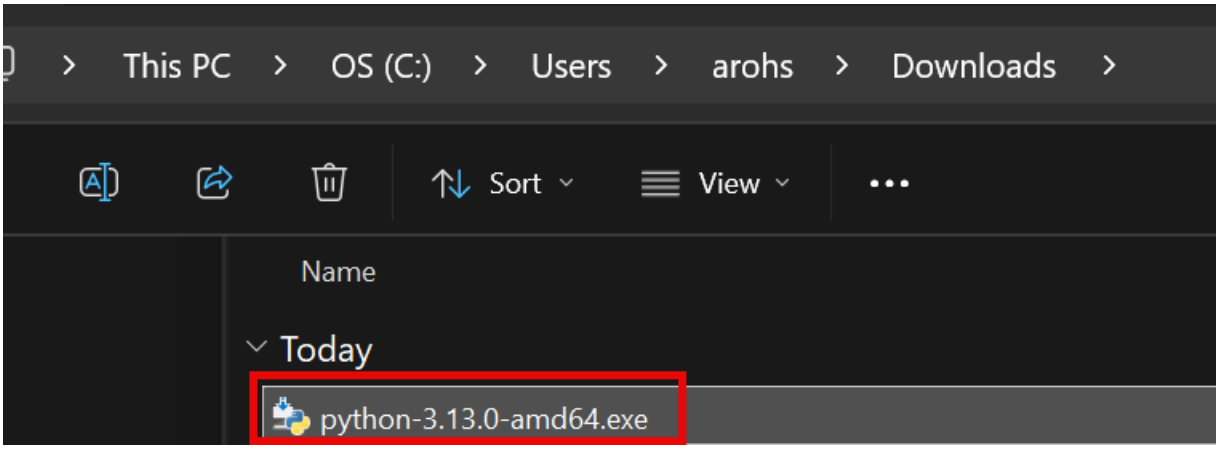
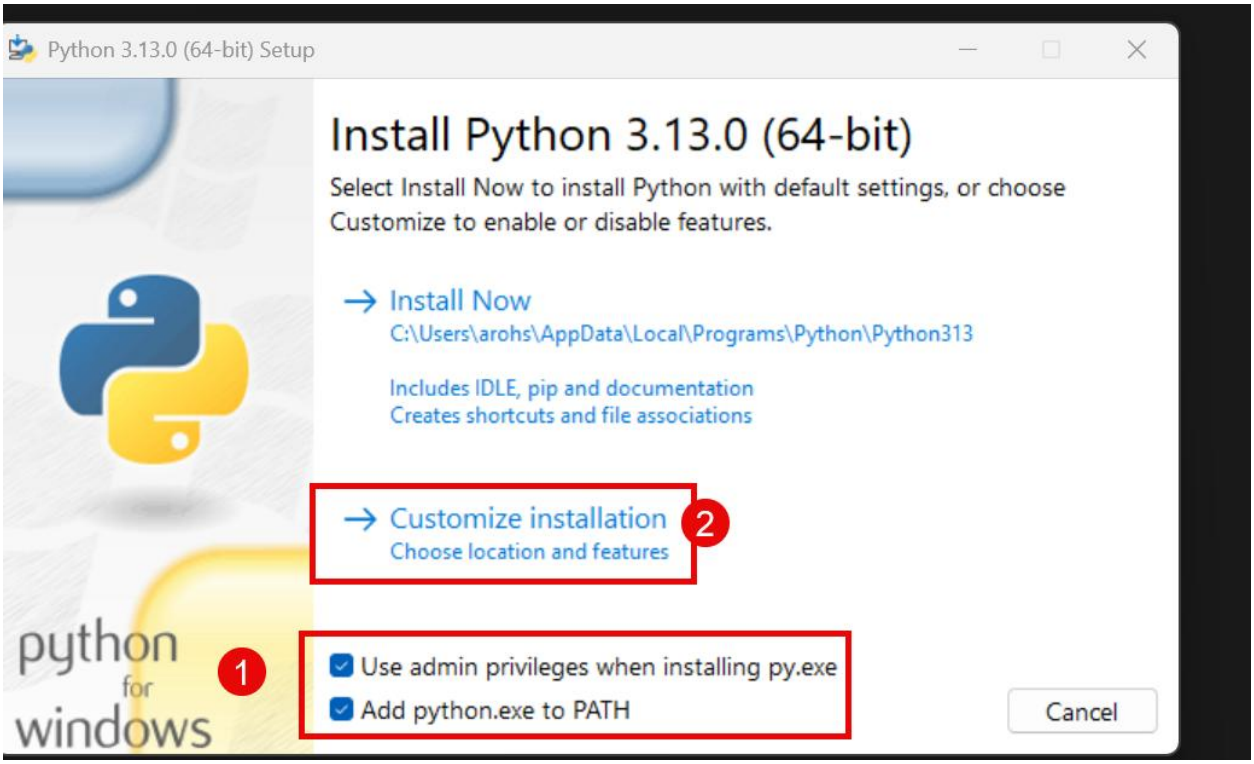
Develop a speech-to-text app using Python & Azure OpenAI

Prerequisites

1. Python

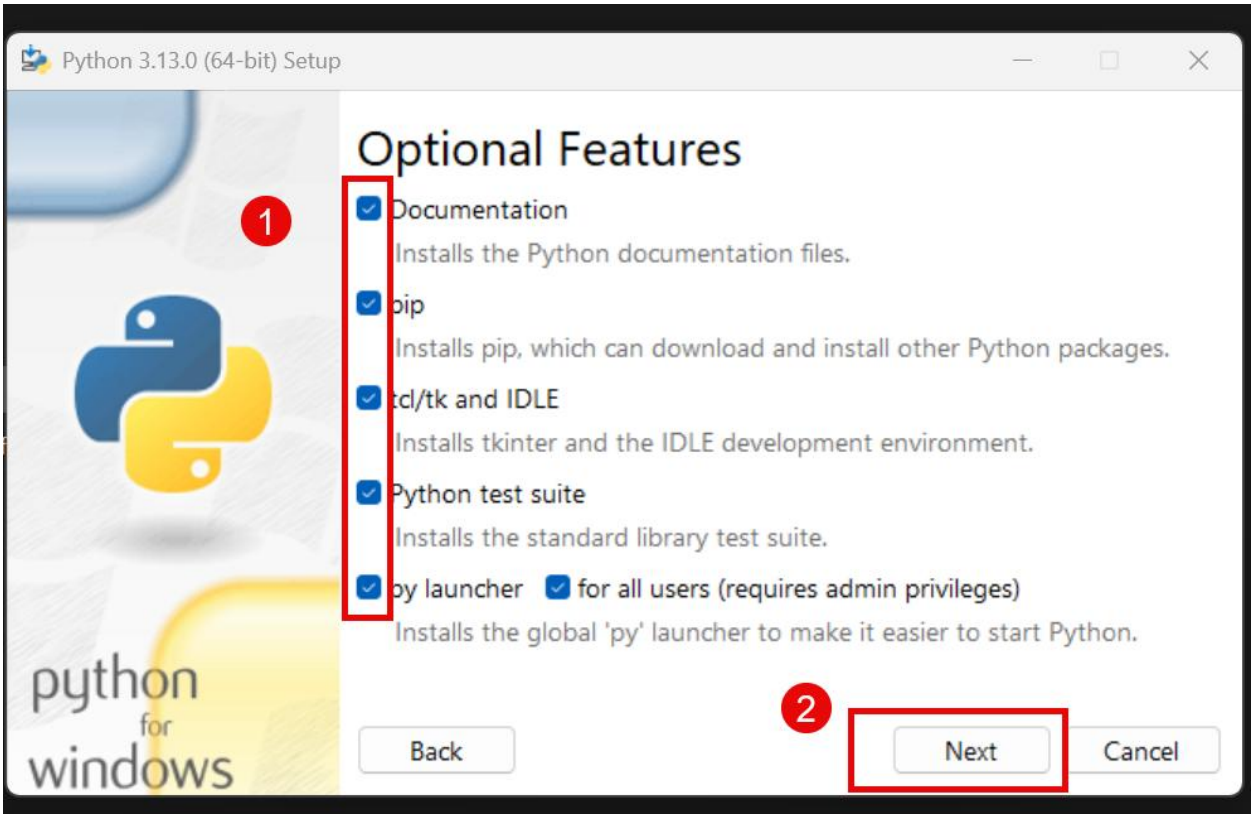
https://www.python.org/downloads/

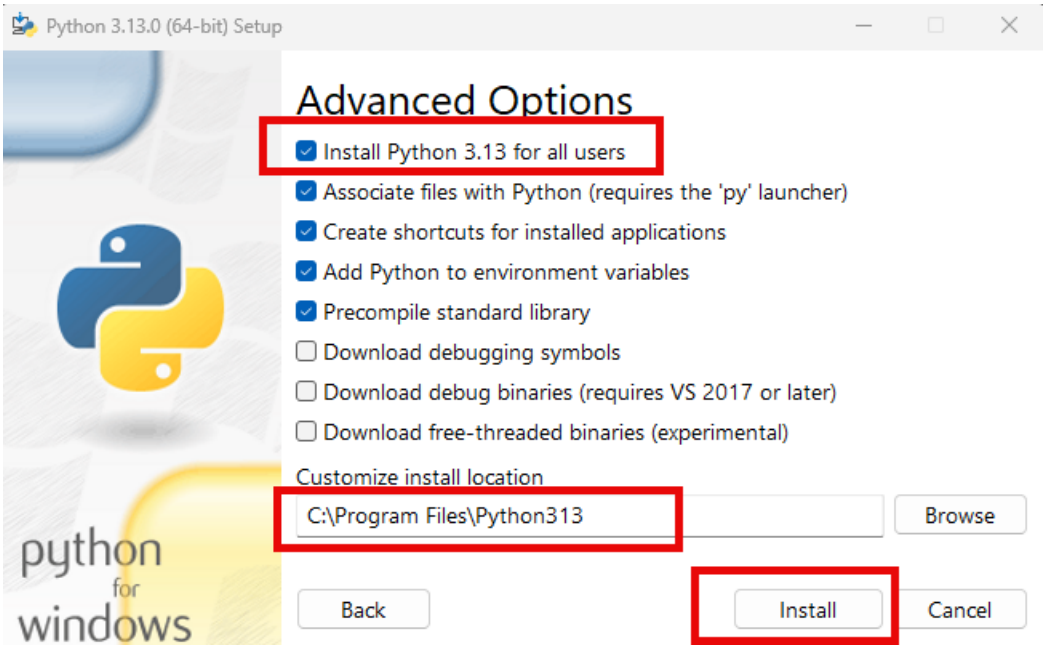


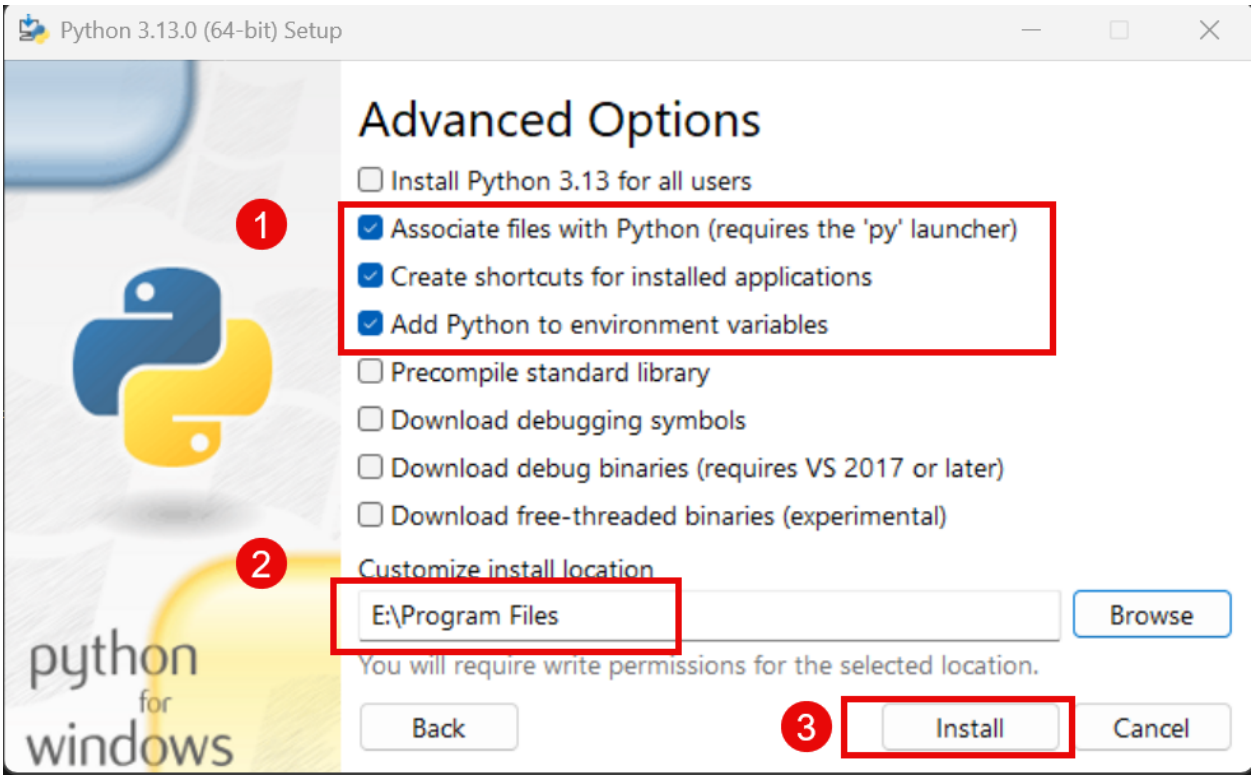


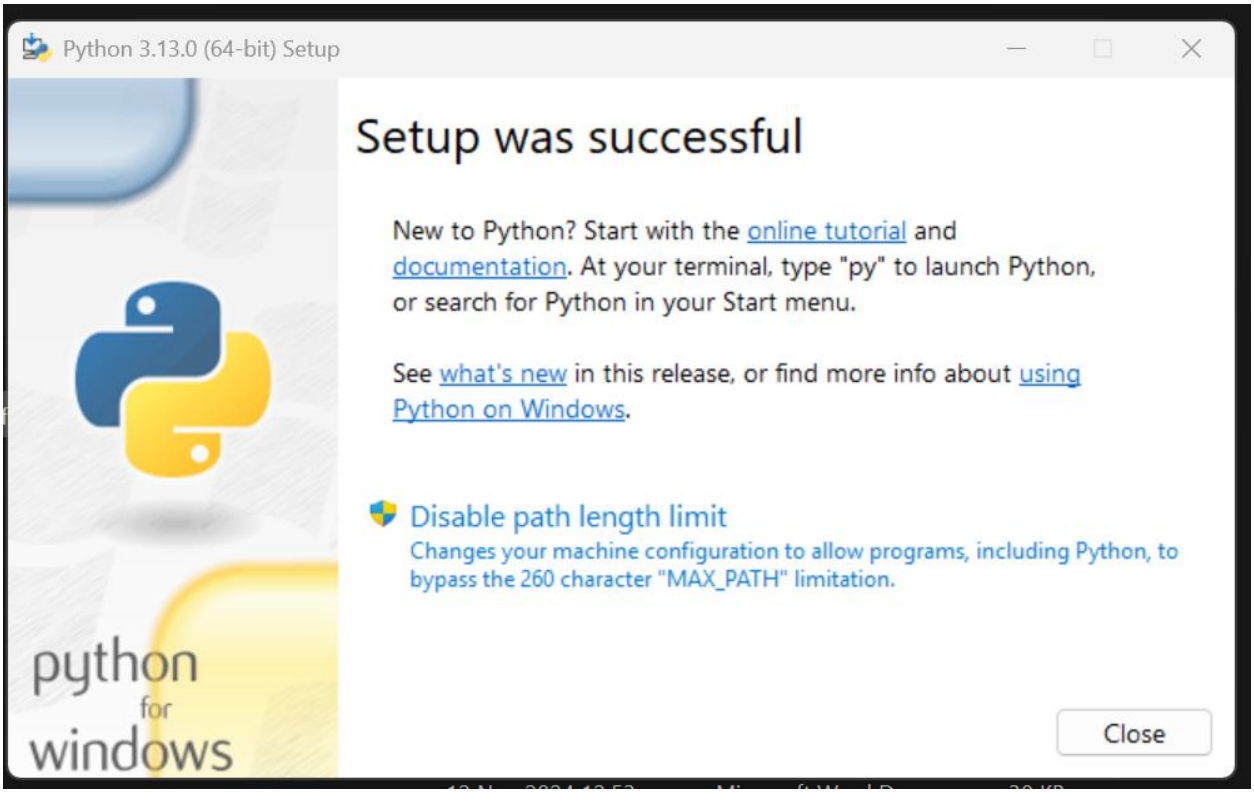
Optional Features

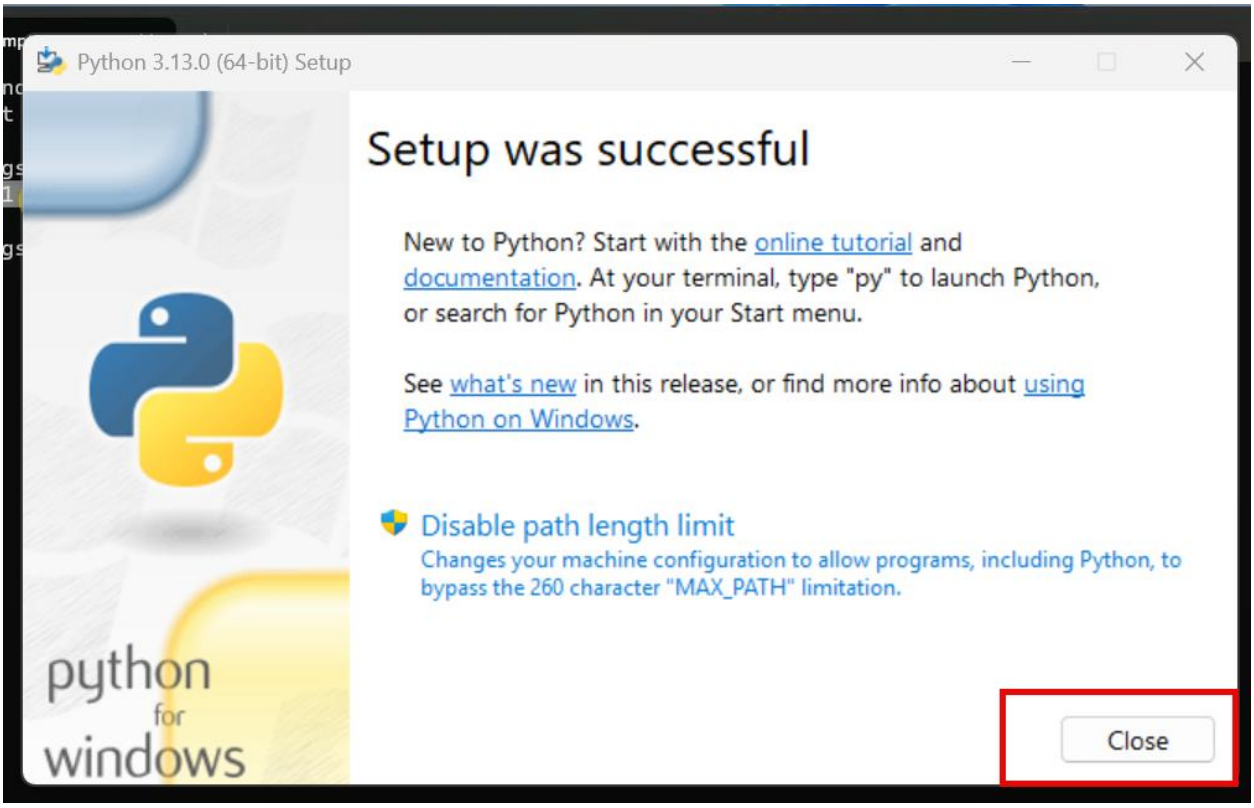
Optional Features

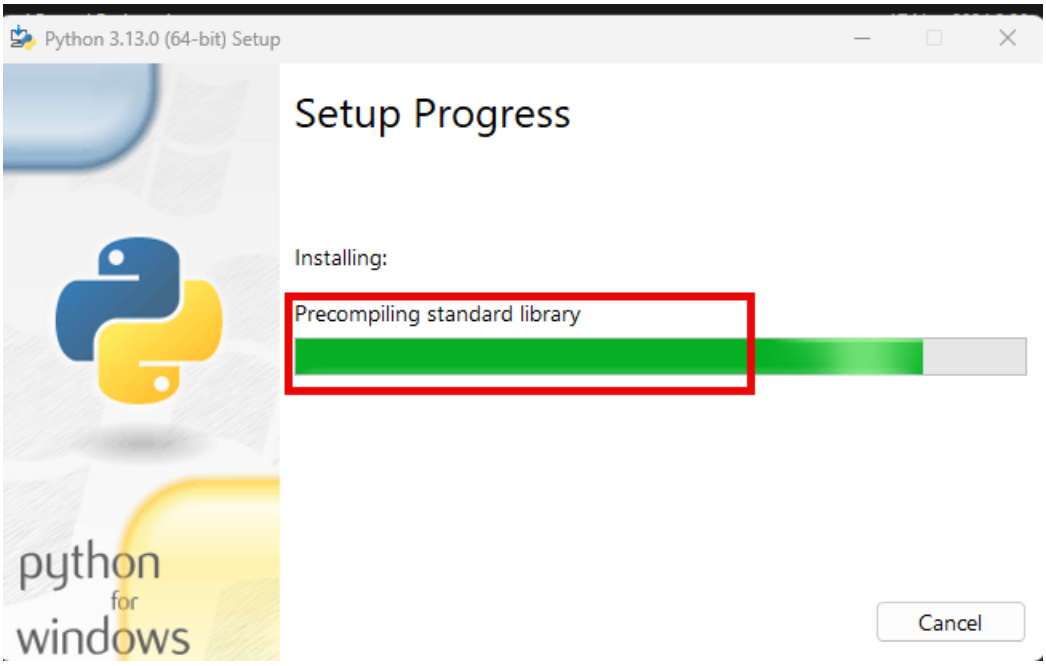




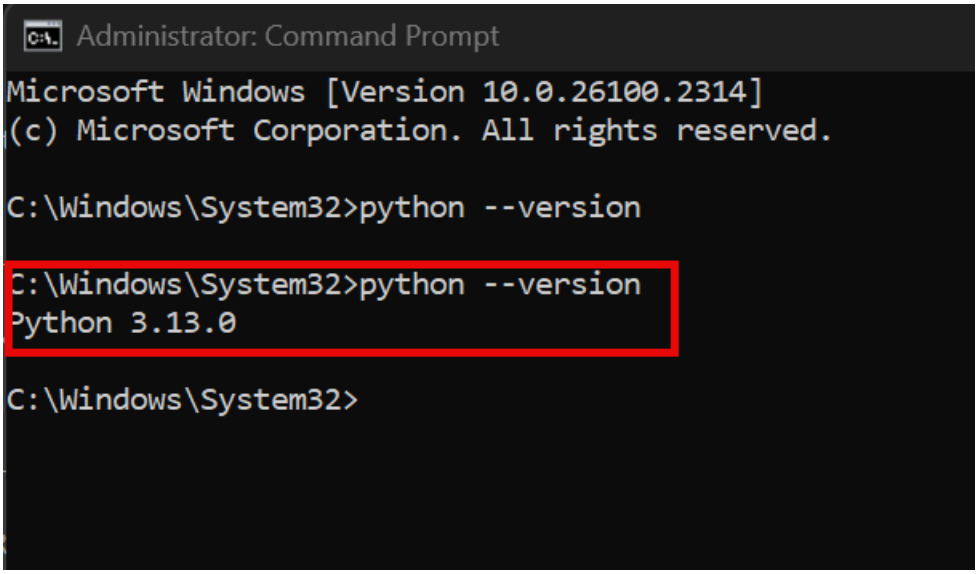

Python 3.13.0 (64-bit) Setup
Advanced Options
Install Python 3.13 for all users
1
Associate files with Python (requires the 'py' launcher)
Create shortcuts for installed applications
Add Python to environment variables
Precompile standard library
Download debugging symbols
Download debug binaries (requires VS 2017 or later)
Download free-threaded binaries (experimental)
2
Customize install location
E:\Program Files
Browse
You will require write permissions for the selected location.
python
for
windows
Back
3
Install
Cancel


Python 3.13.0 (64-bit) Setup
Setup was successful
New to Python? Start with the online tutorial and documentation. At your terminal, type "py" to launch Python, or search for Python in your Start menu.
See what's new in this release, or find more info about using Python on Windows.
Disable path length limit
Changes your machine configuration to allow programs, including Python, to bypass the 260 character "MAX_PATH" limitation.
python
for
windows
Close

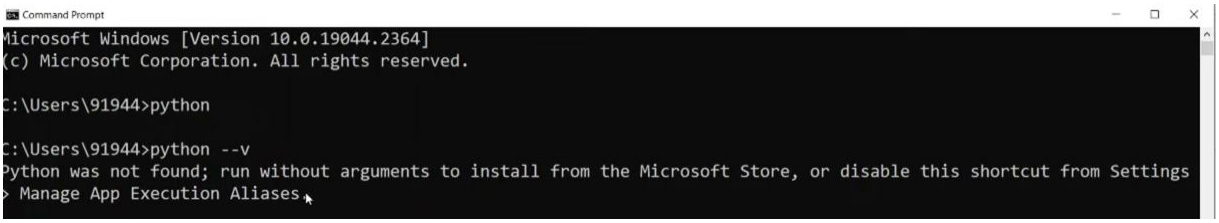




Check Python version



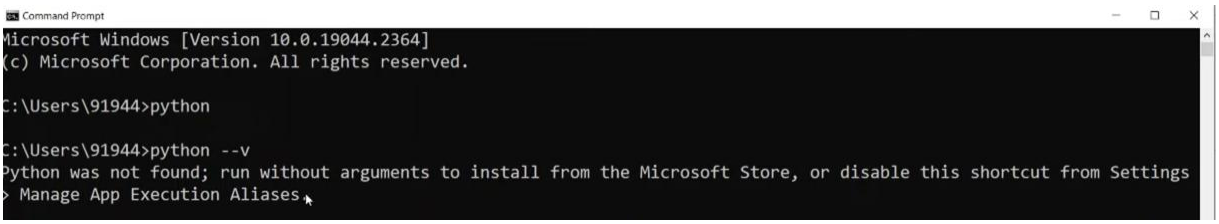
Issue



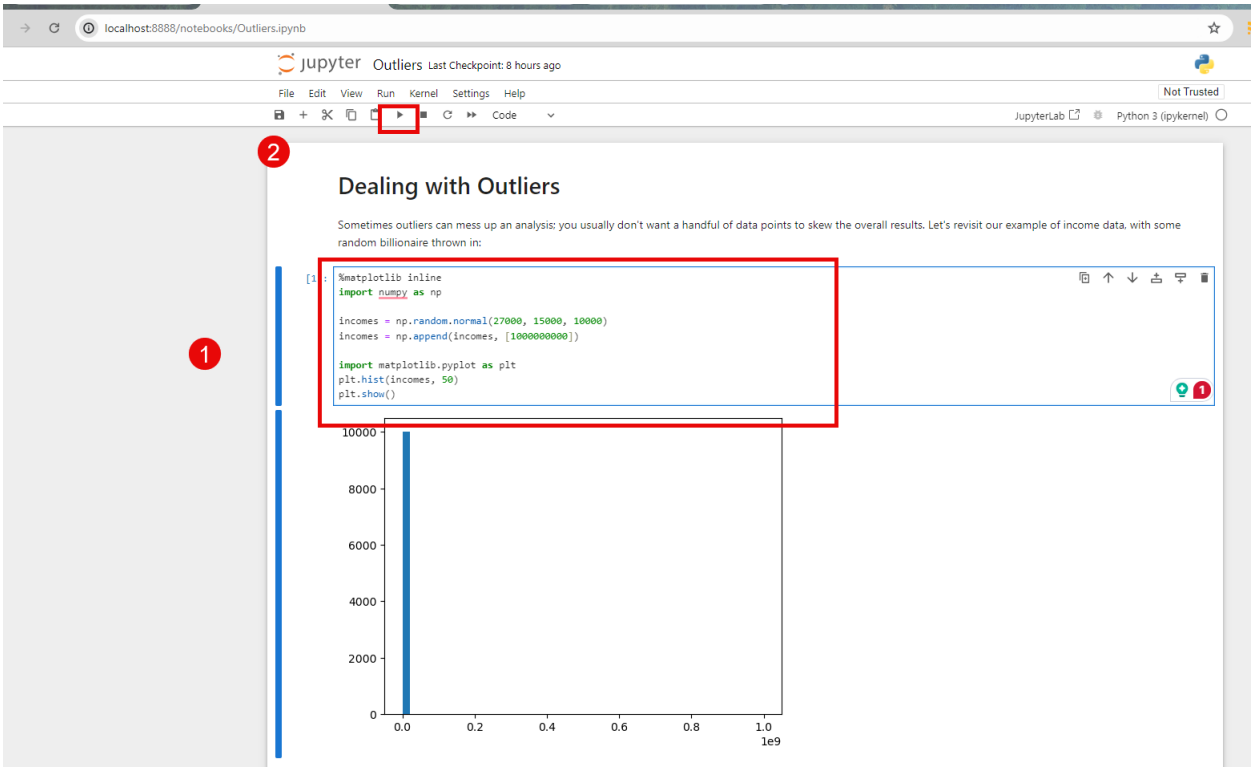
https://www.youtube.com/watch?v=9QrDn_hRSGs

https://www.python.org/downloads/

Issue



https://www.youtube.com/watch?v=9QrDn_hRSGs

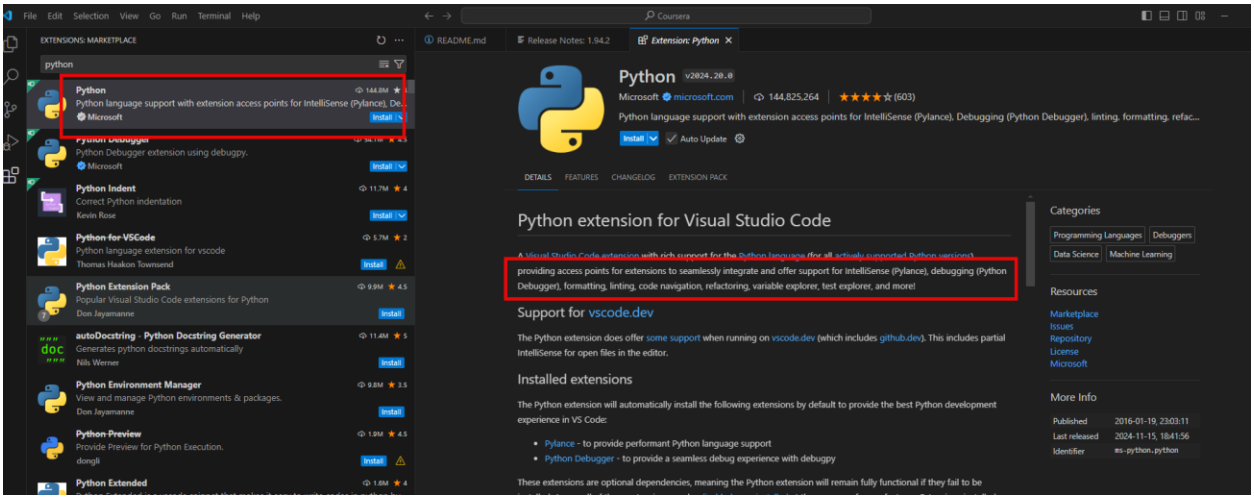


2. VS Code

https://www.youtube.com/watch?v=D2cwvpJSBX4 **Getting Started with Python in VS Code (Official Video)**

VS Code Extension

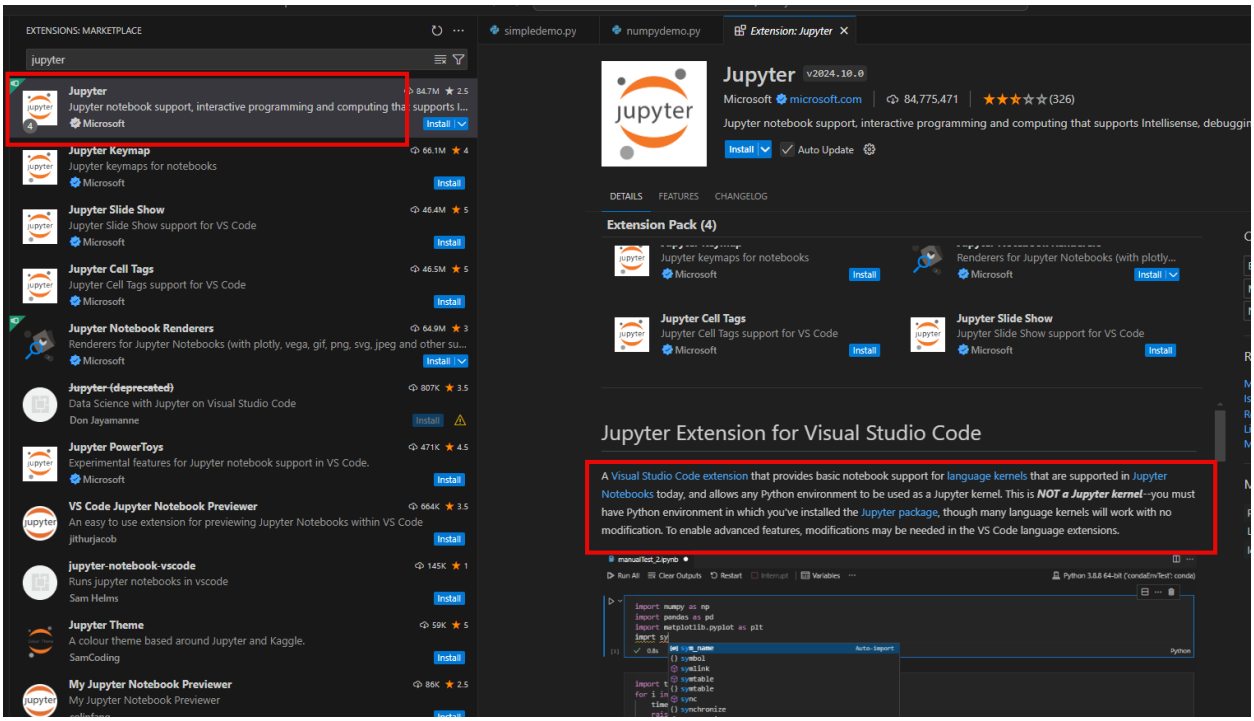
Python Extensions



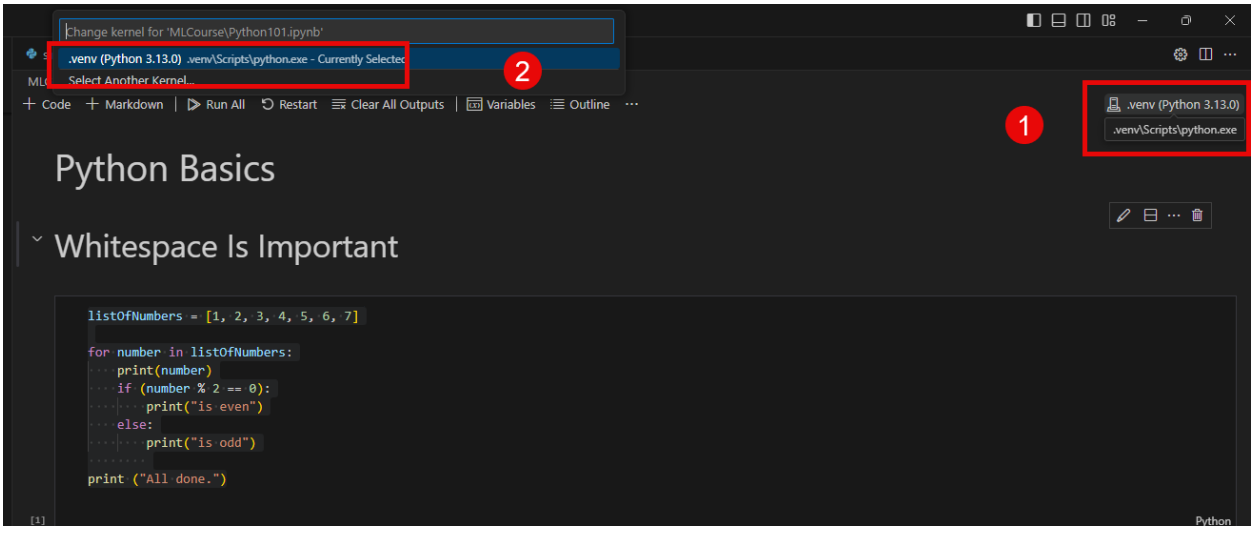
Why VS Code extension?

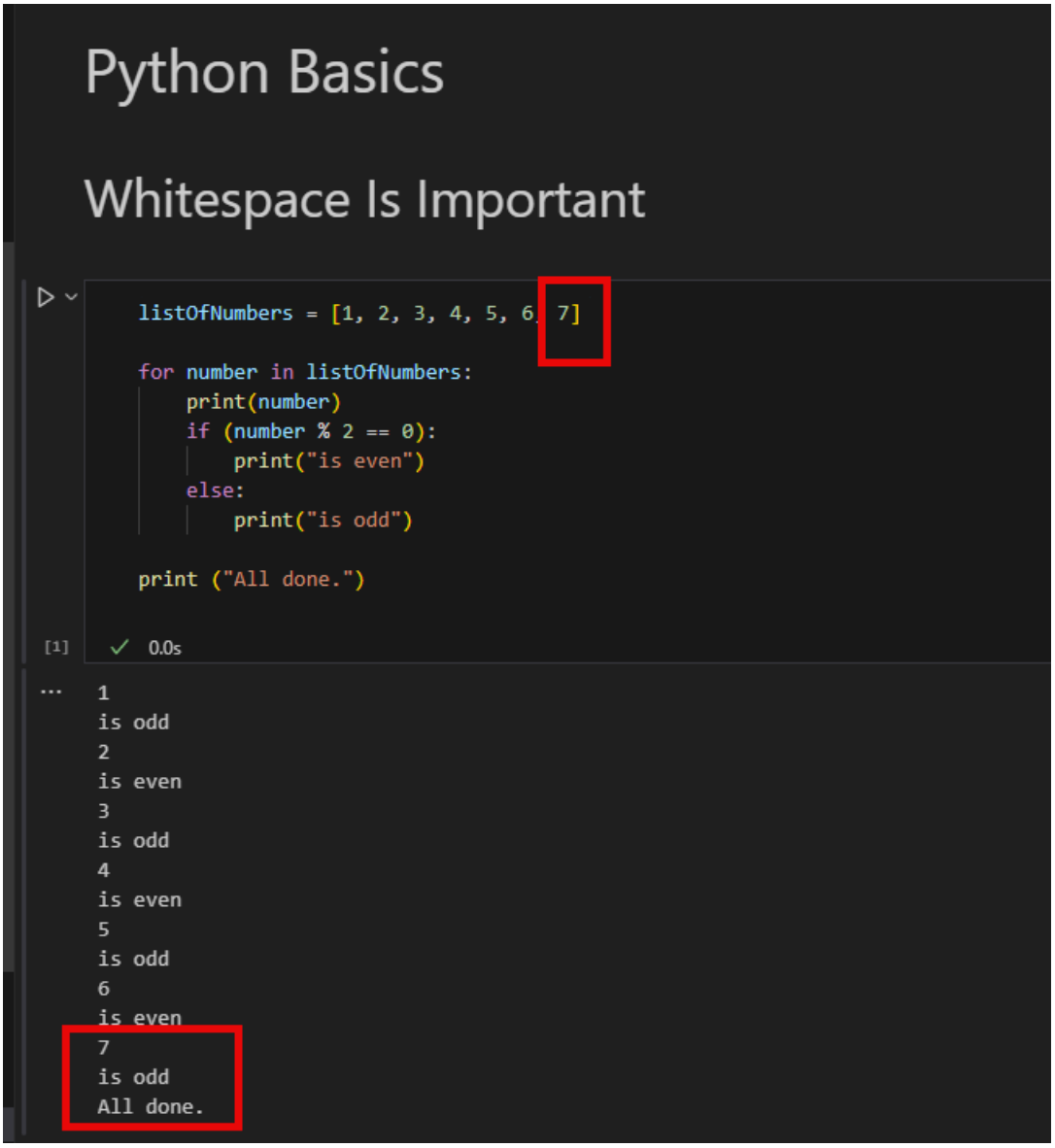
IntelliSense (Pylance), debugging (Python Debugger), formatting, linting, code navigation, refactoring, variable explorer, test explorer, and more!

Jupyter Extension



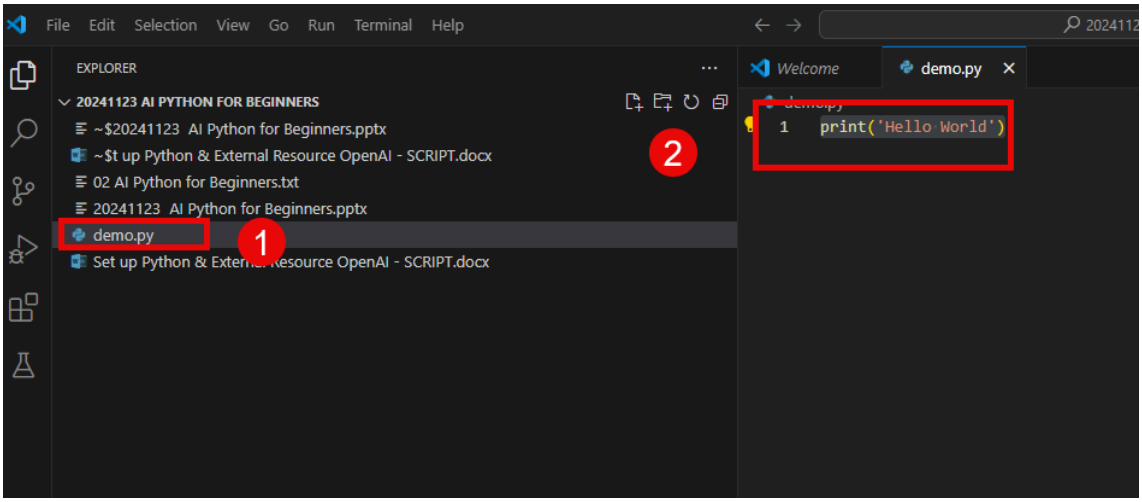
Select the Kernel



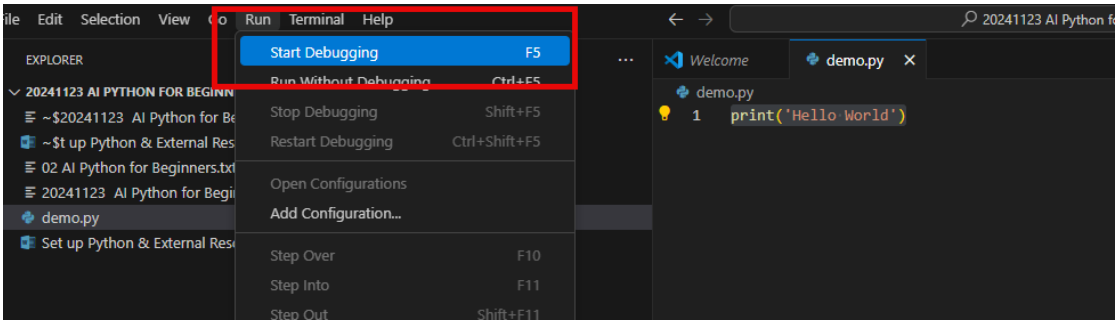


Hello World sample

- Create a folder
- Create a file name demo.py
- Write python code



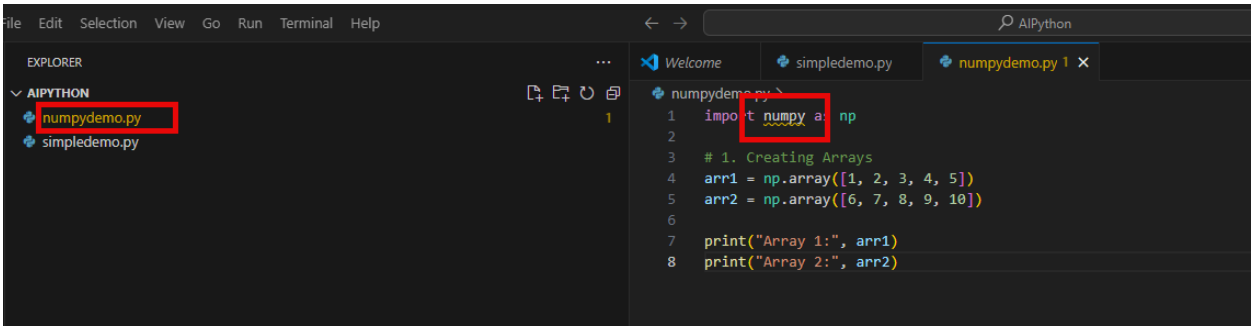
- Run the code by Pressing **F5**



-

Virtual Environments

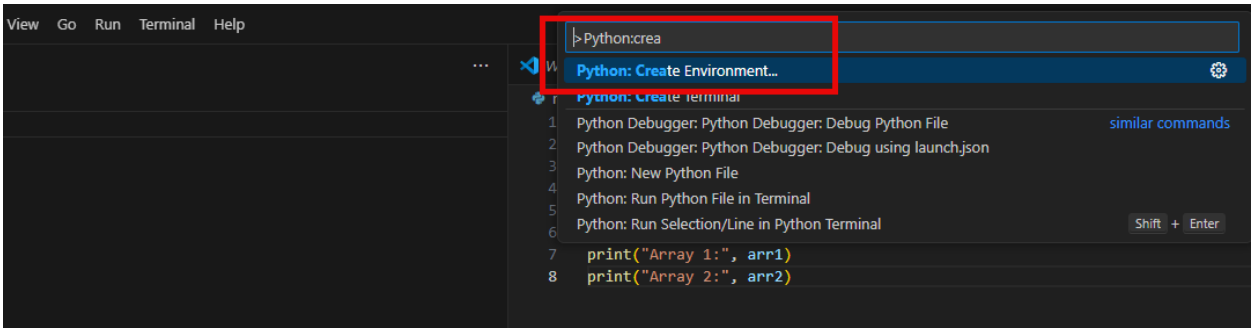
Step 1: Install necessary Python packages such as numpy, openai



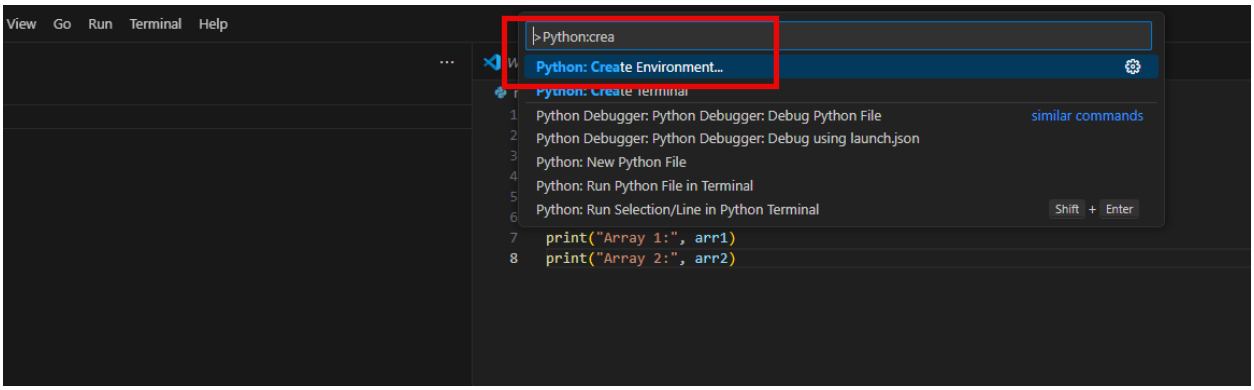
Numpy is not detected

Create a new virtual environment

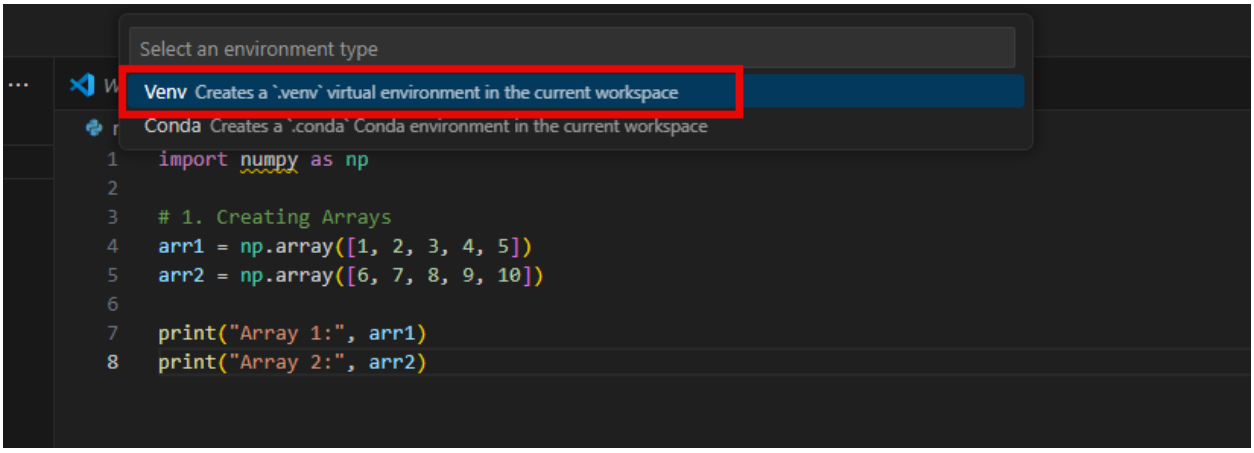
VS Code – Create Environment



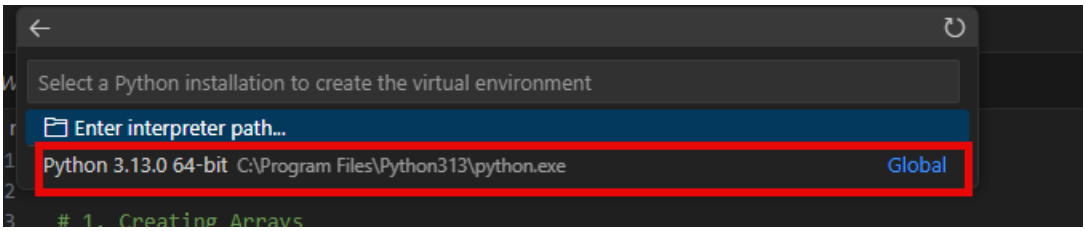
>Python:Create

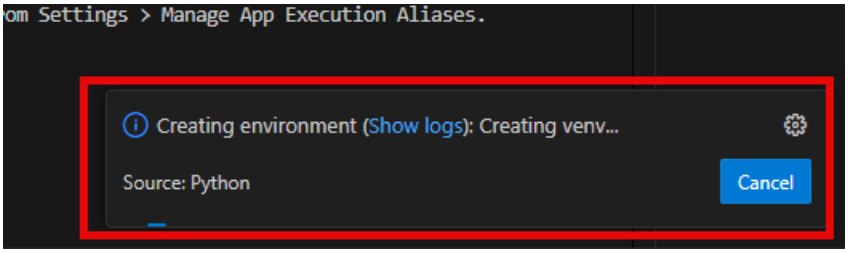


Create new environment with Venv

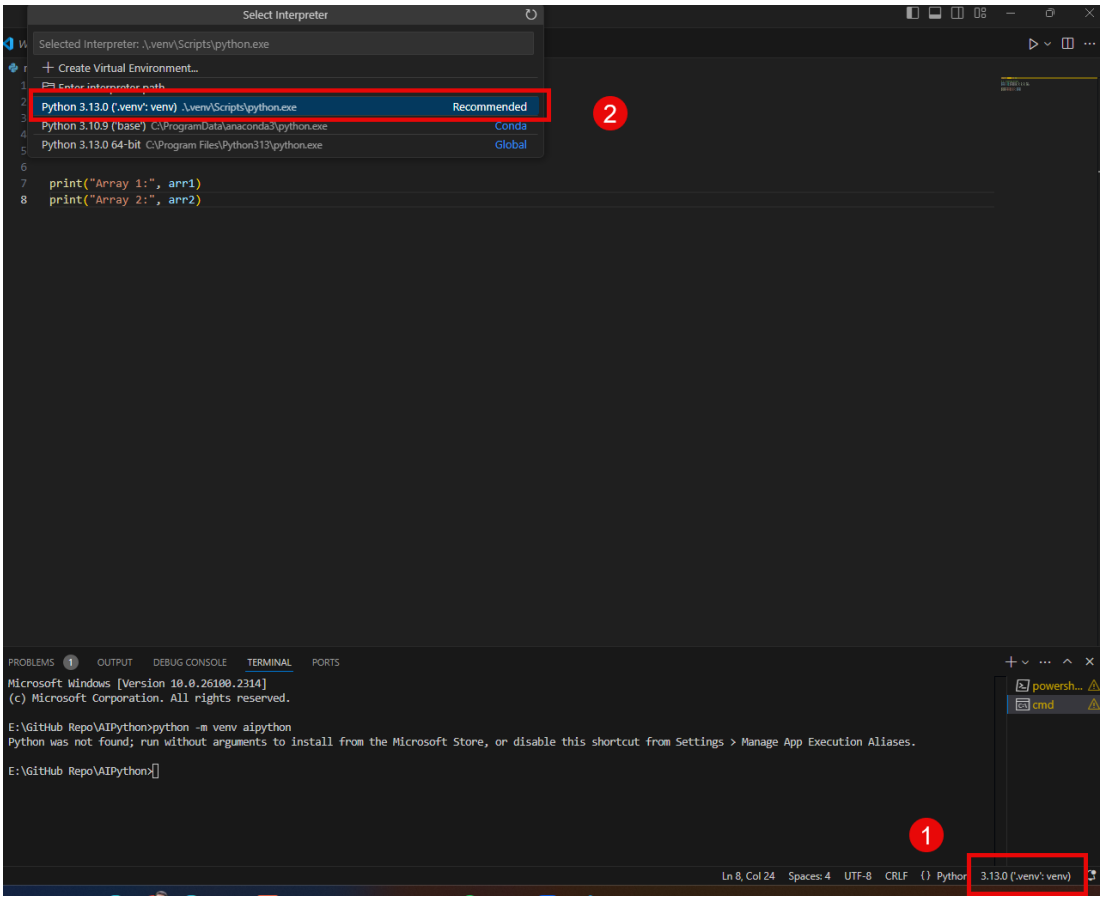


Select the Python interpreter path

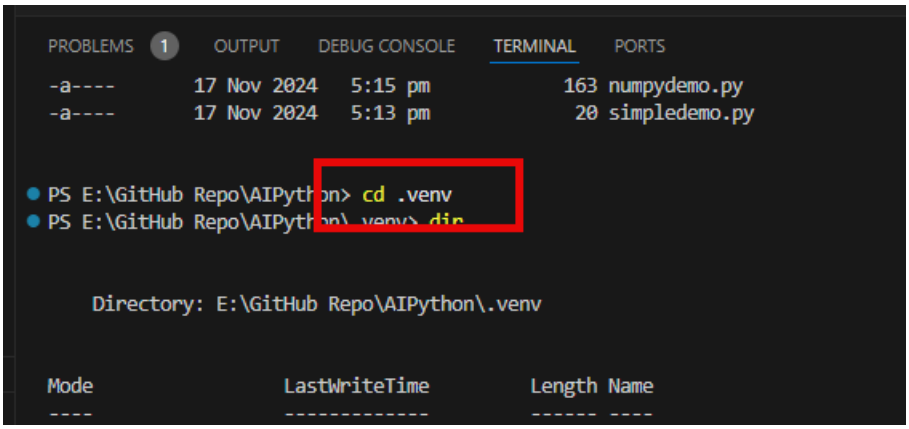




Select the Environment

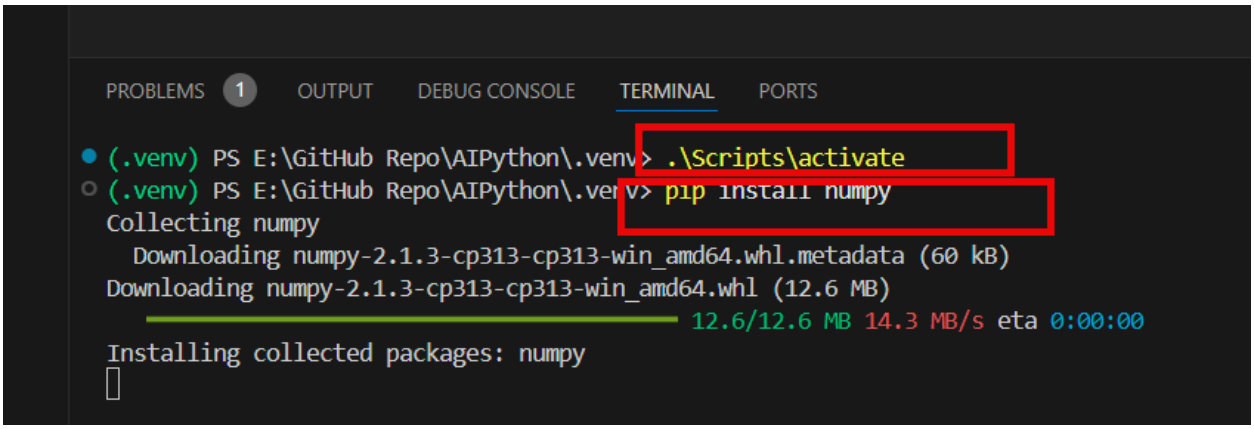


Navigate to .venv



Activate scripts

Install Python packages



Now numpy has been identified
Select the Environment

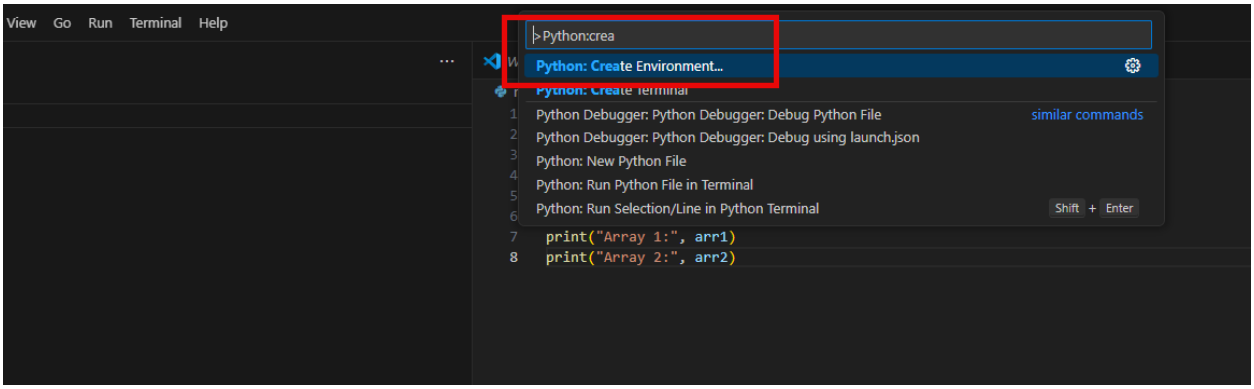
Why to use virtual environment

1. **Dependency Management:** Python virtual environments isolate **project dependencies**, preventing conflicts between packages.

2. **Consistency:** They ensure the same setup across **different development environments**, **avoiding version mismatches**.
3. **Security:** Virtual environments limit the scope of installed packages, reducing the risk of

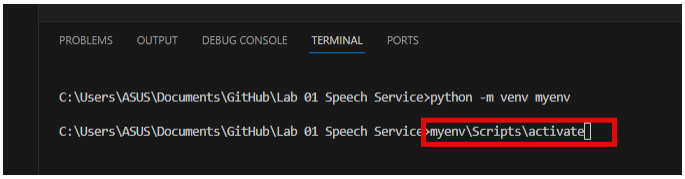
VS Code

>Python:Create



python -m venv myenv

Activate scripts



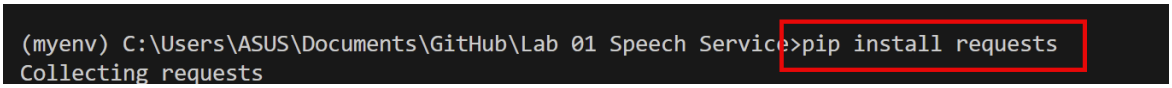
myenv\Scripts\activate (backslash)

Install requests modules

Requests module

Pip install **requests**

allows you to send HTTP requests using Python.



os module – NOT Required

os is already installed on Windows

https://stackoverflow.com/questions/48010748/how-to-install-the-os-module

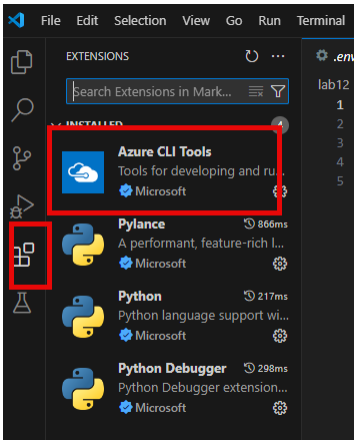
pip install load_dot module

```
(myenv) C:\Users\ASUS\Documents\GitHub\Lab 01 Speech Service>pip install load_dotenv
Collecting load_dotenv
  Downloading load_dotenv-0.1.0-py3-none-any.whl.metadata (1.9 kB)
```

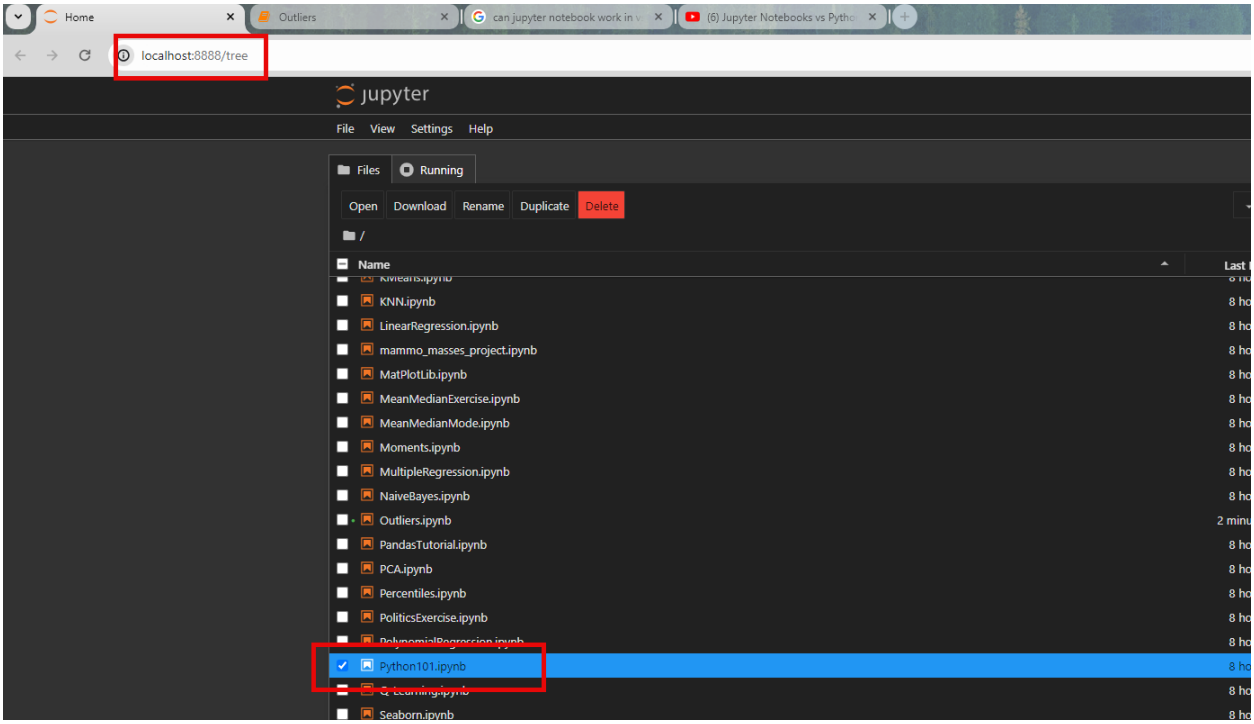
Json – Not Required

https://code.visualstudio.com/

Azure CLI tools

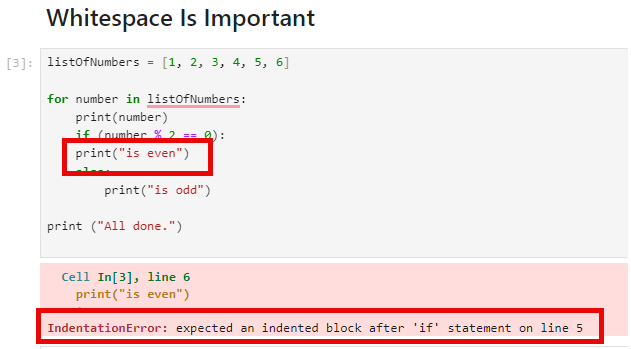


Python 101



Whitespace is important

1. IndentationError
   tabs, spaces to group together

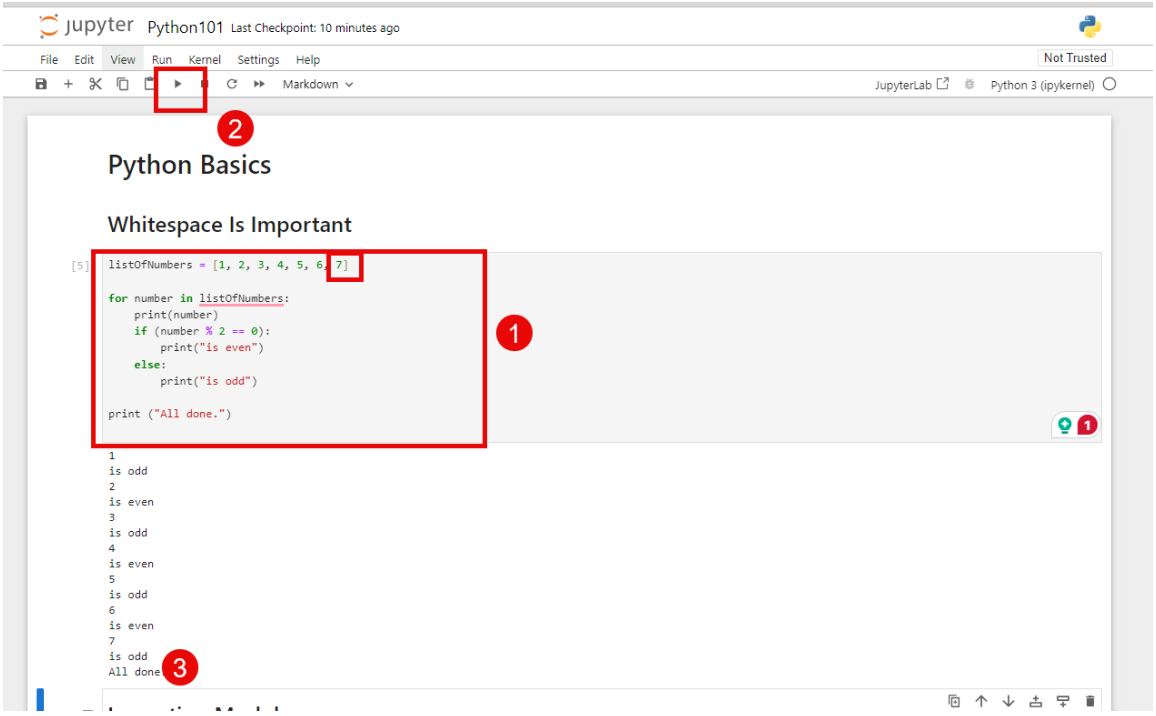


2. **No curly Braces** ({}) like C# and PowerApps

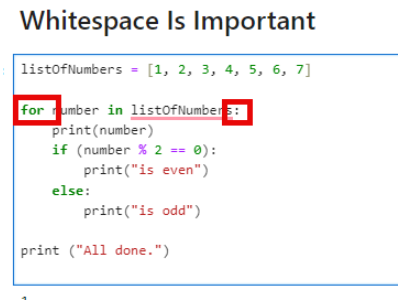
3. listOfNumbers = [1, 2, 3, 4, 5, 6]
   Lists (Just like Array)

4. **No character to terminate** line like C# (;)

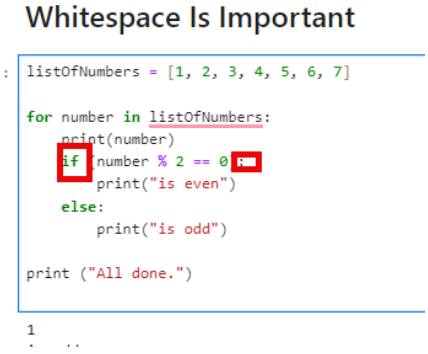
5. Add one more number in list and Execute



6. For statement is similar to other languages but needs : as the end



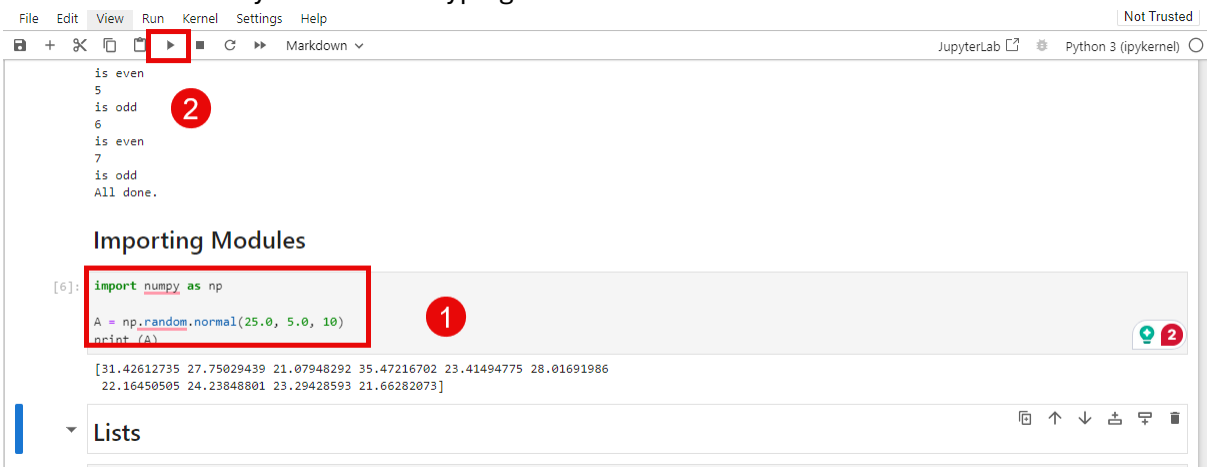
7. Same thing applies to if statement, it needs :



8. We **don't need to define variables ahead of** time nor the **data types**.

9. Python is what's called typed language.
10. But you can cast one variable to another. For e.g. string to integer.

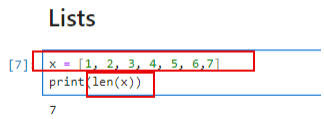
Import module

- Use **import** command (similar of **using** in C#)
- Define alias to save yourself some typing too.
- 

Lists

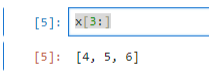
a list is a versatile and **mutable data structure** that can contain a **collection of items**, such as integers, strings, or even other lists

- **len()** (returns the number of elements in that list)



- **slice** part of list by using specific items
  for e.g. x[:3] will extract 1st three item

- x[3:] will last 3 items from the list



- x[-2:] negative from the list.

```
[11]: x[-2:]

[11]: [6, 7]
```

- **extend()**: **Appends the elements** of an iterable (such as another list) to the **end of the list.**

```
[12]: x.extend([7,8])
      x

[12]: [1, 2, 3, 4, 5, 6, 7, 7, 8]
```

- **append()**: Adds an element to the **end of the list.**

```
[8]: x.append(9)
     x

[8]: [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

- List can contain just about any type you want. You can have a list of list.

```
[14]: y = [10, 11, 12]
      listOfLists = [x, y]
      listOfLists

[14]: [[1, 2, 3, 4, 5, 6, 7, 7, 8], [10, 11, 12]]
```

- To retrieve an element of a list, use [] bracket operator

```
[14]: y = [10, 11, 12]
      listOfLists = [x, y]
      listOfLists

[14]: [[1, 2, 3, 4, 5, 6, 7, 7, 8], [10, 11, 12]]

[15]: y[0]

[15]: 10
```

- Sort() - Sorts the elements of the list in **ascending order (0 – 10s)** (by default).

```
16]: z = [3, 2, 1, 0]
     z.sort()
     z

16]: [0, 1, 2, 3]
```

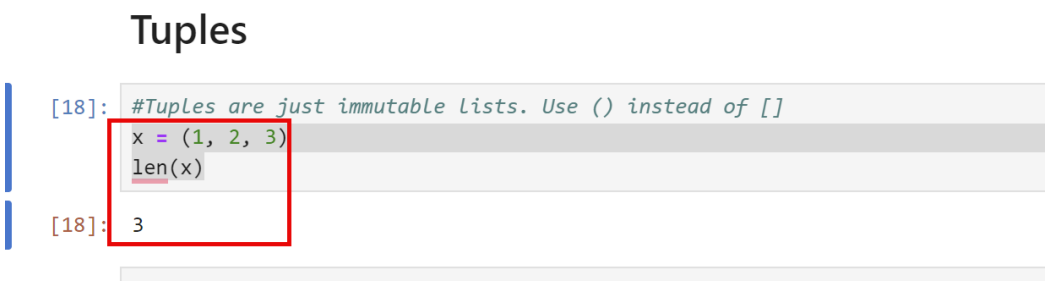
- Reverse() - reverses the order of the elements in the list.

```
[16]: z = [3, 2, 1, 0]
      z.sort()
      z

[16]: [0, 1, 2, 3]

[17]: z.sort(reverse=True)
      z

[17]: [3, 2, 1, 0]
```

Tuples

- tuples are just **immutable lists**. Use () instead of []
  Similar as C# Tuple.

- For e.g.
  x = (1, 2, 3)
  len(x)



- You can reference elements in a tuple in the same way that you would in a list as well.

```
[14]: y = (4, 5, 6)
      y[2]

[14]: 6
```

- You can make a **list of tuples** if you desire.

```
[14]: y = (4, 5, 6)
      y[2]

[14]: 6
```

- Another common use of tuples is in passing around **group of variables that you want to keep together.**

```
[16]: (age, income) = "32,120000".split(',')
      print(age)
      print(income)

      32
      120000
```

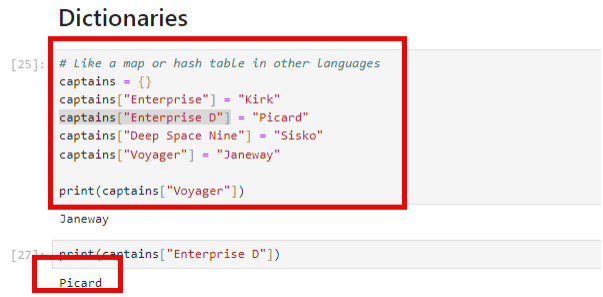
Split is same as C# and PowerApps

Dictionaries

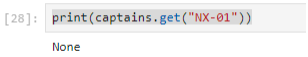
- Useful data structure in Python is the dictionary.
- Another language use has map or hash table. C# the same name Dictionary
- Defined as curly braces {}
-

```
[17]: # Like a map or hash table in other languages
      captains = {}
      captains["Enterprise"] = "Kirk"
      captains["Enterprise D"] = "Picard"
      captains["Deep Space Nine"] = "Sisko"
      captains["Voyager"] = "Janeway"

      print(captains["Voyager"])

      Janeway
```

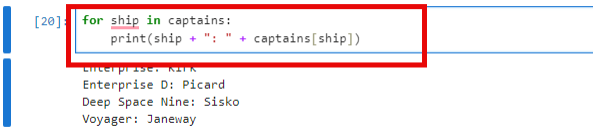
- You can get the value from dictionary as follows:



- If no value from dictionary, then it will be returned as none



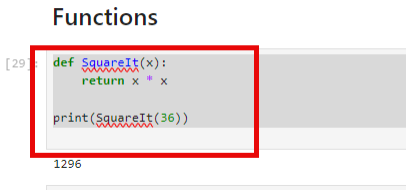
- You Iterate through all dictionary items by using for looping



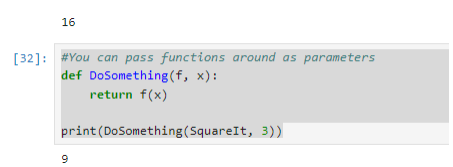
Functions

- To define a function in Python, you use the **def** keyword followed by the *function name* and parentheses containing **any parameters** the function takes

- For e.g.



- Some cool things with Function. You can pass another function to a function as parameter



- Lambda :

- In simpler terms, a lambda function in Python is a small**, anonymous function** that can have any number of parameters but only one expression.

- For e.g.

```
[33]: #Lambda functions let you inline simple functions
      print(DoSomething(lambda x: x * x * x, 3))

      27
```

Boolean

- In Python, the boolean data type represents a binary value, which can be either **True** or **False**.

- Booleans are used to represent the truth values of logical expressions.

Boolean Expressions

```
[34]: print(1 == 3)

      False

[37]: print(True or False)

      True

[40]: print(1 == 3)

      False

[42]: if 1 == 3:
          print("How did that happen?")
      elif 1 > 3:
          print("Yikes")
      else:
          print("All is well with the world")

      All is well with the world
```

Looping

- The **for loop** is used to iterate **over a sequence** (such as a list, tuple, string, or range) or any iterable object.

Looping

```
[28]: for i in range(10):
          print(i)

      0
      1
      2
      3
      4
      5
      6
      7
      8
```

- **Break :** terminates the loop prematurely when a certain condition is met.

```
9
[44]: for x in range(10):
          if (x == 1):
              continue
          if (x > 5):
              break
          print(x)
0
2
3
4
5
```

- while loop - The while loop repeatedly executes a block of code as long as a specified condition is True.

```
5
[30]: x = 0
      while (x < 10):
          print(x)
          x += 1
0
1
```