# Lab 7 - Installing VSCode and Blinking the onboard LED

In this lab, we will install VSCode, a development environment for programming the Raspberry Pi Pico using Micropython. After installing VSCode and configuring the Pico for Micropython we will verify that the Pico can be reprogrammed by blinking the on-board LED.

Table of Contents

[Lab 7 - Installing VSCode and Blinking the onboard LED 1](#_Toc525037801)

[Python 3 1](#_Toc128991757)

[Downloading Python 3 1](#_Toc1014777999)

[Installing Python 3 1](#_Toc1820064276)

[Configuring System Variables for Python 3 3](#_Toc1893006953)

[Visual Studio Code 5](#_Toc1724279260)

[Downloading Visual Studio Code 5](#_Toc2106619225)

[Installing Visual Studio Code 5](#_Toc2130053569)

[Configuring Visual Studio Code 7](#_Toc1764206140)

[Configuring the look of the development Environment 7](#_Toc1993707547)

[MicroPico Extension for Visual Studio Code 7](#_Toc1665873479)

[Installing the Python Extension 8](#_Toc423721434)

[Installing the IntelliCode Extension 8](#_Toc1910064865)

[Installing the Pylance Extension 9](#_Toc610871264)

[Installing the MicroPico Extension 9](#_Toc643406974)

[Install Wokwi Simulator 10](#_Toc442345071)

[Install MicroPython Bootloader onto the Raspberry Pi Pico 10](#_Toc1784252755)

[What is a Bootloader? 10](#_Toc1038595718)

[Downloading the UF2 Bootloader 10](#_Toc1817767012)

[Uploading the Bootloader onto the Raspberry Pi Pico 10](#_Toc2044423478)

[Blink an LED on the Raspberry Pi Pico 11](#_Toc1356925944)

[Create a Project 11](#_Toc119541883)

[Configure MicroPico Extension for Project 12](#_Toc1545044446)

[Verify that Project is Configured for the MicroPico Extension 12](#_Toc1396509738)

[Create a new file 12](#_Toc1132482359)

[Write Code for Running on the Pico 12](#_Toc636980883)

[Uploading the Code onto the Pico 15](#_Toc1028918933)

[Additional Resources 15](#_Toc1724304300)

## Python 3

Micropython is derived from Python 3 so we need to install Python 3 onto our computers to be able to program using Micropython.

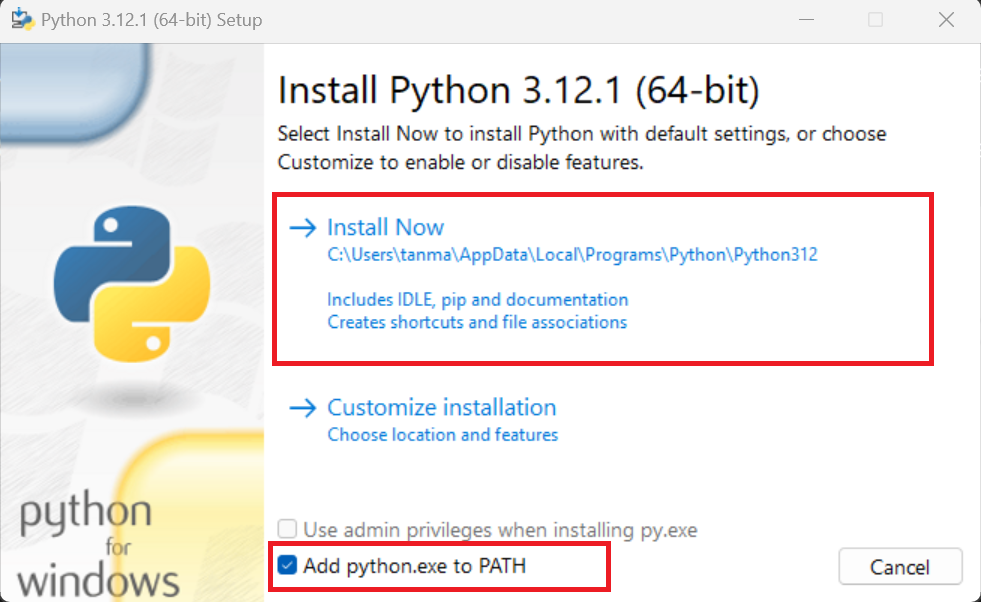
### Downloading Python 3

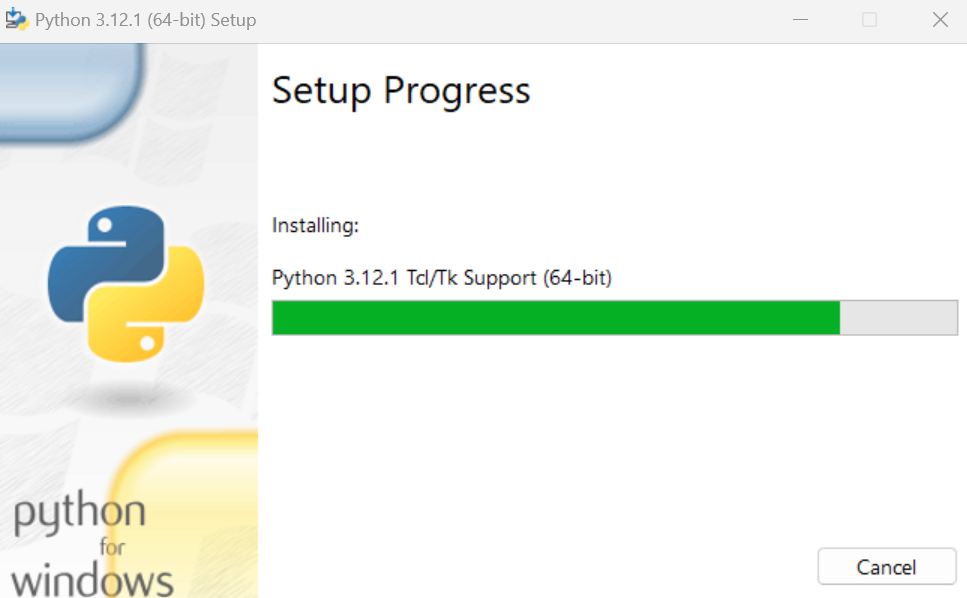
Go to <https://www.python.org/downloads/> and click the link to download the latest version of Python. Be sure to click on the link for the appropriate operating system of the computer that you are using.

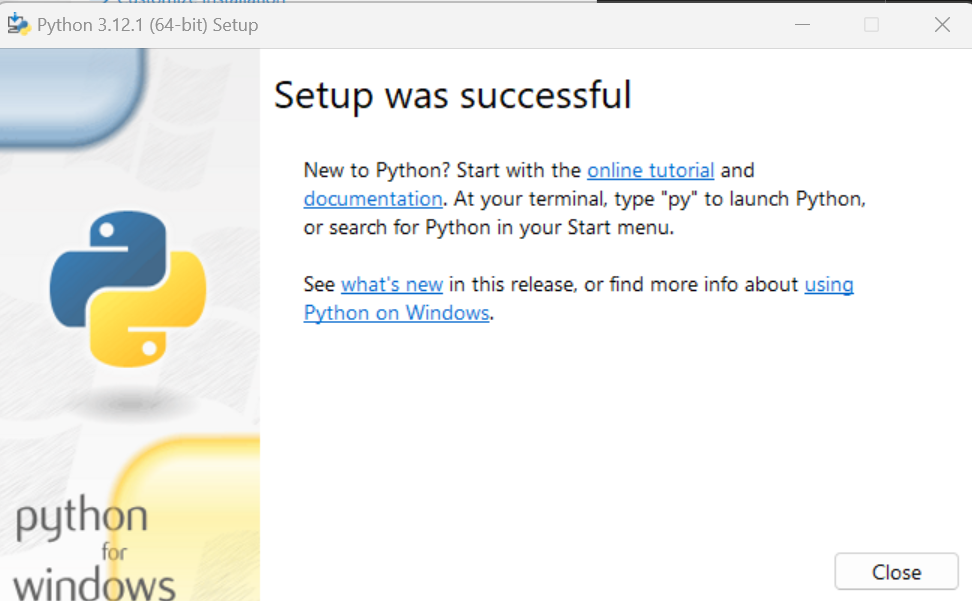


### Installing Python 3

Find the python.exe file and run it.







Copy down the full path directory that you installed Python 3, we will use this information when adding Python 3 to our PATH, more details to follow in the next section. For Example, Prof. Blum installed Python 3 at the following location:

C:\Users\Arielle\Python\Python3

### Configuring System Variables for Python 3

On Windows if we want to be able to run python without changing folders and using the full directory path to where the .exe is located, we can use an alias instead which incorporates the proper directories when the alias is entered, like just typing out “python”. By adding python 3 to your path, you can run python programs by typing out “python” in the command line and then type out the name of the file you want to run .py and the file will run. Otherwise, python will not be recognized.

To add something to your PATH complete the following steps.

Press the Windows key

Type into the search menu “path”

“Edit the system environment variables” should come up as an option as shown below.

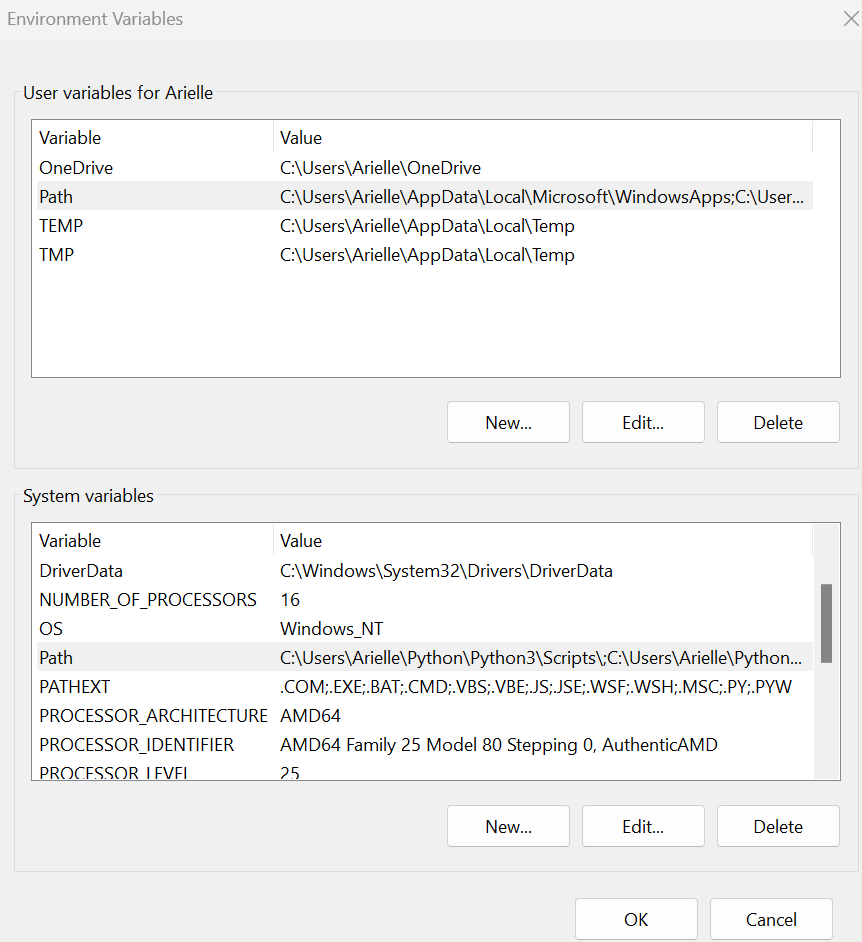
Screenshot of searching for path in the windows search bar.


Click “Edit the system environment variables”

Under System Properties, click the “Advanced” tab.

At the bottom of the “Advanced” tab window there should be a button called “Environment Variables”

The following window should pop up, go down to “System Variables”



Select “Path” and then the “Edit” button

Go to your Windows Explorer and locate the directory that you installed Python 3. Copy this file path

The following window will then pop up, if you don’t already have a Environment Variable for Python then select the “New” button to create a new Environment Variable.

A screenshot of a computer

Description automatically generated

Type out the directory path to your Python 3 installation. Recall that you made note of this file path when you originally installed Python 3 in the Downloading Python 3 section.

For example, Prof. Blum

## Visual Studio Code

There are many different development environments to program the Raspberry Pi Pico. In this course, we will be using Visual Studio Code.

### Downloading Visual Studio Code

Go to <https://code.visualstudio.com/> to download Visual Studio Code

Click on Download for <Insert your operating system>

Select the “Stable” installation

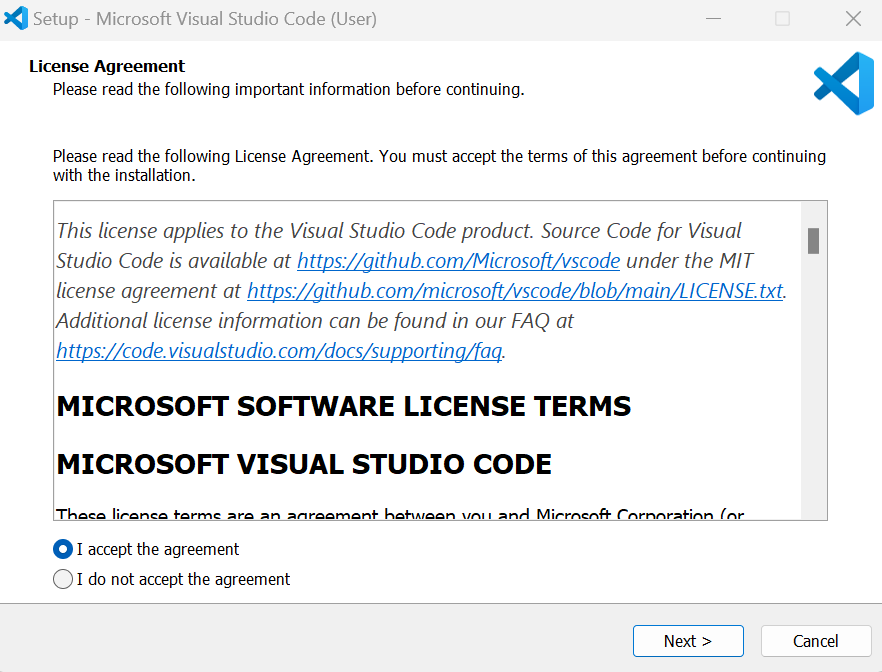
Wait until the download has completed before proceeding to the next step.

### Installing Visual Studio Code

After downloading the file, then go through the process of installing the program on your computer as outlined below:

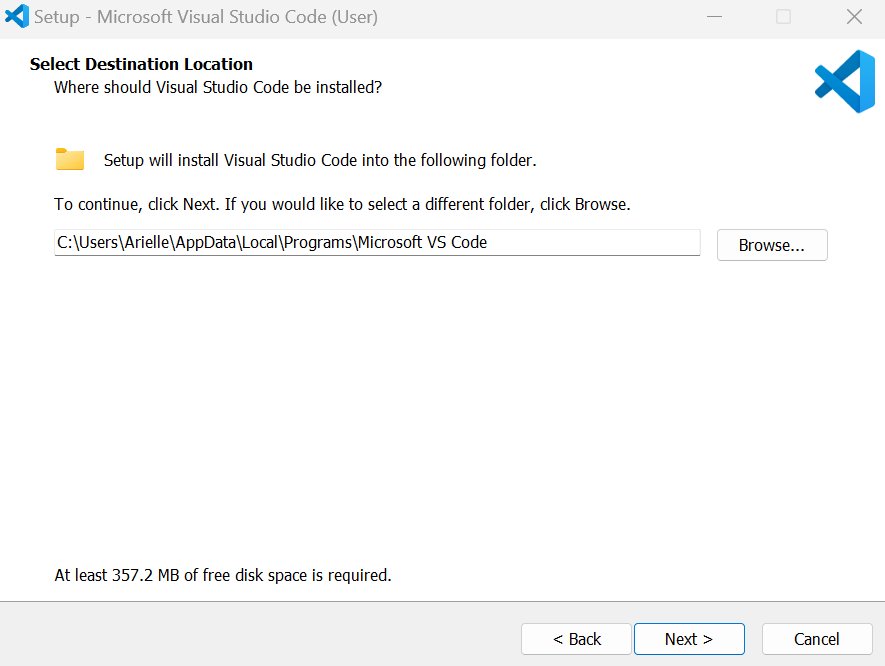
Navigate to your executable file, .exe file and double click the file to start the installation.

The following window will then pop up.



Accept the agreement and click Next

Change the directory of the installation to where you install your other programs by removing the “AppData\Local” from the file path for Windows Users.

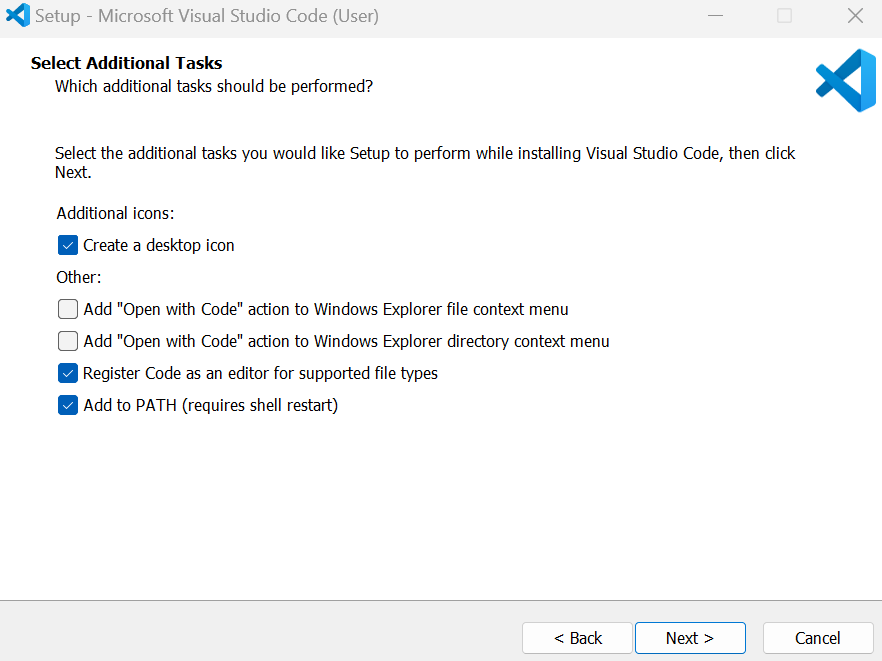


Here is an example of the updated file path:

C:\Users\Arielle\Programs\Microsoft VS Code

Click next

At the following prompt, be sure to select the last 2 boxes as shown below



Then click next until installation begins.

## Configuring Visual Studio Code

In this section, we will configure Visual Studio Code for use with the Raspberry Pi Pico.

Navigate to the installed Visual Studio Code software and start the program.

### Configuring the look of the development Environment

In this course, I will use Dark High Contrast as shown below:

Screenshot of VSCode configuration.


Click on the button for “**Rich support for all your languages**” and find the icon for the **IntelliSense Extension**

### MicroPico Extension for Visual Studio Code

MicroPico is a Visual Studio Code extension designed for using MicroPython with the Raspberry Pi Pico and Pico W. From the website: “This tool streamlines the coding process, providing code highlighting, auto-completion, code snippets, and project management features”

To use MicroPico, we first need to install the following Visual Studio Code extensions which are used to support the MicroPico extension:

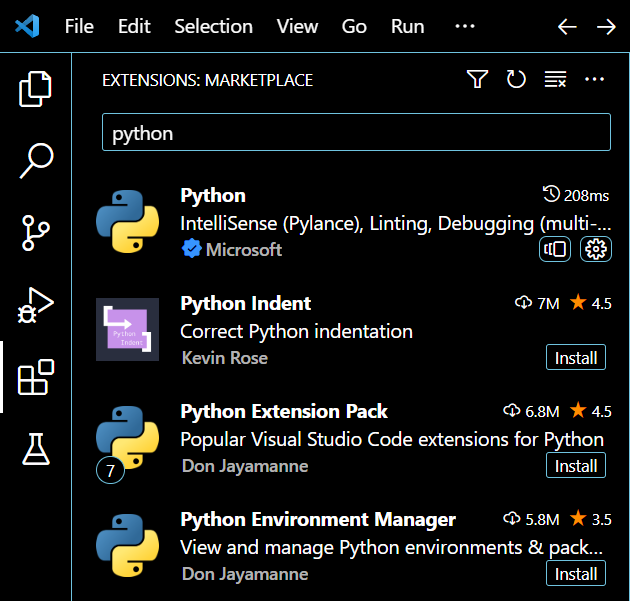
1. **Python**
2. **Intellicode**
3. **Pylance**

### Installing the Python Extension

The “Python” Extension will act as an assistant while we are programming in python and is used as the basis for the MicroPico Extension.

In Visual Studio Code, click on the Extensions Icon, A white and blue squares on a black background

Description automatically generated, and in the search bar under Extensions, type “python”

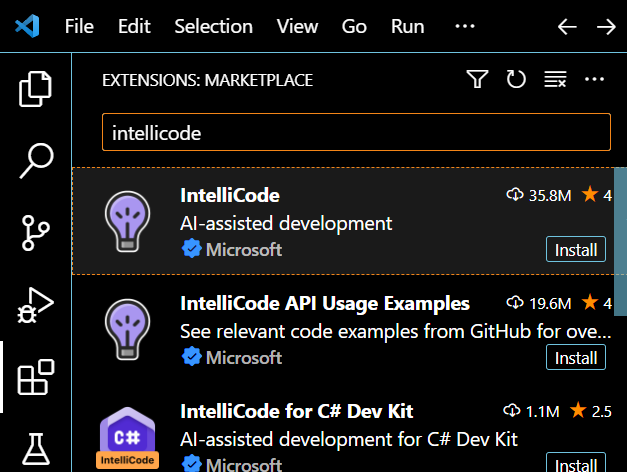


Click on the button to Install the Python Extension.

### Installing the IntelliCode Extension

IntelliCode is an Artificial Intelligence assistance that provides support for programming with features like code completion. The AI detects the context of the code, like whether you want to use a certain variable or function and then suggests that variable or function name.

Find the “IntelliCode” Extension by searching for the extension within the Extension tab in Visual Studio Code, using the search bar to type in IntelliCode.



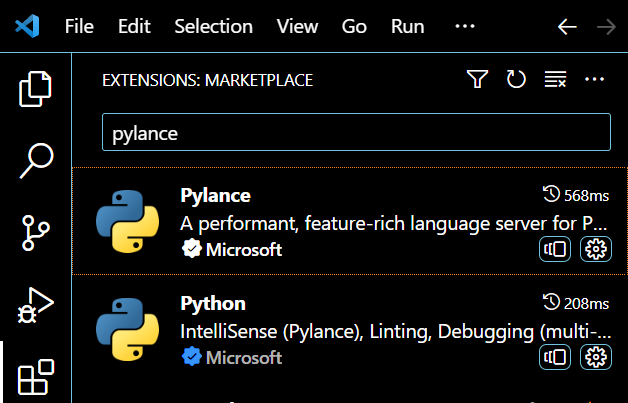
Click on the “Install” button for IntelliCode.

### Installing the Pylance Extension

Pylance is an Extension for a Python Language Server within Visual Studio Code that supports the editing experience, such as implementing auto-complete.

If interested, here is a link to learn more (this is not required, <https://microsoft.github.io/language-server-protocol/overviews/lsp/overview/> )

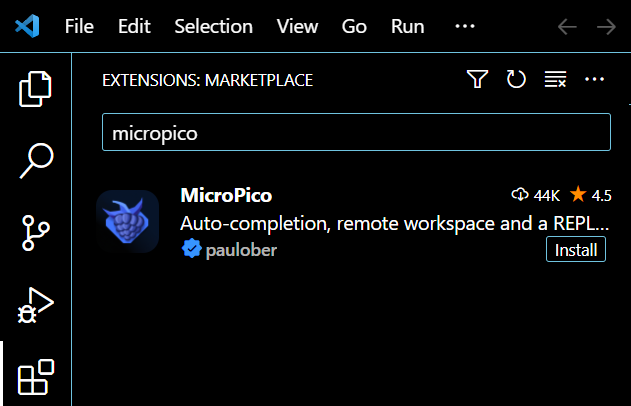
Install the “Pylance” Extension using the same process as described above for the other Extensions.



### Installing the MicroPico Extension

Why use MicroPico Extension?

After installing the previous extensions, you should now be ready to install MicroPico Extension. Go ahead and do so.



### Install Wokwi Simulator

Go to Extensions and then type “Wokwi” to find the simulator extension. Then click to Install the extension.

Congratulations, now we are done with configuring the Extensions for Visual Studio Code. Next, we will configure the physical Raspberry Pi Pico so that we can program it using the MicroPython programming language.

## Install MicroPython Bootloader onto the Raspberry Pi Pico

In this next section, we will physically connect to the Raspberry Pi Pico and configure it so that we can use the MicroPython language to program it.

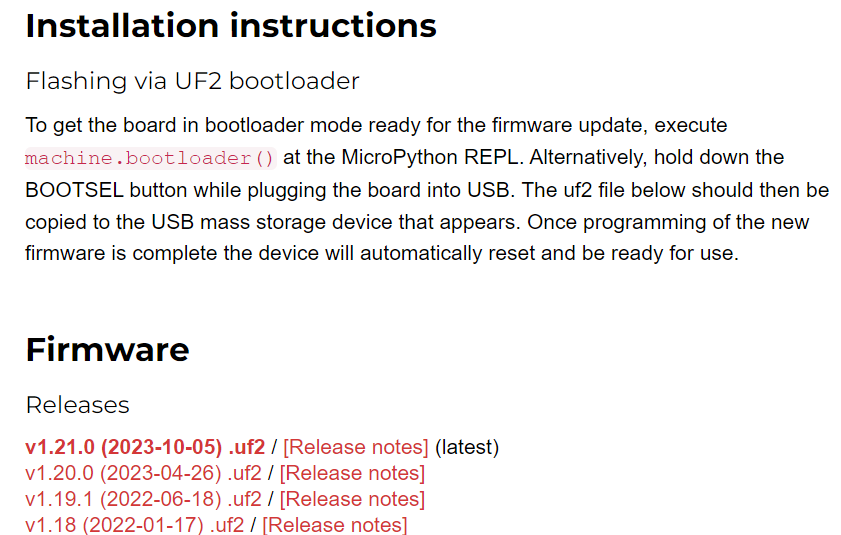
### What is a Bootloader?

**Bootloader** – a bootloader is a program that runs when the Raspberry Pi Pico is turned on. When we want to load new code or programs onto the Pico, the bootloader assists with loading the information sent by your computer into the Pico’s memory.

### Downloading the UF2 Bootloader

Go to <https://micropython.org/download/RPI_PICO/>

Scroll down until you find the Firmware section, under Releases, click to download the latest version of the UF2 bootloader.



At this time, the most recent release is the following: v1.21.0 (2023-10-05) .uf2 but please go ahead and **download the latest** release even if it differs from the one shown in the screenshot above and listed previously.

### Uploading the Bootloader onto the Raspberry Pi Pico

CAUTION: The USB connector on the Pico is fragile, be careful when connecting and disconnecting the USB cable to this connector as the connector can be easily broken.

Be sure to remove the USB cable directly and never pull the cable at an angle as this will break the connector.

Required Materials:

* Raspberry Pi Pico
* Computer with UF2 file downloaded
* USB-B female to USB-B Male Micro cable

There is a button on the Raspberry Pi Pico labeled BOOTSEL, short for “boot selection”. This button is used to place the Pico in boot mode. In boot mode the Pico runs MicroPython. If the button is held down while a USB cable is connected between the Pico and your computer, then the Pico will appear to the computer as a removeable drive. As a removeable drive, new bootloader files can be loaded onto the Pico.

On the Pico, hold down the BOOTSEL button while plugging in the USB cable connected to the Pico into your computer. Release the BOOTSEL button once the Pico is connected to the computer.

Using your computer’s file manager, navigate to the directory where you downloaded the bootloader firmware UF2 file. For me, this is in my Downloads folder.

#### Locating the Pico on your computer

WindowsOS - open a new **File Explorer** window

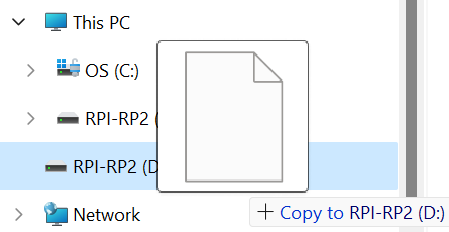
In the left-hand navigation window, scroll down until you see your device listed as a USB mass storage device as RP1-RP2

A black and white text

Description automatically generated

macOS – open a new **Finder** window the Pico should show up as RP1-RP2

Drag the UF2 file to the Pico shown as a mass storage device on your computer.



Once you do this, the RP1-RP2 will disappear.

Unplug the USB cable connected to the Pico from the computer and then plug it back into the computer without holding the BOOTSEL button.

## Blink an LED on the Raspberry Pi Pico

We want to verify that we can connect to the Pico and upload code, to test this, we are going to blink the LED on the Pico. This will also serve to verify that Python 3 and the various Extensions were installed properly.

### Create a Project

In the Explorer, create a new folder called “Lab8\_Blink\_OnBoard\_LED”

### Configure MicroPico Extension for Project

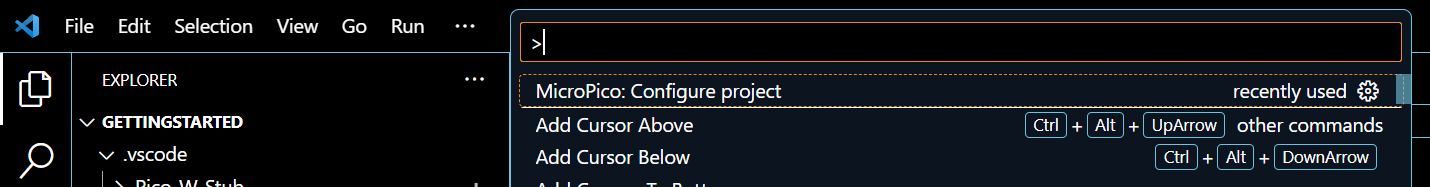
To configure the MicroPico Extension for the project we can use the following hot-keys:

**PC**: Type CTRL + SHIFT + P

**MAC**: Type CMD + SHIFT + P

Then type “MicroPico: Configure project”

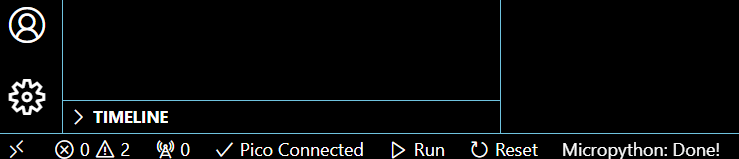
It should look like the screenshot below:



### Verify that Project is Configured for the MicroPico Extension

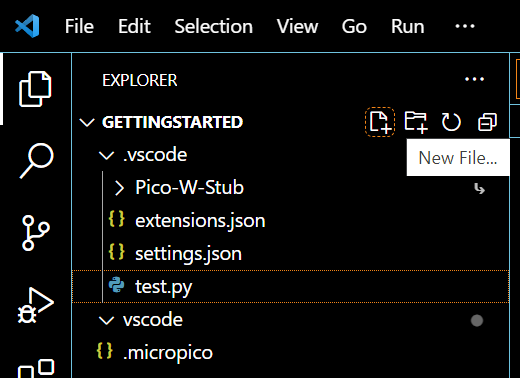
To verify that the project has been configured properly, we want to connect the Pico and ensure that VSCode can identify the board. To do this complete following steps outlined below.

Plug the USB cable connected to the Pico back into the computer and verify that the Pico is connected.



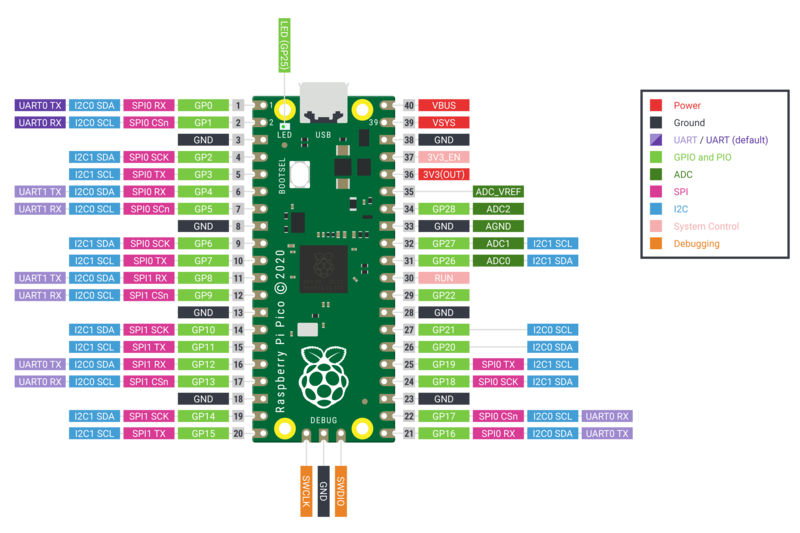
### Create a new file

Now create a file called Lab8.py



### Write Code for Running on the Pico

Our first code to run on the Pico will be blinking the onboard LED. To blink the onboard LED, we will determine the pin that the LED is connected to on the board

Use the pinout to determine the pin that the on-board LED is connected to.

Once you have located which pin is connected to the Raspberry Pi Pico, then we can use the code provided.

Copy the following code into Lab8.py

from machine import Pin

from utime import sleep

pin = Pin("LED", Pin.OUT)

print("LED starts flashing...")

while True:

    try:

        pin.toggle()

        sleep(1) # sleep 1sec

    except KeyboardInterrupt:

        break

pin.off()

print("Finished.")

from machine import Pin

- This line imports the `Pin` class from the `machine` module. The `machine` module is often used in MicroPython for hardware-related operations on microcontrollers.

from utime import sleep

- This line imports the `sleep` function from the `utime` module. The `utime` module provides functions related to time and is used here for creating delays in the program.

pin = Pin("LED", Pin.OUT)

- This line creates a Pin object named `pin` and associates it with the physical pin labeled "LED" on the microcontroller. It also specifies that this pin will be used for output (`Pin.OUT`).

print("LED starts flashing...")

- This line prints a message to the console indicating that the LED is about to start flashing.

while True:

- This line initiates an infinite loop. The code inside this loop will keep executing repeatedly until a break condition is encountered.

try:

- This line begins a try block, indicating that the code inside it might raise an exception, and the program should be prepared to handle it.

pin.toggle()

- This line toggles the state of the `pin` object between high and low. If the LED was on, it will turn it off, and if it was off, it will turn it on.

sleep(1) # sleep 1sec

- This line introduces a delay of 1 second using the `sleep` function. It pauses the program execution for the specified duration.

except KeyboardInterrupt:

- This line starts an exception block specifically for handling the `KeyboardInterrupt` exception. This exception is raised when the user interrupts the program, typically by pressing Ctrl+C.

break

- This line breaks out of the infinite loop when a `KeyboardInterrupt` exception is caught.

pin.off()

- This line turns off the LED by setting the pin state to low.

print("Finished.")

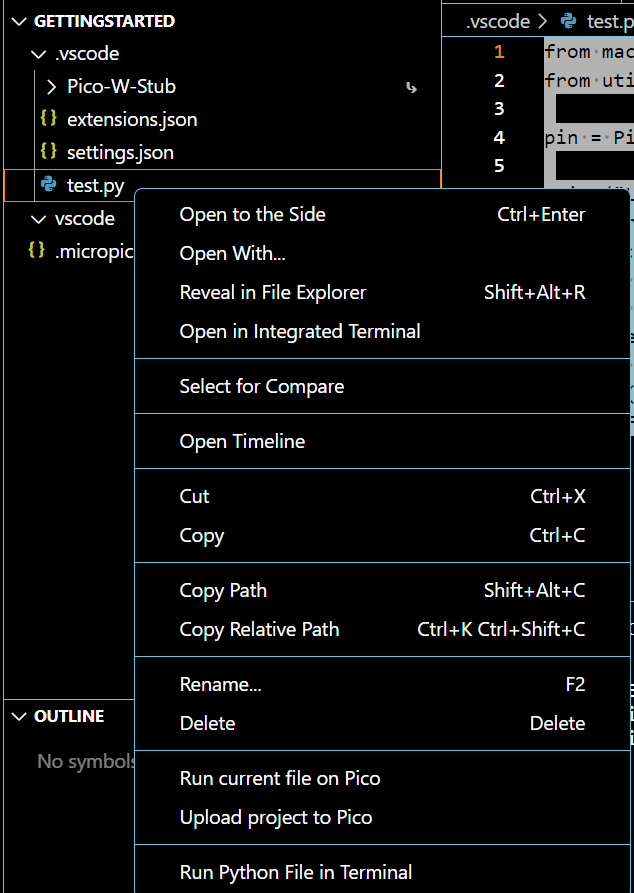
- This line prints a message indicating that the LED flashing has finished.

In summary, this code sets up a loop that toggles the state of an LED (connected to a pin labeled "LED") every second. The loop continues until the user interrupts the program, at which point the LED is turned off, and a finishing message is printed.

### Uploading the Code onto the Pico

Go to the Explorer tab in Visual Studio Code and right-click on the Lab8.py file.

Select “Run current file on Pico” to upload the code



The LED should now be flashing on the Pico.

Verify that the LED is flashing, if it is, congratulations!

dfddsfsfsdfds

## Additional Resources

Raspberry Pi Pico Micropython specific libraries for the RP2040 <https://docs.micropython.org/en/latest/library/rp2.html>

Micropython Quick Reference for the RP2, Raspberry Pi Pico – Documentation for the Hardware-Level Libraries

<https://docs.micropython.org/en/latest/rp2/quickref.html>

Video about Raspberry Pi Pico and Visual Studio - <https://www.youtube.com/watch?v=O6lkYTfcMEg&t=200s>