Lab 01: Introduction

Goals for this lab:

1. Download and install Python
2. Create a .py file
3. Write a simple program that outputs text to the screen
4. Write a simple program that takes in user input and outputs text to the screen
5. Write and execute simple Python programs using the given instructions

General Guidelines:

- Use meaningful variable names
- Use appropriate indentation
- Use comments, especially for the header, variables, and blocks of code.

Example Header:

```
# Author: Your name
# Date: Today's date
# Description: A small description in your own words that describes what
the program does. Any additional information required for the program to
execute should also be provided.
```

1. Organizing Files

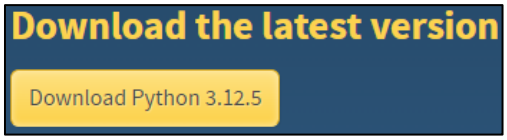
Much of the work in this course, and in programming in general, builds upon itself. Previous programs that you write will oftentimes be useful either for reference or for borrowing pieces of functionality from. Therefore, organizing your files for easy access and lookup will help you later, and the best time to start this organization is at the beginning! It is recommended that you decide on a location on your computer where you will be storing all of the work for this course (labs, in-class coding, homeworks, projects) and create a folder or directory at that location, perhaps named `CS1`. Since each assignment will most likely have multiple parts, it is also recommended that you organize your individual assignments into their own folders/directories. For example, you could create a folder/directory, inside your `CS1` folder/directory, named `Lab1` to store the files you create for this lab. Ultimately, the organization and naming conventions are up to you, so set them up in a way that best helps you find your work in the future.

2. Download and Install Python

In order to develop programs using Python, you will first need to download and install the Python interpreter and its modules. Python is provided for free on the Python.org website, and to download Python, you will first need to go to:

Python Download

This webpage should automatically detect your operating system, and thus you only need to click the large yellow button to start the download.



This will download everything you need for Python. The download should provide a single installation file that you can use to install everything you need. **Note that Python is updated regularly, and the yellow button always provides the most up-to-date version.**

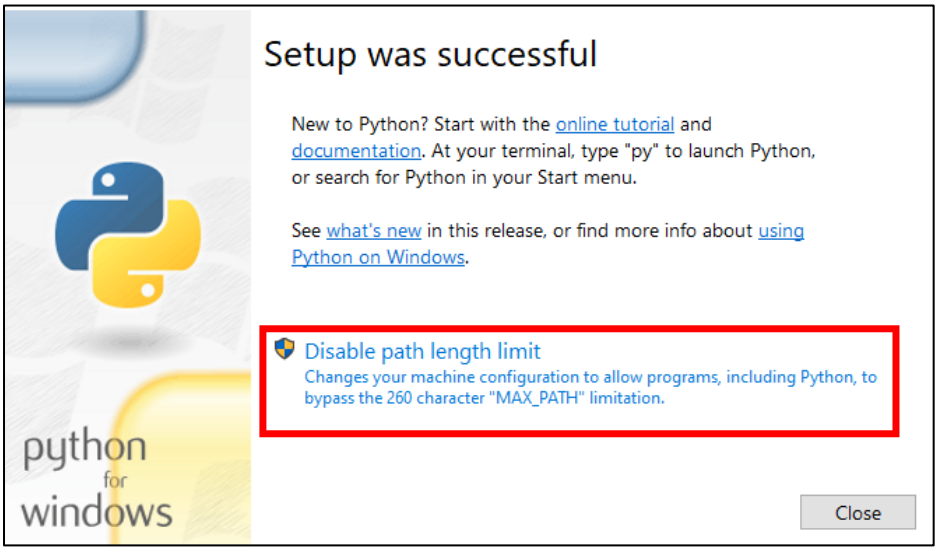
Once the installer has been downloaded, read the **Notes for Windows/MacOS Users** below, and then execute the installer and follow the prompts. Once installation has completed, you are ready to create your first Python program!

Notes for Windows Users:

There are two extra steps you will need to perform to make sure that Python runs on your system as smoothly as possible:

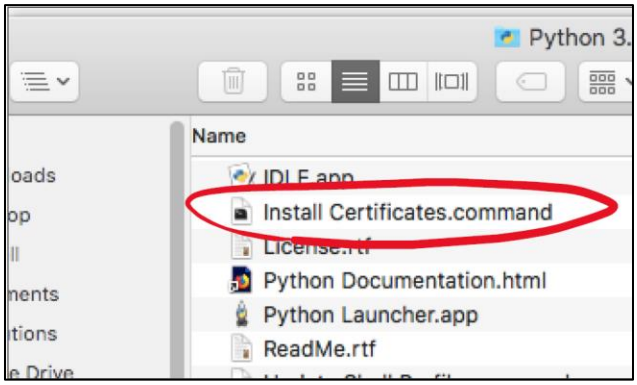
1. When you start the installer, there will be some check boxes at the bottom of the screen. Be sure to check the `Add python.exe to PATH`. This makes sure that Python can be executed from anywhere on your computer using the command line.

2. In order to allow Python to use files that may have longer names and be in deeper folder/directories, it is a good idea to `Disable path length limit` by clicking that button once the installation has finished.



Note for MacOS Users:

1. Once installation has completed, be sure to run the `Install Certificates.command` program in your Python directory/folder.



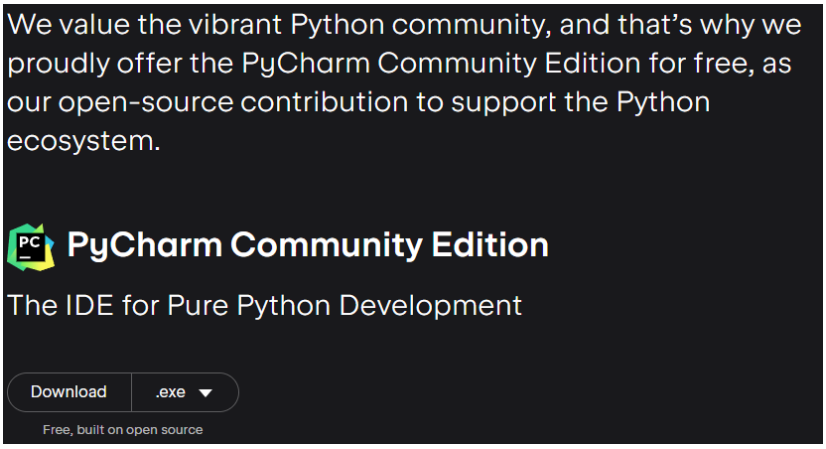
3. Download and Install PyCharm

Now that Python is installed on your system, we still need a way to create and edit programs. While there are multiple strategies and softwares that enable you to do this, for this course we will be using the free PyCharm Integrated Development Environment (IDE). PyCharm will allow us to write and execute programs written in Python, as well as several other features that we will discuss later such as debugging and visualizations.

To use PyCharm, it first must be downloaded and installed. You will need to navigate to spyder-ide.org and download the program:

[Download PyCharm]

This webpage should automatically detect your operating system, and thus you only need to scroll down to the PyCharm Community Edition and click Download. (NOTE: Make sure you download the Community Edition as it is free, unlike the Professional edition.)

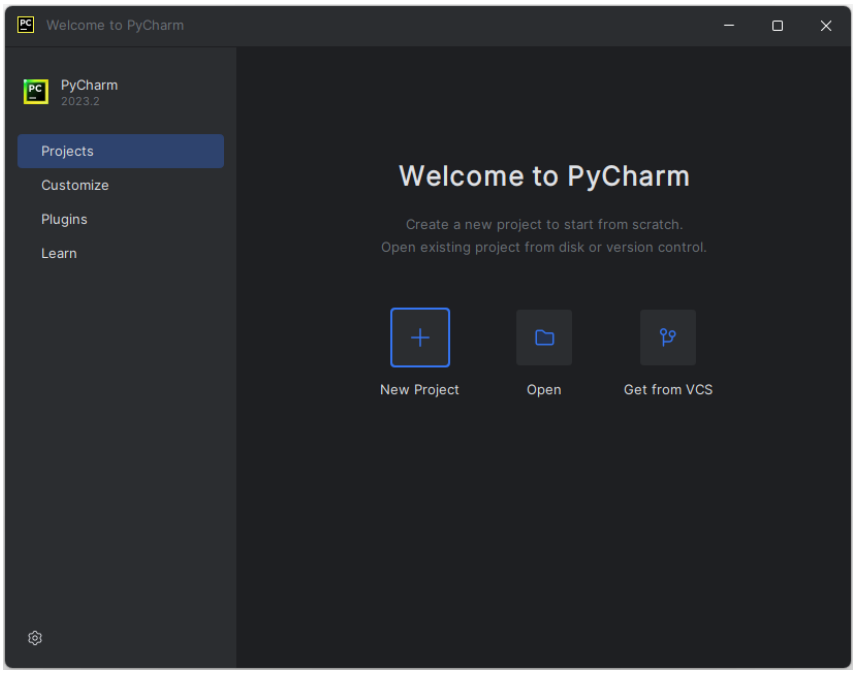


Once the file has finished downloading, run the file and follow the setup steps provided by the program. Using all of the default options provided by the setup program is recommended. The setup may take several minutes to complete.

4. Practicing with PyCharm

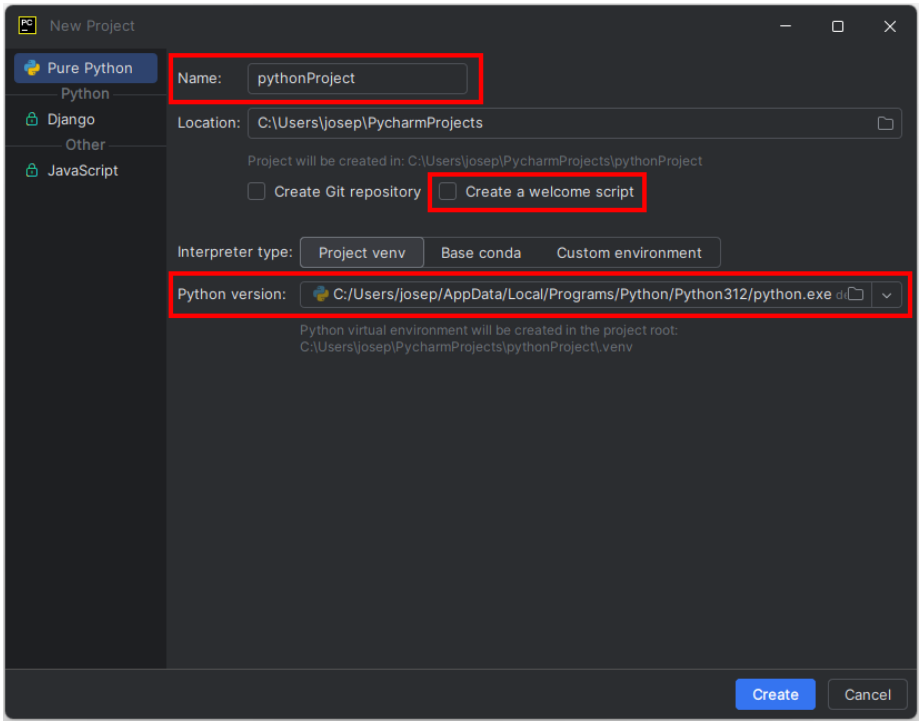
Now that we have Python and PyCharm installed, we need to practice a bit with the IDE. To begin we will create a first Python program, a simple program that outputs the phrase “Hello, world!”. To create your program, perform the following steps:

1. Start the PyCharm program. This will open a window that looks similar to this:

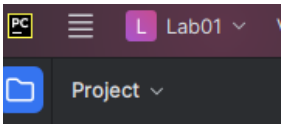


PyCharm is primarily designed for more complex projects, which will be useful later, but we have to do a bit of setup before we can begin coding.

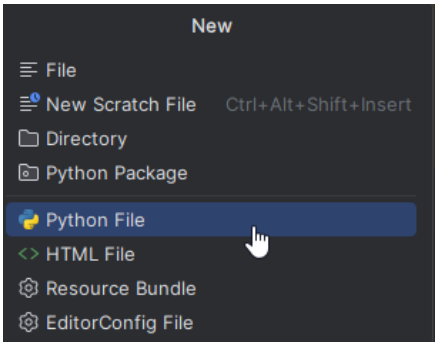
2. Click on the New Project button to start creating a new project. There will be some extra setup the very first time we create a new project, but PyCharm will save the settings for future new projects.



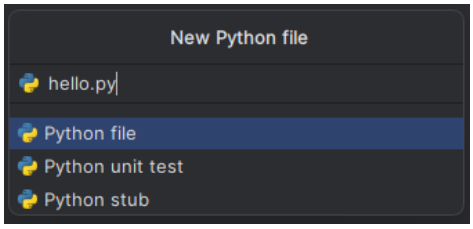
   a. First, to name your project, in the top **Name:** change `pythonProject` to `Lab01`, to create a new folder named `Lab01` for our project
   b. Second, the **Python version:** should be set to `Python312`. If it is not, click the arrow at the right to open a drop-down menu and then select the option with `Python312`
   c. Third, make sure the **Create a welcome script** box is not checked
   d. Finally click the **Create** button to create the project
3. Before you begin coding, you will need to create an empty file to store your program's code.
   a. First, click on the 4 horizontal lines at the top of the window to expand the main menu



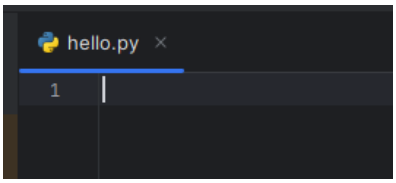
   b. Next, select `File -> New`, and choose Python File from the menu.



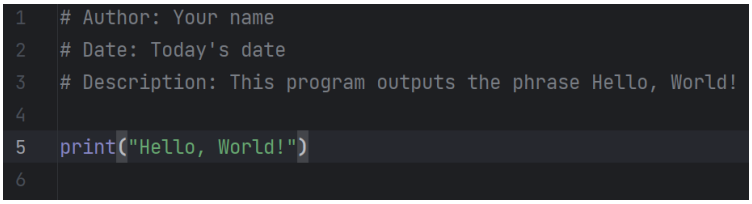
c. In the window that appears type in **hello.py** and press enter to create your file. Note that almost all python files will end with the extension `.py`, so make sure to include it when naming your program files!



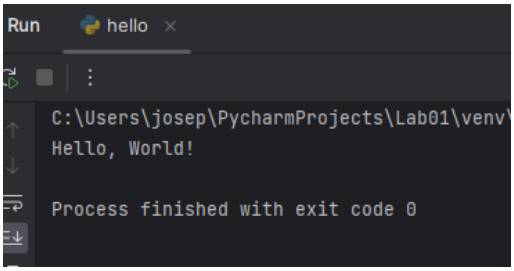
d. You should now have a new window with **hello.py** at the top, and you are ready to code!



4. Type the following text into the window and be sure to enter your name and today's date into the appropriate places in the commented header. Additionally, as you are typing, notice how PyCharm automatically colors the text in different color, and what happens when you close the parenthesis.



5. Now execute the program by going to the Main Menu (the four horizontal lines) and selecting `Run -> Run 'hello.py'`. PyCharm should now show something similar to this:



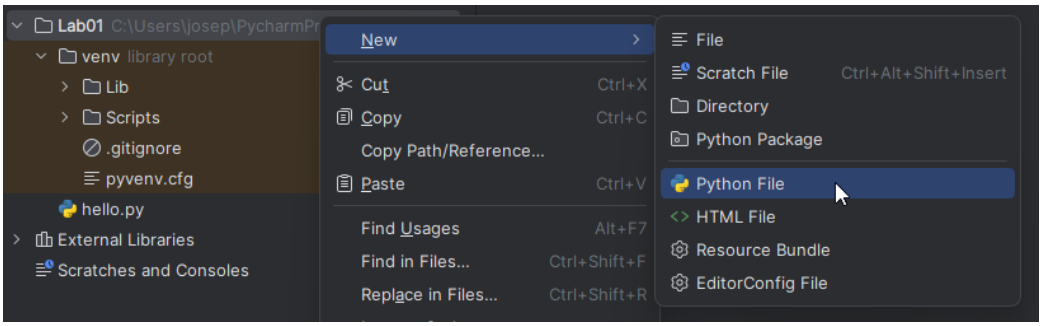
We can see that the program outputs the phrase `Hello, World!` without producing any errors.

Make sure you save your **hello.py** as you will be submitting this file for Lab 01. If you experienced any issues, please let your instructor know so we can sort them out.

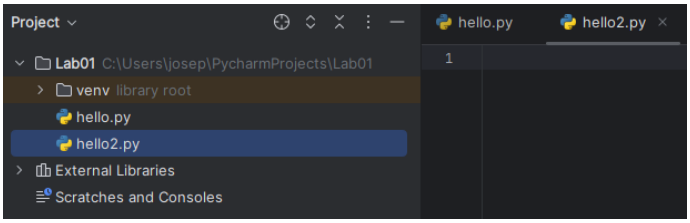
5. Obtaining User Input

Now that we have built our first program, we should practice working with prompting the user for information, storing that information, and then using it.

1. Create a new `.py` file named **hello2.py**, by right clicking on the Lab01 folder, selecting New and then Python File. In the resulting window, name the file **hello2.py** and press enter.



2. You should now have both files in your Lab01 directory, and one tab for each file in the editor area.

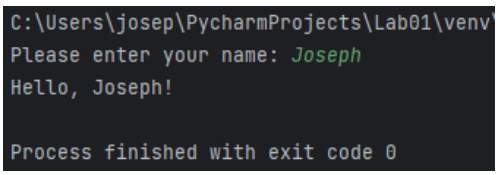


   - Now, copy all of the code from **hello.py** and paste it in **hello2.py**

3. Modify the code in **hello2.py** as shown below, and be mindful of a few things:
   - `userName` is a variable, which allows the program to store data during program execution
   - Be careful with white space (spaces, tabs, new line) as Python uses these to determine how to organize the execution of the instructions you provide
   - Symbols such as (), {}, and [] are used by Python to provide certain functionalities. In this case, () is being used to determine what `print` should output. {} is being used to denote that when `print` outputs the phrase `Hello, {userName}!`, it uses the information stored in the `userName` variable and does not print out the word userName
   - The `f` in `print` configures `print` to output the information stored in any variables found inside {}

```
1  # Author: Your name
2  # Date: Today's date
3  # Description: This program prompts the user for their name and greets them
4  # using that name.
5
6  userName = input("Please enter your name: ")
7  print(f"Hello, {userName}!")
8
```

3. Execute the program. This time, PyCharm will prompt you for your name, which you can type in and press the Enter key. Once you enter your name, your output should look similar to this:

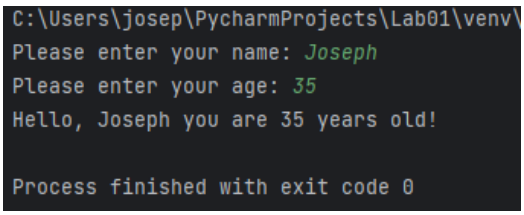


Make sure you save your **hello2.py** as you will be submitting this file for Lab 01.

5. Putting It All Together

You should now have a better grasp of how to write and execute simple programs in Python. For the third program, you will need to:

- Create a new file **hello3.py** that is a copy of your **hello2.py** program using either strategy we have used to create `.py` files so far
- Create a new variable called `userAge` that is used to store the user's age, and prompt the user for their age in a similar way to how you prompted the user for their name
- Modify the hello message by also stating how many years old the user is
- Update the Description in the program header (the top lines in the file that begin with #) to reflect the new functionality you have added
- Your new output should look similar to the following:



Make sure you save your **hello3.py** as you will be submitting this file for Lab 01.

Submission

Once you have completed this lab it is time to turn in your work. Please submit the following three files to the Lab 01 assignment in Canvas:

- **hello.py**
- **hello2.py**
- **hello3.py**

Sample Output

hello.py

```
Hello, World!
```

hello2.py

```
Please enter your name: Viv
Hello, Viv!
```

hello3.py

```
Please enter your name: Tandri
Please enter your age: 22
Hello, Tandri you are 22 years old!
```

Rubric

For each program in this lab, you are expected to make a good faith effort on all requested functionality. Each submitted program will receive a score of either 100, 75, 50, or 0 out of 100 based upon how much of the program you attempted, regardless of whether the final output is correct (See table below). The scores for all of your submitted programs for this lab will be averaged together to calculate your grade for this lab. Keep in mind that labs are practice both for the exams in this course and for coding professionally, so make sure you take the assignments seriously.

Score	Attempted Functionality
100	100% of the functionality was attempted
75	<100% and >=75% of the functionality was attempted
50	<75% and >=50% of the functionality was attempted
0	<50% of the functionality was attempted