

National University of Singapore  
School of Computing  
CS1010S: Programming Methodology  
Semester II, 2022/2023

**Mission 0**  
**Setting Up Python**

Release date: 5<sup>th</sup> January 2023

**Due: 17<sup>th</sup> January 2023, 23:59**

## Required Files

- mission00-template.py

The objective of this mission is to guide you in the installation and setting up of the programming tool, called IDLE, that you will be using for the rest of the semester. In Part 2, we include a simple exercise for you to familiarize yourself with the basics of Python, as well as learn how to submit homework on Coursemology.

This mission consists of only **one** task.

## Part 1: Installing Python 3.10.6 and required packages

Before you start on your quest to master the Python programming language, you have to install the necessary tools. Please follow the following instructions to set up your programming environment for the class.

PIL is required to render images that will be used in the later missions. We will be using PILLOW, which is a modern replacement for PIL. The NumPy package are also required in the later missions.

**Note:** The highest priority package to install is **PILLOW**, as we will be using it over the next few weeks.

Now, please follow the following instructions carefully:

*(For Windows) (For Mac) (For Linux)*

## Troubleshooting

You can contact the teaching staff or check the Coursemology forum if you face any difficulty during the setup/installation process.

Please provide the following information to help us quickly identify your problem:

1. The operating system and version that you are using. MacOS, Windows 10, 32-bit or 64-bit, etc.
2. Which step in the instructions given below did it fail or produce an error?
3. Take a screen shot of the failure or error.

## Download Python

**IMPORTANT:** You will need to be **connected to the Internet** for all the installation steps as the required packages will have to be downloaded.

### Windows Users:

#### Step 1: Install Python for Windows

Head over to <https://www.python.org/downloads/> and you should see something similar to this:

Looking for a specific release?

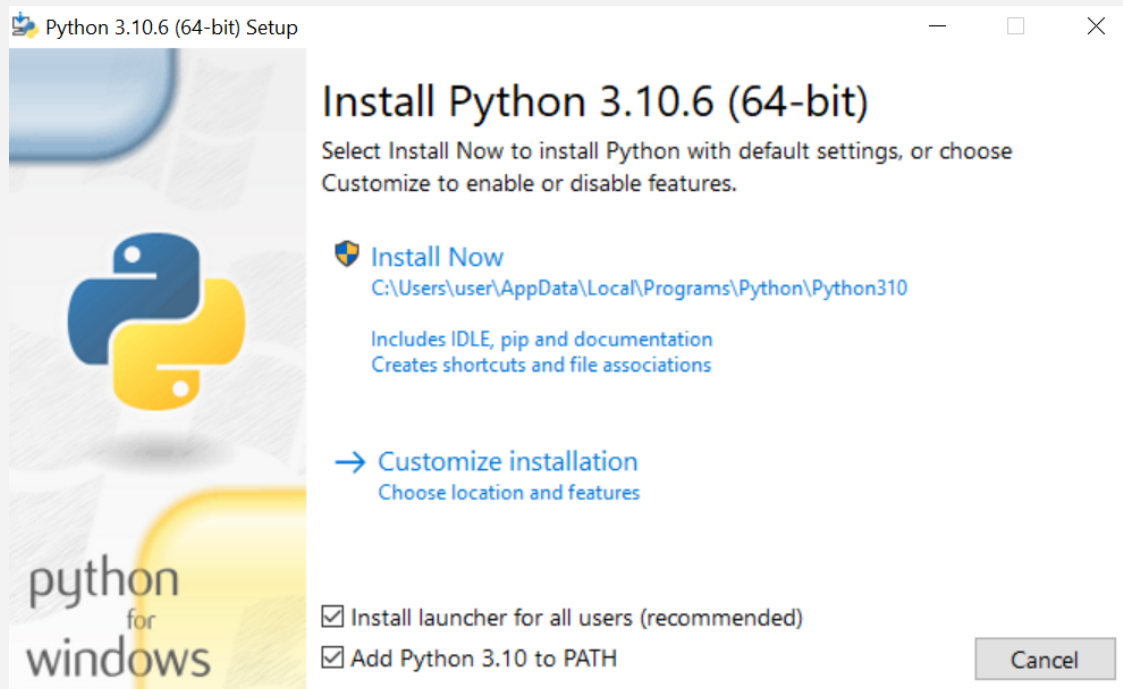
Python releases by version number:

Release version	Release date	
<b>Python 3.10.6</b>	Aug. 2, 2022	<a href="#">Download</a>
Python 3.10.5	June 6, 2022	<a href="#">Download</a>
Python 3.9.13	May 17, 2022	<a href="#">Download</a>

Choose **Python 3.10.6** and then choose **Windows installer (x-bit)** (depends whether you are on 32-bit or 64-bit).

Version	Operating System	Description	MD5
<a href="#">Gzipped source tarball</a>	Source release		a2d1
<a href="#">XZ compressed source tarball</a>	Source release		11d1
<a href="#">macOS 64-bit Intel-only installer</a>	macOS	for macOS 10.9 and later, deprecated	558c
<a href="#">macOS 64-bit universal2 installer</a>	macOS	for macOS 10.9 and later	ff07l
<a href="#">Windows embeddable package (32-bit)</a>	Windows		a54f
<a href="#">Windows embeddable package (64-bit)</a>	Windows		712c
<a href="#">Windows help file</a>	Windows		901c
<b><a href="#">Windows installer (32-bit)</a></b>	Windows		415c
<b><a href="#">Windows installer (64-bit)</a></b>	Windows	Recommended	a09c

The **executable file** should be inside your downloads folder now. Upon running it, you should see something like this:



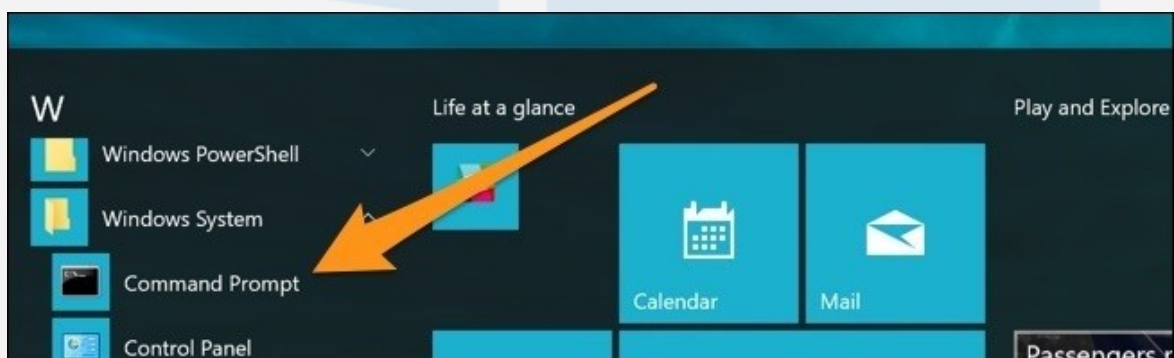
**Important:** You want to be sure to check the box that says **Add Python 3.10 to PATH** as shown to ensure that the interpreter will be placed in your execution path.

You do not need to install for all users if you have no admin access.

Click **Install Now** and it will complete in a few minutes. You should see a new program **IDLE** in your programs menu.

## Step 2: Installing the packages

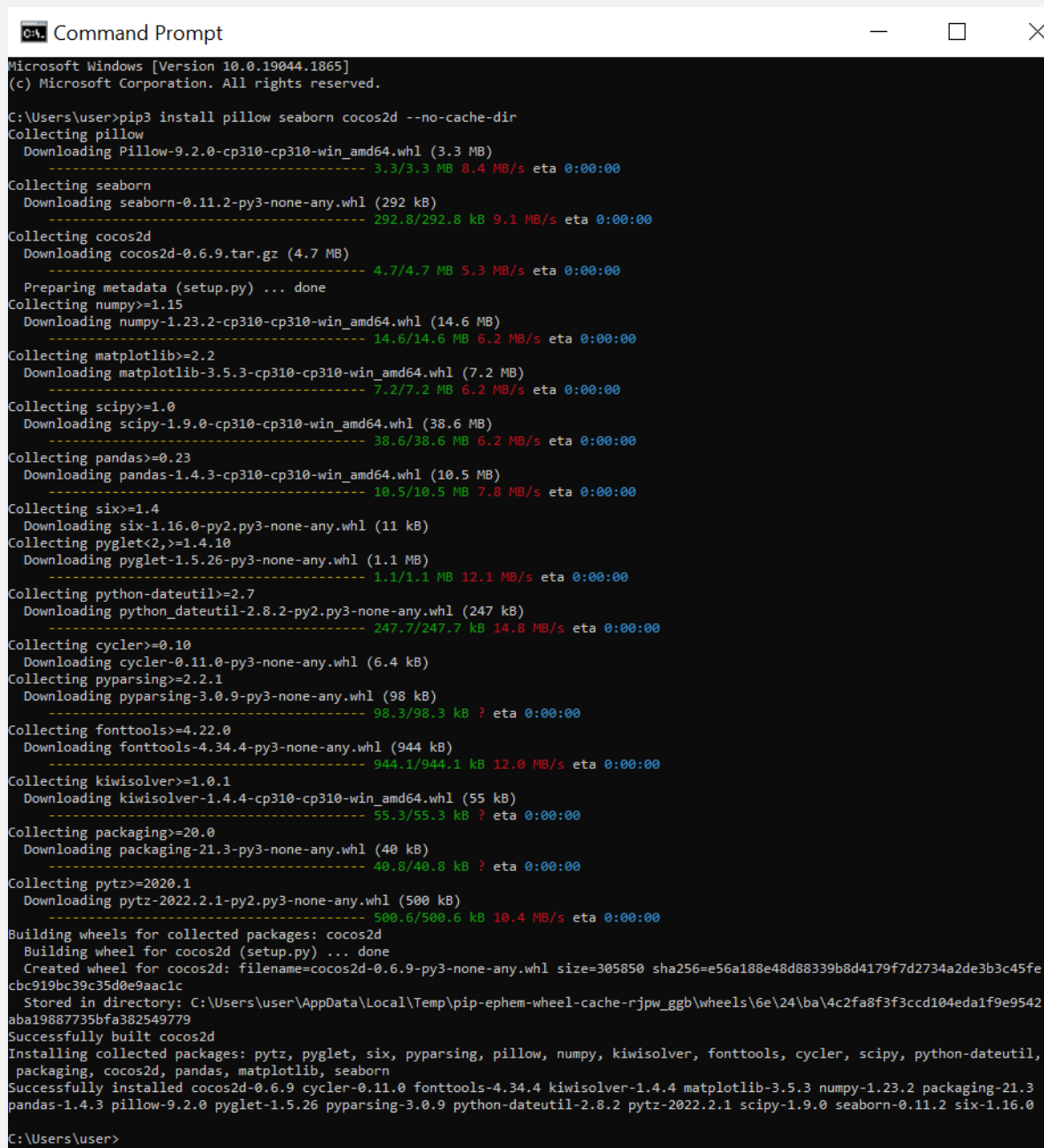
Open the **Command Prompt** from your Start menu.



Enter the following command as one line and press enter:

```
pip3 install pillow seaborn cocos2d --no-cache-dir
```

This will install the packages stated such as pillow and seaborn, plus all other packages that are not stated but they depend on. If successful, you should see a line saying “Successfully installed...” like this:



```

C:\Users\user>pip3 install pillow seaborn cocos2d --no-cache-dir
Collecting pillow
  Downloading Pillow-9.2.0-cp310-cp310-win_amd64.whl (3.3 MB)
    ----- 3.3/3.3 MB 8.4 MB/s eta 0:00:00
Collecting seaborn
  Downloading seaborn-0.11.2-py3-none-any.whl (292 kB)
    ----- 292.8/292.8 kB 9.1 MB/s eta 0:00:00
Collecting cocos2d
  Downloading cocos2d-0.6.9.tar.gz (4.7 MB)
    ----- 4.7/4.7 MB 5.3 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting numpy>=1.15
  Downloading numpy-1.23.2-cp310-cp310-win_amd64.whl (14.6 MB)
    ----- 14.6/14.6 MB 6.2 MB/s eta 0:00:00
Collecting matplotlib>=2.2
  Downloading matplotlib-3.5.3-cp310-cp310-win_amd64.whl (7.2 MB)
    ----- 7.2/7.2 MB 6.2 MB/s eta 0:00:00
Collecting scipy>=1.0
  Downloading scipy-1.9.0-cp310-cp310-win_amd64.whl (38.6 MB)
    ----- 38.6/38.6 MB 6.2 MB/s eta 0:00:00
Collecting pandas>=0.23
  Downloading pandas-1.4.3-cp310-cp310-win_amd64.whl (10.5 MB)
    ----- 10.5/10.5 MB 7.8 MB/s eta 0:00:00
Collecting six>=1.4
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Collecting pyglet<2,>=1.4.10
  Downloading pyglet-1.5.26-py3-none-any.whl (1.1 MB)
    ----- 1.1/1.1 MB 12.1 MB/s eta 0:00:00
Collecting python-dateutil>=2.7
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
    ----- 247.7/247.7 kB 14.8 MB/s eta 0:00:00
Collecting cyclr>=0.10
  Downloading cyclr-0.11.0-py3-none-any.whl (6.4 kB)
Collecting pyparsing>=2.2.1
  Downloading pyparsing-3.0.9-py3-none-any.whl (98 kB)
    ----- 98.3/98.3 kB ? eta 0:00:00
Collecting fonttools>=4.22.0
  Downloading fonttools-4.34.4-py3-none-any.whl (944 kB)
    ----- 944.1/944.1 kB 12.0 MB/s eta 0:00:00
Collecting kiwisolver>=1.0.1
  Downloading kiwisolver-1.4.4-cp310-cp310-win_amd64.whl (55 kB)
    ----- 55.3/55.3 kB ? eta 0:00:00
Collecting packaging>=20.0
  Downloading packaging-21.3-py3-none-any.whl (40 kB)
    ----- 40.8/40.8 kB ? eta 0:00:00
Collecting pytz>=2020.1
  Downloading pytz-2022.2.1-py2.py3-none-any.whl (500 kB)
    ----- 500.6/500.6 kB 10.4 MB/s eta 0:00:00
Building wheels for collected packages: cocos2d
  Building wheel for cocos2d (setup.py) ... done
  Created wheel for cocos2d: filename=cocos2d-0.6.9-py3-none-any.whl size=305850 sha256=e56a188e48d88339b8d4179f7d2734a2de3b3c45fecbc919bc39c35d0e9aac1c
    Stored in directory: C:\Users\user\AppData\Local\Temp\pip-ephem-wheel-cache-rjpw_ggb\wheels\6e\24\ba\4c2fa8f3f3ccd104eda1f9e9542aba19887735bfa382549779
Successfully built cocos2d
Installing collected packages: pytz, pyglet, six, pyparsing, pillow, numpy, kiwisolver, fonttools, cyclr, scipy, python-dateutil, packaging, cocos2d, pandas, matplotlib, seaborn
Successfully installed cocos2d-0.6.9 cyclr-0.11.0 fonttools-4.34.4 kiwisolver-1.4.4 matplotlib-3.5.3 numpy-1.23.2 packaging-21.3 pandas-1.4.3 pillow-9.2.0 pyglet-1.5.26 pyparsing-3.0.9 python-dateutil-2.8.2 pytz-2022.2.1 scipy-1.9.0 seaborn-0.11.2 six-1.16.0
C:\Users\user>

```

You may ignore the yellow text if any. However, if you see a bunch of red text, the packages might not be installed properly. Please take a look at the section next page. As a last resort, take a screenshot and check the Coursemology forums for assistance.

Otherwise, you may skip the extra troubleshooting part and proceed to *the next part*.

## More Troubleshooting

Here are some commonly found issues that might help you. **Please read them carefully as your problem might be included below.**

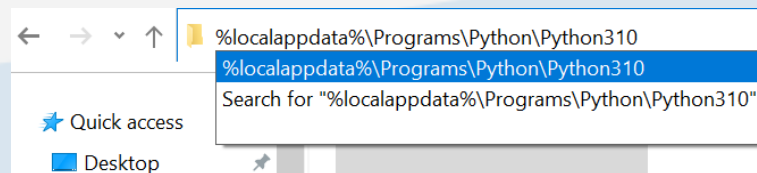
### 1. Checking if Python is installed

Your Python path should be in the form of

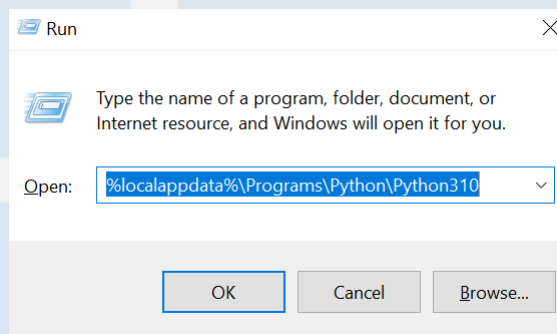
C:\Users\USERNAME\AppData\Local\Programs\Python\Python310

where USERNAME is your computer's username. Here's how to navigate you there.

- Method 1: Go to Windows Explorer, and type  
%localappdata%\Programs\Python\Python310 on the current directory.



- Method 2: Key in Win+R, and then type  
%localappdata%\Programs\Python\Python310.



### 2. pip or pip3 is not recognized as an internal or external command

If this issue happens, try interchanging pip3 with pip or pip3.10.

If the error still persists, you can try prepending the pip commands with either python -m or py -m. For example,

```
python -m pip3 install pillow seaborn cocos2d --no-cache-dir
```

For reference, please take a look at [this Medium article](#).

### 3. Checking if pip is installed

With your command prompt, try running

```
pip3 --version or pip --version
```

If you have more than one version of Python, try running pip3.10 --version.

If either one of these commands produce an output similar to the one below, then you are good to go.

```
C:\Users\user>pip --version
pip 22.2.2 from C:\Users\user\AppData\Local\Programs\Python\Python310\lib\site-packages\pip (python 3.10)
```

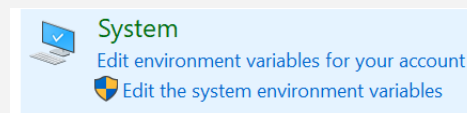
If all commands fail, try prepending these commands with either python -m or py -m.

Alternatively you can run either `python -m ensurepip` or `py -m ensurepip`. The output should be something similar to this.

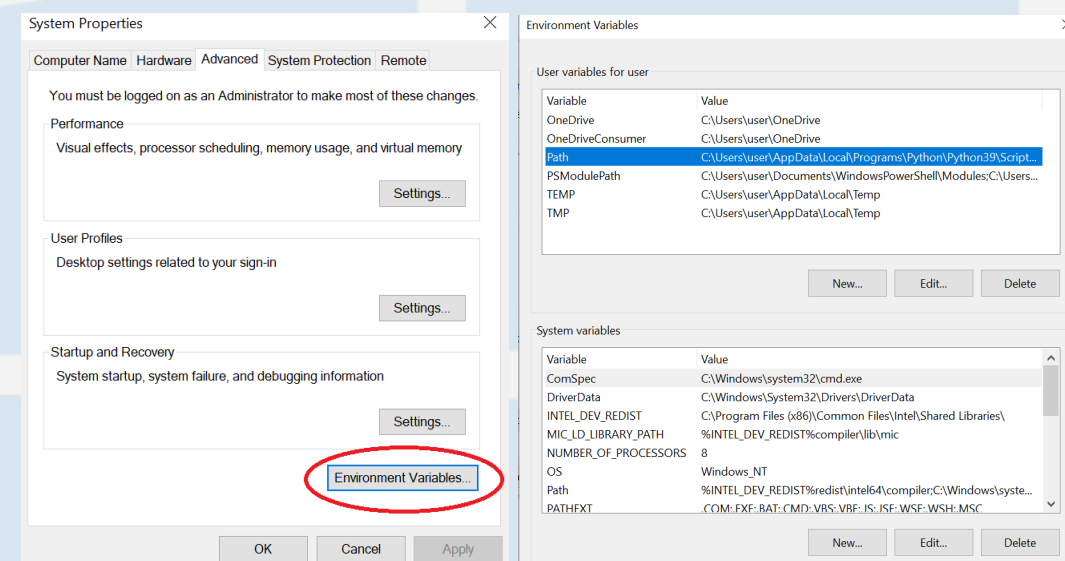
```
C:\Users\user>py -m ensurepip
Looking in links: c:\Users\user\AppData\Local\Temp\tmpfm1kxnmz
Requirement already satisfied: setuptools in c:\users\user\appdata\local\programs\python\python310\lib\site-packages (63.2.0)
Requirement already satisfied: pip in c:\users\user\appdata\local\programs\python\python310\lib\site-packages (22.2.2)
```

#### 4. Adding Python to PATH manually

Go to Control Panel, and find this option by using the search bar.



Open Edit the system environment variables, then Environment Variables. For each step, your display should be like the ones below.



Select Path from User variables for user, then Edit. You can add Python310's path by clicking New. Then, key in

%localappdata%\Programs\Python\Python310

and

%localappdata%\Programs\Python\Python310\Scripts

To check if Python's path is already added to the PATH variable, go back to the command prompt and type `echo %PATH%`. It will give you a list of directories added to the PATH variable. Make sure the copied path is already there.

#### 5. Checking if the packages are installed

One of these commands should work on your command prompt.

- `pip freeze`
- `pip3 freeze`
- `pip3.10 freeze`

What these commands do is checking all the Python packages you've installed so far. The result should be a list with a format of

```
<package-name>==<package-version>
```

For example, `scipy==1.7.3`. With this, you can see whether the required packages are installed correctly.

Alternatively, you can also use `pip list` instead of `pip freeze`.

## 6. Upgrading pip

During installing the packages, you might receive a warning in yellow that pip must be upgraded. This is not necessary as long as you can work on with the Python packages normally. However, should you need it, then run this line on the command prompt.

```
pip --install upgrade pip
```

If this fails, try prepending this command with either `python -m` or `py -m`.

## 7. Subprocess error when installing packages

As of now, this problem arises due to the incorrect Python installer chosen (between 32-bit and 64-bit). For example, your laptop is Windows 64-bit but the 32-bit Python installer is chosen. As a result, you might see something like this when installing the packages.

```
note: This error originates from a subprocess, and is likely not a problem with pip.
error: subprocess-exited-with-error

x Getting requirements to build wheel did not run successfully.
  exit code: 1
  -> See above for output.

note: This error originates from a subprocess, and is likely not a problem with pip.
```

The solution is to simply switch the installer from 32-bit to 64-bit or vice versa and uninstall the previous one to avoid conflicts with the new installer.

To uninstall Python, see the section **Uninstalling other versions of Python**.

## 8. Issues with wheels during installation

This might occur to some of you as the wheel package is not installed by default. Some issues that fall into this category include getting one of these messages when trying to install the packages.

- "Could not build wheels for <package name> since package wheel is not installed"

This is an error and will stop the installation process before it finishes properly.

- "Using legacy 'setup.py install' for <package name>, since package 'wheel' is not installed"

This is not an error but will very likely slow down the installation process.

The solution is to simply modify the installation command with the wheel package installed first.

For example,

```
pip3 install wheel pillow seaborn cocos2d --no-cache-dir
```

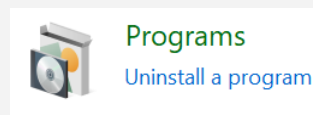
Same as the other cases, if this fails you might need to prepend the command with `python -m` or `py -m`, and/or changing `pip3` to either `pip` or `pip3.10`.



## 9. Uninstalling other versions of Python

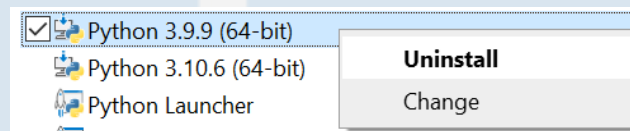
**Note:** This process is optional. However, one might easily confuse between different instances of Python that co-exist inside the computer. An example issue would be unable to import the packages on IDLE although the installation process ran smoothly, only to find out that the IDLE version does not match.

Go to Control Panel and look for this option.



Then, simply find the Python program that is to be uninstalled, click right and uninstall.

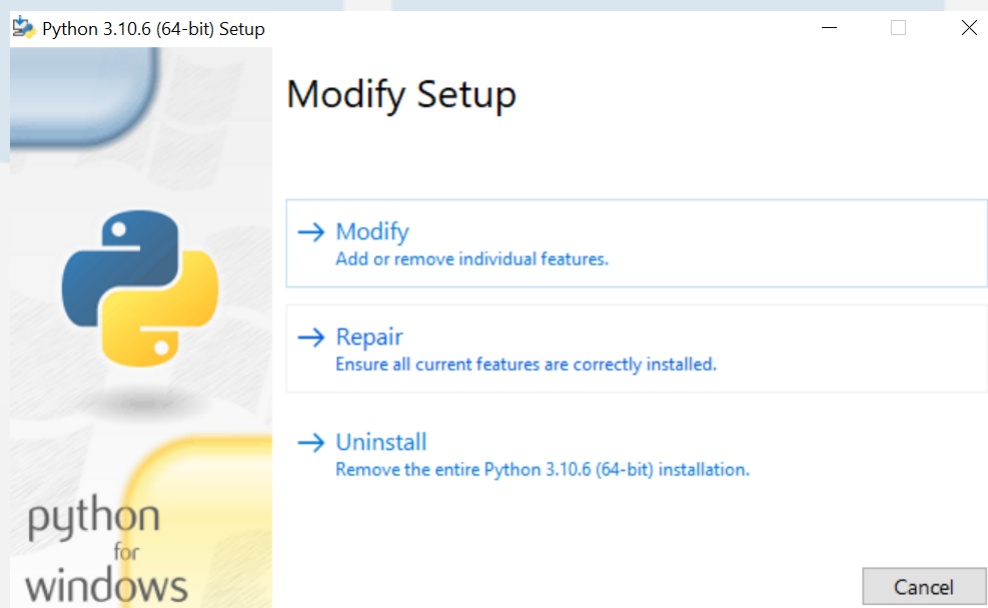
For example, the picture below shows how to uninstall Python 3.9 since there are both Python 3.9 and 3.10.



## 10. Reinstalling Python

Reinstalling Python might be needed if you have two or more instances of Python (e.g. Anaconda, Microsoft Store) and many errors appear during package installation due to certain clashes, which forces you to uninstall/delete them. Here's what to do **suppose you have to reinstall Python 3.10.6 again.**

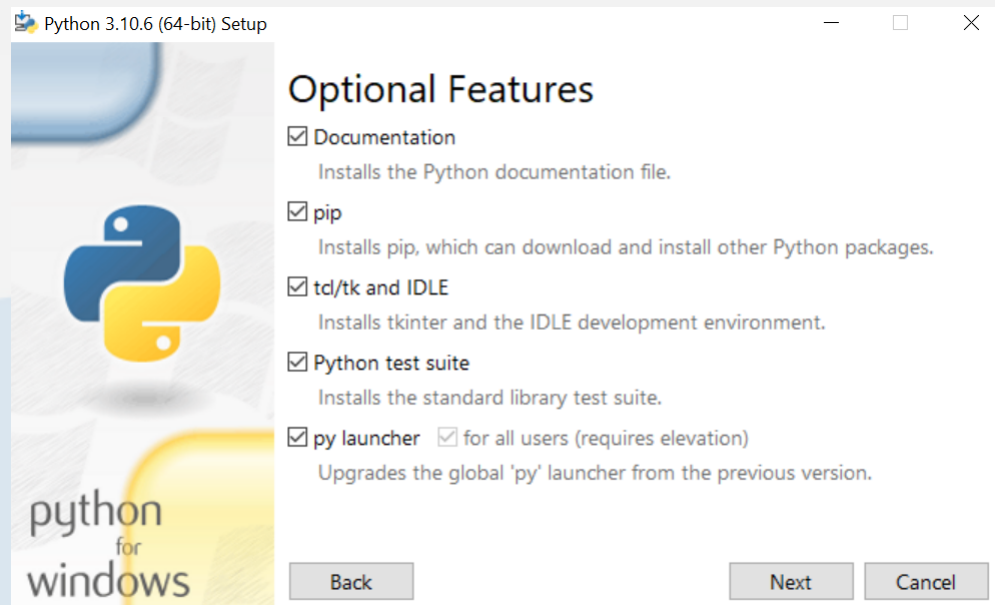
- Reopen the **executable file** that you obtained from Python's website. This should be the display the moment you open it.



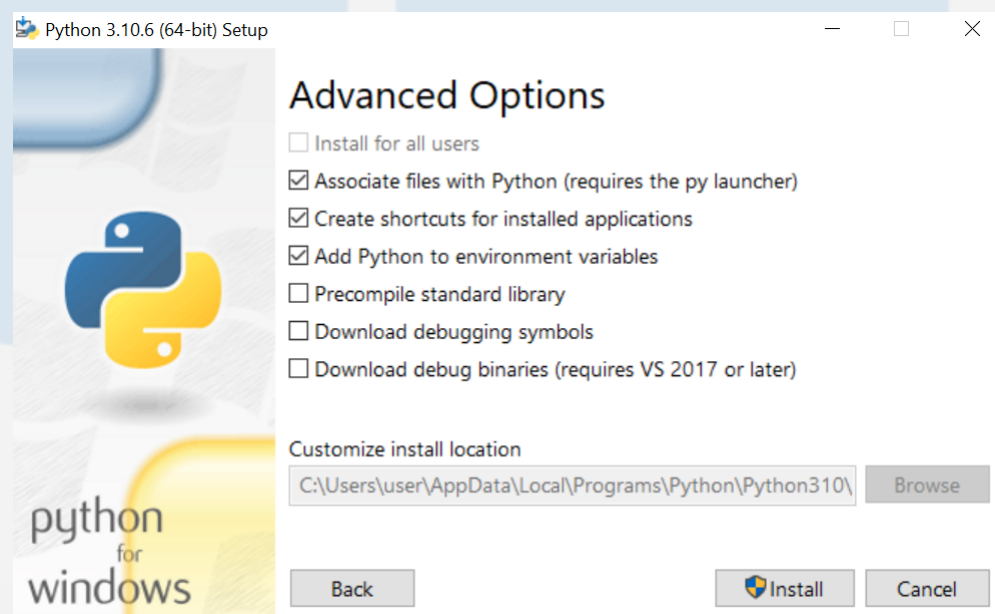
Note that you can also uninstall Python as seen in the last option above.



- Click Modify and check off **everything**. The last option is unchecked by default, so ensure that this is also checked off.



- Click Next, and tick **the first three tickable options**. It is optional to tick the last three. Finally, click Install.



Now that you have finished self-troubleshooting your installation, you may proceed to *the next part*.

**Mac Users:****Step 1: Install Python for Mac**

Download Python 3.10.6 **macOS 64-bit universal2 installer** from <https://www.python.org/downloads> and run the pkg installer.

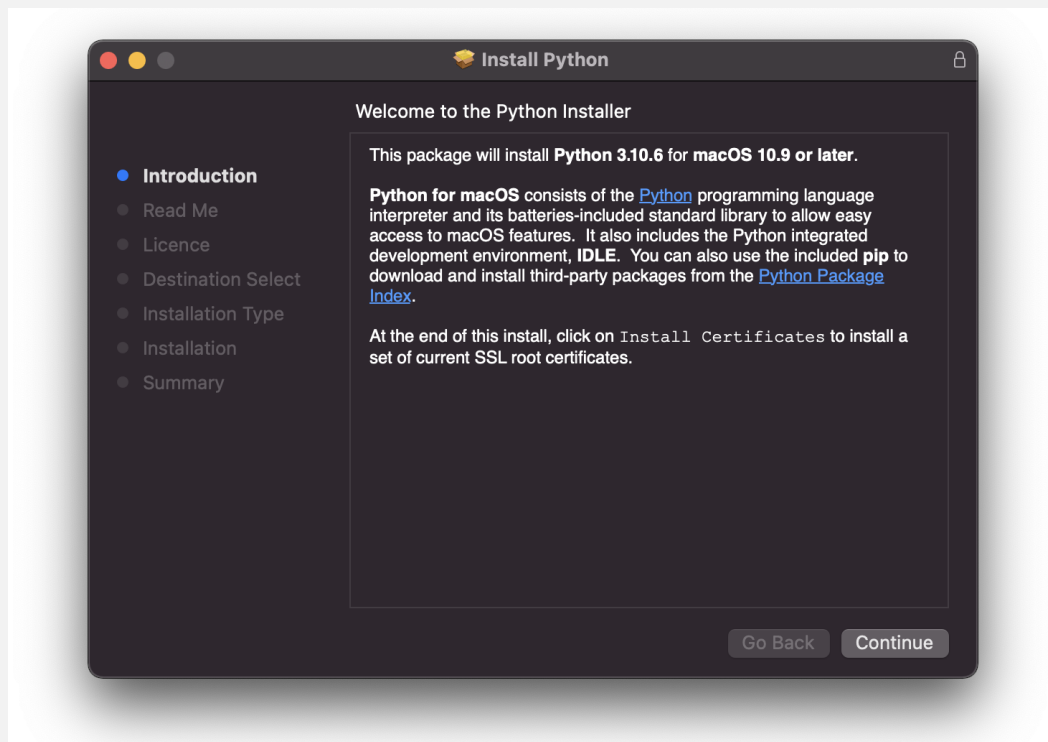
Looking for a specific release?

Python releases by version number:

Release version	Release date	
<b>Python 3.10.6</b>	Aug. 2, 2022	<a href="#">Download</a>
Python 3.10.5	June 6, 2022	<a href="#">Download</a>
Python 3.9.13	May 17, 2022	<a href="#">Download</a>

Version	Operating System	Description	MD5 Sum
<a href="#">Gzipped source tarball</a>	Source release		a2da2a456c0
<a href="#">XZ compressed source tarball</a>	Source release		11d12076311
<a href="#">macOS 64-bit Intel-only installer</a>	macOS	for macOS 10.9 and later, deprecated	558d424cd54
<b><a href="#">macOS 64-bit universal2 installer</a></b>	macOS	for macOS 10.9 and later	ff07b39be8e
<a href="#">Windows embeddable package (32 bit)</a>	Windows		a54f24cee83

Once installation is completed, you should see that **IDLE** is available from your finder.



## Step 2: Installing the packages

Run the following commands in your terminal. (You can find the terminal by clicking on Finder on the dock, Go > Utilities > Terminal.)

```
pip3 install pillow seaborn cocos2d --no-cache-dir
```

If you see a bunch of red text, the packages might not be installed properly. Please take a look at the section next page. As a last resort, take a screenshot and check the Coursemology forums for assistance.

Otherwise, you may skip the extra troubleshooting part and proceed to *the next part*.

## More Troubleshooting

Here are some commonly found issues that might help you. **Please read them carefully as your problem might be included below.**

### 1. Some packages cannot compile correctly

Attempting to run the installation command might sometimes result in compilation errors for M1 devices. Here are some possible errors that fall under this category:

- Incompatible architecture

```
Traceback (most recent call last):
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/numpy/core/__init__.py", line 23, in <module>
    from . import multiarray
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/numpy/core/multiarray.py", line 10, in <module>
    from . import overrides
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/numpy/core/overrides.py", line 6, in <module>
    from numpy.core._multiarray_umath import (
ImportError: dlopen(/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/numpy/core/_multiarray_umath.cpython-310-darwin.so, 0x0002): tried: '/Applications/Python 3.10/IDLE.app/Contents/Frameworks/_multiarray_umath.cpython-310-darwin.so' (no such file), '/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/numpy/core/_multiarray_umath.cpython-310-darwin.so' (no such file), '/usr/lib/_multiarray_umath.cpython-310-darwin.so' (mach-o file, but is an incompatible architecture (have 'arm64', need 'x86_64'))

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/Users/nicholas/Downloads/mission00-template.py", line 304, in <module>
    from numpy import *
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/numpy/__init__.py", line 140, in <module>
    from . import core
  File "/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/numpy/core/__init__.py", line 49, in <module>
    raise ImportError(msg)
ImportError:

IMPORTANT: PLEASE READ THIS FOR ADVICE ON HOW TO SOLVE THIS ISSUE!

Importing the numpy C-extensions failed. This error can happen for
many reasons, often due to issues with your setup or how NumPy was
installed.

We have compiled some common reasons and troubleshooting tips at:
    https://numpy.org/devdocs/user/troubleshooting-importerror.html

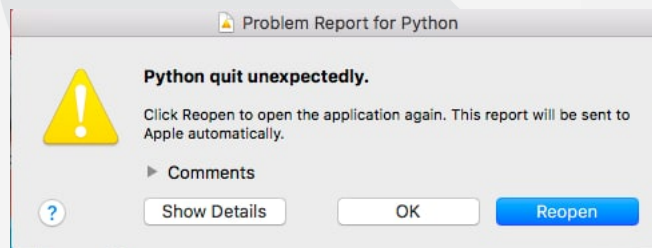
Please note and check the following:

  * The Python version is: Python3.10 from "/Library/Frameworks/Python.framework/Versions/3.10/bin/python3.10"
  * The NumPy version is: "1.23.1"

and make sure that they are the versions you expect.
Please carefully study the documentation linked above for further help.

Original error was: dlopen(/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/numpy/core/_multiarray_umath.cpython-310-darwin.so, 0x0002): tried: '/Applications/Python 3.10/IDLE.app/Contents/Frameworks/_multiarray_umath.cpython-310-darwin.so' (no such file), '/Library/Frameworks/Python.framework/Versions/3.10/lib/python3.10/site-packages/numpy/core/_multiarray_umath.cpython-310-darwin.so' (no such file), '/usr/lib/_multiarray_umath.cpython-310-darwin.so' (mach-o file, but is an incompatible architecture (have 'arm64', need 'x86_64'))
```

- Python quitting unexpectedly



- Subprocess error while installing dependencies

```
Installing build dependencies ... done
Getting requirements to build wheel ... error
error: subprocess-exited-with-error

× Getting requirements to build wheel did not run successfully.
  exit code: 1
  [61 lines of output]
  The Meson build system
  Version: 0.62.2
  Source dir: /private/var/folders/7j/kvhb5m5j0q94991wlx3p8hh80000gn/T/pip-i
  stall-4y038pnj/scipy_8622e6cb8d3e499d92fb5d21dbbc11f7
  Build dir: /private/var/folders/7j/kvhb5m5j0q94991wlx3p8hh80000gn/T/pip-in
  stall-4y038pnj/scipy_8622e6cb8d3e499d92fb5d21dbbc11f7/.mesonpy-9y0lvbx/build
  Build type: native build
  Project name: SciPy
  Project version: 1.9.0
  C compiler for the host machine: cc (clang 12.0.5 "Apple clang version 12.
  0.5 (clang-1205.0.22.11)")
  C linker for the host machine: cc ld64 650.9
  C++ compiler for the host machine: c++ (clang 12.0.5 "Apple clang version
  12.0.5 (clang-1205.0.22.11)")
  C++ linker for the host machine: c++ ld64 650.9
```

Please follow these steps for an alternative installation method.

- (a) Install Homebrew by running

```
/bin/bash -c "$(curl -fsSL
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

**as one line.**

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

- (b) You will be prompted to input your sudo password. It won't be shown when you're typing it, so make sure you are keying in the correct one.

```
==> Checking for `sudo` access (which may request your password)...
Password:
```

- (c) Read and follow any additional instructions to run on the Terminal window.

- (d) Run

```
brew install openblas gfortran
```

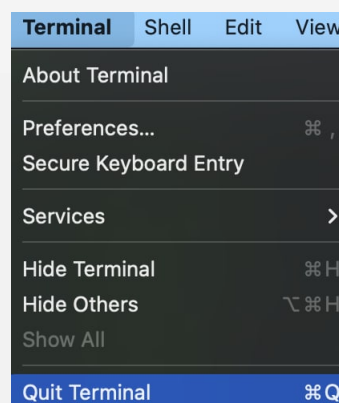
This should install openblas and gfortran from Homebrew.

- (e) Run

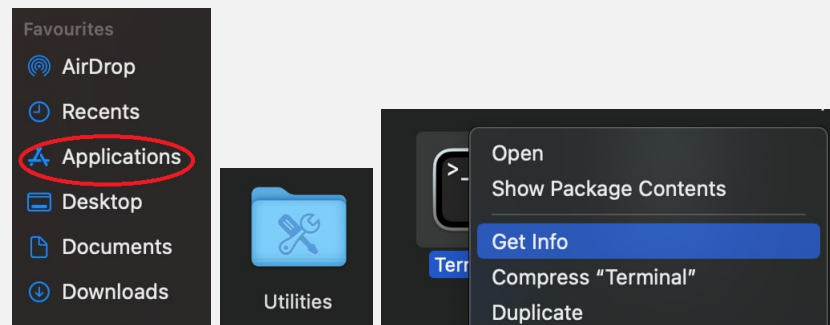
```
pip3.10 freeze | xargs pip3.10 uninstall -y
```

This will uninstall your previously installed Python packages because we will use a new one.

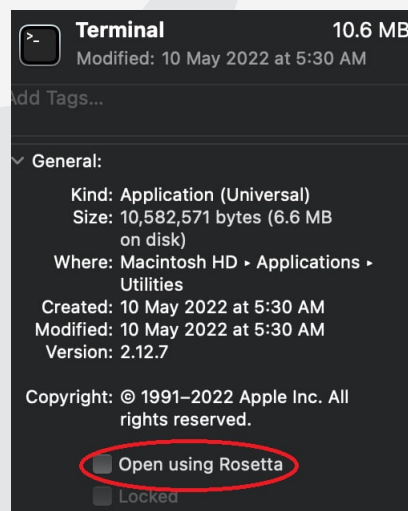
- (f) If you receive a prompt to upgrade your pip version, please do so by running the command shown on the screen.
- (g) Quit the Terminal application. **Quitting Terminal is different than merely closing the Terminal window. Make sure you don't have any Terminal windows open.** You can do this by doing Command + Q or following the picture below.



- (h) Locate Terminal in your Finder (User -> Applications -> Utilities). These pictures will guide you as you navigate yourself through the files.



- (i) Select Terminal and press Command + I **OR** click right + select Get Info as shown above. **Do not have any Terminal windows open yet.**
- (j) **Check** the “Open with Rosetta” option.



(k) Reopen Terminal.

(l) Run

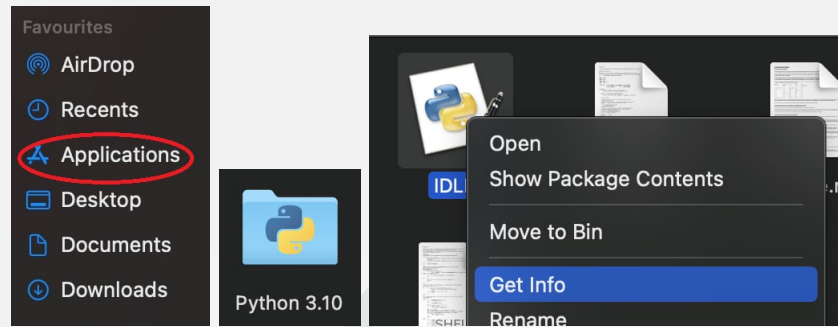
```
export OPENBLAS=/opt/homebrew/opt/openblas/lib/
```

(m) Run

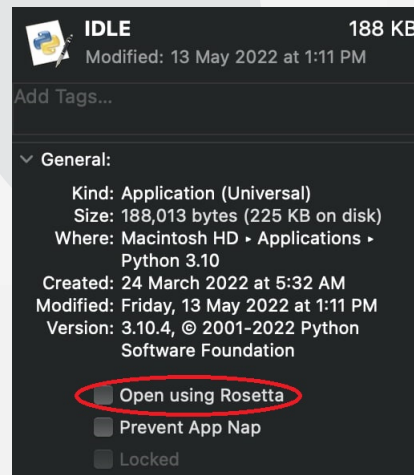
```
pip3.10 install pybind11 cython pythran pillow seaborn cocos2d
--no-cache-dir
```

**as one line.** Pay attention to the spellings, spacings, and the dashes.

- (n) Locate Terminal in your Finder (User -> Applications -> Utilities), just like before. Select Terminal and press Command + I **OR** click right + select Get Info.
- (o) This time, **uncheck** the “Open with Rosetta” option.
- (p) Quit the IDLE application if it is still open.



- (q) Locate IDLE in your Finder (User -> Applications -> Python3.10). Select IDLE and press Command + I **OR** click right + select Get Info as shown above.
- (r) **Check** the “Open with Rosetta” option.



- (s) Try and run the import statements from Mission 0 again.

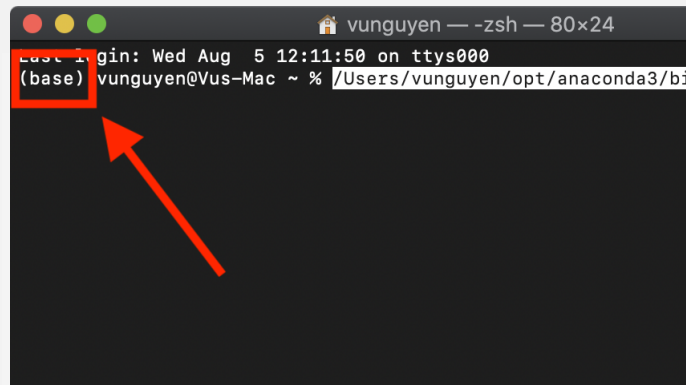


## 2. Packages installed correctly but unable to import or

### Previously installed Anaconda/Conda

Sometimes due to multiple versions of Python installation present, IDLE will not import packages from the correct version. This issue is especially pertinent to users who have previously installed Anaconda.

If you are unsure about whether you have Anaconda installed, open a new Terminal window and look out for (base) on the left.



To remove Anaconda, follow the guide at <https://docs.anaconda.com/anaconda/install/uninstall/>.

**Beware of the `rm -rf` command. Incorrect usage might result in PERMANENT and IRRECOVERABLE loss of data.**

Additionally, open `.zshrc` or `.bashrc` and remove all traces of Anaconda. Check if the removal process is under control by closing all Terminal instances and reopening them.

However, for any reason you are/want/like to keep Anaconda around, you do not have to remove them. Simply run `deactivate` to exit from the Anaconda base environment every time you launch Terminal. You can then follow the installation instructions in Mission 0 again.

## 3. Bouncing rocket icon appears in dock when importing packages

This is normal, you can ignore it.

This might happen when one tries to import `cocos` which will interact with the Python graphical user interface (GUI).

Source: [Stack Overflow](#)

Now that you have finished self-troubleshooting your installation, you may proceed to the next part.

### Linux Users:

If you are *really* on Linux, then you are 1337 and will know how to install Python :) Just kidding. You can approach the TA for help or any tutor familiar with Linux.

## Editing Python Files

The default behaviour of double clicking on a Python file **executes** the content of the Python file. You should see the a command line window briefly open, and close when Python finishes executing the file.

In order to **edit** a Python file, you will need to use the **IDLE** application.

- Windows Users: Right click on the Python file > **Edit with IDLE**
- Mac Users: Right click on the Python file > **Open With > IDLE**

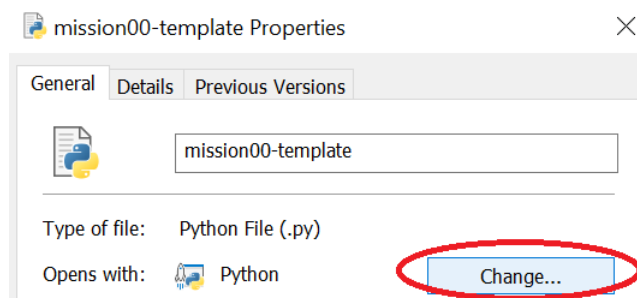
The content of the Python file should now appear in **IDLE**. You can then make changes to the file, and execute it.

To **execute** the Python file, go to **Run > Run Module**. The output of your Python file should then appear in the Shell Window.

**For Windows users, if these instructions still fail, do read the guide below.** Otherwise, you are also recommended to try it out.

The following guide enables you to open IDLE instead of executing a Python file upon double-clicking the file. Here are the steps to do so.

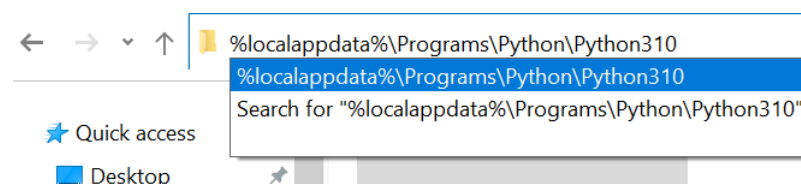
1. Right click any Python file, such as mission00-template.py.
2. Try one of the two alternatives:
  - Select “Open With”, then click on “Choose another app”.
  - Select “Properties”, and then on the “Opens with:” section, select “Change”.



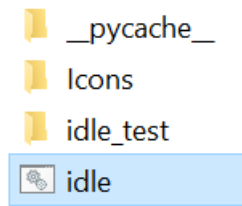
3. Scroll down all the way to find the “Look another app on the PC” option. You might need to click on the “More apps” option to be able to see more options.
4. Type

%localappdata%\Programs\Python\Python310\Lib\idlelib

on the address bar, just like this.



5. There should a file named `idle.bat` (see the highlighted file in the picture below). Click on it so that the files can be opened in IDLE automatically.



6. If applicable, do click “Apply” to save your changes.

7. You should be able to open IDLE by double-clicking the file!

If even doing the above guide doesn’t work, do reinstall Python again by following the steps stated at the Windows troubleshooting section: **Reinstalling Python**.

## Part 2: The Task

Consider the following Python expressions. Your job is to predict the output when each expression is evaluated in IDLE. The first three has been done for you as an example.

Before checking your answers with IDLE, write down briefly your guess of what the interpreter would **display** when the expressions are evaluated **sequentially** as listed. If you do not expect any output, you may write “Nothing” and if you expect an error, you may write “Error”.

Now, run the code by removing the `#` in the front of the respective lines in the template file. You should leave error-causing lines in their commented out state by adding `#` to the front of such lines to allow IDLE to skip processing them. (Or by selecting the area that you would like to comment out and then pressing IDLE’s hot-key `alt+3` for Windows, `control+3` for MacOS)

```
4 Welcome to CS1010S!
5 This line will cause an error.
6 But, the line below will not!
7 print(42)
8
9 ##This is commented. Let's drag the
10 ##first three lines above!
```

Before commenting

```
4 ##Welcome to CS1010S!
5 ##This line will cause an error.
6 ##But, the line below will not!
7 print(42)
8
9 ##This is commented. Let's drag the
10 ##first three lines above!
```

After commenting

Commenting with IDLE’s hot-key

After you have checked your answers, if any expression has evaluated differently from your expected answer, write down what it evaluated to **below** your expected answer. **Please note that you will be graded only on your final answer (If you have no corrections to make, your expected answer will be your final answer).**

However, if any expression would generate an error message after running, please **specify the type of error** (such as `TypeError`) together with the **error message** in your final answer (see Example 3). You do not need to include the full error output. The required answer is usually only the *last line* of the error output.

Please use the template file provided in **mission00-template.py** instead of copying the code snippet from this PDF file.

**Note:** Lines that begin with a `#` are comments (text that do not affect the Python execution). To execute a particular expression, ensure it does not begin with a `#`.

# Example 1:

# My expected result is zero but upon printing the result is 0.

```
print(0)
```

# expected answer: zero

# final answer: 0

# Example 2:

# My expected result is 1 which turns out to be correct after I run the  
# print statements, so I don't need to do anything for the final answer.

```
print(1)
```

# expected answer: 1

# final answer:

```
# Example 3:
# This print statement results in an error,
# so I need to comment it and put the specific error as shown.
#print(a)
# expected answer: NameError: name 'a' is not defined
# final answer:

# Now, it's your turn :)

print(42)
# expected answer:
# final answer:

print(0000)
# expected answer:
# final answer:

print("the force!")
# expected answer:
# final answer:

print("Hello World")
# expected answer:
# final answer:

print(6 * 9)
# expected answer:
# final answer:

print(2 + 3)
# expected answer:
# final answer:

print(2 ** 4)
# expected answer:
# final answer:

print(2.1**2.0)
# expected answer:
# final answer:

print(15 > 9.7)
# expected answer:
# final answer:

print((5 + 3) ** (5 - 3))
# expected answer:
# final answer:
```

```
print(--4)
# expected answer:
# final answer:

print(1 / 2)
# expected answer:
# final answer:

print(1 / 3)
# expected answer:
# final answer:

print(1 / 0)
# expected answer:
# final answer:

print(7 / 3 == 7 / 3.0)
# expected answer:
# final answer:

print(3 * 6 == 6.0 * 3.0)
# expected answer:
# final answer:

print(11 % 3)
# expected answer:
# final answer:

print(2 > 5 or (1 < 2 and 9 >= 11))
# expected answer:
# final answer:

print(3 > 4 or (2 < 3 and 9 > 10))
# expected answer:
# final answer:

print("2" + "3")
# expected answer:
# final answer:

print("2" + "3" == "5")
# expected answer:
# final answer:

print("2" <= "5")
# expected answer:
# final answer:
```

```
print("2 + 3")
# expected answer:
# final answer:

print("May the force" + " be " + "with you")
# expected answer:
# final answer:

print("force"*2)
# expected answer:
# final answer:

print('daw' in 'padawan')
# expected answer:
# final answer:

a, b = 3, 4 # Do not comment or remove this line

print(a)
# expected answer:
# final answer:

print(b)
# expected answer:
# final answer:

a, b = b, a # Do not comment or remove this line

print(a)
# expected answer:
# final answer:

print(b)
# expected answer:
# final answer:

print(red == 44)
# expected answer:
# final answer:

red, green = 44, 43 # Do not comment or remove this line

print(red == 44)
# expected answer:
# final answer:

print(red = 44)
# expected answer:
# final answer:
```



```
print("red is 1") if red == 1 else print("red is not 1")
# expected answer:
# final answer:

print(red - green)
# expected answer:
# final answer:

purple = red + green # Do not comment or remove this line

print("purple")
# expected answer:
# final answer:

print("purple"[7])
# expected answer:
# final answer:

print(red + green != purple + purple / purple - red % green)
# expected answer:
# final answer:

print(green > red)
# expected answer:
# final answer:

print("green bigger") if green > red else print("red equal or bigger")
# expected answer:
# final answer:

print(green + 5)
# expected answer:
# final answer:

print(round(3.8))
# expected answer:
# final answer:

print(int(3.8))
# expected answer:
# final answer:

print(int("3.8"))
# expected answer:
# final answer:
```

```

# Run these lines of code before proceeding to the next question!
# Do not comment these lines or remove it from your submission!
def f(n):
    if n == 1: return 1
    return n + f(n - 1)

print(f(4))
# expected answer:
# final answer:

print(f(f(2)))
# expected answer:
# final answer:

print(f(0))
# expected answer:
# final answer:

d = {1: 2} # Do not comment or remove this line

print(d[1])
# expected answer:
# final answer:

print(d[2])
# expected answer:
# final answer:

d[2] = "apple" # Do not comment or remove this line

print(d[2])
# expected answer:
# final answer:

#####
# The following 7 questions are to ensure that you have #
# installed all the packages correctly: #
# - PILLOW          - matplotlib          - scipy #
# - seaborn         - numpy              - pygamelet #
# - cocos #
# #
# Just uncomment the line "from <package> import *", #
# run the line, and observe the output. #
# #
# If there is no output, the packages have been installed #
# correctly, so answer "Nothing" to let us know that it's #
# working properly. Otherwise, if you see some errors, do #
# refer to the troubleshooting guide in the PDF file. #
#####

```

```
from PIL import *
# expected answer:
# final answer:

from matplotlib import *
# expected answer:
# final answer:

from scipy import *
# expected answer:
# final answer:

from seaborn import *
# expected answer:
# final answer:

from numpy import *
# expected answer:
# final answer:

from pygamelet import *
# expected answer:
# final answer:

from cocos import *
# expected answer:
# final answer:
```

To submit your work to Coursemology, complete **mission00-template.py** and **copy all the contents from the file** into the box on the mission page, and click “Save Draft.” You can continue to make changes to your submission if you wish.

If you want to check whether your submission passes the test cases provided, click “Run Code”. The autograder provided will evaluate your answer and tell whether you fail some test cases or not. **By doing this, you can still continue to amend your submission.**

A teal rectangular button with the text "RESET ANSWER" in white capital letters.A pink rectangular button with the text "RUN CODE" in white capital letters.A teal rectangular button with the text "SAVE DRAFT" in white capital letters.A pink rectangular button with the text "FINALISE SUBMISSION" in white capital letters.

Once you are satisfied with your solution, click “Finalise Submission.” **Note that once your submission is finalised, it is considered to be submitted and cannot be changed.** If you need to undo this action, you will have to email your tutor or the lecturer. You will not be allowed to undo your *finalised* submission, or a penalty might be imposed. **Please do not finalise your submission until you are sure that you want to submit your solutions for grading.**