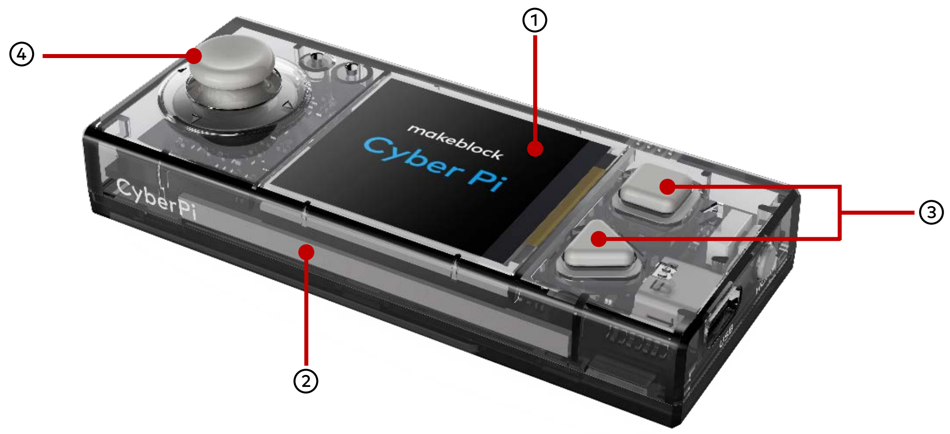
Lesson 1~2 Python Quizzes

Worksheet

K-W-L Chart

What I Know	What I Wonder	What I Learnt

New Commands



`import cyberpi`

① **Screen:** `cyberpi.display`, `cyberpi.console`

`cyberpi.display.clear()`

`cyberpi.console.print()`

`cyberpi.console.println()`

② **LED Strip:** `cyberpi.led`

`cyberpi.led.off()`

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

`cyberpi.led.on()`

`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")`

③ **Buttons A and B:** `cyberpi.controllor.is_press()`

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

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

④ **Joystick:** `cyberpi.controller.is_press()`

`cyberpi.controller.is_press(“up”)`

`cyberpi.controller.is_press(“down”)`

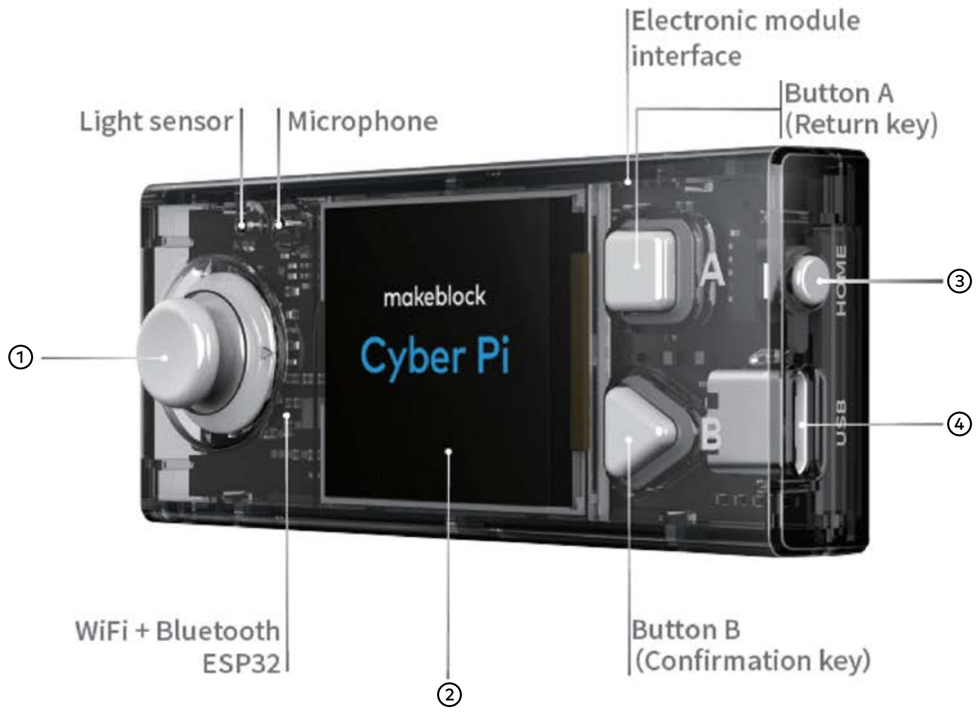
`cyberpi.controller.is_press(“right”)`

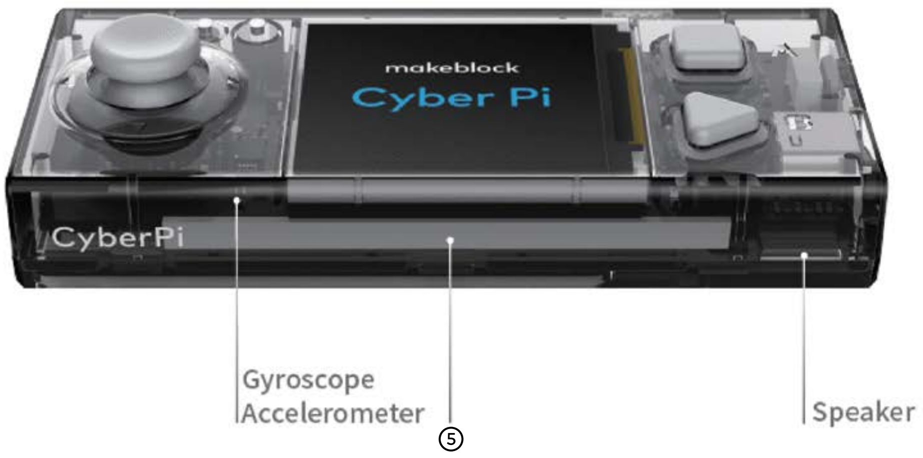
`cyberpi.controller.is_press(“left”)`

`cyberpi.controller.is_press(“middle”)`

Procedures

Features of CyberPi





Step 1 Predict

Read the program.

```python
import cyberpi

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

cyberpi.console.println("A - True")
cyberpi.console.println("B - False")

cyberpi.led.on(255, 255, 255)
cyberpi.console.println("Python is a compiled language.")
cyberpi.console.println("Your Answer:")

while True:

    if cyberpi.controller.is_press("a"):
        cyberpi.led.on(255, 0, 0)
        cyberpi.console.println("Incorrect.")
        cyberpi.console.println("Correct Answer: False")
        break

    if cyberpi.controller.is_press("b"):
        cyberpi.led.on(0, 255, 0)
        cyberpi.console.println("Correct!")
        break
```

Questions:

- Which module is imported? 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?

Step 2 Run

Run the program.

Step 3 Investigate

Investigate the physical components and API code samples of the screen, the LED strip, the buttons, and the joystick.

Step 4 Modify and Make

Task 1: Modify the quiz and the answer in the example program.

Task 2: Modify the lighting effects.

Tip:

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.

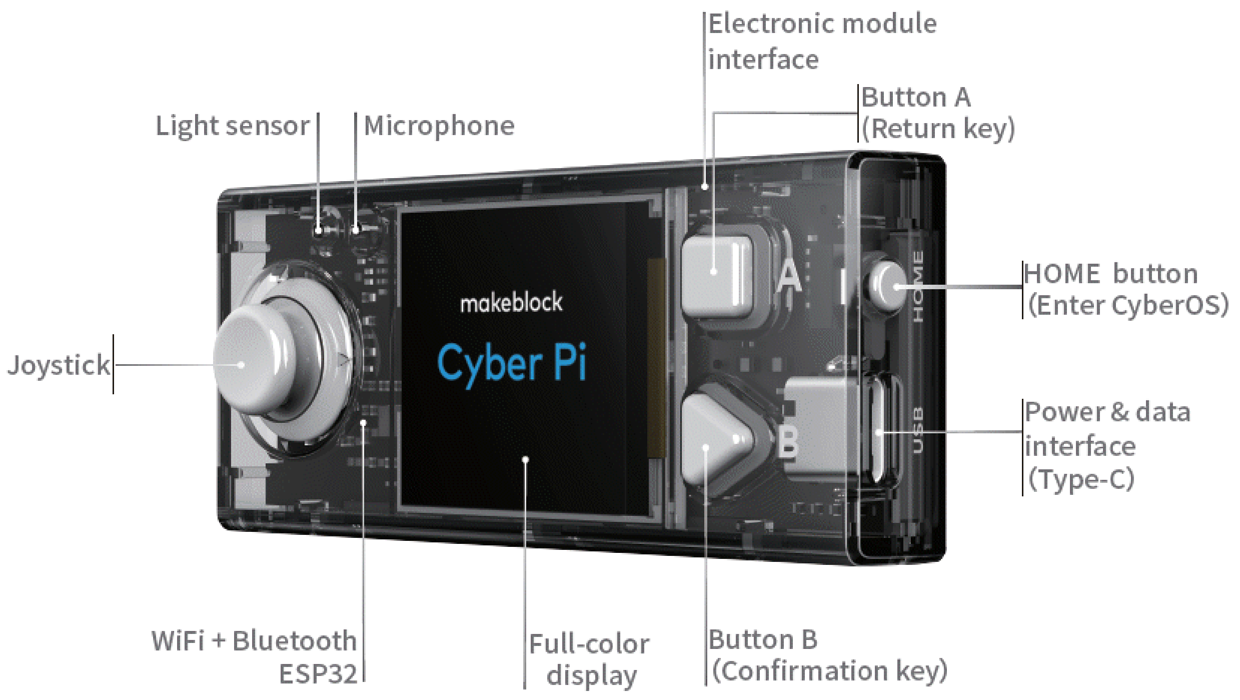
Lesson 3 Data Protection and Passwords

Worksheet

K-W-L Chart

What I Know	What I Wonder	What I Learnt

Features of CyberPi



Procedures

Step 1 Predict

Read the program.

```python
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.")
```

Questions:

- 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.

Step 2 Run

Run the program.

Step 3 Investigate

Answer the questions.

Step 4 Modify

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

Task 2: Add some lighting effects as the indicator. Use what you have learnt to program CyberPi.

Tip:

```
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.

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.

Step 5 Make

Develop a bank card reader

- 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.

Lesson 4 Normal Distribution

Worksheet

K-W-L Chart

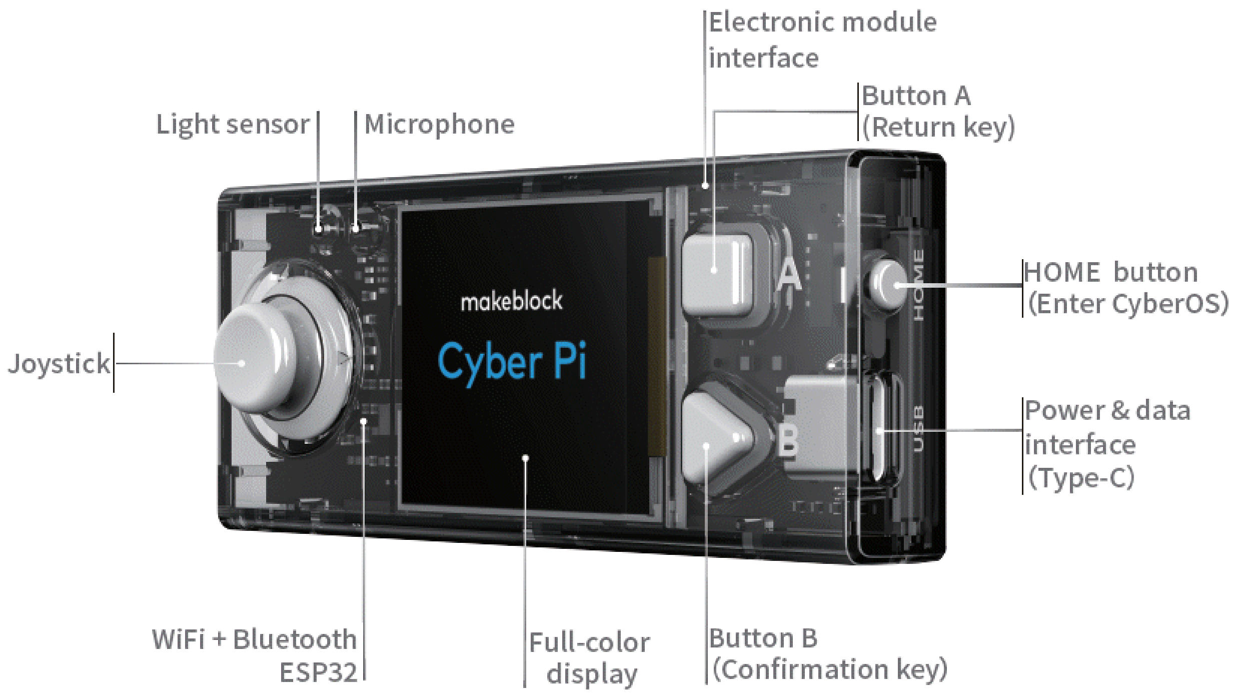
What I Know	What I Wonder	What I Learnt

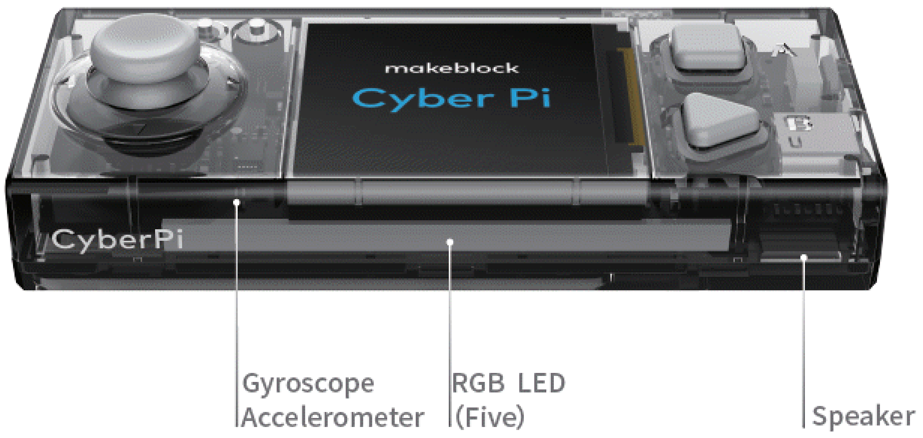
New Commands

`cyberpi.display.set_brush()`

`cyberpi.barchart.add()`

Features of CyberPi





Procedures

Step 1 Predict

Read the program.

```python
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]
count_list = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

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)
    count_list[sum_index] += 1

cyberpi.display.set_brush(128, 0, 0)
cyberpi.barchart.add(round(count_list[0]/n * 200, 2))
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))
```

Step 2 Run

Run the program.

Step 3 Investigate

Investigate 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.

Step 4 Modify and Make

Task: Simulate the experiment of tossing two coins together. Calculate all the possible outcomes in this experiment and visualise the distribution with a bar chart.

Head, Head, Head	

Lesson 5　Data Storage

Worksheet

K-W-L Chart

What I Know	What I Wonder	What I Learnt

New Commands

`import psutil`

`psutil.cpu_percent()`

`psutil.cpu_count()`

`psutil.cpu_freq()`

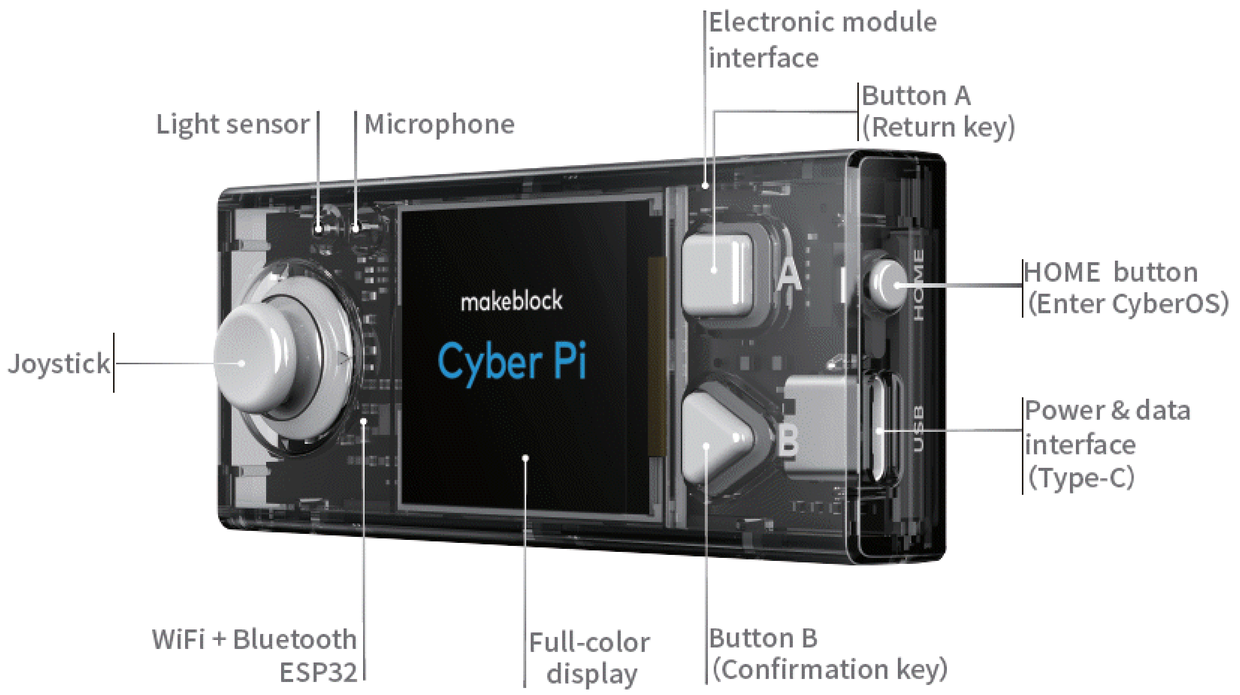
`psutil.virtual_memory()`

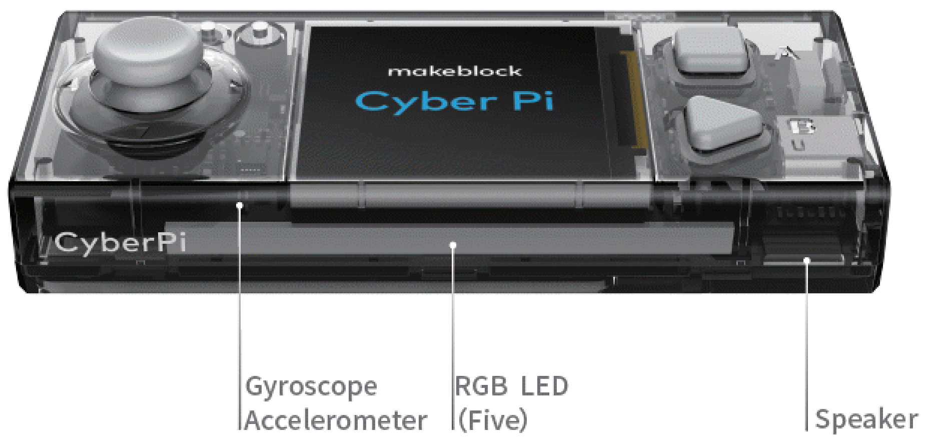
`cyberpi.linechart.add()`

`cyberpi.chart.set_name()`

`cyberpi.audio.play_tone()`

Features of CyberPi





Procedures

Step 1 Predict

Read the program.

```python
import cyberpi
import psutil

cyberpi.chart.clear()
while True:
    CPU = psutil.cpu_percent()
    mem = psutil.virtual_memory()
    mem_p = mem.percent
    cyberpi.display.set_brush(0, 0, 255)
    cyberpi.linechart.add(int(CPU))
    cyberpi.display.set_brush(255, 255, 0)
    cyberpi.linechart.add(int(mem_p))
    print("CPU:", CPU, "% Memory:", mem_p, "%")
```

Step 2 Run

Run the program.

Step 3 Investigate

Investigate 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.

Step 4 Modify and Make

Task: Add control structures and light effects to the example program. Define the thresholds of the alarm and the corresponding alarm indicators.

Note: 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

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

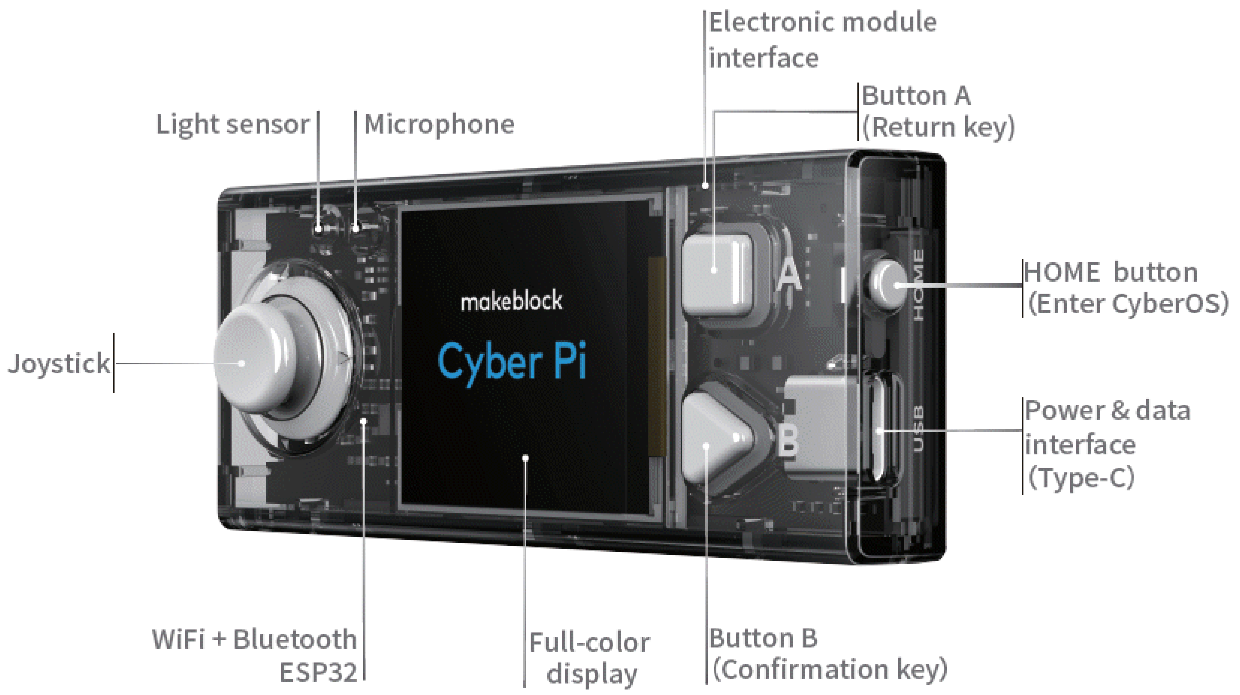
For the teacher:

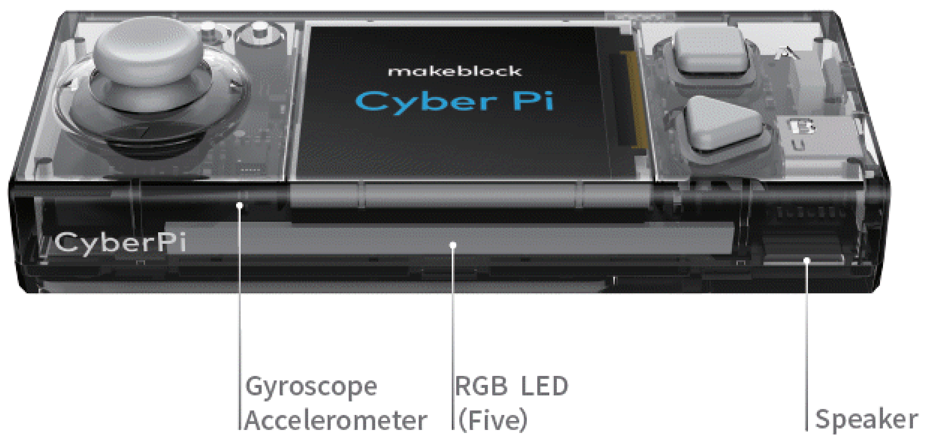
- A laptop or desktop with mBlock Python code editor installed
- A CyberPi device
- A Type-C cable
- Lesson plan
- Worksheet

For students:

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

Features of CyberPi





Example Program

```python
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.

```python
import cyberpi
from pynput.keyboard import Key, Listener
cyberpi.audio.set_vol(50)

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":
        cyberpi.led.on("red")
        bar1()
    if key.char == "2":
        cyberpi.led.on("orange")
        bar2()
    if key.char == "3":
        cyberpi.led.on("yellow")
        bar3()
    if key.char == "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:
    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.

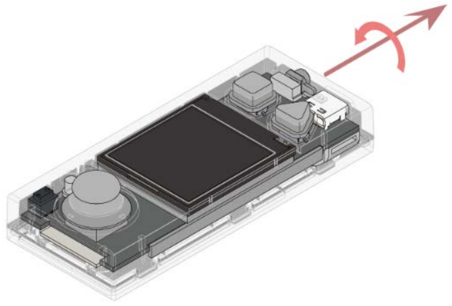
Example 6-1

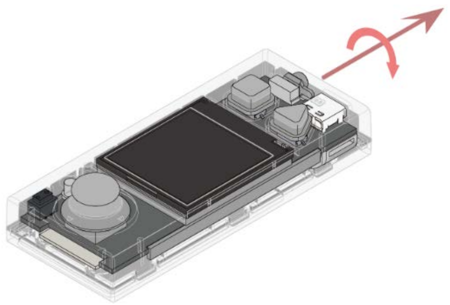
```python
import cyberpi
from pynput.mouse import Button, Controller
# Control the mouse

mouse = Controller()

while True:
    if cyberpi.is_tiltleft():
        # Tilt CyberPi left
        mouse.move(-5, 0)
        # Move the mouse cursor relative to current position
        print(mouse.position)

    if cyberpi.is_tiltright():
        # Tilt CyberPi right
        mouse.move(5, 0)
        print(mouse.position)

    if cyberpi.controller.is_press("a"):
        # Press CyberPi's Button A
        mouse.press(Button.left)
        mouse.release(Button.left)
        # 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

```python
if cyberpi.is_tiltforward():
    mouse.move(0, -5)

if cyberpi.is_tiltback():
    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

```python
if cyberpi.controller.is_press("b"):
    mouse.press(Button.right)
    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

```python
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

```python
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