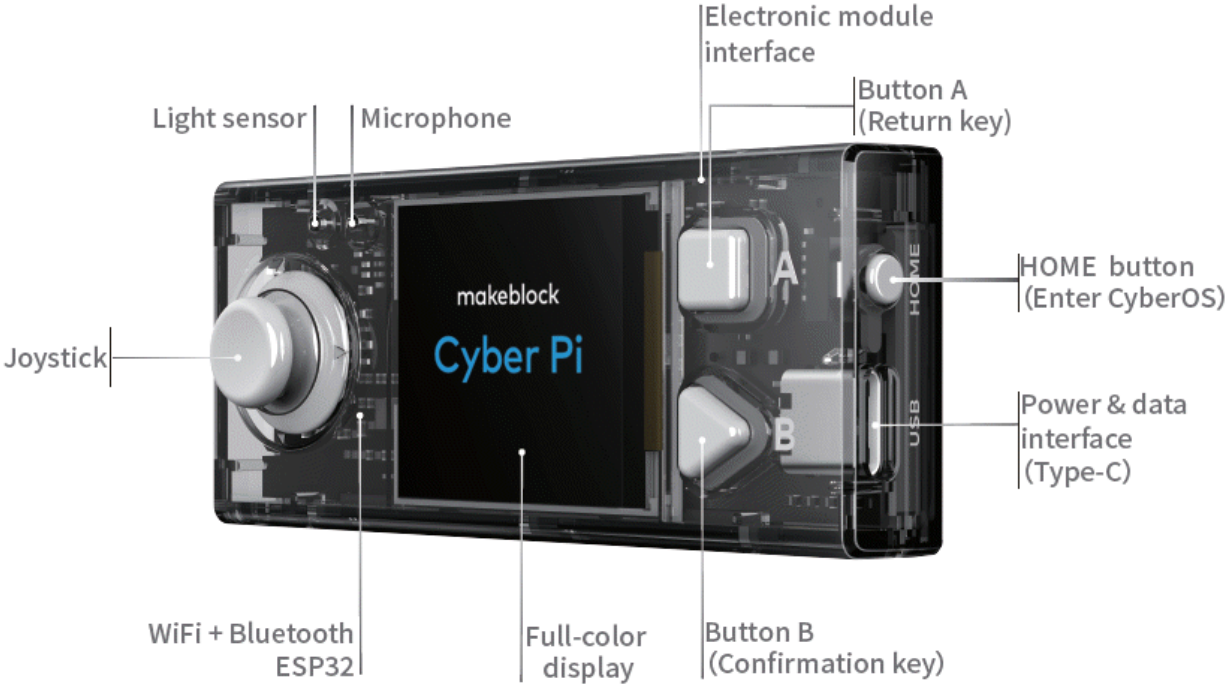
  (Available from https://python.mblock.cc/)
- A CyberPi device
- A Type-C cable
- Lesson plan
- Worksheet

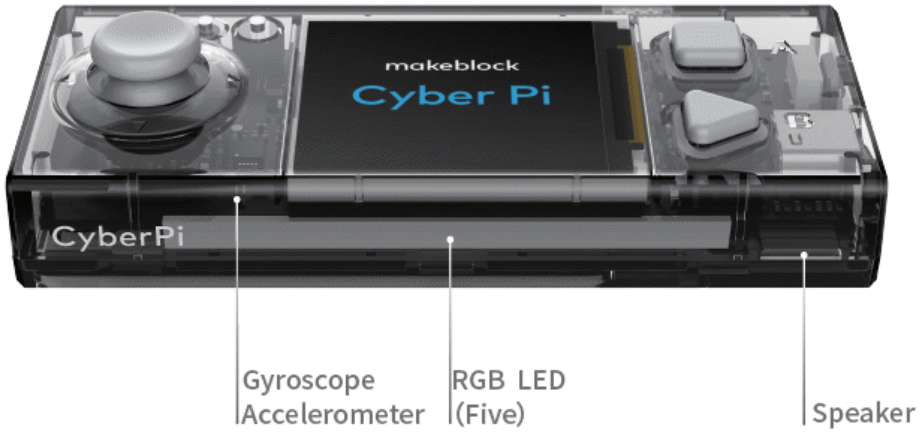
For students:

- 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

```
import cyberpi

cyberpi.display.clear()
t = 0
while True:
    print("Create a password")
    pin_1 = input("Type password: ")
    pin_2 = input("Type password again: ")
    if pin_2 == pin_1:
        print("Success!")
        break
    else:
        print("Passwords don't match. Try again.")

print("Sign in your account")
cyberpi.console.println("Sign in")
while t < 3:
# Have 3 attempts
# If t < 3 is true, iterate the loop body
    pin = input("Password: ")
    cyberpi.console.print("Password: ")
    cyberpi.console.println(pin)

    # Verify the input password:
    if pin == pin_1:
    # Passwords match
        cyberpi.console.println("Success!")
        break
    else:
    # Passwords don't match
        cyberpi.console.println("Incorrect. Try again.")
        t += 1
        # Reduce the number of attempts
        if t == 3:
            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 Learnt
• Concept of the API • Import **cyberpi** • Display screen of CyberPi • LED strip of CyberPi • Buttons of CyberPi • Joystick of CyberPi		

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

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

```
import cyberpi

cyberpi.display.clear()
t = 0

while True:
    print("Create a password")
    pin_1 = input("Type password: ")
    pin_2 = input("Type password again: ")
    if pin_2 == pin_1:
        print("Success!")
        break
    else:
        print("Passwords don't match. Try again.")


print("Sign in your account")
cyberpi.console.println("Sign in")

while t < 3:
    pin = input("Password: ")
    cyberpi.console.print("Password: ")
    cyberpi.console.println(pin)
    if pin == pin_1:
        cyberpi.console.println("Success!")
        break
    else:
        cyberpi.console.println("Incorrect. Try again.")
        t += 1
        if t == 3:
            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 this function: A user is given 3 attempts to enter the account password. If the user fails 3 times, the user cannot enter the password any more.

Section 3 Run (3 minutes)

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

- Ask students to think about the above 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 Learnt
• Import cyberpi • Display screen of CyberPi • LED strip of CyberPi • Buttons of CyberPi • Joystick of CyberPi	• How does CyberPi interact with the computer (e.g. send and receive data)? • How to display text CyberPi's screen?	

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.

```
while True:
    print("Create a password")
    pin_1 = input("Type password: ")
    pin_2 = input("Type password again: ")
    if pin_2 == pin_1:
        print("Success!")
        break
    else:
        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 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 specified times the loop is executed isn't specified on request. In short, the **'while True'** means loop forever.

In the **'while <controlling expression> (is True)'** (e.g. **'while t < 3:'**), however, as it 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 in to the account, and therefore, the controlling expression is **'t < 3'**.

```
while t < 3:
# Have 3 attempts
    pin = input("Password: ")
    # Type the password
    cyberpi.console.print("Password: ")
    cyberpi.console.println(pin)
    # Display the password on the screen
    # Verify the input password:
    if pin == pin_1:
    # Passwords match
        cyberpi.console.println("Success!")
        break
    else:
    # Passwords don't match
        cyberpi.console.println("Incorrect. Try again.")
        t += 1
        # 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 Summarise 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 tasks as follow:

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

Example – Task 1

```
import cyberpi

cyberpi.display.clear()
t = 0
while True:
    user_id = input("Create a username: ")
    # Create a username
    print("Create a password")
    pin_1 = input("Type password: ")
    pin_2 = input("Type password again: ")
    if pin_2 == pin_1:
        print("Success!")
        break
    else:
        print("Passwords don't match. Try again.")

print("Sign in your account")
cyberpi.console.println("Sign in")
while t < 3:
    user = input("Username: ")
    # Type the username
    cyberpi.console.print("Username: ")
    cyberpi.console.println(user)
    pin = input("Password: ")
    cyberpi.console.print("Password: ")
    cyberpi.console.println(pin)
    if user == user_id and pin == pin_1:
        # Verify the username
        cyberpi.console.println("Success!")
        break
    else:
        cyberpi.console.println("Incorrect. Try again.")
        t += 1
        if t == 3:
            cyberpi.console.println("Too many failed attempts.")
```

- **Task 2:** Add some lighting effects as the indicator. Use what you have learnt 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 LED strip as an indicator to remind 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 three conditions as follows:

  The input username is incorrect;

  The input password is incorrect;

  Both the username and password are incorrect.

  Example – Task 4

```
import cyberpi

cyberpi.display.clear()
cyberpi.led.off()
t = 0
while True:
    user_id = input("Create a username: ")
    print("Create a password")
    pin_1 = input("Type password: ")
    pin_2 = input("Type password again: ")
    if pin_2 == pin_1:
        print("Success!")
        break
    else:
        print("Passwords don't match. Try again.")
print("Sign in your account")
cyberpi.console.println("Sign in")
cyberpi.led.on("white", id="all")

while t < 5:
    user = input("Username: ")
    cyberpi.console.print("Username: ")
    cyberpi.console.println(user)
    pin = input("Password: ")
    cyberpi.console.print("Password: ")
    cyberpi.console.println(pin)
    if user == user_id and pin == pin_1:
        cyberpi.console.println("Success!")
        cyberpi.led.play(name="rainbow")
        break
    elif user != user_id and pin == pin_1:
        cyberpi.console.println("Incorrect username. Try again.")
        t += 1
        cyberpi.led.off(id=t)
    elif user == user_id and pin != pin_1:
        cyberpi.console.println("Incorrect password. Try again.")
        t += 1
        cyberpi.led.off(id=t)
    elif user != user_id and pin != pin_1:
        cyberpi.console.println("Both incorrect. Try again.")
        t += 1
        cyberpi.led.off(id=t)
    if t == 5:
        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 bank accounts, they will be asked to use the card reader as the security measure. For example, they need to enter the PIN with the keypad on the card reader and receive a random verification code or passcode number to confirm the transaction operation.
- Explain how the card reader project works as well as the requirement 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 or the verification code is 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

```
import cyberpi, random

cyberpi.display.clear()

while True:
    if cyberpi.controller.is_press("a"):
        code = random.randint(1000, 9999)
        cyberpi.console.print("Verification code: ")
        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.

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

  - The above script is to confirm the transaction information displayed on the screen. The sender should check the receiver and transit money and then confirm the transaction operation 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 Summarise the key points learnt in this lesson.

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

What I Know	What I Wonder	What I Learnt
• Import cyberpi • Display screen of CyberPi • LED strip of CyberPi • Buttons of CyberPi • Joystick of CyberPi	• How does CyberPi interact with the computer (e.g. send and receive data)? • How to display text CyberPi's screen?	• How to use variables to transmit data between the computer and CyberPi • How to display text on the screen • Importance of password security

Lesson 4 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

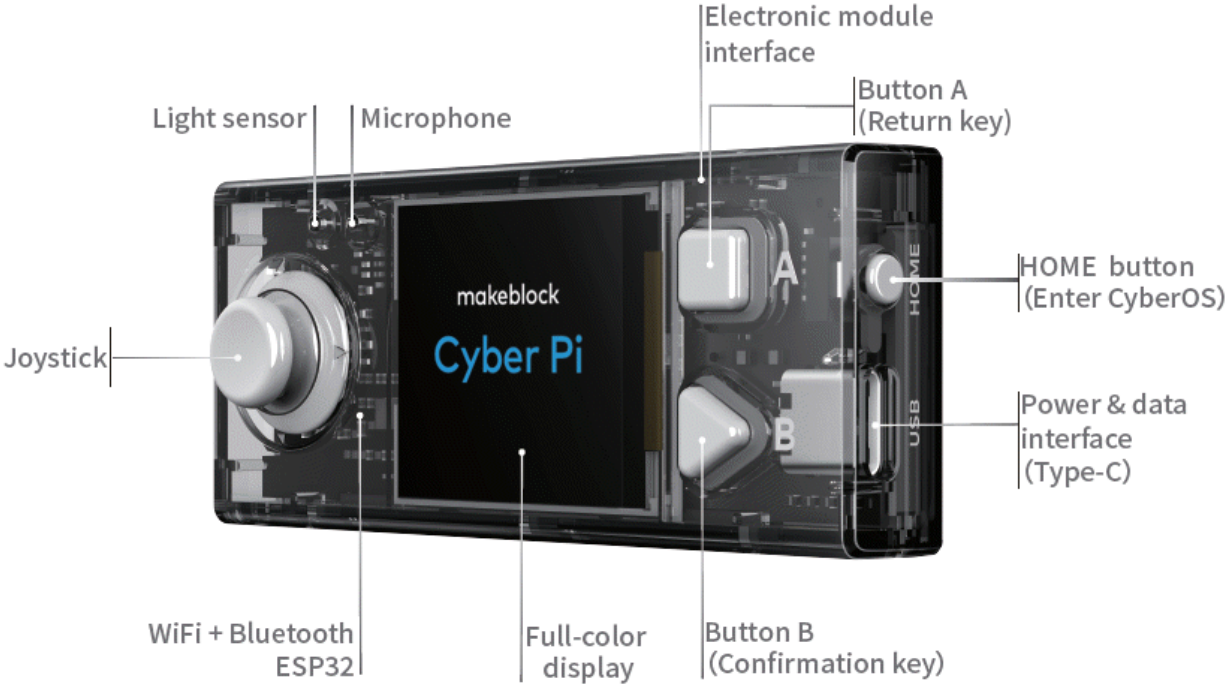
For the teacher:

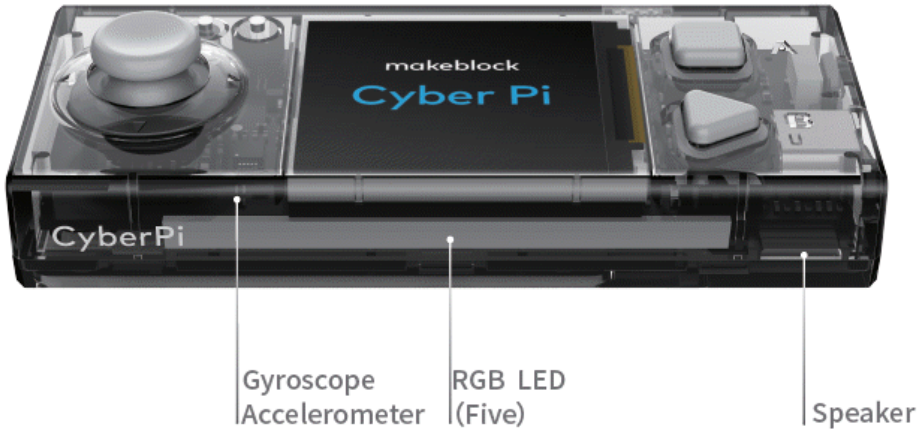
- A laptop or desktop with mBlock Python editor installed
- A CyberPi device
- A Type-C cable
- Lesson plan
- Worksheet

For students:

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

Features of CyberPi





Example Program

```
import cyberpi, random

cyberpi.display.clear()
n = int(input("The number of times to roll 2 dice: "))
t = 0
sum_list = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
# Outcomes of the sum of two dice numbers
count_list = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
# Count the frequency of the sum in the sample space
while t < n:
    dice_x = random.randint(1, 6)
    dice_y = random.randint(1, 6)
    print("(", dice_x, ",", dice_y, ")")
    result = dice_x + dice_y
    t += 1
    sum_index = sum_list.index(result)
    # Identify the index of the sum in the list 'sum_list'
    count_list[sum_index] += 1
    # Add '1' to the value of the corresponding item

cyberpi.display.set_brush(128, 0, 0)
# Define the colour based on RGB colour model
cyberpi.barchart.add(round(count_list[0]/n * 200, 2))
# Create a bar
cyberpi.display.set_brush(220, 20, 60)
cyberpi.barchart.add(round(count_list[1]/n * 200, 2))
cyberpi.display.set_brush(255, 0, 0)
cyberpi.barchart.add(round(count_list[2]/n * 200, 2))
cyberpi.display.set_brush(205, 92, 92)
cyberpi.barchart.add(round(count_list[3]/n * 200, 2))
cyberpi.display.set_brush(233, 150, 122)
cyberpi.barchart.add(round(count_list[4]/n * 200, 2))
cyberpi.display.set_brush(255, 69, 0)
cyberpi.barchart.add(round(count_list[5]/n * 200, 2))
cyberpi.display.set_brush(255, 165, 0)
cyberpi.barchart.add(round(count_list[6]/n * 200, 2))
cyberpi.display.set_brush(255, 215, 0)
cyberpi.barchart.add(round(count_list[7]/n * 200, 2))
cyberpi.display.set_brush(240, 230, 140)
cyberpi.barchart.add(round(count_list[8]/n * 200, 2))
cyberpi.display.set_brush(255, 255, 0)
cyberpi.barchart.add(round(count_list[9]/n * 200, 2))
cyberpi.display.set_brush(154, 205, 50)
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.

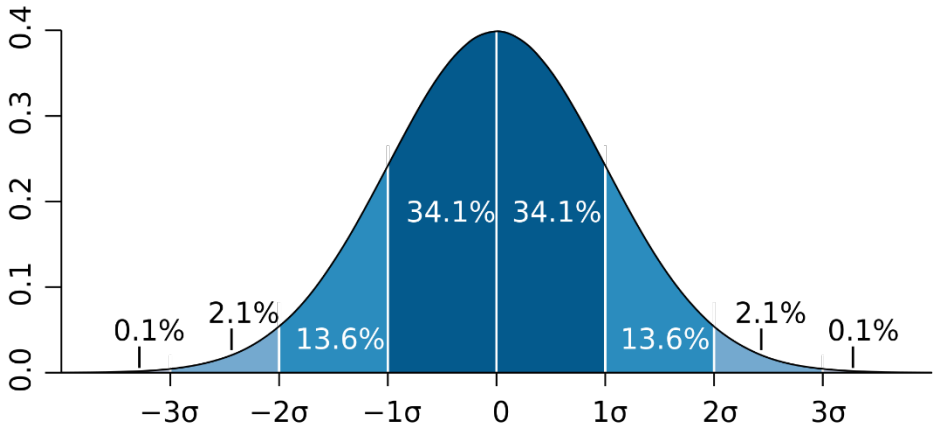
What I Know	What I Wonder	What I Learnt
• Import **random** • Import **cyberpi** • Display screen of CyberPi • **'cyberpi.console'** • **'cyberpi.display'**		

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 summarise 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 enough efficient 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 Learnt
• Import **random** • Import **cyberpi** • Display screen of CyberPi • **'cyberpi.console'** • **'cyberpi.display'**	• What if I need to use more than one module, what should I do in Python? • How to create data charts on CyberPi's screen?	

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)

```
import cyberpi, random

cyberpi.display.clear()
n = int(input("The number of times to throw up 2 coins: "))
event_list = [0, 0, 0]

for i in range(0, n):
    coin_x = random.randint(0, 1)
    coin_y = random.randint(0, 1)
    # '0' represents 'head', '1' represents 'tail'
    print("(", coin_x, ",", coin_y, ")")
    if coin_x == 0 and coin_y == 0:
        # 2 heads
        event_list[0] += 1
    elif coin_x == 1 and coin_y == 1:
        # 2 tails
        event_list[2] += 1
    else:
        # 1 head, 1 tail
        event_list[1] += 1

cyberpi.display.set_brush(255, 0, 0)
cyberpi.barchart.add(round(event_list[0]/n * 200, 2))
cyberpi.display.set_brush(0, 255, 0)
cyberpi.barchart.add(round(event_list[1]/n * 200, 2))
cyberpi.display.set_brush(0, 0, 255)
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 '' and use the 'random.choice' function to randomly select one of the outcomes.

- Ask students to think about this question: Toss a coin 3 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 Heads) = P(HHH) = 1/8

  P(2 Heads) = P(HHT) + P(HTH) + P(THH) = 3/8

  P(1 Head) = P(HTT) + P(THT) + P(TTH) = 3/8

  P(0 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

```
import cyberpi

cyberpi.display.clear()

# 3 Heads:
cyberpi.display.set_brush(255, 65, 0)
cyberpi.barchart.add(1/8 * 200)

# 2 Heads:
cyberpi.display.set_brush(255, 100, 0)
cyberpi.barchart.add(3/8 * 200)

# 1 Head:
cyberpi.display.set_brush(255, 165, 0)
cyberpi.barchart.add(3/8 * 200)

# 0 Head:
cyberpi.display.set_brush(255, 215, 0)
cyberpi.barchart.add(1/8 * 200)
```

Step 5.2 Summarise 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').

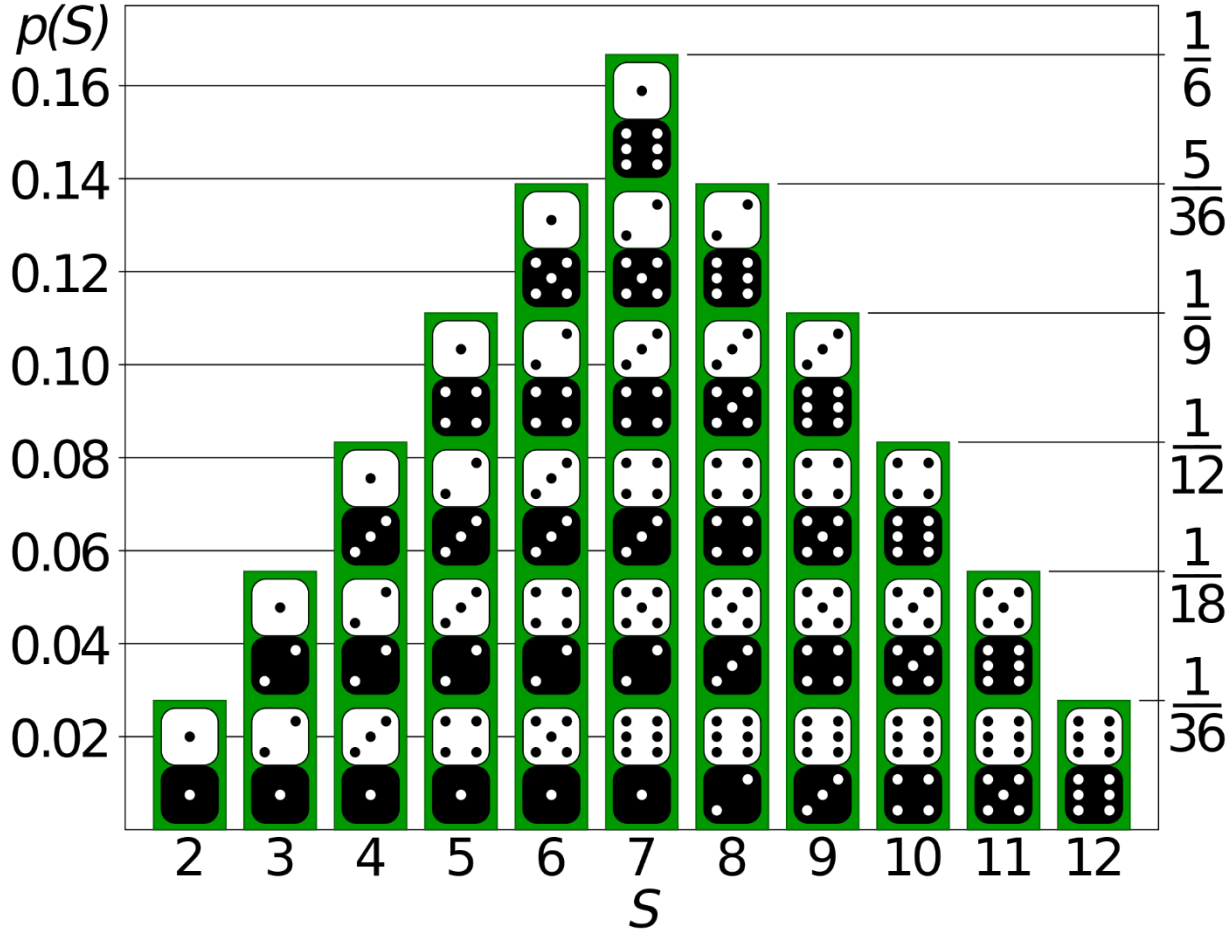
- Add a numeric value to the bar by using this syntax:

  `cyberpi.barchart.add()`

Section 6 Recap (5 minutes)

Step 6.1 Summarise the key points learnt 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 Learnt'** column of the **K-W-L chart**.

What I Know	What I Wonder	What I Learnt
• Import **cyberpi**, **random** • Display screen of CyberPi • **'cyberpi.console'** • **'cyberpi.display'**	• What if I need to use more than one module, what should I do in Python? • How to create data charts on CyberPi's screen?	• How to use lists to store a group of data • How to modify data stored in a list • How to graph a bar chart on CyberPi's screen

Lesson 5 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 and 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 tradeoffs associated with computing technologies that affect people's everyday activities and career options.

Preparation

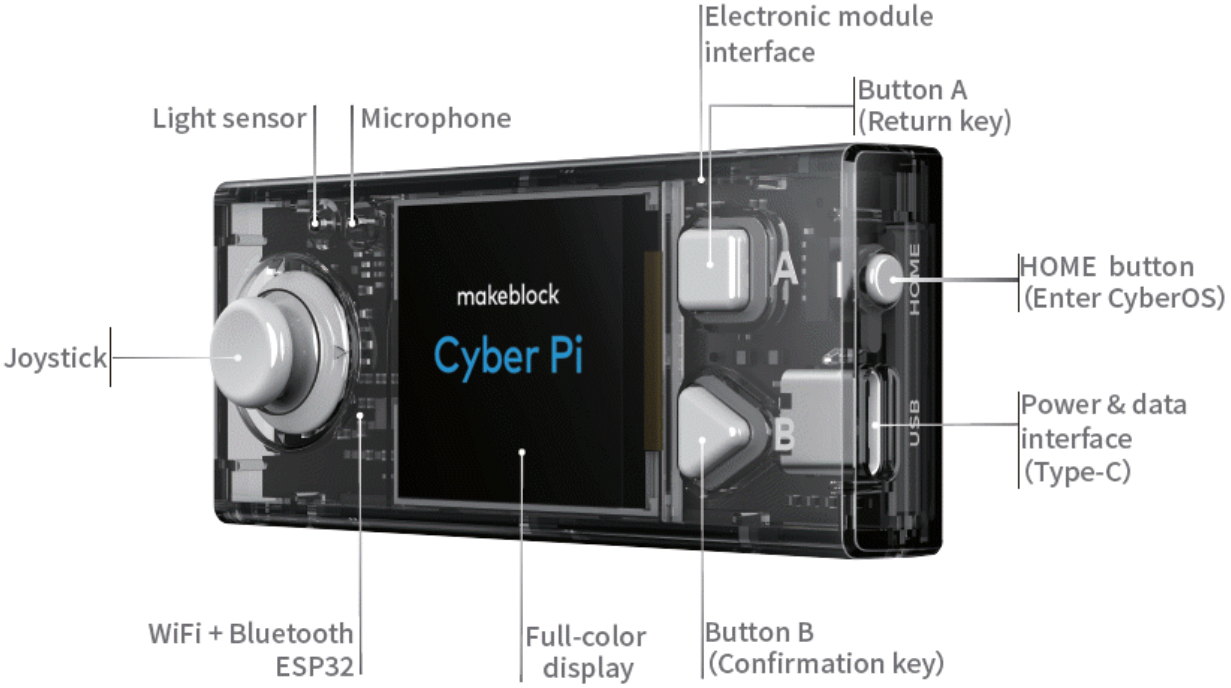
For the teacher:

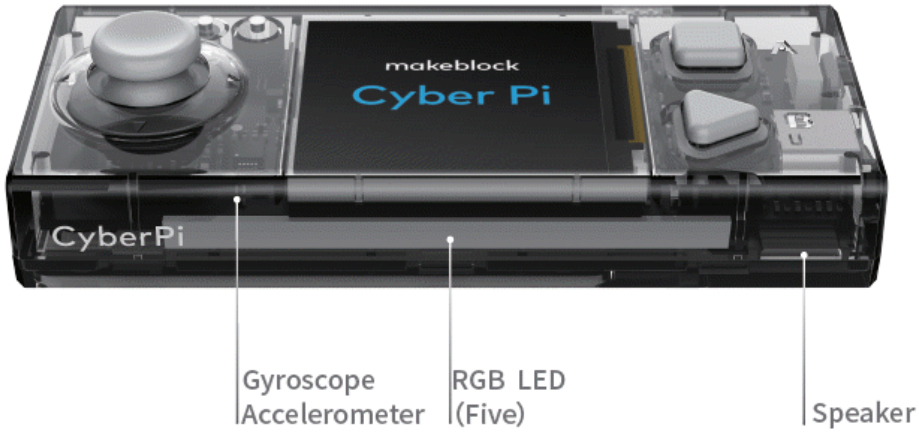
- A laptop or desktop with mBlock Python editor installed
- A CyberPi device
- A Type-C cable
- Lesson plan
- Worksheet

For students:

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

Features of CyberPi



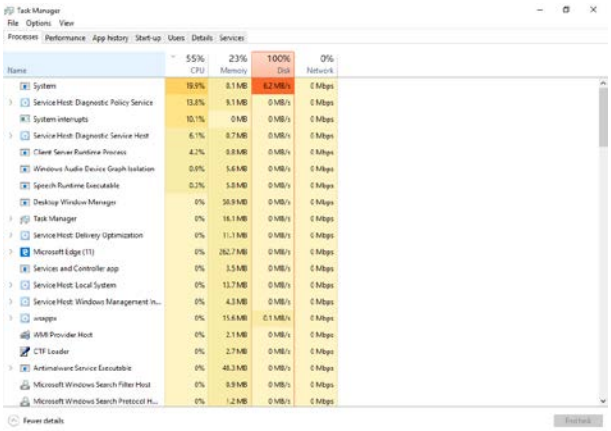


Example Program

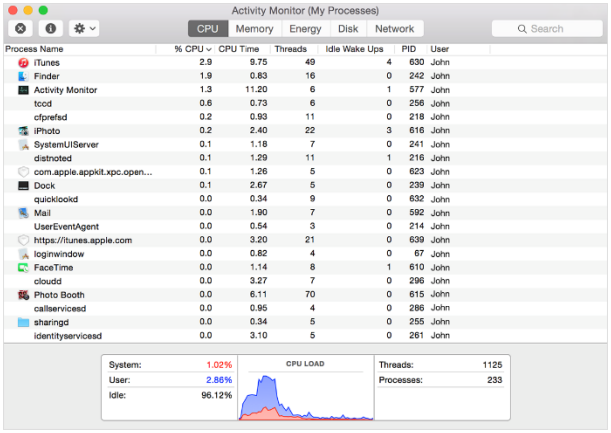
```
1.  import cyberpi, psutil
2.  # Import both 'cyberpi' and 'psutil' modules
3.
4.  cyberpi.chart.clear()
5.
6.  while True:
7.      CPU = psutil.cpu_percent()
8.      # Monitor the CPU usage
9.      mem = psutil.virtual_memory()
10.     mem_p = mem.percent
11.     # Monitor the memory usage
12.
13.     # 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.

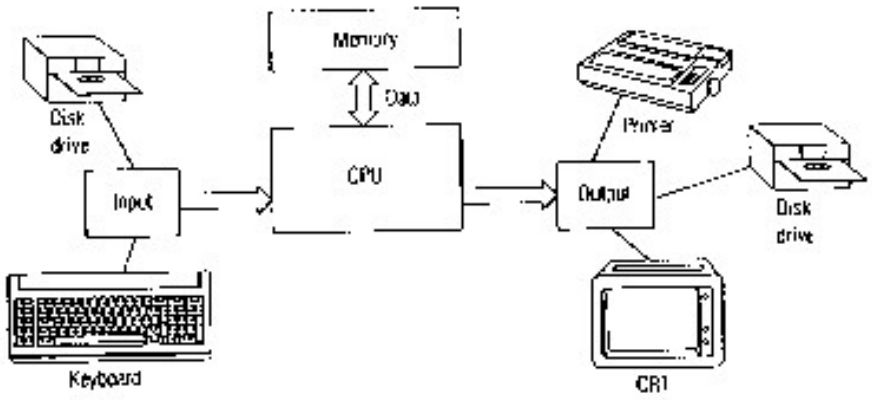
What I Know	What I Wonder	What I Learnt
• Import **cyberpi**, **random** • Display screen of CyberPi • **'cyberpi.display'** • **'cyberpi.barchart'**		

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: © Oregon State University)

However, if this is your students' first time to learn the computer components, use the above figure to explain each component. Highlight the components as follows:

- **CPU**

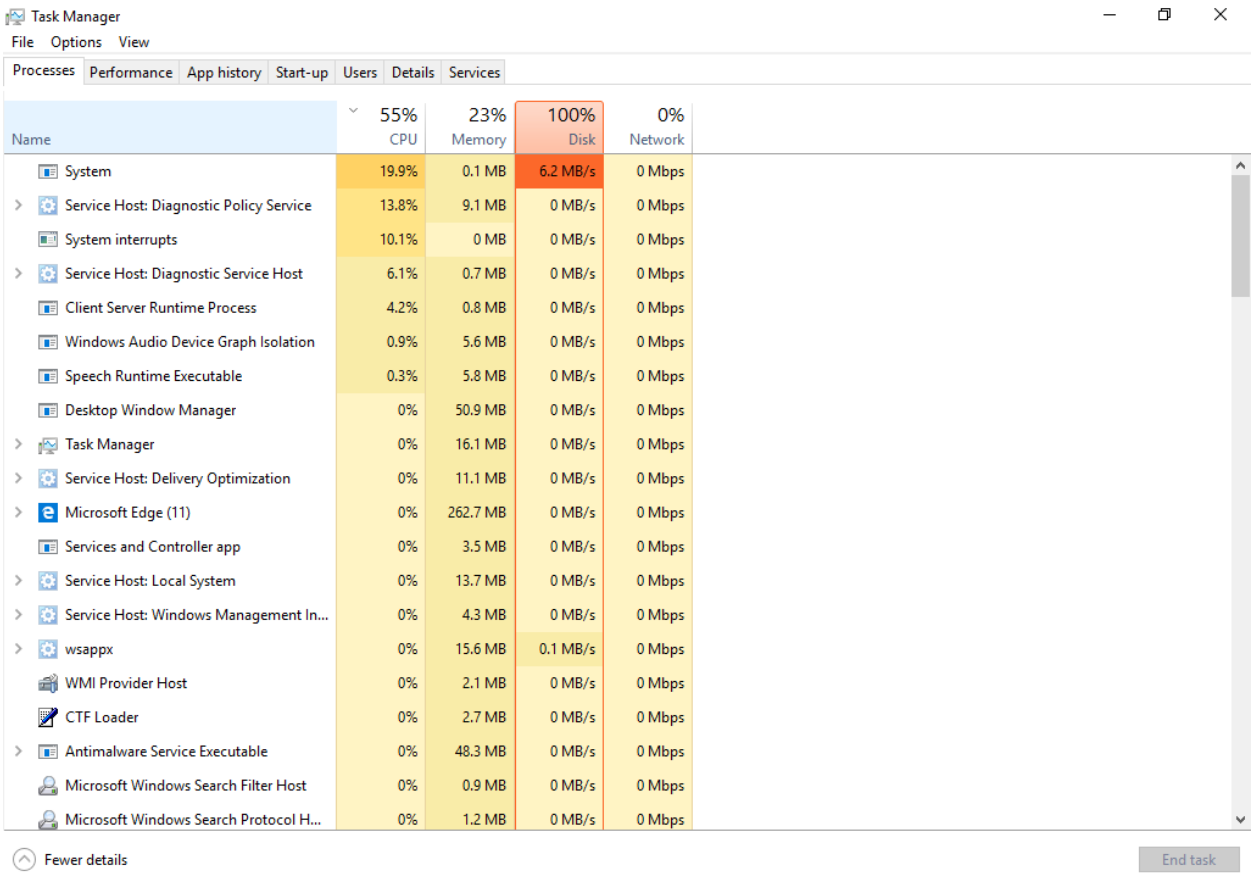
  CPU stands for 'Central Processing Unit'. It is the chip that contains all the circuitries for performing arithmetic and logic operations and directing data to and from memory.

- **Memory**

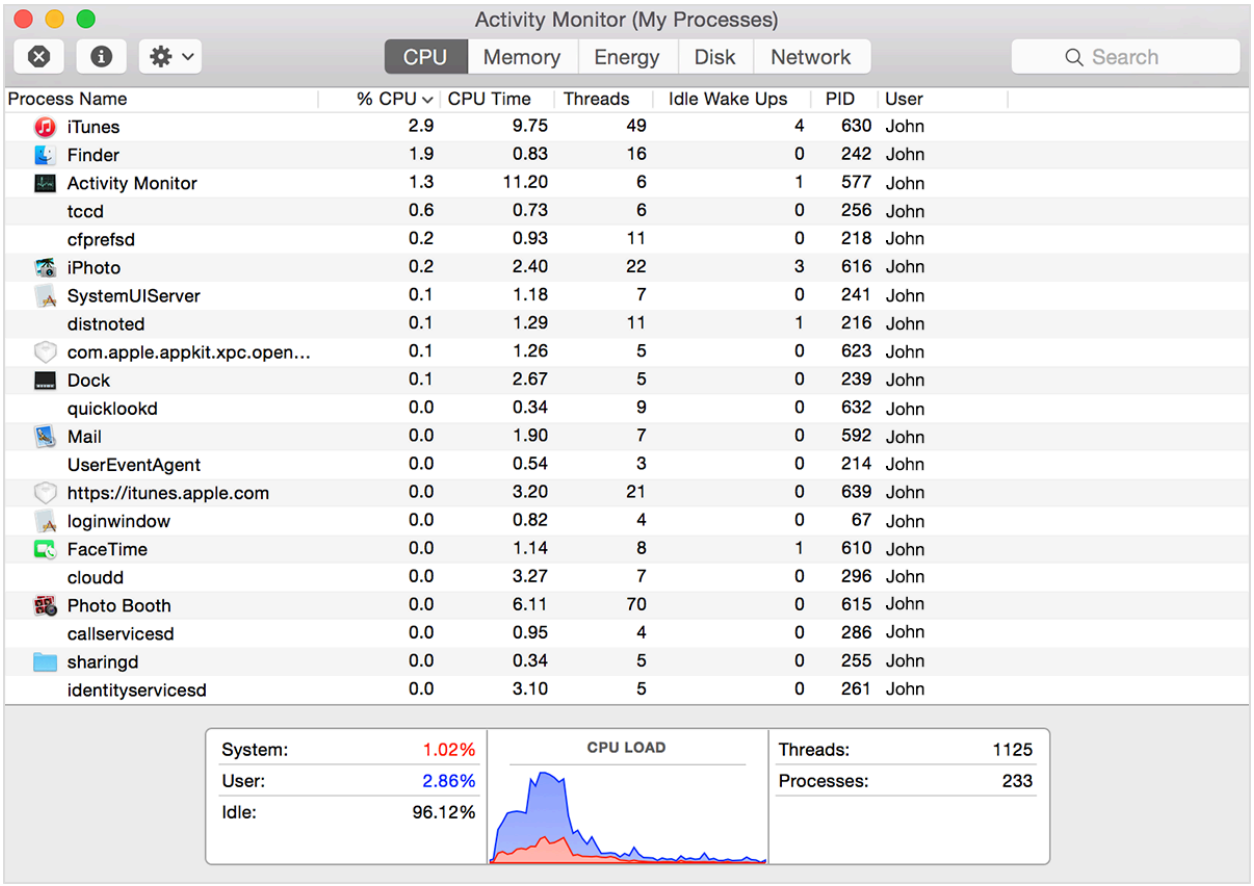
  Like a human brain, computer memory is the storage space in the computer system.

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 colour 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 Learnt
• Import cyberpi, random • Display screen of CyberPi • 'cyberpi.display' • 'cyberpi.barchart'	• How can I create other types of charts using Python? • What is going on inside my computer? How does the CPU usage affect it?	

Section 4 Investigate (20 minutes)

Step 4.1 Invite volunteer students to report and share their findings.

- Students should identify the functions as follows:
  - 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 see 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 see 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 given instantly to processes, the memory usage in percentage, etc.

To read 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 colour 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 see 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 it because a computer might excel at some tasks but not do so well at others.

- **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 amount 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

```
import psutil

speed = psutil.cpu_freq()
max_speed = speed.max
print("Maximum Clock Speed:", max_speed, "Mhz")
min_speed = speed.min
print("Minimum Clock Speed:", min_speed, "Mhz")
```

- **Explain the cache size:** 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 would explore them in the future lesson.
- Have students discuss the features of lower and higher CPU performances based on the above information. Ask them to summarise 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).

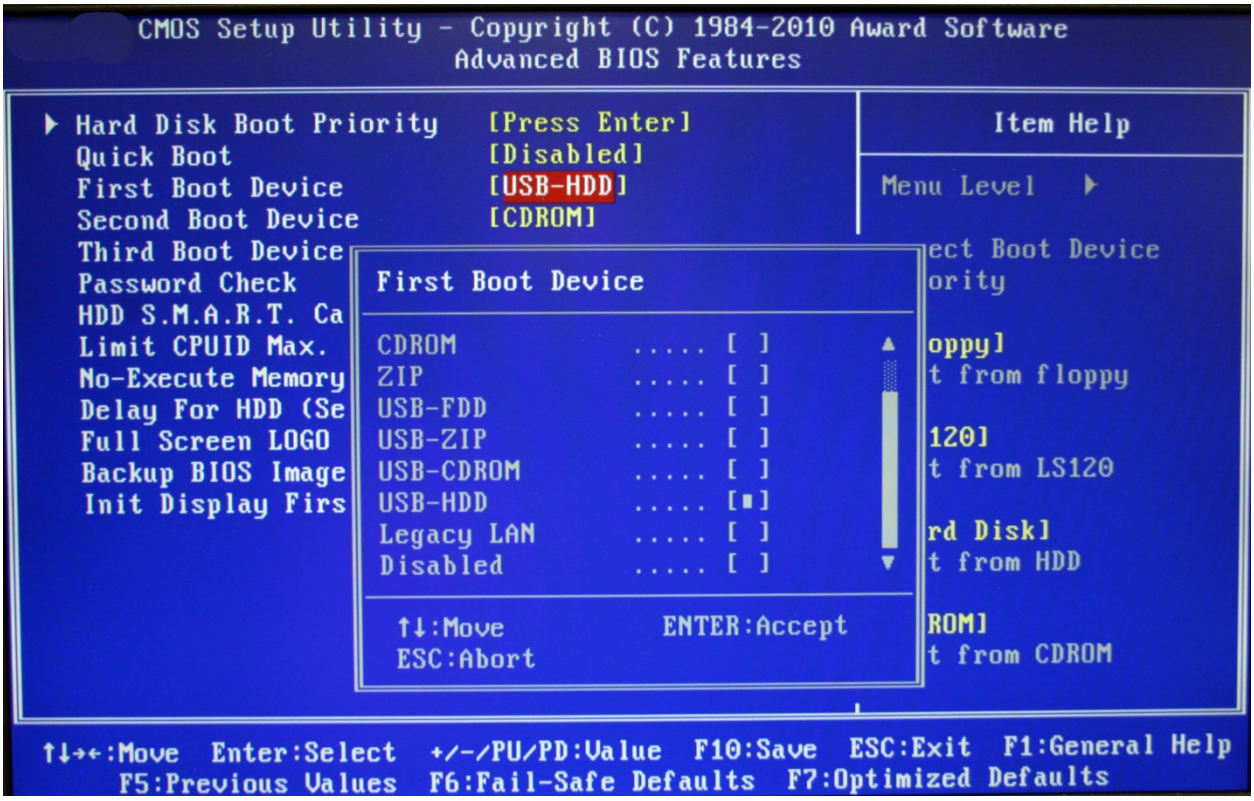
- **Explain the RAM:** Random Access Memory, or the RAM, stores user programs that control what the CPU does including the data used by these programs and the results of operations performed by these programs. RAM is accessible to the user. The memory size of RAM affects computer performance.

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 the 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 frequency of the buzzer in the 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 strings. 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 Summarise 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

- Summarise 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 Learnt'** column of the K-W-L chart.

What I Know	What I Wonder	What I Learnt
• Import **cyberpi**, **random** • Display screen of CyberPi • **'cyberpi.display'** • **'cyberpi.barchart'**	• How can I create other types of charts using Python? • What is going on inside my computer? How does the CPU usage affect it?	• Import **cyberpi**, **psuti**l • **'cyberpi.linechart'** • The roles of the CPU and computer memory • Factors that affect the CPU performance

Lesson 6 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, 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

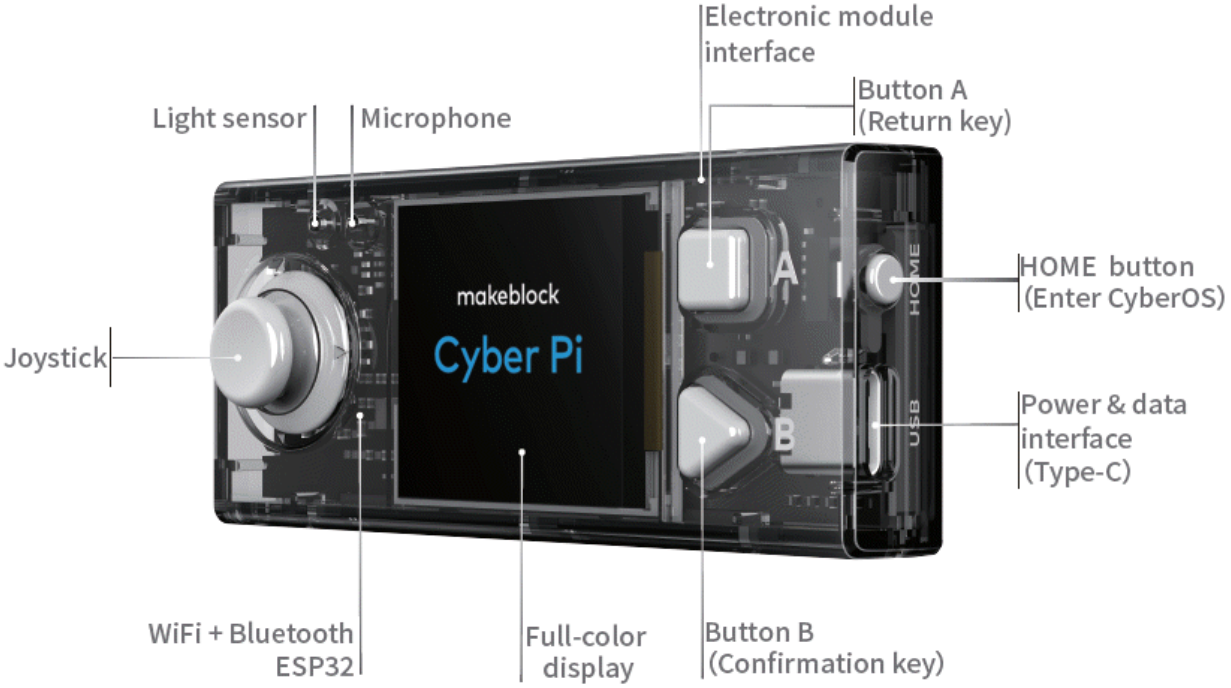
For the teacher:

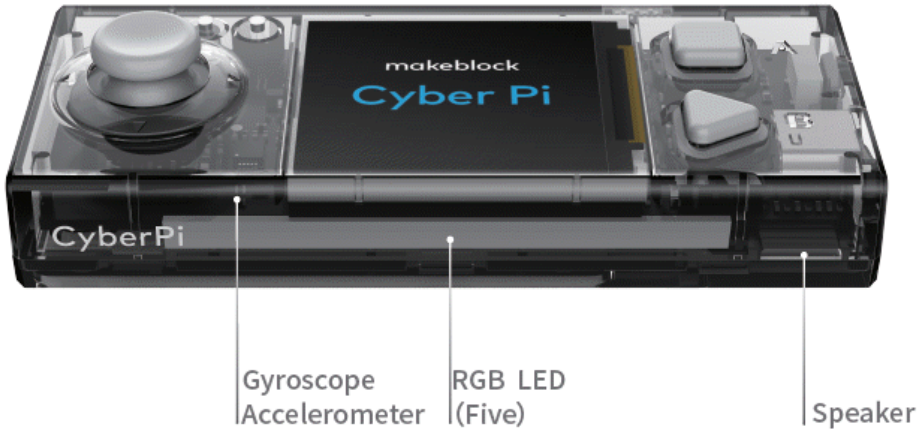
- A laptop or desktop with mBlock Python code editor installed
- A CyberPi device
- A Type-C cable
- Lesson plan
- Worksheet

For students:

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

Features of CyberPi





Example Program

```
import cyberpi
from pynput.keyboard import Key, Listener
# Monitor the keyboard input
cyberpi.audio.set_vol(50)
# Adjust the volume on CyberPi
def bar1():
    cyberpi.audio.play_music(60, 0.2)
    cyberpi.audio.play_music(64, 0.2)
    cyberpi.audio.play_music(67, 0.2)
def bar2():
    cyberpi.audio.play_music(64, 0.2)
    cyberpi.audio.play_music(65, 0.2)
    cyberpi.audio.play_music(69, 0.2)
def bar3():
    cyberpi.audio.play_music(64, 0.2)
    cyberpi.audio.play_music(67, 0.2)
    cyberpi.audio.play_music(71, 0.2)
def bar4():
    cyberpi.audio.play_music(65, 0.2)
    cyberpi.audio.play_music(69, 0.2)
    cyberpi.audio.play_music(72, 0.2)
def on_press(key):
    if key.char == "1":
        # A key produces a character value '1'
        cyberpi.led.on("red")
        bar1()
    if key.char == "2":
        # A key produces a character value '2'
        cyberpi.led.on("orange")
        bar2()
    if key.char == "3":
        # A key produces a character value '3'
        cyberpi.led.on("yellow")
        bar3()
    if key.char == "4":
        # A key produces a character value '4'
        cyberpi.led.on("green")
        bar4()
def on_release(key):
    cyberpi.led.off()
    pass
with Listener(on_press=on_press, on_release=on_release) as listener:
    # Collect events until released
    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 Learnt
• The **'psutil'** module • Import **cyberpi**, **psutil**		

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 works or products by combining or editing existing materials or works. Sometimes, remix culture is also called read-write culture, which indicates the cultural artefacts may not be considered as the original work of someone and hence the '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 of work 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 can do 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-organises 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 against their predictions.

- Instruct students to write down their prediction and annotate the code.
- Have students fill out the **'What I Wonder'** column of the **K-W-L chart** after running the example program. If students get some ideas about what they would learn in this lesson, ask them to try to write down some main points.

What I Know	What I Wonder	What I Learnt
• The **'psutil'** module • Import **cyberpi**, **psutil**	• What is meant by the word **'def'**? • How to play sounds or music through CyberPi?	

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 in the text.
- **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 we want the functions that can monitor the keyboard input.
- Instruct students to utilise **'pynput'** to control the mouse. Demonstrate an example program as follow:
  - Send the file to students and ask them to run the program first.

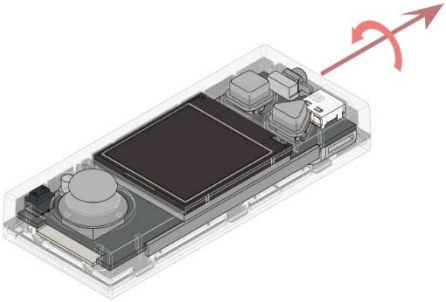
Example 6-1

```
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
```

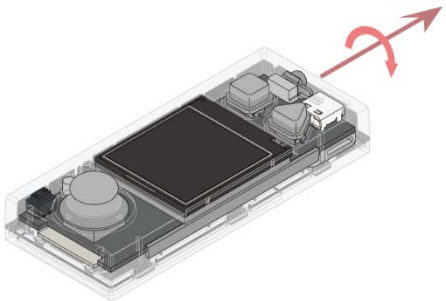
- Then ask students to add new functions that make the mouse cursor move up and down by tilting CyberPi forward and backward.

  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)
```

- **Summarise:** 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, organised, 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 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 points as follows:
  - 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 working individually to modify the example program by completing the task below:

- **Task 1:** Modify the parameters in the example program. Use letters instead 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

```
def bar1():
    cyberpi.led.on("green", id=1)
    cyberpi.audio.play_music(72, 0.4)
    cyberpi.audio.play_music(67, 0.4)
    cyberpi.led.on("green", id=2)
    cyberpi.audio.play_music(64, 0.2)
    cyberpi.audio.play_music(64, 0.1)
    cyberpi.audio.play_music(65, 0.1)
    cyberpi.audio.play_music(67, 0.4)
    cyberpi.led.on("green", id=3)
    cyberpi.audio.play_music(72, 0.1)
    cyberpi.audio.play_music(71, 0.1)
    cyberpi.audio.play_music(72, 0.1)
    cyberpi.audio.play_music(74, 0.1)
    cyberpi.audio.play_music(72, 0.1)
    cyberpi.audio.play_music(71, 0.1)
    cyberpi.audio.play_music(72, 0.1)
    cyberpi.audio.play_music(74, 0.1)
    cyberpi.led.on("green", id=4)
    cyberpi.audio.play_music(72, 0.1)
    cyberpi.audio.play_music(71, 0.1)
    cyberpi.audio.play_music(72, 0.1)
    cyberpi.audio.play_music(74, 0.1)
    cyberpi.audio.play_music(72, 0.4)


def on_press(key):
    if key.char == "1":
        bar1()
```

Example 6-6

```
def bar2():
    cyberpi.led.on("orange", id=1)
    cyberpi.audio.play_music(71, 0.1)
    cyberpi.audio.play_music(69, 0.1)
    cyberpi.audio.play_music(68, 0.1)
    cyberpi.audio.play_music(69, 0.1)
    cyberpi.led.on("orange", id=2)
    cyberpi.audio.play_music(72, 0.2)
    time.sleep(0.2)
    cyberpi.audio.play_music(74, 0.1)
    cyberpi.audio.play_music(72, 0.1)
    cyberpi.audio.play_music(71, 0.1)
    cyberpi.audio.play_music(72, 0.1)
    cyberpi.led.on("orange", id=3)
    cyberpi.audio.play_music(76, 0.2)
    time.sleep(0.2)
    cyberpi.audio.play_music(77, 0.1)
    cyberpi.audio.play_music(76, 0.1)
    cyberpi.audio.play_music(75, 0.1)
    cyberpi.audio.play_music(76, 0.1)
    cyberpi.led.on("orange", id=4)
    cyberpi.audio.play_music(83, 0.1)
    cyberpi.audio.play_music(81, 0.1)
    cyberpi.audio.play_music(80, 0.1)
    cyberpi.audio.play_music(81, 0.1)
    cyberpi.audio.play_music(83, 0.1)
    cyberpi.audio.play_music(81, 0.1)
    cyberpi.audio.play_music(80, 0.1)
    cyberpi.audio.play_music(81, 0.1)
    cyberpi.led.on("orange", id=5)
    cyberpi.audio.play_music(84, 0.2)


def on_press(key):
    if key.char == "2":
        bar2()
```

Section 6 Recap (5 minutes)

Step 6.1 Summarise the key points learnt in this lesson:

- Summarise why and how to define functions.
- Summarise how to control and monitor the input device by using the **'pynput'** module.
- Have students fill out the **'What I Learnt'** column of the **K-W-L chart**.

What I Know	What I Wonder	What I Learnt
● The **'psutil'** module ● Import **cyberpi**, **psutil**	● What is meant by the word **'def'**? ● How to play sounds or music through CyberPi?	● The '**pynput'** module ● Import **cyberpi**, **pynput** ● Functions ● Audio features of CyberPi