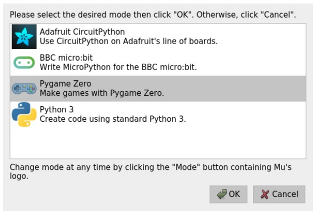
Your first program

When you start the Mu-Editor, you can pick a mode: choose "PyGame Zero".


Please select the desired mode then click "OK". Otherwise, click "Cancel".

Adafruit CircuitPython
Use CircuitPython on Adafruit's line of boards.

BBC micro:bit
Write MicroPython for the BBC micro:bit.

Pygame Zero
Make games with Pygame Zero.

Python 3
Create code using standard Python 3.

Change mode at any time by clicking the "Mode" button containing Mu's logo.

Mode If you have already chosen a different mode, click on the "Mode" button in the toolbar to get it again:

In the editor's window, you can remove the placeholder

```
# Write code here :-)
```

and type your first code:

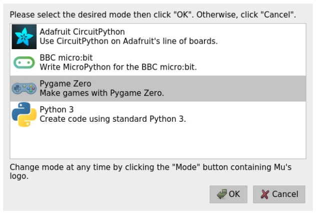
```
WIDTH = 640
HEIGHT = 480

def update(dt):
    pass

def draw():
    screen.fill((128, 0, 0))
```

You should see a dark red window popping up: it's your program running!

Stop You can close it by clicking on the stop button in the Mu-editor toolbar:

Your first program

When you start the Mu-Editor, you can pick a mode: choose "PyGame Zero".


Please select the desired mode then click "OK". Otherwise, click "Cancel".

Adafruit CircuitPython
Use CircuitPython on Adafruit's line of boards.

BBC micro:bit
Write MicroPython for the BBC micro:bit.

Pygame Zero
Make games with Pygame Zero.

Python 3
Create code using standard Python 3.

Change mode at any time by clicking the "Mode" button containing Mu's logo.

Mode If you have already chosen a different mode, click on the "Mode" button in the toolbar to get it again:

In the editor's window, you can remove the placeholder

```
# Write code here :-)
```

and type your first code:

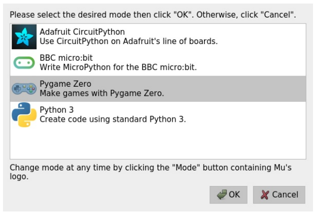
```
WIDTH = 640
HEIGHT = 480

def update(dt):
    pass

def draw():
    screen.fill((128, 0, 0))
```

You should see a dark red window popping up: it's your program running!

Stop You can close it by clicking on the stop button in the Mu-editor toolbar:

Your first program

When you start the Mu-Editor, you can pick a mode: choose "PyGame Zero".


Please select the desired mode then click "OK". Otherwise, click "Cancel".

Adafruit CircuitPython
Use CircuitPython on Adafruit's line of boards.

BBC micro:bit
Write MicroPython for the BBC micro:bit.

Pygame Zero
Make games with Pygame Zero.

Python 3
Create code using standard Python 3.

Change mode at any time by clicking the "Mode" button containing Mu's logo.

Mode If you have already chosen a different mode, click on the "Mode" button in the toolbar to get it again:

In the editor's window, you can remove the placeholder

```
# Write code here :-)
```

and type your first code:

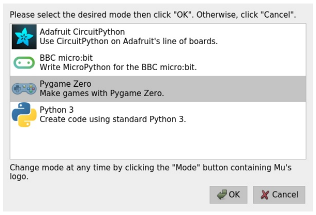
```
WIDTH = 640
HEIGHT = 480

def update(dt):
    pass

def draw():
    screen.fill((128, 0, 0))
```

You should see a dark red window popping up: it's your program running!

Stop You can close it by clicking on the stop button in the Mu-editor toolbar:

Your first program

When you start the Mu-Editor, you can pick a mode: choose "PyGame Zero".


Please select the desired mode then click "OK". Otherwise, click "Cancel".

Adafruit CircuitPython
Use CircuitPython on Adafruit's line of boards.

BBC micro:bit
Write MicroPython for the BBC micro:bit.

Pygame Zero
Make games with Pygame Zero.

Python 3
Create code using standard Python 3.

Change mode at any time by clicking the "Mode" button containing Mu's logo.

Mode If you have already chosen a different mode, click on the "Mode" button in the toolbar to get it again:

In the editor's window, you can remove the placeholder

```
# Write code here :-)
```

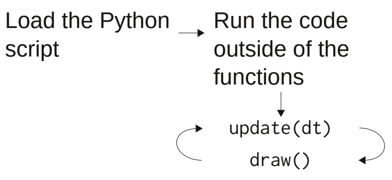
and type your first code:

```
WIDTH = 640
HEIGHT = 480

def update(dt):
    pass

def draw():
    screen.fill((128, 0, 0))
```

You should see a dark red window popping up: it's your program running!

Stop You can close it by clicking on the stop button in the Mu-editor toolbar:

What does the code you just wrote?



```python
WIDTH = 640
HEIGHT = 480
```

The two variable `WIDTH` and `HEIGHT` are defined outside of all functions: they are global variables that can be accessed from everywhere.
The two variables contain the width and height of the window and are"magically" used by PyGame to set the size of the game.

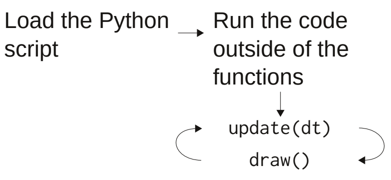
```python
def update(dt):
    pass
```

The `def` keyword starts the definition of a function. A function is a set of actions wrapped together. The `update(dt)` function has the name `update` and receives the argument `dt`. It gets called very often by PyGame Zero: you will write in there the code that modifies the state of your game.
Currently, our game does "nothing", so we just write there the void command `pass` (because in Python a function cannot be left empty).

```python
def draw():
    screen.fill((128, 0, 0))
```

The `draw` function is run just after the `update(dt)` function. In this first program, it simply calls the `fill()` function from the PyGame Zero module called `screen` to set the background of the window to a dark red: the argument we are passing to the `screen.fill` function is a tuple with the red, green and blue composant of the color.

What does the code you just wrote?



```python
WIDTH = 640
HEIGHT = 480
```

The two variable `WIDTH` and `HEIGHT` are defined outside of all functions: they are global variables that can be accessed from everywhere.
The two variables contain the width and height of the window and are"magically" used by PyGame to set the size of the game.

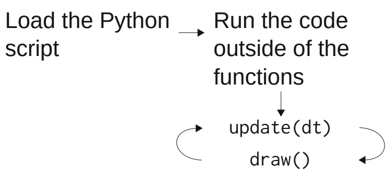
```python
def update(dt):
    pass
```

The `def` keyword starts the definition of a function. A function is a set of actions wrapped together. The `update(dt)` function has the name `update` and receives the argument `dt`. It gets called very often by PyGame Zero: you will write in there the code that modifies the state of your game.
Currently, our game does "nothing", so we just write there the void command `pass` (because in Python a function cannot be left empty).

```python
def draw():
    screen.fill((128, 0, 0))
```

The `draw` function is run just after the `update(dt)` function. In this first program, it simply calls the `fill()` function from the PyGame Zero module called `screen` to set the background of the window to a dark red: the argument we are passing to the `screen.fill` function is a tuple with the red, green and blue composant of the color.

What does the code you just wrote?



```python
WIDTH = 640
HEIGHT = 480
```

The two variable `WIDTH` and `HEIGHT` are defined outside of all functions: they are global variables that can be accessed from everywhere.
The two variables contain the width and height of the window and are"magically" used by PyGame to set the size of the game.

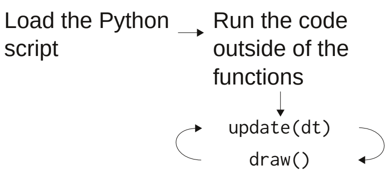
```python
def update(dt):
    pass
```

The `def` keyword starts the definition of a function. A function is a set of actions wrapped together. The `update(dt)` function has the name `update` and receives the argument `dt`. It gets called very often by PyGame Zero: you will write in there the code that modifies the state of your game.
Currently, our game does "nothing", so we just write there the void command `pass` (because in Python a function cannot be left empty).

```python
def draw():
    screen.fill((128, 0, 0))
```

The `draw` function is run just after the `update(dt)` function. In this first program, it simply calls the `fill()` function from the PyGame Zero module called `screen` to set the background of the window to a dark red: the argument we are passing to the `screen.fill` function is a tuple with the red, green and blue composant of the color.

What does the code you just wrote?



```python
WIDTH = 640
HEIGHT = 480
```

The two variable `WIDTH` and `HEIGHT` are defined outside of all functions: they are global variables that can be accessed from everywhere.
The two variables contain the width and height of the window and are"magically" used by PyGame to set the size of the game.

```python
def update(dt):
    pass
```

The `def` keyword starts the definition of a function. A function is a set of actions wrapped together. The `update(dt)` function has the name `update` and receives the argument `dt`. It gets called very often by PyGame Zero: you will write in there the code that modifies the state of your game.
Currently, our game does "nothing", so we just write there the void command `pass` (because in Python a function cannot be left empty).

```python
def draw():
    screen.fill((128, 0, 0))
```

The `draw` function is run just after the `update(dt)` function. In this first program, it simply calls the `fill()` function from the PyGame Zero module called `screen` to set the background of the window to a dark red: the argument we are passing to the `screen.fill` function is a tuple with the red, green and blue composant of the color.