



## Reno Technology Academy

Multnomah University Reno/Tahoe

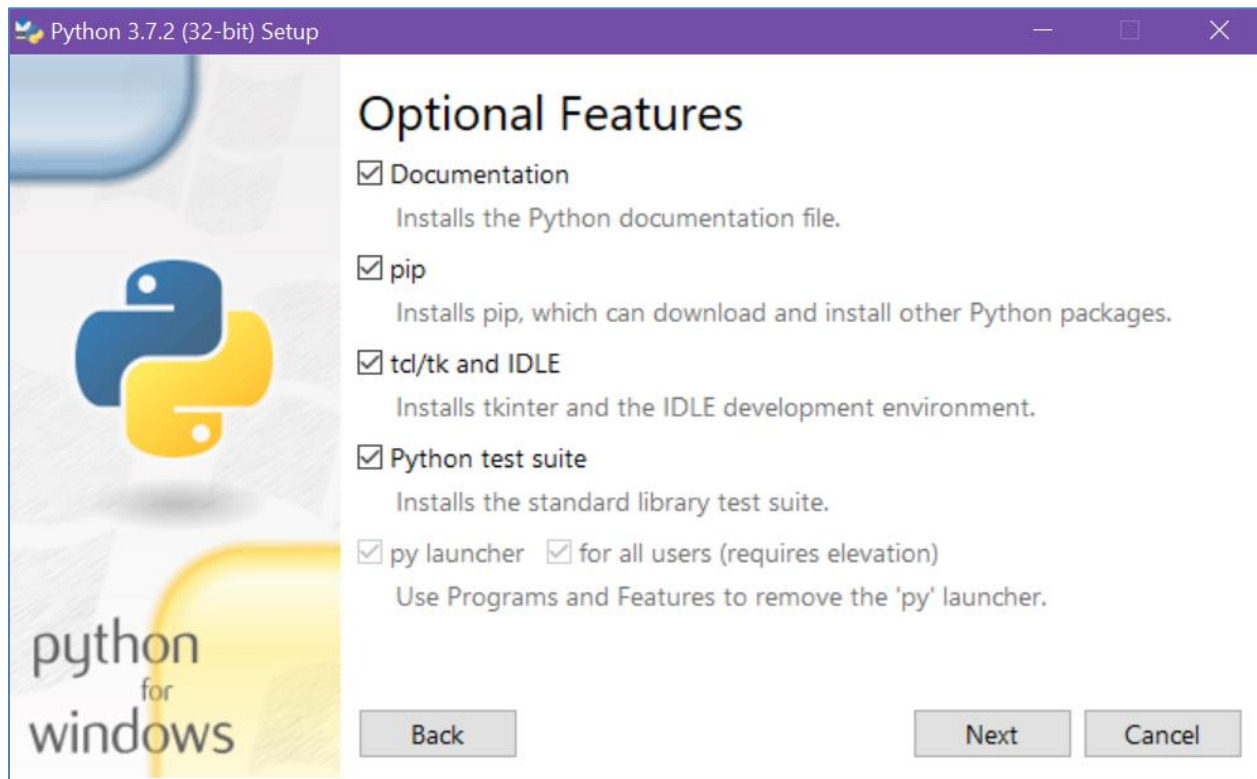
CIS104: Coding in Python

### Lesson 1

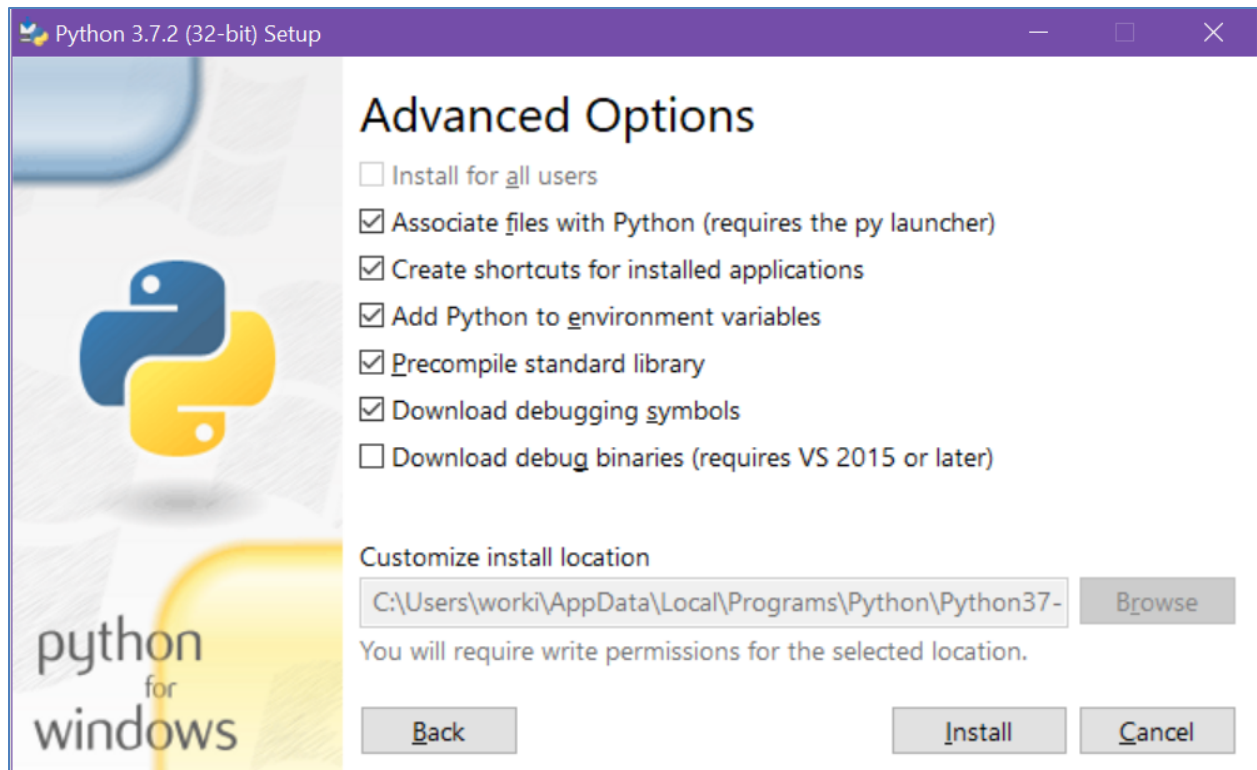
#### Lab 1: Setup Python, Hello World in IDLE

Python Installation: <https://www.python.org/downloads/> Download latest Python 3 version.

Install all the optional Features.

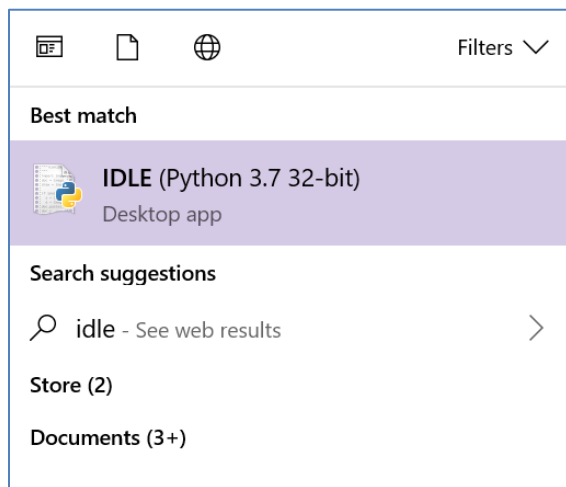


Install the following Advanced Options



Click Install.

Run IDLE: Click on the Start Menu and type "IDLE":



Open new file: File > New File.

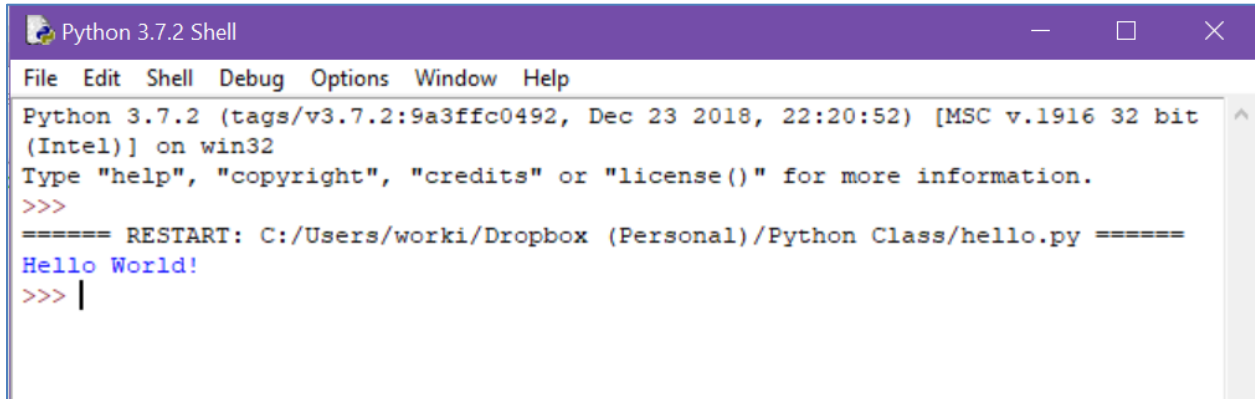
Type

```
print("Hello World!")
```

Save file: File > Save > Choose location and name.

Run: Run > Run Module.

You should see your output displayed in the Python Shell:

A screenshot of a Windows application window titled "Python 3.7.2 Shell". The window has a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main text area shows the following content: "Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit (Intel)] on win32", "Type 'help', 'copyright', 'credits' or 'license()' for more information.", a prompt ">>>", a separator line "===== RESTART: C:/Users/worki/Dropbox (Personal)/Python Class/hello.py =====", the output "Hello World!" in blue, and another prompt ">>> |".

```
Python 3.7.2 Shell
File Edit Shell Debug Options Window Help
Python 3.7.2 (tags/v3.7.2:9a3ffc0492, Dec 23 2018, 22:20:52) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:/Users/worki/Dropbox (Personal)/Python Class/hello.py =====
Hello World!
>>> |
```

## Lab 2: Setup Visual Studio Code (VS Code), Hello World in VS Code

VS Code Installation: <https://code.visualstudio.com/download>

Follow the tutorial at <https://code.visualstudio.com/docs/python/python-tutorial> -- Stop at **Configure and run the debugger**

*From the tutorial:*

### Prerequisites

To successfully complete this tutorial, you must do the following:

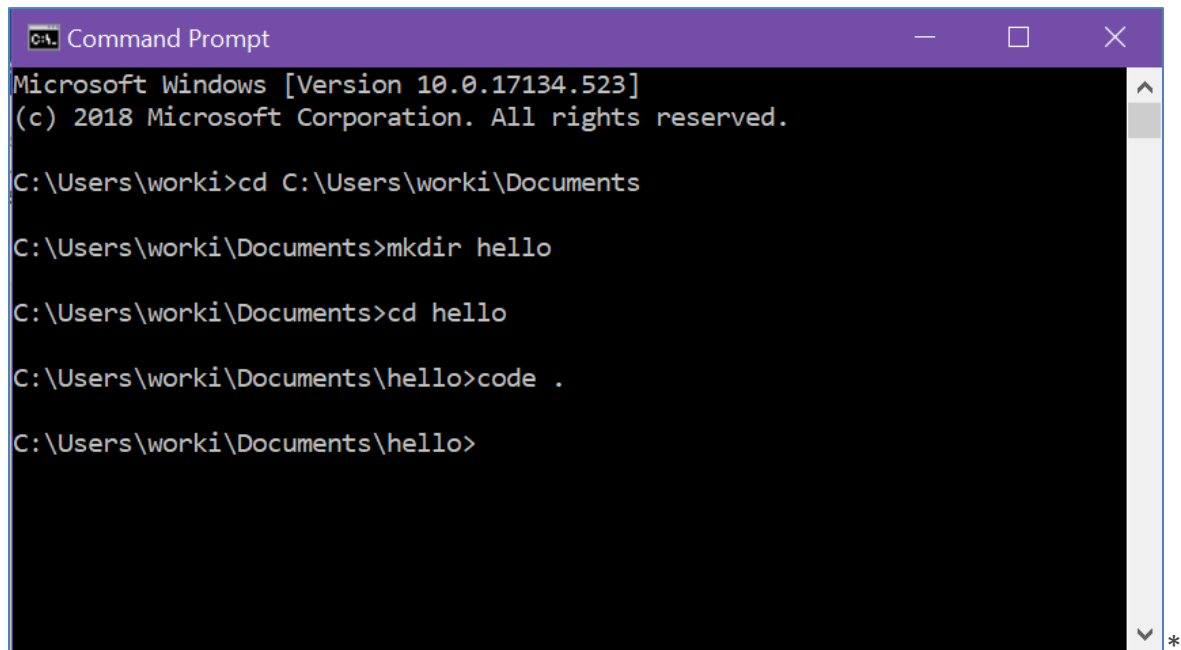
1. Install the [Python extension for VS Code](#).
2. Install a version of Python 3 (for which this tutorial is written). Options include:
  - (All operating systems) A download from [python.org](https://python.org); you can typically use the **Download Python 3.7.1** button that appears first on the page (or whatever is the latest version).
  - (Linux) The built-in Python 3 installation works well, but to install other Python packages you must install **pip** with **get-pip.py**.
  - (macOS) An installation through [Homebrew](#) on macOS using **brew install python3** (the system install of Python on macOS is not supported).
  - (All operating systems) A download from [Anaconda](#) (for data science purposes).
3. On Windows, make sure the location of your Python interpreter (that is, the folder where it's installed, like **c:\python32**) is included in your PATH environment variable. You can check this by running **path** at the command prompt. If the Python interpreter's folder isn't included, open Windows Settings, search for "environment", select **Edit environment variables for your account**, then edit the **Path** variable to include that folder.

4. On MacOS, make sure the location of your VS Code installation is included in your PATH environment variable. See [the setup instructions](#) for more information.

## Start VS Code in a project (workspace) folder

At a command prompt (Start menu > type "cmd" > Command Prompt)\* or terminal, (navigate to the folder you want to create the folder in, e.g. if you want to create the project in C:\Users\{username}\Documents, type "cd C:\Users\{username}\Documents"\*, create an empty folder called "hello", navigate into it, and open VS Code (`code`) in that folder (`.`) by entering the following commands:

```
mkdir hello  
cd hello  
code .
```



```
Command Prompt  
Microsoft Windows [Version 10.0.17134.523]  
(c) 2018 Microsoft Corporation. All rights reserved.  
  
C:\Users\worki>cd C:\Users\worki\Documents  
  
C:\Users\worki\Documents>mkdir hello  
  
C:\Users\worki\Documents>cd hello  
  
C:\Users\worki\Documents\hello>code .  
  
C:\Users\worki\Documents\hello>
```

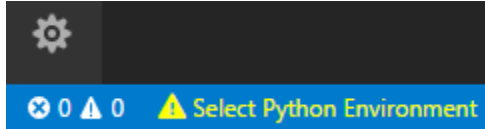
By starting VS Code in a folder, that folder becomes your "workspace". VS Code stores settings that are specific to that workspace in `.vscode/settings.json`, which are separate from user settings that are stored globally.

Alternately, you can run VS Code through the operating system UI, then use **File > Open Folder** to open the project folder.

## Select a Python interpreter

Python is an interpreted language, and in order to run Python code and get Python IntelliSense, you must tell VS Code which interpreter to use.

From within VS Code, select a Python 3 interpreter by opening the **Command Palette** (**Ctrl+Shift+P**), start typing the **Python: Select Interpreter** command to search, then select the command. You can also use the **Select Python Environment** option on the Status Bar if available (it may already show a selected interpreter, too):



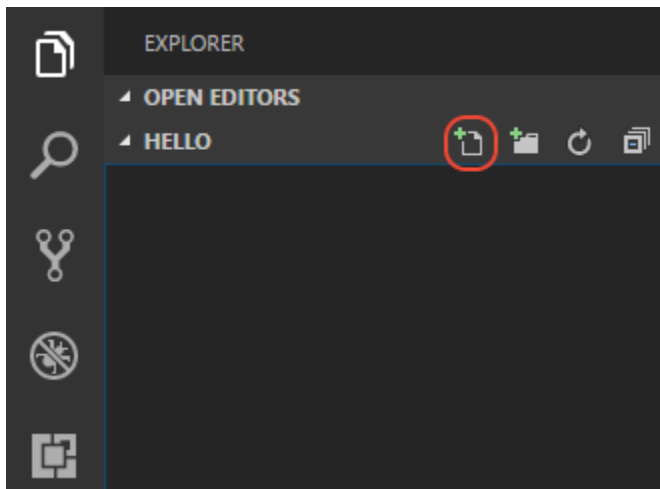
The command presents a list of available interpreters that VS Code can find automatically. If you don't see the desired interpreter, see [Configuring Python environments](#).

Selecting an interpreter sets the `python.pythonPath` value in your workspace settings to the path of the interpreter. To see the setting, select **File > Preferences > Settings (Code > Preferences > Settings on macOS)**, then select the **Workspace Settings** tab.

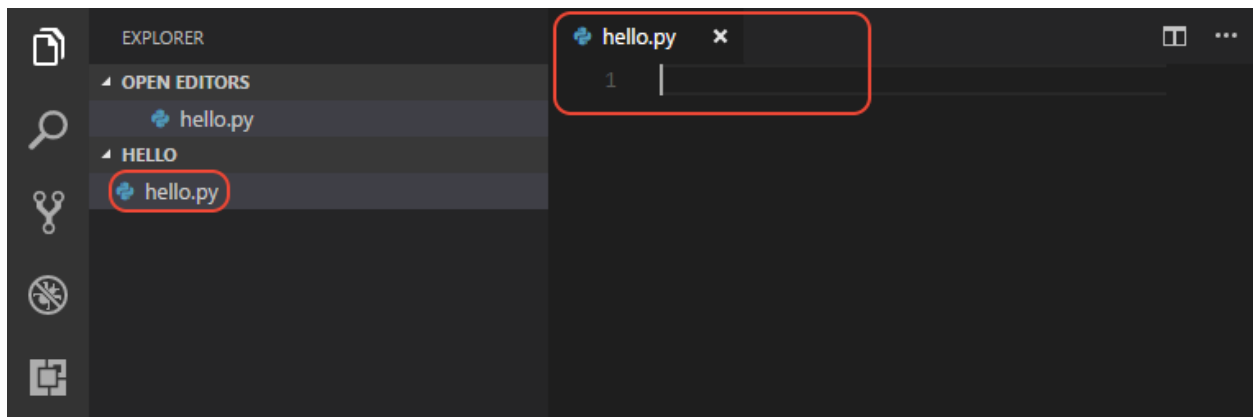
**Note:** If you select an interpreter without a workspace folder open, VS Code sets `python.pythonPath` in your user settings instead, which sets the default interpreter for VS Code in general. The user setting makes sure you always have a default interpreter for Python projects. The workspace settings lets you override the user setting.

## Create a Python Hello World source code file

From the File Explorer toolbar, click the New File button on the `hello` folder:



Name the file `hello.py`, and it automatically opens in the editor:

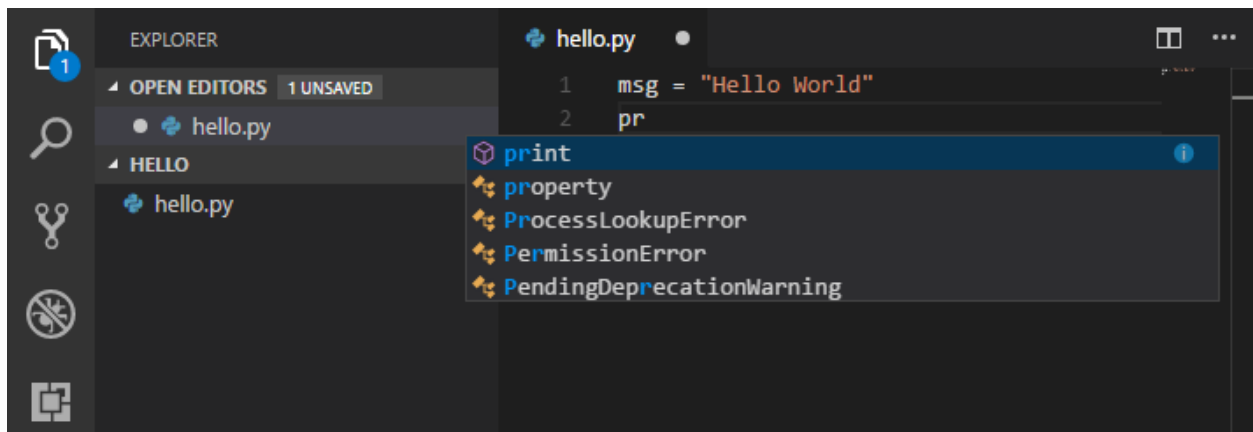


By using the `.py` file extension, you tell VS Code to interpret this file as a Python program, so that it evaluates the contents with the Python extension and the selected interpreter.

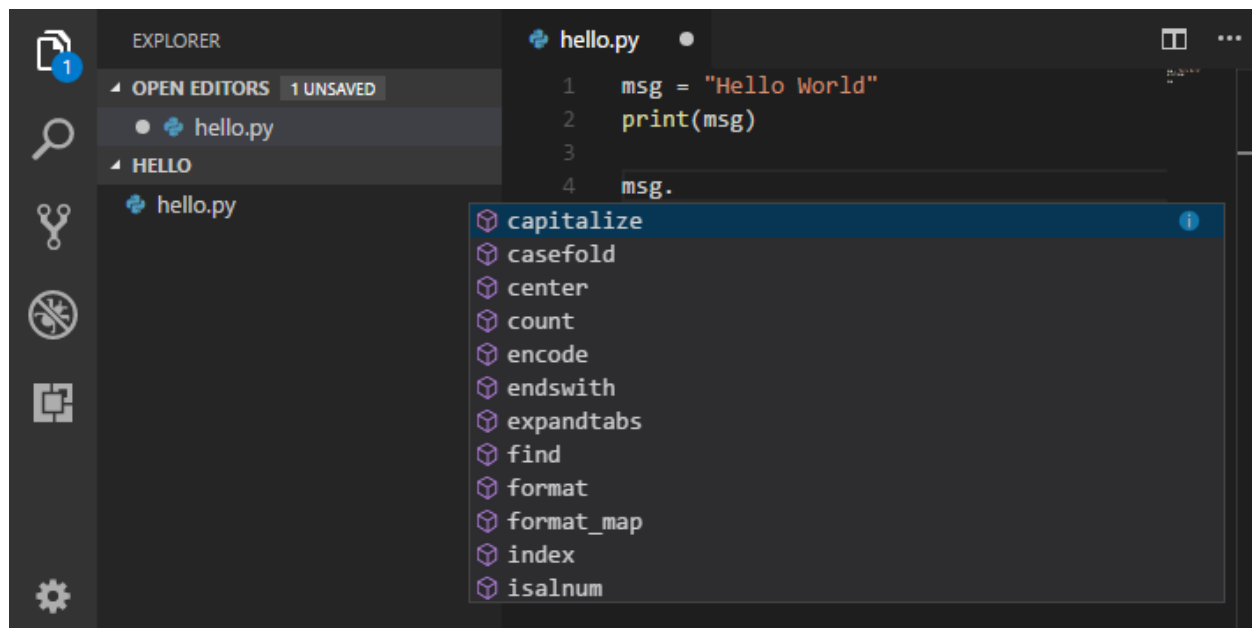
Next, start entering the following source code if using Python 3:

```
msg = "Hello World"
print(msg)
```

When you start typing `print`, notice how [IntelliSense](#) presents auto-completion options.



IntelliSense and auto-completions work for standard Python modules as well as other packages you've installed into the environment of the selected Python interpreter. It also provides completions for methods available on object types. For example, because the `msg` variable contains a string, IntelliSense provides string methods when you type `msg.`:

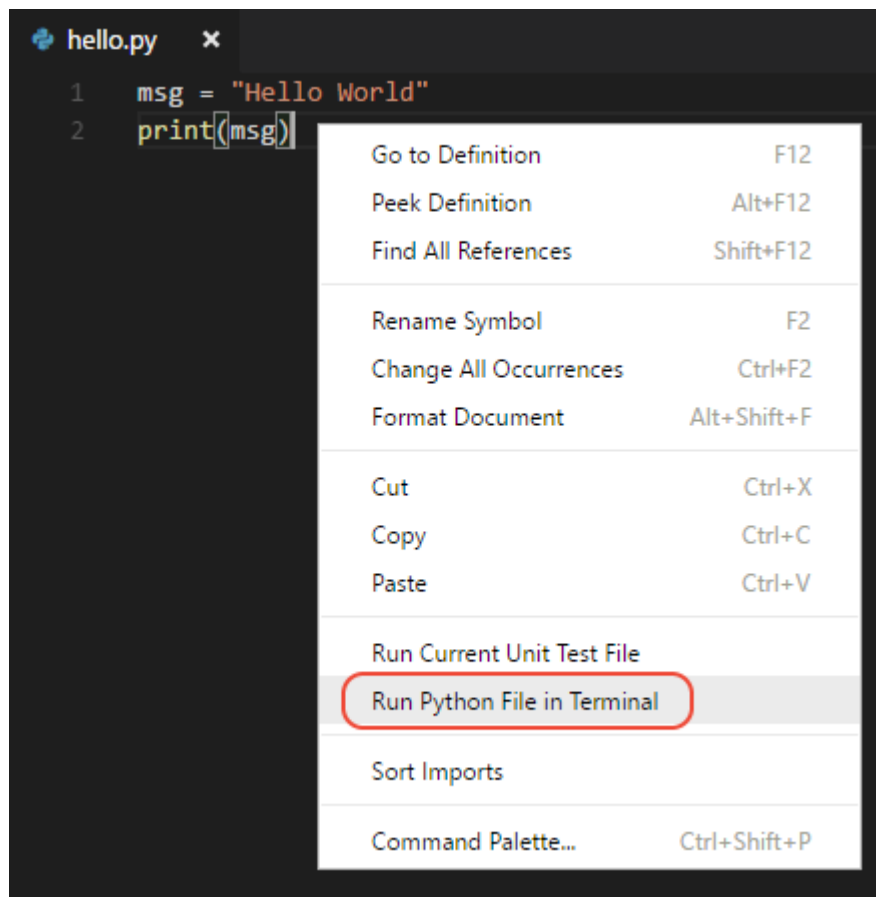


Feel free to experiment with IntelliSense some more, but then revert your changes so you have only the `msg` variable and the `print` call, and save the file ([Ctrl+S](#)).

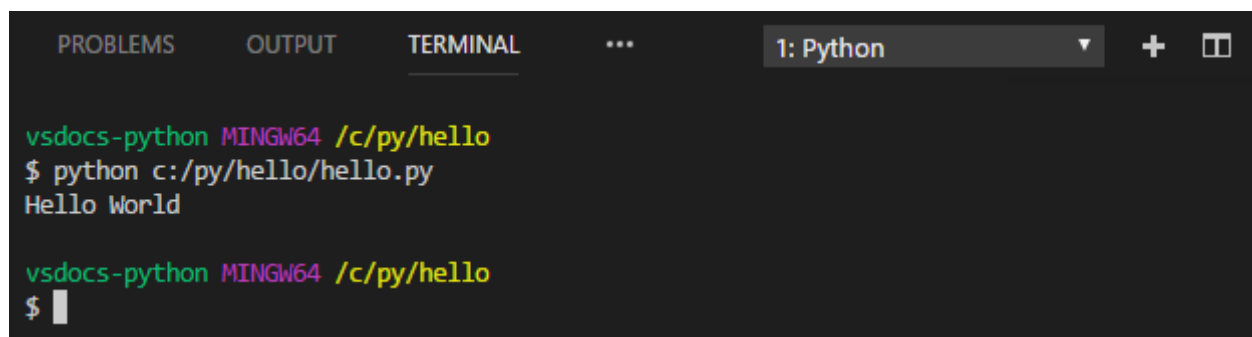
For full details on editing, formatting, and refactoring, see [Editing code](#). The Python extension also has full support for [Linting](#).

## Run Hello World

It's simple to run `hello.py` with Python. Right-click in the editor and select **Run Python File in Terminal**(which saves the file automatically):



The command opens a terminal panel in which your Python interpreter is automatically activated, then runs `python3 hello.py` (macOS/Linux) or `python hello.py` (Windows):



There are two other ways you can run Python within VS Code:

- Select one or more lines, then press `Shift+Enter` or right-click and select **Run Selection/Line in Python Terminal**. This command is very convenient for testing just a part of a file.
- Use the **Python: Start REPL** command to open a REPL terminal for the currently selected Python interpreter. In the REPL you can then enter and run lines of code one at a time.

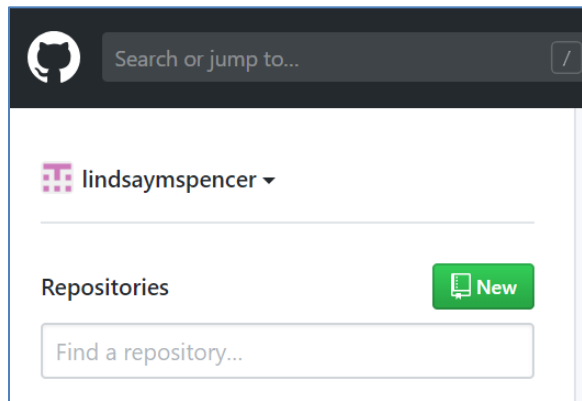
\*Additional material NOT in the online version.



## Lab 3: Create a Repository and check a Project into GitHub

### Create the repository

Assuming you have a GitHub account, login and create a new repository



Name the repository "CIS104" then select "Create repository".


## Create a new repository


A repository contains all the files for your project, including the revision history.

---

Owner

Repository name \*


 lindsaym Spencer ▾

/ CIS104 


Great repository names are short and memorable. Need inspiration? How about **fuzzy-guacamole**.

Description (optional)

---

☒  **Public**

Anyone can see this repository. You choose who can commit.

☐  **Private**


You choose who can see and commit to this repository.

---

☐ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▾

Add a license: **None** ▾ 

---

**Create repository**

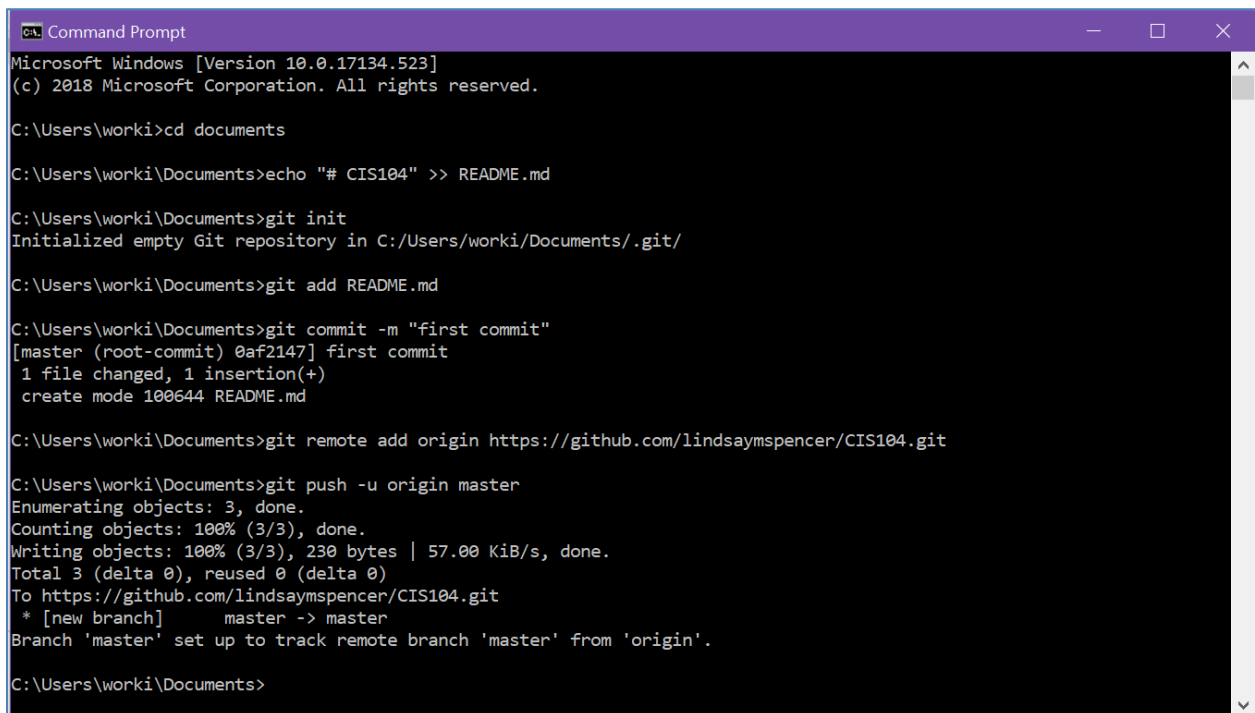
## Link the repository to a folder

Open the command prompt (Start > cmd > Command Prompt)\* and navigate to the folder where you want the repository to be (cd {folder location})\*. Follow the directions from your repository for creating a new repository

### ...or create a new repository on the command line

```
echo "# CIS104" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/lindsayspencer/CIS104.git <-- Your URL will be different!
git push -u origin master
```

Your command prompt should look something like this when you're through



```
Command Prompt
Microsoft Windows [Version 10.0.17134.523]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\worki>cd documents

C:\Users\worki\Documents>echo "# CIS104" >> README.md

C:\Users\worki\Documents>git init
Initialized empty Git repository in C:/Users/worki/Documents/.git/

C:\Users\worki\Documents>git add README.md

C:\Users\worki\Documents>git commit -m "first commit"
[master (root-commit) 0af2147] first commit
1 file changed, 1 insertion(+)
create mode 100644 README.md

C:\Users\worki\Documents>git remote add origin https://github.com/lindsayspencer/CIS104.git

C:\Users\worki\Documents>git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 230 bytes | 57.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/lindsayspencer/CIS104.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.

C:\Users\worki\Documents>
```

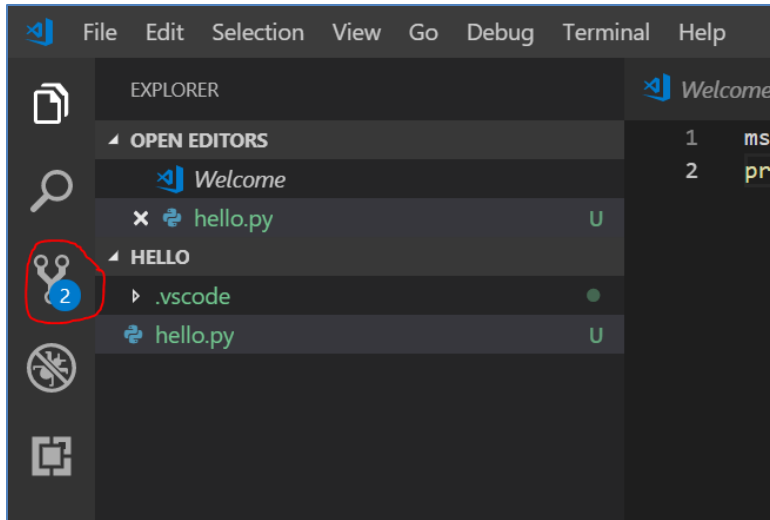
Now you have a linked GitHub repository and a folder on your computer.

## Create a project in the repository

Navigate to inside the folder (cd {folder name}), and create a new Python project (as in Lab 2).

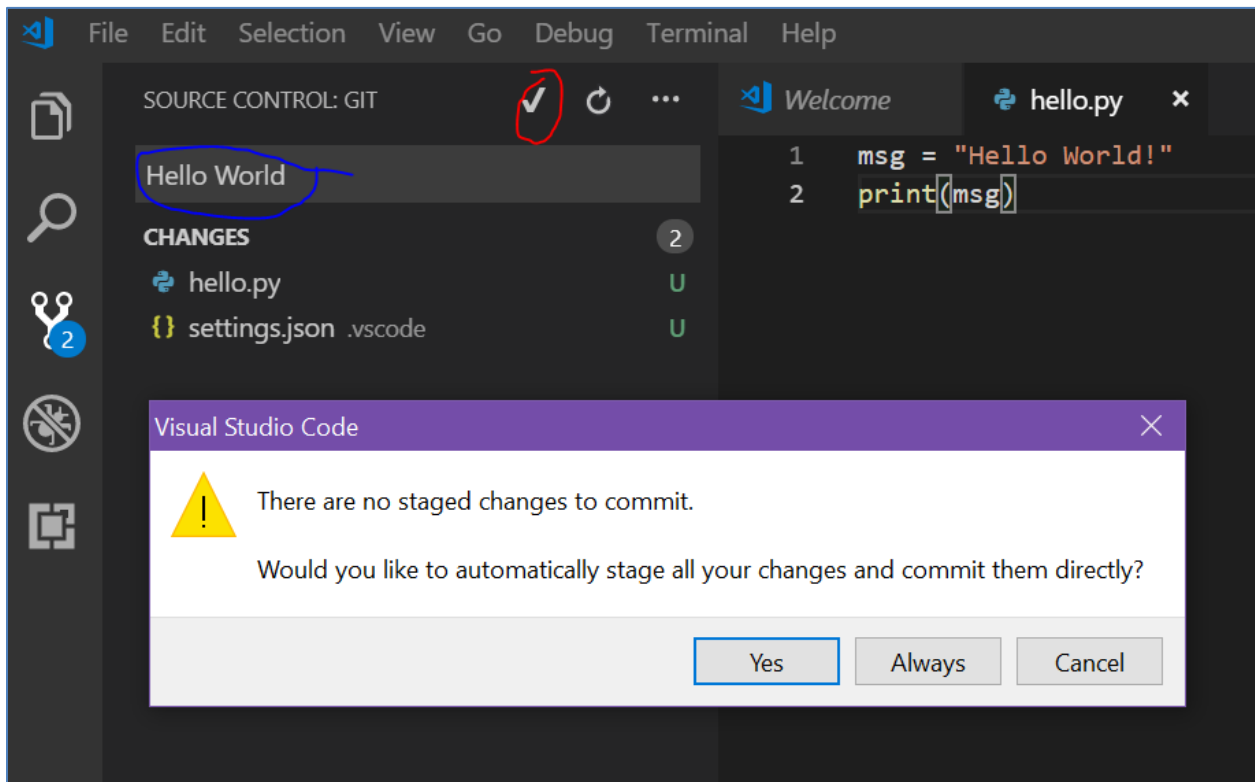
## Commit changes

After you have created the hello.py file, you should see



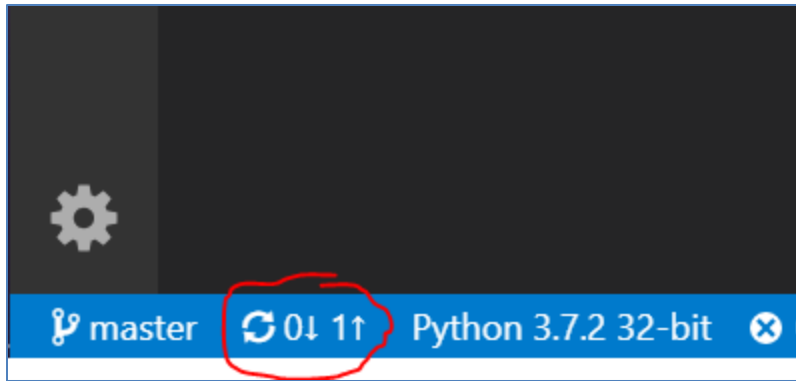
Click on the circled icon.

Enter a commit message in the blank (blue circle) and click the checkmark (red circle). Click “Yes” in the pop-up.



### Synchronize changes

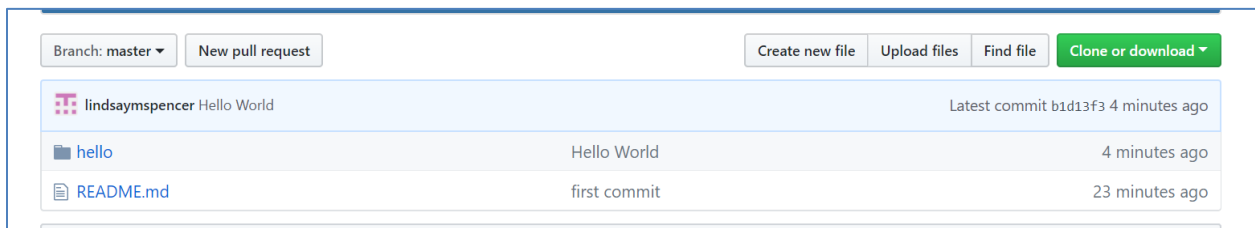
At the bottom left of the application, click on the circled icon



Click “OK”. You may be asked for your GitHub account username and password.

## Submit assignment

Your GitHub account online should now contain your “hello” project.



Click on the “hello” folder.

Grab the URL from the browser (e.g. <https://github.com/lindsayspencer/CIS104/tree/master/hello>) and submit it via Populi in Lesson 1 Lab.

\*See Lab 2: Start VS Code in a project (workspace) folder for more details.

## Readings:

### *Learning Python*

Chapter 3: How You Run Programs p. 43

The Interactive Prompt pp. 43-45

What Not to Type: Prompts and Comments pp. 48-49

Running Code Interactively, Why the Interactive Prompt? pp. 49-51

Usage Notes: The Interactive Prompt pp. 52-54

System Command Lines and Files pp. 54-59

Clicking File Icons pp. 62-66

The IDLE User Interface pp. 73-79

Debugging Python Code pp. 85-87

Chapter Summary p. 85

Chapter 4: Introducing Python Object Types p.95

Python's Core Data Types pp. 97-108

Chapter 5: Numeric Types pp.137-179

Chapter 7: String Fundamentals pp. 195-246

Floating Point Arithmetic: Issues and Limitations:

<https://docs.python.org/2/tutorial/floatingpoint.html>

## Homework (10 points)

All homework files can be added to your "hello" project from Lab 3. After you commit and sync the changes, resubmit the same URL in Lesson 1 Homework. I would suggest committing each file when you finish each part. You can sync the commits at the end. Feel free to commit and sync as many times as necessary. A commit/sync doesn't mean the project is finished.

### Part 1:

Write an application (named H1P1.py) which prompts the user for a few bits of information about themselves and outputs a greeting to the console:

- Prompt the user to enter their first and last name, each into different variables (named "first\_name" and "last\_name").
- Prompt the user to enter their age (variable named "age").
- Prompt the user to express their confidence in programmers between 0-100% (variable named "confidence").
- Calculate the age of the user in dog years (1 human year = 7 dog years), (variable name "dog\_age").
- Output a greeting in the form: "Hello {first\_name} {last\_name}, nice to meet you! You might be {age} in human years, but in dog years you are {dog\_age}."
- (Extra Credit - 1 point – requires If statements from Learning Python Ch. 10) If their confidence is less than 50% (0.5), output the statement: "I agree, programmers can't be trusted!", otherwise output the statement "Writing good code takes hard work!".

### Part 2:

*Programming Principles and Practice Using C++* Chapter 3 Exercise 2 (p.85)

Write a program (named H1P2.py) [in Python] that converts from miles to kilometers. Your program should have a reasonable prompt for the user to enter a number of miles. Hint: There are 1.609 kilometers to the mile.

### Part 3:

*Programming Principles and Practice Using C++* Chapter 3 Exercise 11 (pp.86-7), slightly modified.

Write a program (named H1P3.py) that prompts the user to enter some number of pennies (1-cent coins), nickels (5-cent coins), dimes (10-cent coins), quarters (25-cent coins), half dollars (50-cent coins), and one-dollar coins (100-cent coins). Query the user separately for the number of each size coin, e.g. "How many pennies do you have?" Then your program should print out something like this:

You have 23 pennies.

You have 17 nickels.

You have 14 dimes.

You have 7 quarters.

You have 3 half dollars.

You have 0 one dollars.

The value of all of your coins is \$5.73.

*(Extra Credit - 1 point – requires If statements from Learning Python Ch. 10)* If only one of a coin is reported, make the output grammatically correct, e.g. 14 dimes and 1 dime (not 1 dimes).