

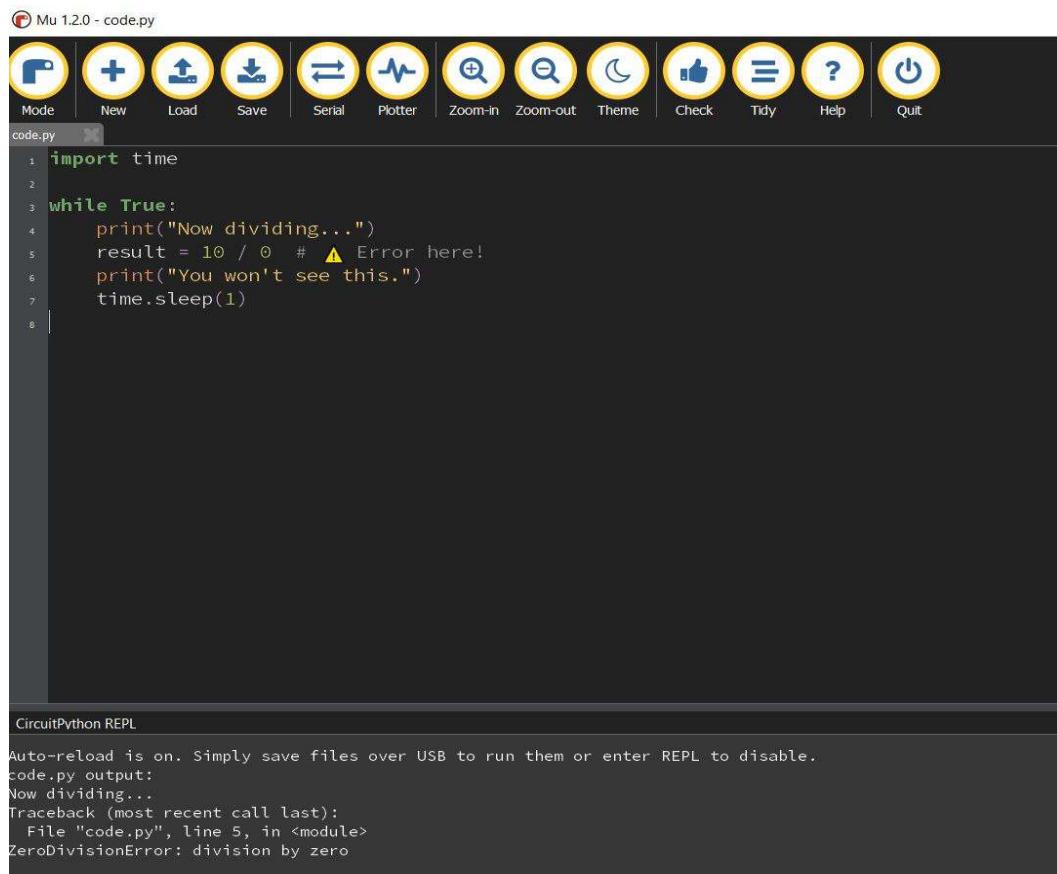
Try-Except block in CircuitPython

Objective

Understand how to use try-except blocks inside a loop in CircuitPython to prevent crashes and handle errors gracefully.

Without Try-Except

Let us execute this code in MuEditor and observe the output.



The screenshot shows the Mu Editor interface with a dark theme. The toolbar at the top has icons for Mode, New, Load, Save, Serial, Plotter, Zoom-in, Zoom-out, Theme, Check, Tidy, Help, and Quit. The code editor window shows a file named 'code.py' with the following content:

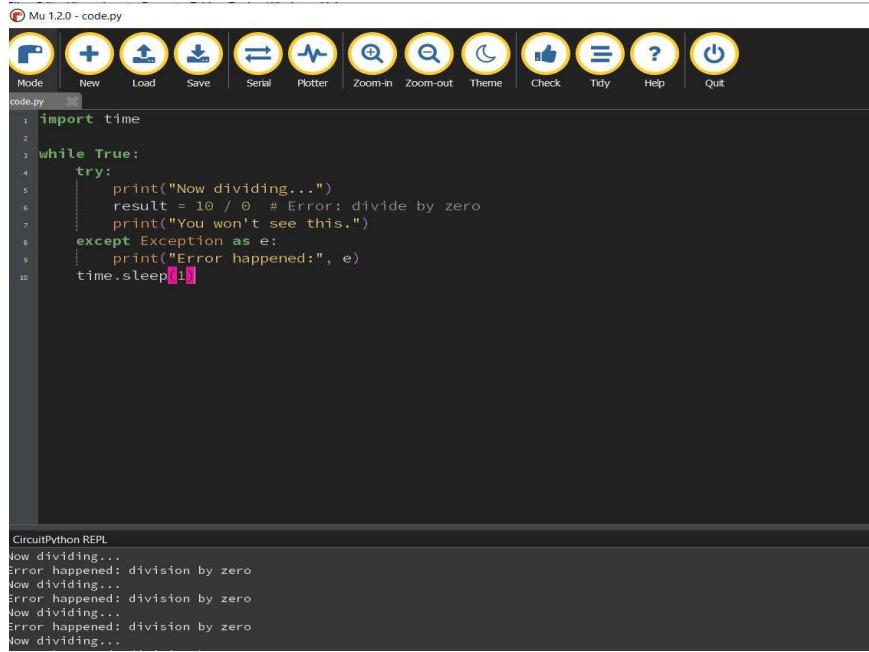
```
1 import time
2
3 while True:
4     print("Now dividing...")
5     result = 10 / 0 # ▲ Error here!
6     print("You won't see this.")
7     time.sleep(1)
8
```

Below the code editor is the CircuitPython REPL window, which displays the following output:

```
CircuitPython REPL
Auto-reload is on. Simply save files over USB to run them or enter REPL to disable.
code.py output:
Now dividing...
Traceback (most recent call last):
  File "code.py", line 5, in <module>
ZeroDivisionError: division by zero
```

In this version, the program crashes after the first error, and the loop stops completely.

Code Example with Try-Except



The screenshot shows the Mu 1.2.0 IDE interface. The top bar has a 'Mu 1.2.0 - code.py' tab. Below the bar are various icons for file operations (Mode, New, Load, Save, Serial, Plotter, Zoom-in, Zoom-out, Theme, Check, Tidy, Help, Quit). The main area shows a Python script named 'code.py' with the following code:

```
1 import time
2
3 while True:
4     try:
5         print("Now dividing...")
6         result = 10 / 0 # Error: divide by zero
7         print("You won't see this.")
8     except Exception as e:
9         print("Error happened:", e)
10    time.sleep(1)
```

The bottom area shows the 'CircuitPython REPL' output:

```
Now dividing...
Error happened: division by zero
Now dividing...
Error happened: division by zero
Now dividing...
Error happened: division by zero
Now dividing...
```

Now let us wrap the statement inside a try-except block and observe what happens

Code Explanation

`import time`

Imports the `time` module. Used for adding delay between servo movements.

`import board:`

Imports the `board` module to use specific hardware pins. For example, `board.A1` refers to pin A1.

`import pwmio`

Imports `pwmio` to enable PWM output. Required for controlling servo motors.

`from adafruit_motor import servo`

Imports the servo class from Adafruit motor library. Allows easy control of standard servo angles

`my_servo = servo.Servo(pwm)"`

Creates a Servo object connected to the PWM signal. Enables angle control from 0° to 180°.

`import time`

Loads the time module

`while True:`

Creates an infinite loop — this block will repeat forever.

try:

Begins a try block, which means "try this and see if an error happens."

print("Now dividing...")

Shows this message before doing the calculation.

result = 10 / 0 # Error: divide by zero

Still causes an error, but now it will be caught safely.

print("You won't see this.")

This is skipped if an error occurs.

except Exception as e:

Catches the error and stores the details in a variable e.

print("Error happened:", e)

Displays the error message without crashing the program.

time.sleep(1)

Program waits 1 second, then repeats the loop safely.

Expected Output

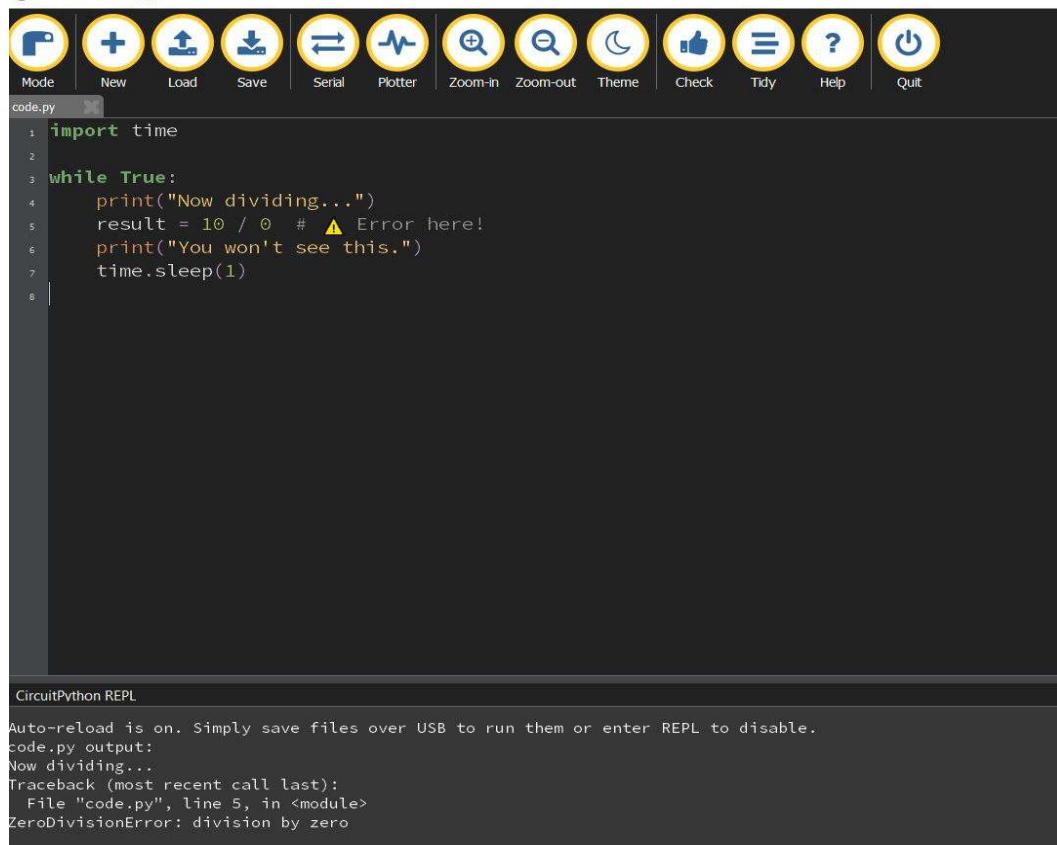
The message 'Now dividing...' is printed.

Then the error occurs (division by zero), and the program jumps to the except block.

'You won't see this.' is skipped because of the error.

The error message is printed, and the loop continues.

¶ Mu 1.2.0 - code.py



The image shows the Mu 1.2.0 IDE interface. At the top is a toolbar with 14 icons: Mode, New, Load, Save, Serial, Plotter, Zoom-in, Zoom-out, Theme, Check, Tidy, Help, and Quit. Below the toolbar is a code editor window titled "code.py" containing the following Python code:

```
1 import time
2
3 while True:
4     print("Now dividing...")
5     result = 10 / 0 # ▲ Error here!
6     print("You won't see this.")
7     time.sleep(1)
8
```

Below the code editor is a "CircuitPython REPL" window showing the output of the code execution. The output includes a warning about auto-reload, the code output, a traceback, and the error message "ZeroDivisionError: division by zero".

```
Auto-reload is on. Simply save files over USB to run them or enter REPL to disable.
code.py output:
Now dividing...
Traceback (most recent call last):
  File "code.py", line 5, in <module>
ZeroDivisionError: division by zero
```