

The background of the slide is a complex, abstract pattern composed of numerous triangles in various shades of purple, blue, and black. The triangles are arranged in a way that creates a sense of depth and movement, with some triangles appearing to point towards the viewer and others away. The overall effect is a modern, digital aesthetic.

# **AI LAB WEEK 1**

**Dr. Mubashir Ahmad**

# HOW TO INSTALL PYTHON

## Installing Anaconda on Windows

This section details the installation of the Anaconda distribution of Python on Windows 10. I think the Anaconda distribution of Python is the best option for problem solvers who want to use Python. Anaconda is free (although the download is large which can take time) and can be installed on school or work computers where you don't have administrator access or the ability to install new programs. Anaconda comes bundled with about 600 packages pre-installed including NumPy, Matplotlib and SymPy. These three packages are very useful for problem solvers and will be discussed in subsequent chapters.

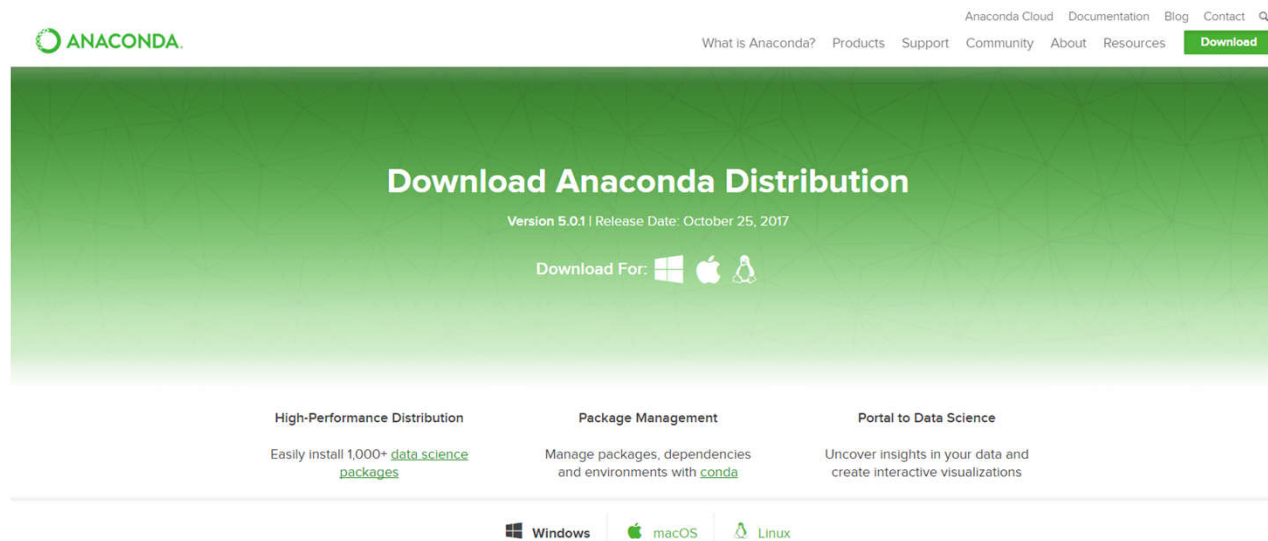
# STEPS:

1. Visit [Anaconda.com/downloads](https://anaconda.com/downloads)
2. Select Windows
3. Download the **.exe** installer
4. Open and run the **.exe** installer
5. Open the **Anaconda Prompt** and run some Python code

# 1. VISIT THE ANACONDA DOWNLOADS PAGE

Go to the following link: [Anaconda.com/downloads](https://anaconda.com/downloads)

The Anaconda Downloads Page will look something like this:



## 2. SELECT WINDOWS



Windows



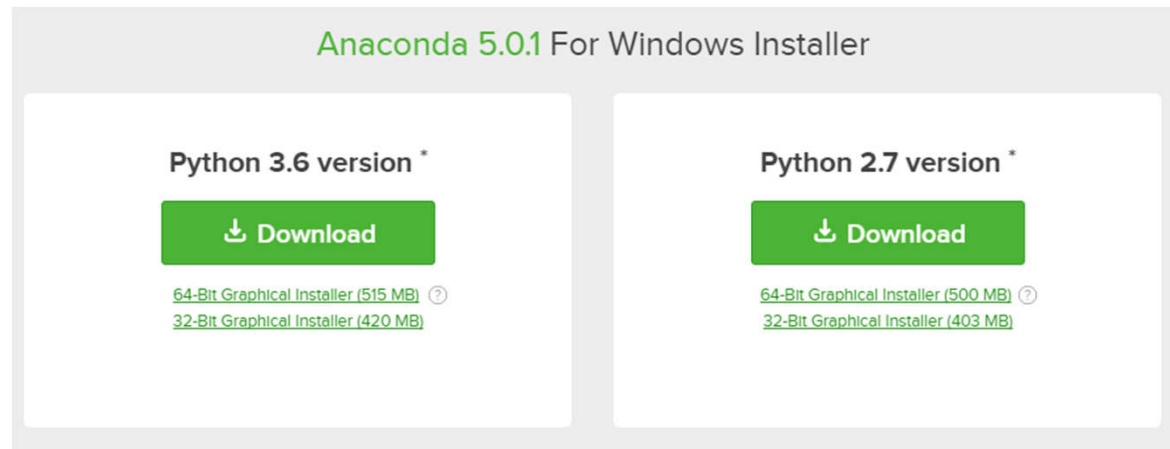
macOS



Linux

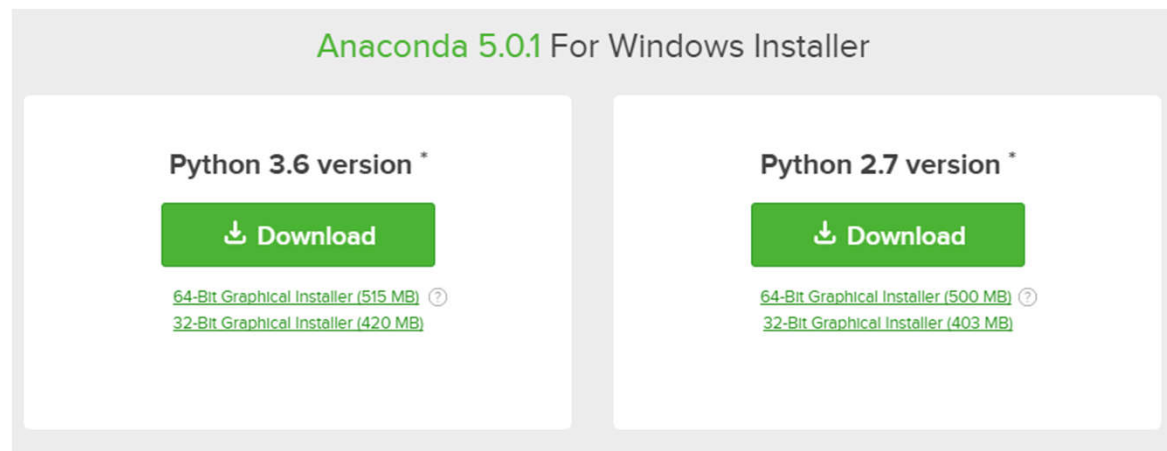
# 3. DOWNLOAD

Download the most recent Python 3 release. At the time of writing, the most recent release was the Python 3.6 Version. Python 2.7 is legacy Python. For problem solvers, select the Python 3.6 version. If you are unsure if your computer is running a 64-bit or 32-bit version of Windows, select 64-bit as 64-bit Windows is most common.

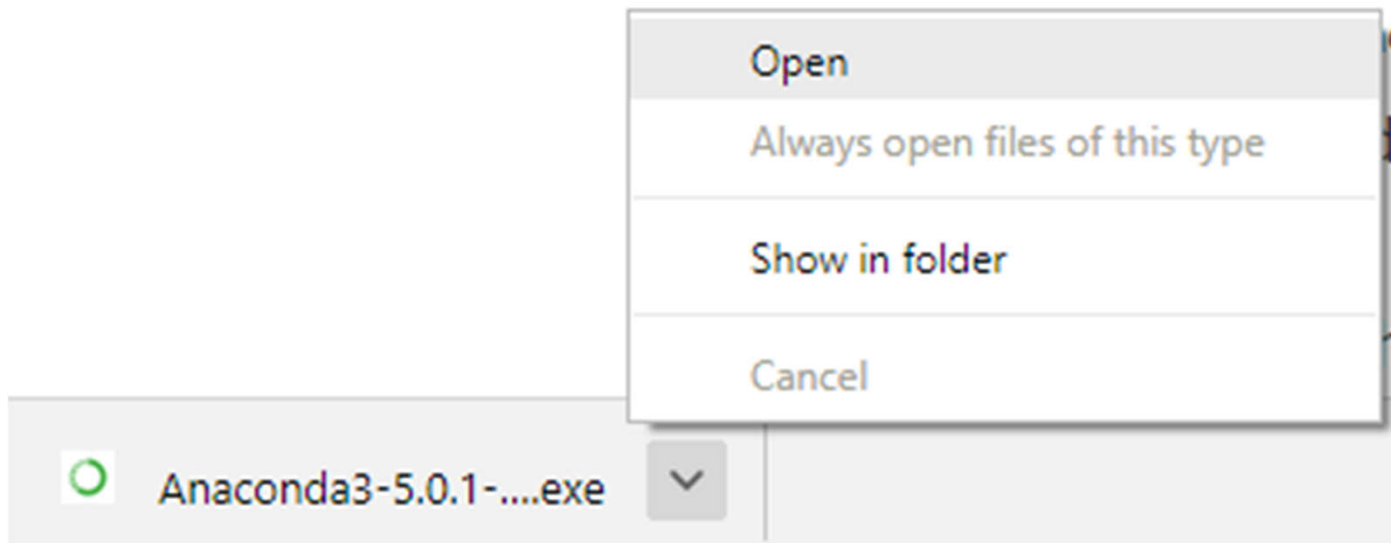


# 3. DOWNLOAD

Download the most recent Python 3 release. At the time of writing, the most recent release was the Python 3.6 Version. Python 2.7 is legacy Python. For problem solvers, select the Python 3.6 version. If you are unsure if your computer is running a 64-bit or 32-bit version of Windows, select 64-bit as 64-bit Windows is most common.



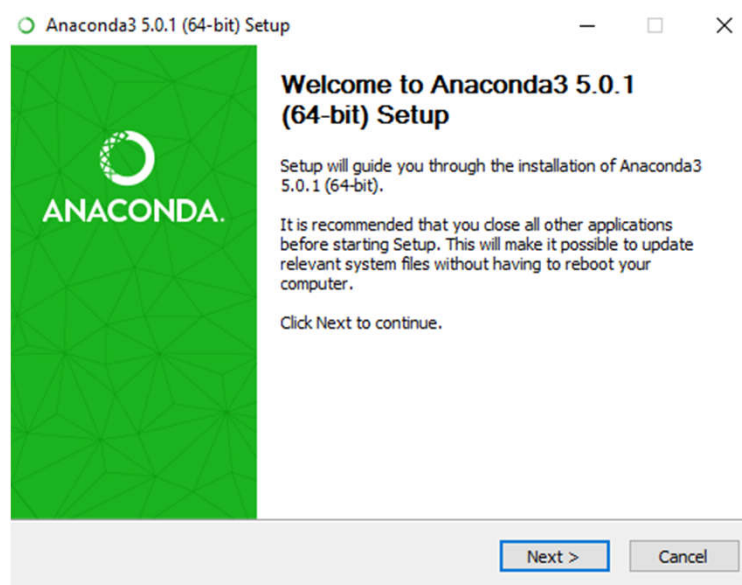
# 4. OPEN AND RUN THE INSTALLER



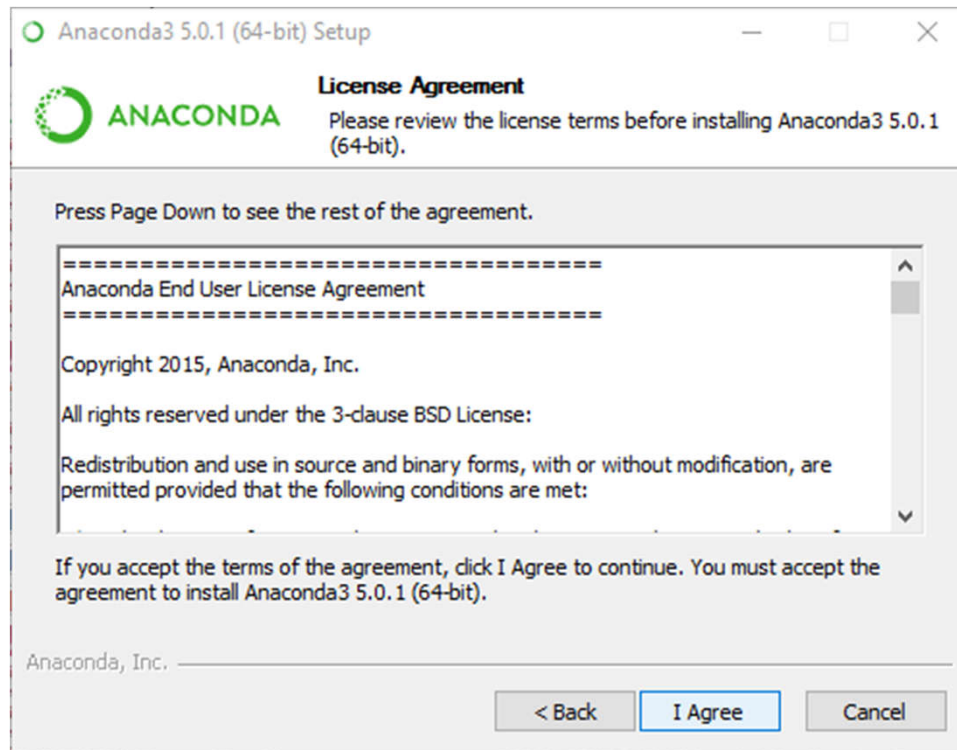


# 4. OPEN AND RUN THE INSTALLER

At the beginning of the install, you need to click **Next** to confirm the installation.

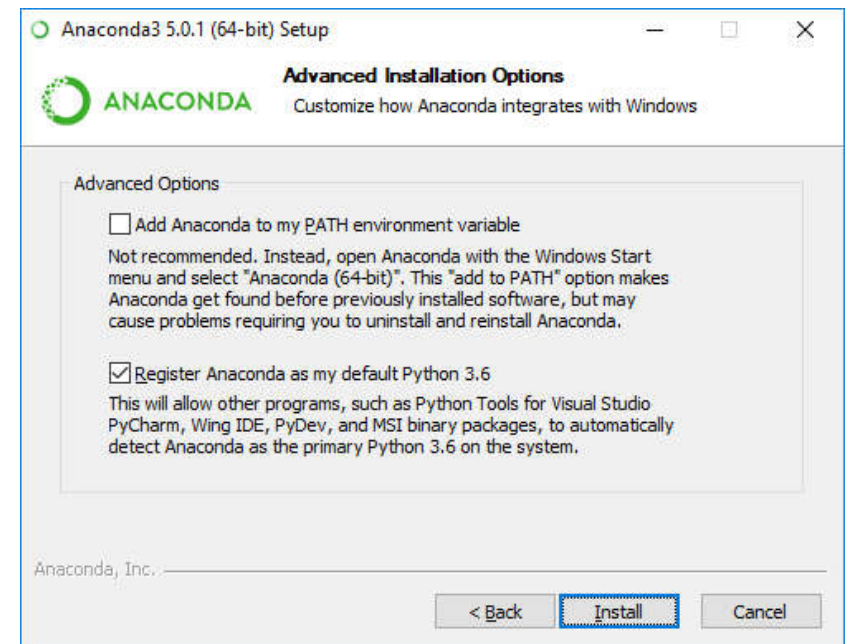


# 4. OPEN AND RUN THE INSTALLER

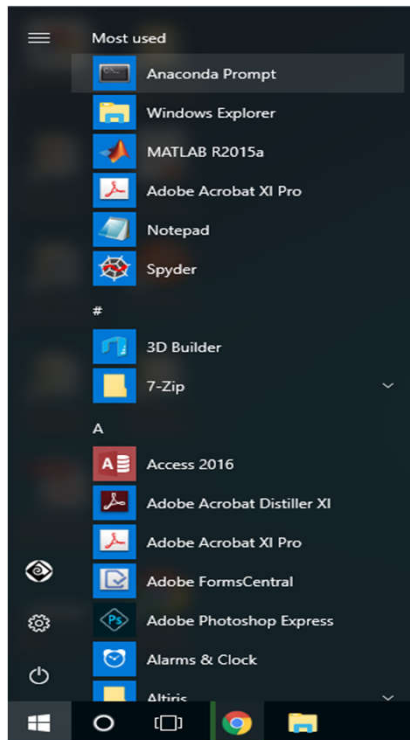


# 4. OPEN AND RUN THE INSTALLER

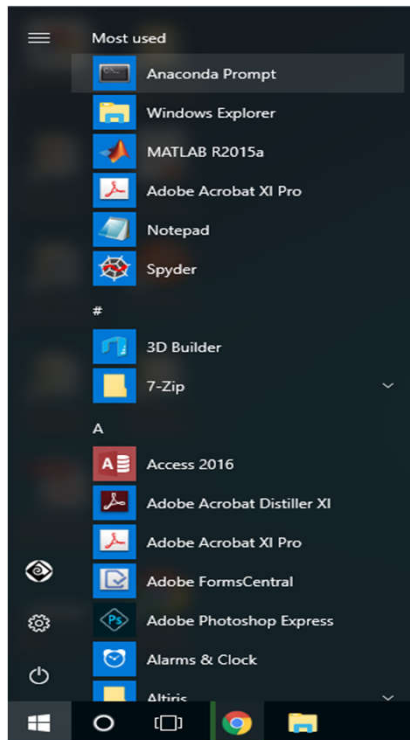
At the Advanced Installation Options screen, I recommend that you **do not** check “Add Anaconda to my PATH environment variable”



# 5. OPEN THE ANACONDA PROMPT FROM THE WINDOWS START MENU



# 5. OPEN THE ANACONDA PROMPT FROM THE WINDOWS START MENU





Home



Environments



Learning



Community

NEW

Documentation

Anaconda Blog



You  
Tube



All applications ▾

on

base (root) ▾

Channels



Launch

Launch



Qt Console

↗ 5.4.2

PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.



Spyder

5.4.3

Scientific PYTHON Development Environment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features



C:\Users\Mubashir\.spyder-py3

Name	Date Modified
> autosave	3/25/2023 11:31 PM
> config	3/23/2023 12:00 PM
> lsp_paths	3/23/2023 12:00 PM
> plugins	3/23/2023 12:00 PM
> spyder.lock	3/5/2024 10:02 PM
help	3/25/2023 11:54 PM
history_internal.py	3/5/2024 10:03 PM
history.py	3/5/2024 10:04 PM
langconfig	3/23/2023 12:00 PM
onlinehelp	3/25/2023 11:54 PM
path	3/5/2024 10:03 PM
pdb_history.sqlite	3/5/2024 10:03 PM
template.py	3/23/2023 12:00 PM
untitled0.py	3/5/2024 10:06 PM

C:\Users\Mubashir\.spyder-py3\variables.py

variables.py X

```

1 x = 5
2 y = "John"
3 print(x)
4 print(y)
5
6

```

Name	Type	Size	Value
x	int	1	5
y	str	4	John

Variable Explorer Plots



variables.py

Console 1/A X

```

In [1]: runfile('C:/Users/Mubashir/.spyder-py3/temp.py', wdir='C:/Users/Mubashir/.spyder-py3')
5
John

In [2]: runfile('C:/Users/Mubashir/.spyder-py3/variables.py', wdir='C:/Users/Mubashir/.spyder-py3')
5
John

In [3]:

```

history.py

```

main.py', wdir='F:/Users/WeChat Files/wxid_0faygccbxgee22/FileStorage/File/2023-03')

## ---(Tue Mar 5 22:03:07 2024)---
runfile('C:/Users/Mubashir/.spyder-py3/temp.py', wdir='C:/Users/Mubashir/.spyder-py3')
runfile('C:/Users/Mubashir/.spyder-py3/variables.py', wdir='C:/Users/Mubashir/.spyder-py3')

```

History Help

# WHAT IS PYTHON?

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.



# WHAT CAN PYTHON DO?

- Python can be used on a server to create web applications.
- Python can be used alongside software to create workflows.
- Python can connect to database systems. It can also read and modify files.
- Python can be used to handle big data and perform complex mathematics.
- Python can be used for rapid prototyping, or for production-ready software development.

# **PYTHON SYNTAX COMPARED TO OTHER PROGRAMMING LANGUAGES**

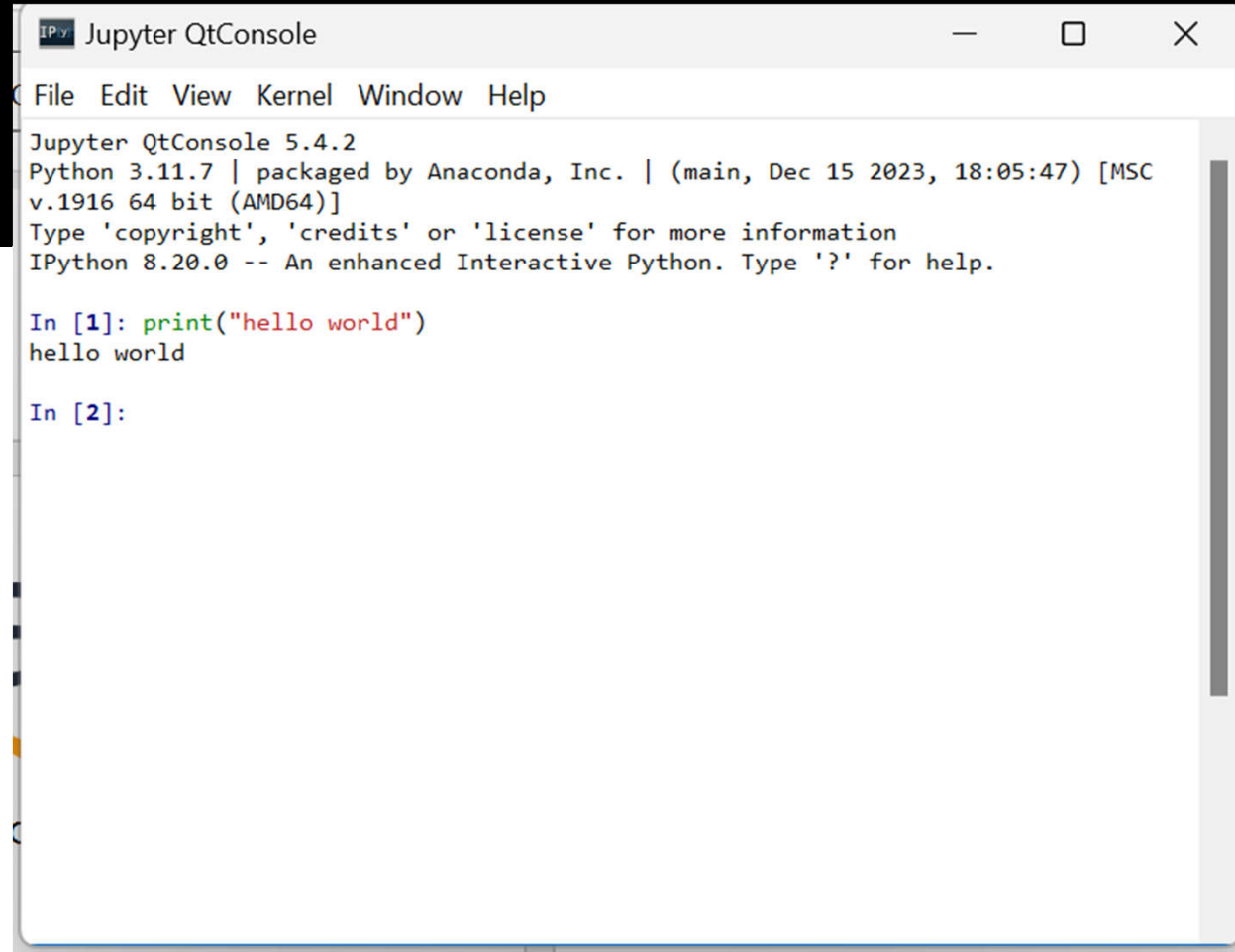
- Python was designed for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

# EXAMPLE

## Example

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

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





The screenshot shows a Jupyter QtConsole window with the following content:



```
Jupyter QtConsole 5.4.2  
Python 3.11.7 | packaged by Anaconda, Inc. | (main, Dec 15 2023, 18:05:47) [MSC  
v.1916 64 bit (AMD64)]  
Type 'copyright', 'credits' or 'license' for more information  
IPython 8.20.0 -- An enhanced Interactive Python. Type '?' for help.  
  
In [1]: print("hello world")  
hello world  
  
In [2]:
```

 Home

 Environments

 Learning

 Community

-  
- Documentation
- Anaconda Blog



All applications ▾ on base (root) ▾ Channels 

Launch

Launch



JupyterLab

4.0.11

An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.



Notebook

7.0.8

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.





Filter files by name



 Name  Last Modified

■ Favorites	last ye. ▲
■ Jedi	11 months ag
■ Links	last ye.
■ Music	last ye.
■ myenv	11 months ag
■ OneDrive	5 months ag
■ OneDrive...	11 days ag
■ Pictures	7 months ag
■ Pycharm...	9 months ag
■ Searches	3 months ag
■ venv	4 years ag
■ Videos	9 months ag
■ VirtualBo...	11 months ag
■ workspace	last ye.

Untitled.ipynb










 Code
 


Notebook   Python 3 (ipykernel) 

```
[1]: print("hello world")
```

hello world

[ ]:

Simple ☐ 0  1  Python 3 (ipykernel) | Idle

Mode: Edit  Ln 1, Col 1 Untitled.ipynb 1 

# CREATING A COMMENT

## Example

```
#This is a comment  
print("Hello, World!")
```

Try it Yourself »

## Example

```
print("Hello, World!") #This is a comment
```

Try it Yourself »

## Example

```
#print("Hello, World!")  
print("Cheers, Mate!")
```

## Example

```
"""  
This is a comment  
written in  
more than just one line  
"""  
print("Hello, World!")
```

# PYTHON OPERATORS

Operator	Name	Example
+	Addition	$x + y$
-	Subtraction	$x - y$
*	Multiplication	$x * y$
/	Division	$x / y$
%	Modulus	$x \% y$
**	Exponentiation	$x ** y$
//	Floor division	$x // y$

## Example

```
print(10 + 5)
```

# PYTHON OPERATORS

```
x = 2  
y = 5  
  
print(x ** y) #same as 2*2*2*2*2
```

---

```
x = 15  
y = 2  
  
print(x // y)  
  
#the floor division // rounds the result down to the nearest whole number
```



# PYTHON ASSIGNMENT OPERATORS

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3

# PYTHON COMPARISON OPERATORS

Operator	Name	Example
<code>==</code>	Equal	<code>x == y</code>
<code>!=</code>	Not equal	<code>x != y</code>
<code>&gt;</code>	Greater than	<code>x &gt; y</code>
<code>&lt;</code>	Less than	<code>x &lt; y</code>
<code>&gt;=</code>	Greater than or equal to	<code>x &gt;= y</code>
<code>&lt;=</code>	Less than or equal to	<code>x &lt;= y</code>

# PYTHON LOGICAL OPERATORS

Operator	Description	Example
and	Returns True if both statements are true	<code>x &lt; 5 and x &lt; 10</code>
or	Returns True if one of the statements is true	<code>x &lt; 5 or x &lt; 4</code>
not	Reverse the result, returns False if the result is true	<code>not(x &lt; 5 and x &lt; 10)</code>

# PYTHON IDENTITY OPERATORS

Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

# PYTHON BITWISE OPERATORS

Operator	Name	Description	Example
&	AND	Sets each bit to 1 if both bits are 1	x & y
	OR	Sets each bit to 1 if one of two bits is 1	x   y
^	XOR	Sets each bit to 1 if only one of two bits is 1	x ^ y
~	NOT	Inverts all the bits	~x
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off	x << 2
>>	Signed right shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off	x >> 2

# READING INPUT FROM THE KEYBOARD

Python

```
1 >>> number = input("Enter a number: ")
2 Enter a number: 50
3 >>> print(number + 100)
4 Traceback (most recent call last):
5   File "<stdin>", line 1, in <module>
6   TypeError: must be str, not int
7
8 >>> number = int(input("Enter a number: "))
9 Enter a number: 50
10 >>> print(number + 100)
11 150
```

Python

```
>>> first_name = "Winston"
>>> last_name = "Smith"

>>> print("Name:", first_name, last_name)
Name: Winston Smith
```

# **INBUILT DATA STRUCTURES IN PYTHON**

**Python has four non-primitive inbuilt data structures namely Lists, Dictionary, Tuple and Set.**

# LIST

## Example

A list with strings, integers and boolean values:

```
list1 = ["abc", 34, True, 40, "male"]
```

Lists are used to store multiple items in a single variable.

Lists are created using square brackets:

## Example

Create a List:

```
thislist = ["apple", "banana", "cherry"]  
print(thislist)
```

## Example

String, int and boolean data types:

```
list1 = ["apple", "banana", "cherry"]  
list2 = [1, 5, 7, 9, 3]  
list3 = [True, False, False]
```



# LIST

## Example

Sort the list alphabetically:

```
thislist = ["orange", "mango", "kiwi", "pineapple", "banana"]  
thislist.sort()  
print(thislist)
```

## Example

Join two list:

```
list1 = ["a", "b", "c"]  
list2 = [1, 2, 3]  
  
list3 = list1 + list2  
print(list3)
```

## Example

Append list2 into list1:

```
list1 = ["a", "b", "c"]  
list2 = [1, 2, 3]  
  
for x in list2:  
    list1.append(x)  
  
print(list1)
```

## Example

Use the `extend()` method to add list2 at the end of list1:

```
list1 = ["a", "b", "c"]  
list2 = [1, 2, 3]  
  
list1.extend(list2)  
print(list1)
```

# LIST METHODS

<code>append()</code>	Adds an element at the end of the list
<code>clear()</code>	Removes all the elements from the list
<code>copy()</code>	Returns a copy of the list
<code>count()</code>	Returns the number of elements with the specified value
<code>extend()</code>	Add the elements of a list (or any iterable), to the end of the current list
<code>index()</code>	Returns the index of the first element with the specified value
<code>insert()</code>	Adds an element at the specified position
<code>pop()</code>	Removes the element at the specified position
<code>remove()</code>	Removes the item with the specified value
<code>reverse()</code>	Reverses the order of the list
<code>sort()</code>	Sorts the list

# PYTHON TUPLES

Tuples are used to store multiple items in a single variable.

Tuple is one of 4 built-in data types in Python used to store collections of data, the other 3 are [List](#), [Set](#), and [Dictionary](#), all with different qualities and usage.

A tuple is a collection which is ordered and **unchangeable**.

Tuples are written with round brackets.

# PYTHON TUPLES

## Example

Create a Tuple:

```
thistuple = ("apple", "banana", "cherry")  
print(thistuple)
```

## Ordered

When we say that tuples are ordered, it means that the items have a defined order, and that order will not change.

## Unchangeable

Tuples are unchangeable, meaning that we cannot change, add or remove items after the tuple has been created.

## Allow Duplicates

Since tuples are indexed, they can have items with the same value:

# PYTHON TUPLES

## Example

Tuples allow duplicate values:

```
thistuple = ("apple", "banana", "cherry", "apple", "cherry")  
print(thistuple)
```

# ACCESS TUPLES

## Example

Print the second item in the tuple:

```
thistuple = ("apple", "banana", "cherry")  
print(thistuple[1])
```

## Example

Print the last item of the tuple:

```
thistuple = ("apple", "banana", "cherry")  
print(thistuple[-1])
```

## Example

Return the third, fourth, and fifth item:

```
thistuple = ("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")  
print(thistuple[2:5])
```

# ACCESS TUPLES

## Example

This example returns the items from the beginning to, but NOT included, "kiwi":

```
thistuple = ("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")
print(thistuple[:4])
```

## Example

This example returns the items from "cherry" and to the end:

```
thistuple = ("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")
print(thistuple[2:])
```

## Example

Check if "apple" is present in the tuple:

```
thistuple = ("apple", "banana", "cherry")
if "apple" in thistuple:
    print("Yes, 'apple' is in the fruits tuple")
```

## Example

Convert the tuple into a list to be able to change it:

```
x = ("apple", "banana", "cherry")
y = list(x)
y[1] = "kiwi"
x = tuple(y)

print(x)
```

# TUPLES

## Example

Convert the tuple into a list, add "orange", and convert it back into a tuple:

```
thistuple = ("apple", "banana", "cherry")
y = list(thistuple)
y.append("orange")
thistuple = tuple(y)
```

## Example

Convert the tuple into a list, remove "apple", and convert it back into a tuple:

```
thistuple = ("apple", "banana", "cherry")
y = list(thistuple)
y.remove("apple")
thistuple = tuple(y)
```

## Example

The `del` keyword can delete the tuple completely:

```
thistuple = ("apple", "banana", "cherry")
del thistuple
print(thistuple) #this will raise an error because the tuple no longer exists
```



# TUPLES

## Example

Iterate through the items and print the values:

```
thistuple = ("apple", "banana", "cherry")
for x in thistuple:
    print(x)
```

```
apple
banana
cherry
```

## Example

Multiply the fruits tuple by 2:

```
fruits = ("apple", "banana", "cherry")
mytuple = fruits * 2

print(mytuple)
```

## Example

Join two tuples:

```
tuple1 = ("a", "b", "c")
tuple2 = (1, 2, 3)

tuple3 = tuple1 + tuple2
print(tuple3)
```

# SET

Sets are used to store multiple items in a single variable.

Set is one of 4 built-in data types in Python used to store collections of data, the other 3 are [List](#), [Tuple](#), and [Dictionary](#), all with different qualities and usage.

A set is a collection which is *unordered*, *unchangeable\**, and *unindexed*.

Sets are written with curly brackets.

# SET

```
thisset = {"apple", "banana", "cherry"}  
print(thisset)
```

# Note: the set list is unordered, meaning: the items will appear in a random order.

# Refresh this page to see the change in the result.

```
{'apple', 'banana', 'cherry'}
```

# SET

## Duplicates Not Allowed

Sets cannot have two items with the same value.

### Example

Duplicate values will be ignored:

```
thisset = {"apple", "banana", "cherry", "apple"}  
  
print(thisset)
```

### Example

`True` and `1` is considered the same value:

```
thisset = {"apple", "banana", "cherry", True, 1, 2}  
  
print(thisset)
```

# SET

## Example

`False` and `0` is considered the same value:

```
thisset = {"apple", "banana", "cherry", False, True, 0}
print(thisset)
```

## Example

Get the number of items in a set:

```
thisset = {"apple", "banana", "cherry"}
print(len(thisset))
```

# PYTHON DICTIONARIES

## Dictionary

Dictionaries are used to store data values in key:value pairs.

A dictionary is a collection which is ordered\*, changeable and do not allow duplicates.

# PYTHON DICTIONARIES

## Example

Create and print a dictionary:

```
thisdict = {  
    "brand": "Ford",  
    "model": "Mustang",  
    "year": 1964  
}  
print(thisdict)
```

## Example

Print the "brand" value of the dictionary:

```
thisdict = {  
    "brand": "Ford",  
    "model": "Mustang",  
    "year": 1964  
}  
print(thisdict["brand"])
```

# PYTHON DICTIONARIES

## Example

Duplicate values will overwrite existing values:

```
thisdict = {  
    "brand": "Ford",  
    "model": "Mustang",  
    "year": 1964,  
    "year": 2020  
}  
print(thisdict)
```