NORTHERN ILLINOIS UNIVERSITY

PHYSICS DEPARTMENT

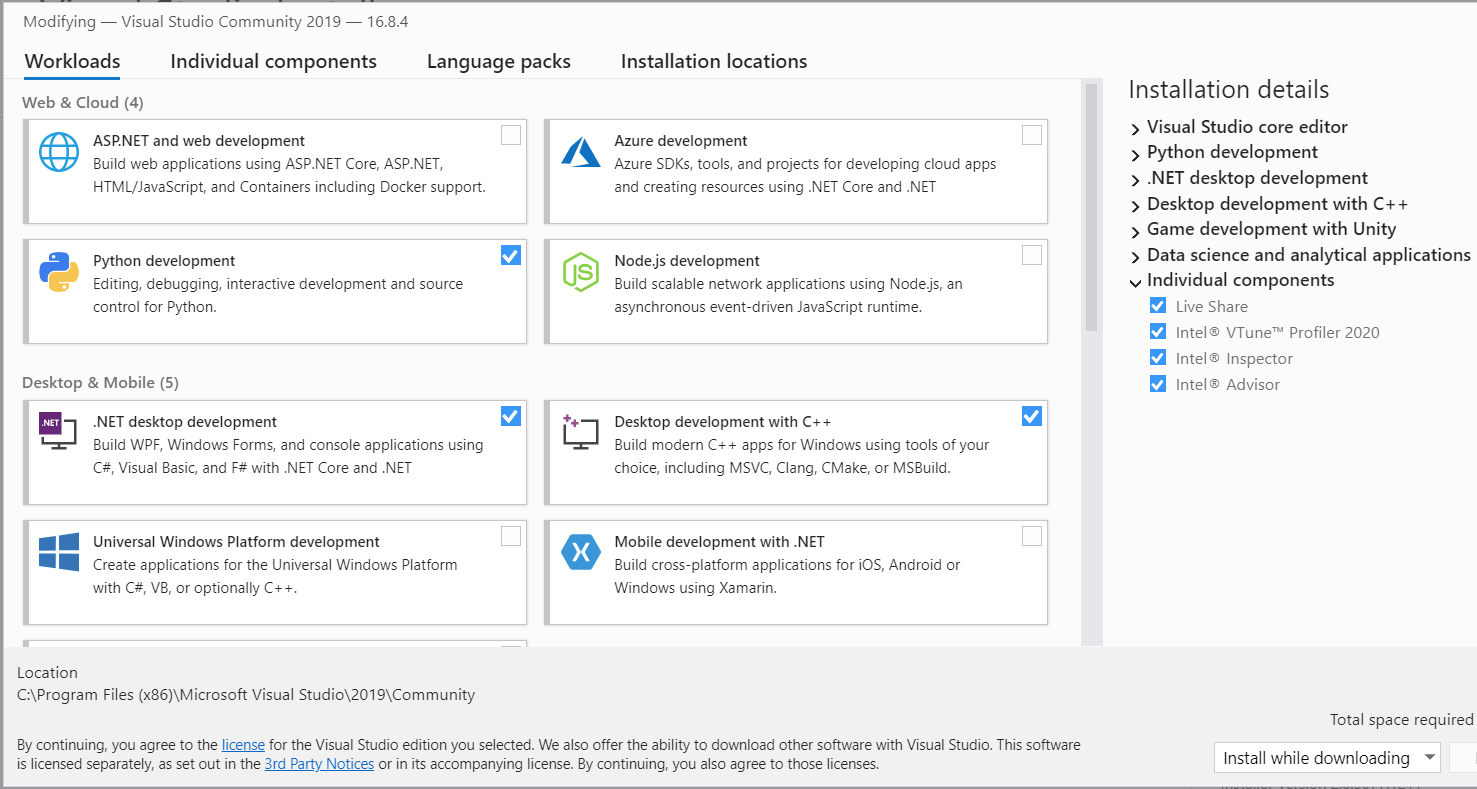
Physics 374 – Junior Physics Lab Fall 2023

Python Tutorial #1

**Installation of Python**

The latest version of Python is 3.11.5 which can be found at <https://www.python.org>. A good integrated development environment (IDE) to develop Phython code is Microsoft Visual Community 2022 which can be found at (*note: the code, like Python, is also free*) <https://visualstudio.microsoft.com/vs/community>. MAC users are recommended to install (*the free*) Microsoft Visual Code (version 1.8.1) as the IDE (go to <https://code.visualstudio.com>).

Install Microsoft Visual Community (or Visual Code for Mac users). When I install it on a Windows system, I generally select the following additional features (they are not installed by default): (1) Python development and (2) Desktop development with C++. If desired, you can install other packages (like the Visual Basic package selected below).

Note: the Python package is optional since you will already have a version installed.

Python is a dynamic language (executed by an interpreter) that cannot be compiled into machine code statically (like C++ and Fortran). Python 3.11.5 is an interpreter that is needed to execute Python commands. There are many Python packages written to do various calculations (for example, numerical integration, least square fit to a line, etc.) To install a Python package, we need to have *pip* installed. *Pip* documentation can be found at:

<https://pip.pypa.io/en/stable/quickstart/>

Fortunately, *pip* is installed automatically when Python 3.11.5 is installed. On Windows, we need to open a command window to install a package. In Windows 11, we can either use the normal Command Prompt window or the PowerShell window. Click on the Start button and type “powershell” and select the Windows PowerShell application. In PowerShell type:

py -m pip install --upgrade pip

To upgrade *pip*. If you get a warning that a Python Scripts directory is not on PATH, add the directory to PATH by editing the system environment variables. Documentation can be found at:

<https://www.architectryan.com/2018/03/17/add-to-the-path-on-windows-10/>

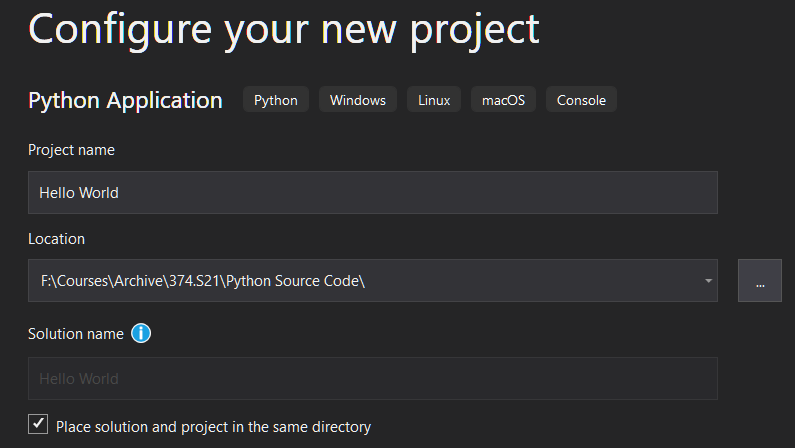
To add the Scripts directory to PATH: (1) click on the Start button and type “env”, then select “Edit the system environment variables”, (2) select the “Advanced” tab, (3) click the “Environment Variables…” button, (4) Under the “System Variables” section (the lower half), find the row with “Path” in the first column, and click “Edit…”, (5) click “New” and type (or copy and paste) the new path. For instance, it should look similar to:

C:\Users\UserName\AppData\Roaming\Python\Python311\Scripts

depending upon what your “username” is. You can always find out which directories are in your PATH by typing: $env:PATH in your PowerShell window.

**Writing Code in Python**

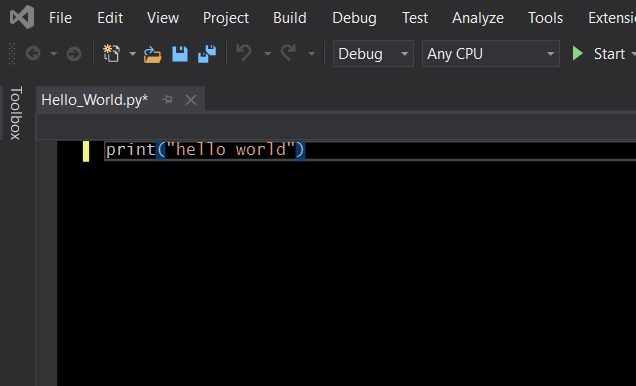
Now that Python has been installed, we can now start writing Python code. Start the Microsoft Visual Studio application, and click “continue without code”. Click on “File” 🡪 “New” 🡪 “Project…”. Under “All languages” select “Python”. Then select the “Python Application” row, and then select “Next”. Name the “Project name” as “Hello World”. Select the Browse button to put the project in a directory of your own choosing.



Browser button to select location of project

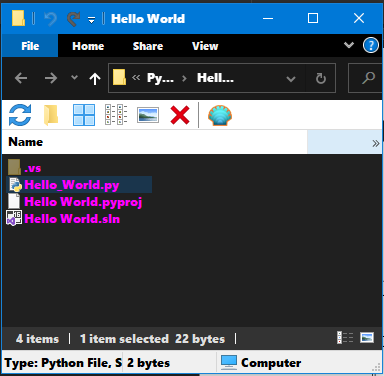
Then select “Create”.

In the *Hello\_World.py* window, type: print("hello world")



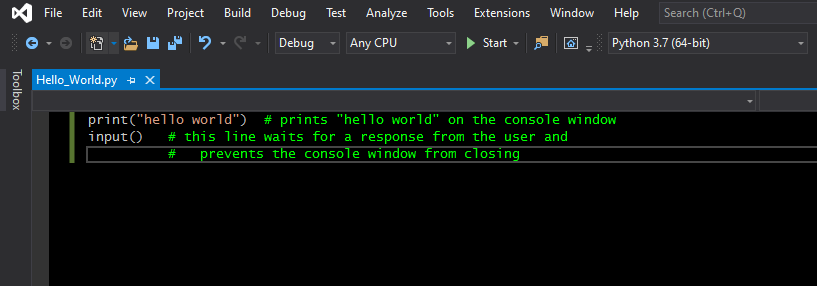
“Start” button runs the code

Click on the “Start” button (green arrow triangle), and a console window should appear with the phrase “hello world”. You have now written your 1st Python application. If you go to the folder where your Python code lies, you should see the following:



The \*.py file is the source code, the \*.sln file is the Solution file (containing the project). Double clicking on the Solution file opens the Python project. Notice that there is no executable file.

Notice that the console window simply flashes and then disappears. We need a way to force the window to stay on the screen until we wish to close it. We can do this by typing the line, “input()”, below the line “print(“hello world”):



The input() function is waiting for the user to press the carriage-return (or enter) key to continue running the program.

**Writing Python GUI with Tcl/Tk**

The previous section showed how to write code for command line applications that rely on a console for operation. In this section we will write code that uses the window environment of Windows, Mac, and Linux operating systems. Thus, we will write a windows-driven rather than a command-line driven program using the widgets available in the Tcl/Tk package called *tkinter*. There is a good tutorial on the web located at:

<https://learn.sparkfun.com/tutorials/python-gui-guide-introduction-to-tkinter/hello-world>

Read Shawn Hymel’s tutorial called “Python GUI Guide: Introduction to Tkinter” on the webpage above. The code below is based on his tutorial. Create a new Python program called *Hello.py*, and copy and paste the code below and run it:

import tkinter as tk # import the tkinter module (or package)

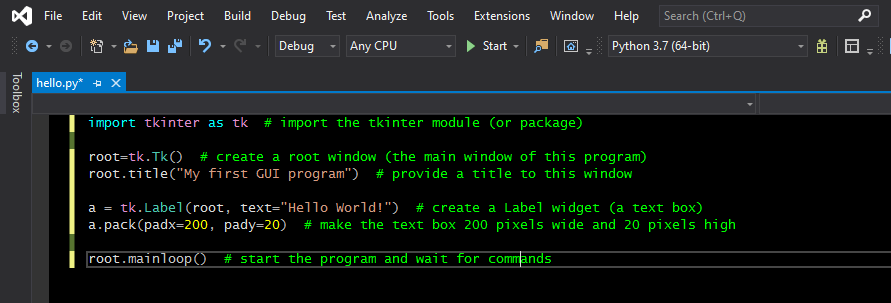
root=tk.Tk() # create a root window (the main window of this program)

root.title("My first GUI program") # provide a title to this window

a = tk.Label(root, text="Hello World!") # create a Label widget (a text box)

a.pack(padx=200, pady=20) # make the text box 200 pixels wide and 20 pixels high

root.mainloop() # start the program and wait for commands



Upload to Blackboard your source code (\*.py) of your Python programs *Hello\_World* and *Hello*. You will see an assignment on Blackboard called **Python Tutorial #1**.