Python Programming

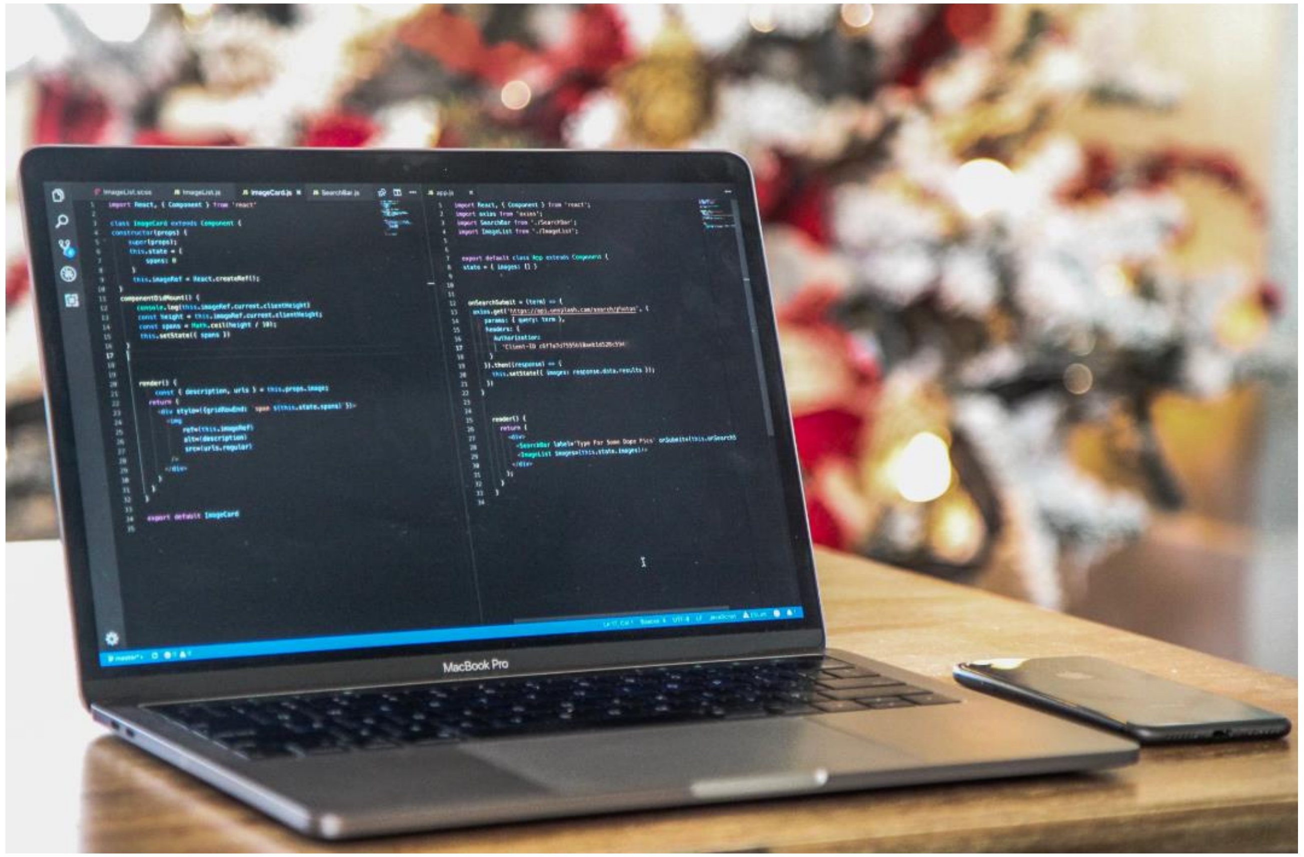
Module 1: Python Fundamentals

Lesson 1: Getting Started

Neba Nfonsang
University of Denver

Lesson 1: Getting Started with Python

- My Goal is that at the end of this course, you would be able to say, “I learned a lot”.
- You will gain skills in programming, data management, data analysis in Python and more…



Contents

Lesson 1: Getting Started with Python

Why Choose Python?

- **Readability**: Python code and syntax is clean, simple, elegant and readable, shorter, prone to less errors, easy to maintain compared to code written in C++, Java, etc.
- Python syntax enhances productivity and software quality

Which of these codes are easy to write or read?

Java code for printing “Hello world”

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World");
    }
}
```

Python code for printing “Hello world”

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

Why Choose Python?

- **Easy to Learn**: Python syntax is user friendly, more intuitive and closely resembles human language syntax.
- It is an ideal first programming language to learn, recommended by most authors and programmers.
- **Open Source**: Python is completely free compared to other software.
- It’s license allows Python to be reproduced, modified, and distributed even without making modified versions open source.

Why Choose Python?

- **Well Supported and Documented**: Python comes loaded with a standard library and is supported by several external libraries, making python versatile and useful for almost everything. Python's Standard Library is well documented and organized.
- **Integration:** Python makes it easy to integrate incompatible software components together. You can use Python to connect to a database, extract data, write and visualize the data in Excel using Python.

Why Choose Python?

- **Popular and Relevant**: The first version of google was implemented in Python.
- Yahoo, Facebook, NASA and others, use Python for various reasons.
- Python is used by education institutions, businesses and government departments and agencies.
- **Runs of Every Platform**. Python is a high-level language and is therefore not platform specific. It an be used in Windows, Mac OS and Linux.

What Really is Python?

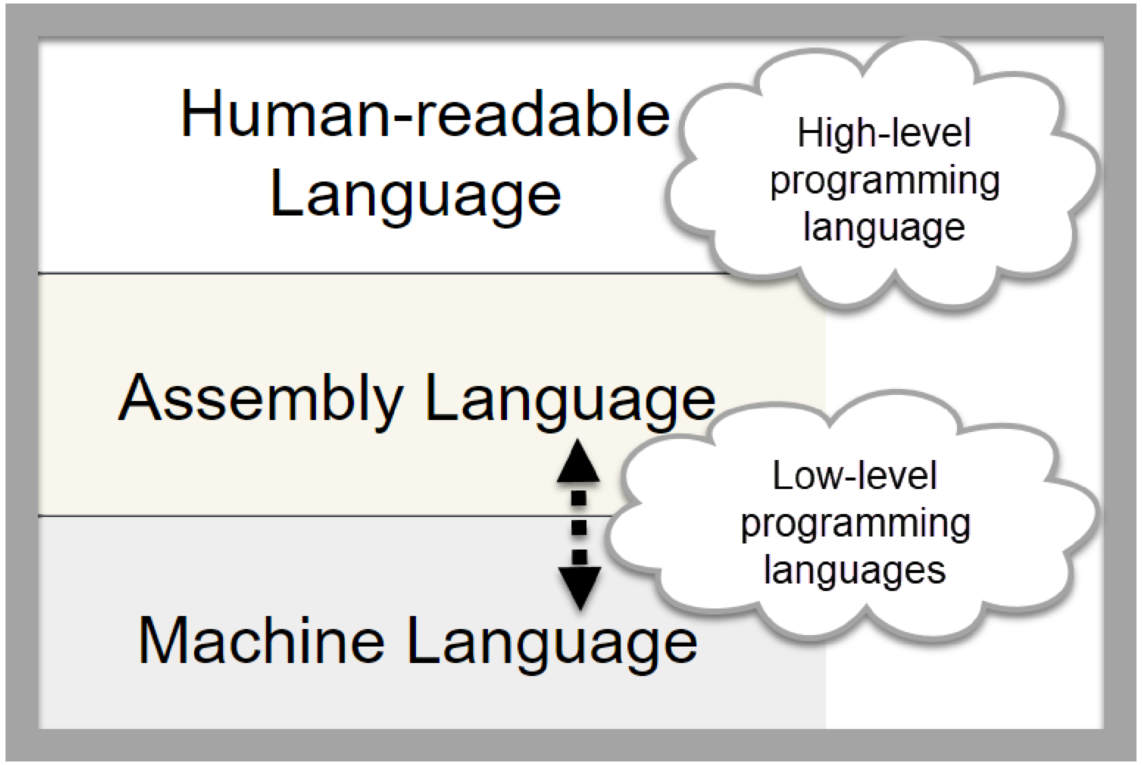
- Python is a general-purpose, high-level, interpreted, object-oriented, and dynamic programming language.

  Let’s break down each of these words:

- **General-Purpose:** Programming languages such as R are more specialized for statistical data analysis.
- Python can be used for almost anything. Python is used for software and web development, data analysis, etc.

What Really is Python?

- **High-Level**: A high-level language is a human-readable language compared to a low-level language such as assembly language and machine language, which consists of binary numbers (0 or 1).


Human-readable Language
High-level programming language
Assembly Language
Low-level programming languages
Machine Language

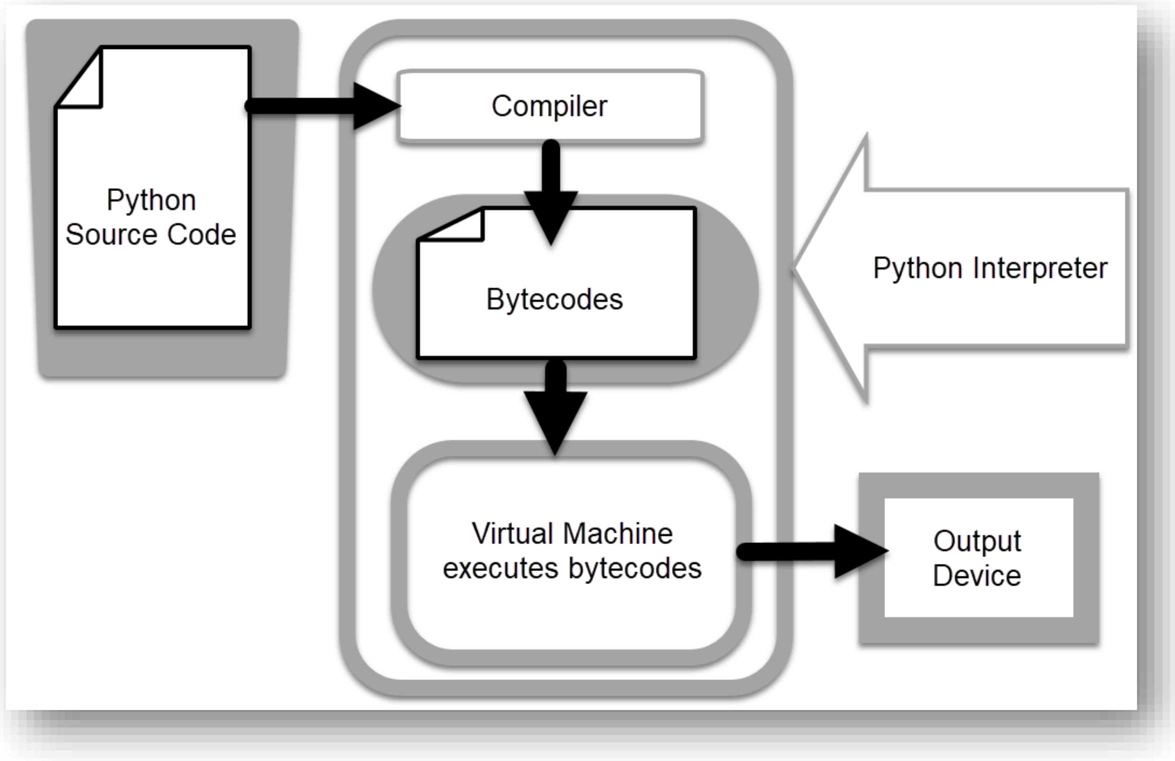
What Really is Python

- **Interpreted**: Python and other languages such as Ruby and Perl are called interpreted languages because they are executed by an interpreter.
- A Python file(.py) or source code is run, it is compiled into bytecode instructions, then executed sequentially from top to bottom by the Python virtual Machine.

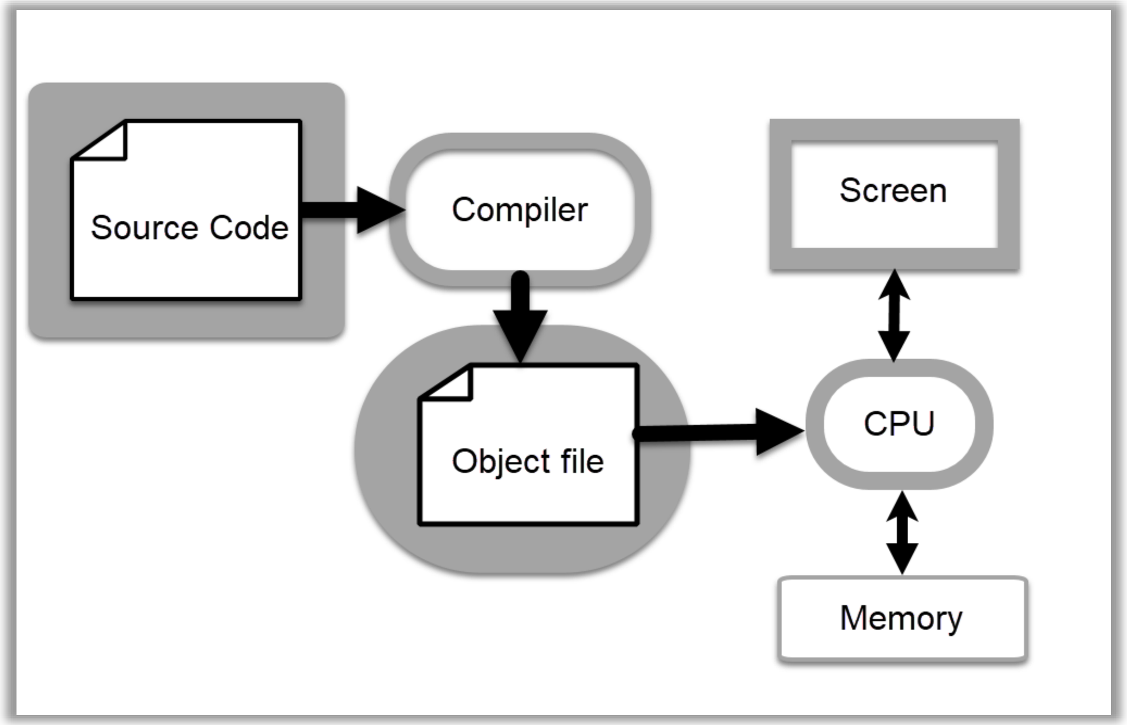
- **Compilers**: Languages such as C and C++ (though high level) are called **compiled** languages because they are executed through a compiler. A compiler converts source code into an object file (machine code).
- Object files are linked to Direct Link Libraries to form executable files, executed by the CPU.

What Really is Python?

How an interpreter executes source code



How a compiler executes source code



What Really is Python?

- **Object-Oriented:** Python is an object-oriented language because it can be used for object-oriented programming (OOP), a programming paradigm (an approach of writing programs) where classes are used to create objects.
- **Dynamic**: Python is a dynamic language. This implies you don’t have to declare variable types when creating variables.
- A Python variable technically don’t have a specific type as the same variable can be assigned to different object types at different points in the program.

Brief History of Python

- The Python language was conceived by Guido van Rossum in the late 1980s.
- In 1983, Guido joint a team that was building a language called ABC, which has much influence on how Python was Created.
- Guido was an implementer in this project.

https://gvanrossum.github.io/

Brief History of Python

- The ABC project was not a big success so Guido moved to another project called Amoeba, which later had a need for a scripting language.
- Guido then started designing a scripting language that had some of the properties of ABC that he liked.
- He used indentation to group code, created data types such as a dictionary, a list, strings, and numbers.

Brief History of Python

- Guido says " I think my most innovative contribution to Python's success was making it easy to extend".

- Python's extensibility (a weakness to the ABC language) was partly due to Python's various data types and Python's flexibility of allowing other programmers to build in their own object types. https://www.artima.com/intv/python.html)

Brief History of Python

- As such, the Python language was born and the first version released in 1991, named “Python” after Guido’s favorite comedy “*The Monty Python’s Flying Circus*”

From history, lets move forward…

Key Definitions

- **Program**: Programming languages are used to write programs. A program is a set of instructions used for accomplishing a specific task.
- You could write a simple program that plays your favorite youtube music after every two hours when you are studying – music break☺

- A program is made up of a single or multiple lines of code or statements that specify how a computer should perform a task or solve a problem.

Key Definitions

- **Algorithm:** An algorithm is a set of instructions (rules) or a process to be followed in order to solve a problem.
- The instructions in an algorithm need to be in the right order. Think of an algorithm as a recipe.
- To bake a cake, you need to follow the recipe in the right order.
- Python statements in a program are executed sequentially, from top to bottom, so your statements need to be logically organized.

Key Definitions

- **Syntax:** A program is structured according to the rules of the programming language. This program structure and rules defining how the program is structured constitute a syntax.

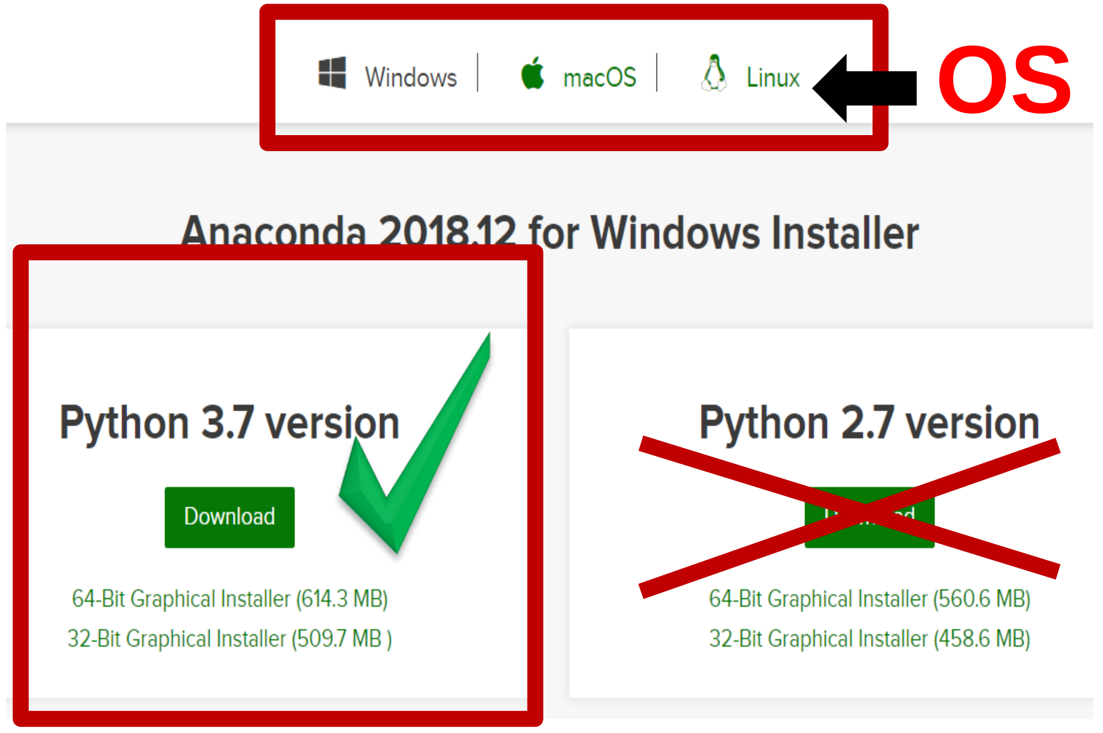
- **Bug**: A bug is an error introduced into the program when the program is not correctly written. The process of finding and fixing a bug is called **debugging.** A program with a bug will malfunction – crash or produce incorrect results.

Installing Python

- There are several Python distributions out there.
- You could install a stand-alone version of Python from Python’s official website: https://www.python.org/downloads/
- We will use this stand-alone Python software later in the course to create Python scripts and modules.
- For now, it is recommended that Python 3.7 be installed through Anaconda: https://www.anaconda.com/distribution/#download-section

Install Python 3.7 using the Anaconda Installer

- You should install Python 3.7 for this course through Anaconda
- Anaconda is an open source Python/R distribution. It is also a package/environment manager.
- To download Python 3.7, first select the right OS. Then click the right (64 or 32-Bit) graphical Installer for your OS



Anaconda 2018.12 for Windows Installer

Python 3.7 version

Download

64-Bit Graphical Installer (614.3 MB)

32-Bit Graphical Installer (509.7 MB)

~~Python 2.7 version~~

~~64-Bit Graphical Installer (560.6 MB)~~

~~32-Bit Graphical Installer (458.6 MB)~~

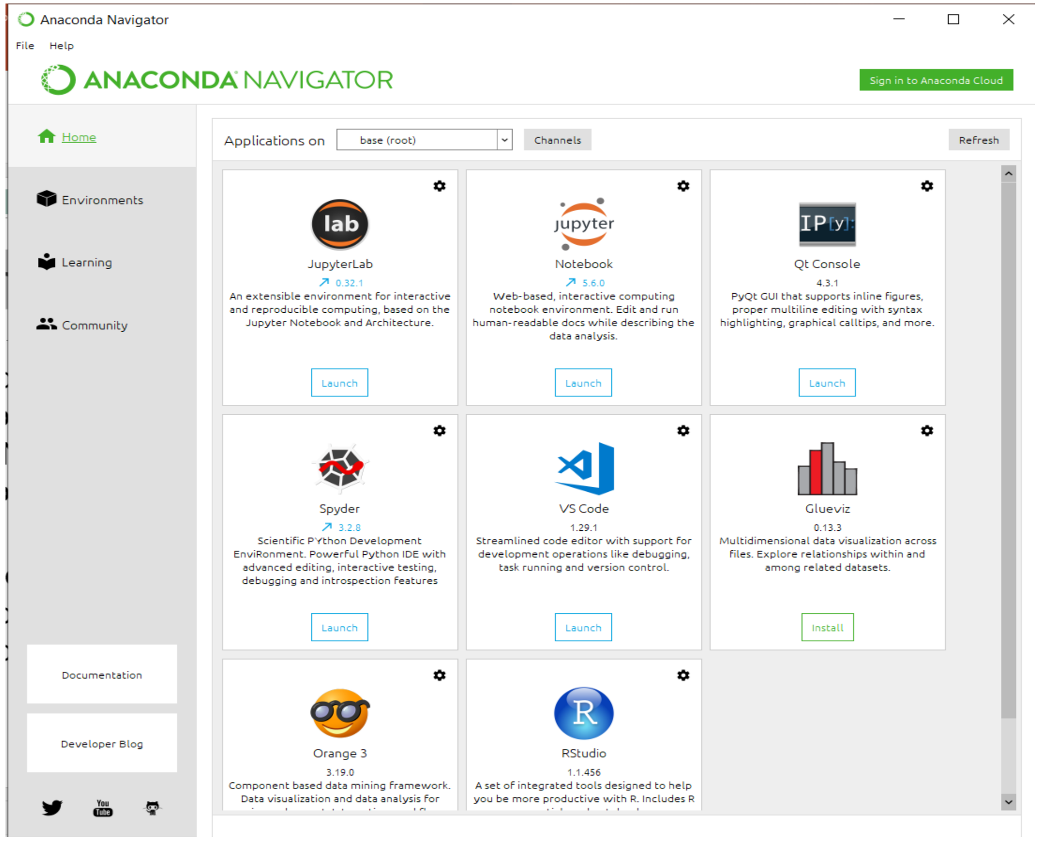
https://www.anaconda.com/distribution/#download-section

The Anaconda Navigator

- **Anaconda Navigator**: This is a graphical user interface (GUI) included in Anaconda distribution that allows you to launch applications easily, install packages and manage environments, etc. without using a command-line interface.
- Anaconda comes loaded with many applications such as Jupyter Notebook, Spyder, R and so on.
- You can access these applications by first launching the Anaconda Navigator.

The Anaconda Navigator

- You can launch the Anaconda Navigator by typing the word "Anaconda Navigator" into your computer's search box.
- Then click the Anaconda Navigator desktop app to launch it.
- The interface would be as shown in this figure.



JupyterLab
0.32.1
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook
5.6.0
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Qt Console
4.3.1
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Spyder
3.2.8
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

VS Code
1.29.1
Streamlined code editor with support for development operations like debugging, task running and version control.

Glueviz
0.13.3
Multidimensional data visualization across files. Explore relationships within and among related datasets.

Orange 3
3.19.0
Component based data mining framework. Data visualization and data analysis for

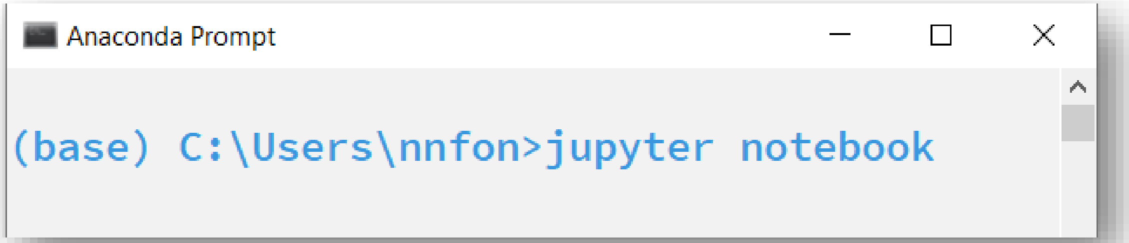
RStudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R

Jupyter Notebook

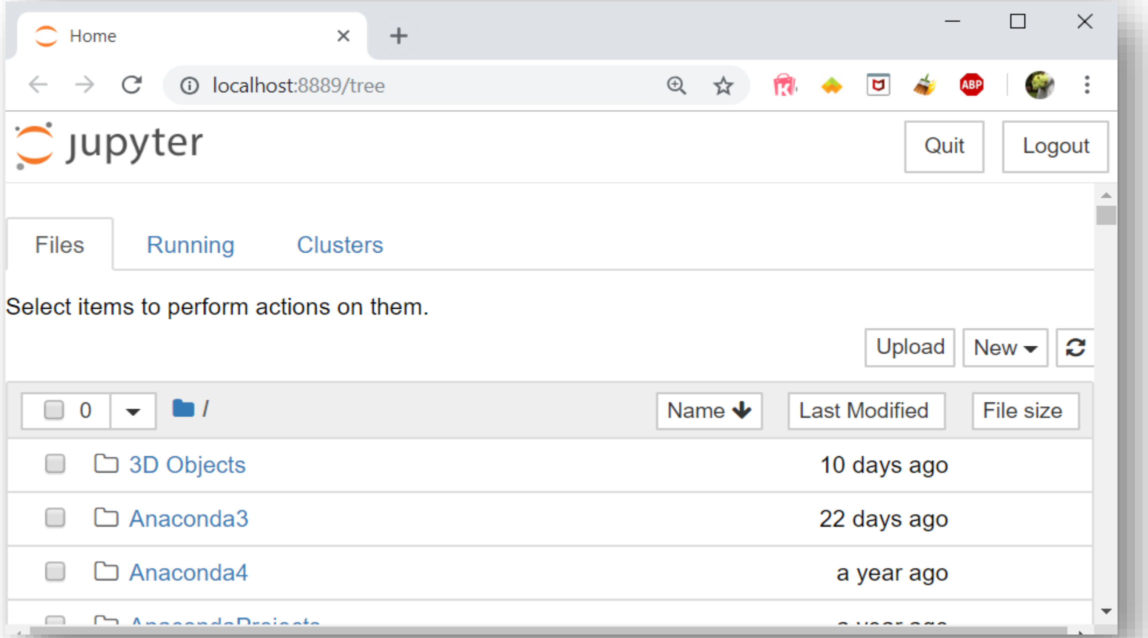
- Jupyter Notebook is a web-based interactive computing notebook environment.
- It is used for writing, editing and running computer code.
- In this course, Jupyter Notebook will be used as the primary environment for writing and running Python code
- Click the Jupyter Notebook icon on the Anaconda Navigator to launch the Notebook.
- To launch the Notebook through Anaconda prompt (terminal in Mac or Linux) type **jupyter notebook** in the prompt or terminal and hit the enter key on your keyboard.

Jupyter Notebook

- Though Jupyter Notebook is a web-based application, it does not run on the internet.
- When the Notebook is launched, it opens locally on your browser. The browser page as shown, is the Notebook home page, also known as the dashboard or directory tree.



```
(base) C:\Users\nnfon>jupyter notebook
```

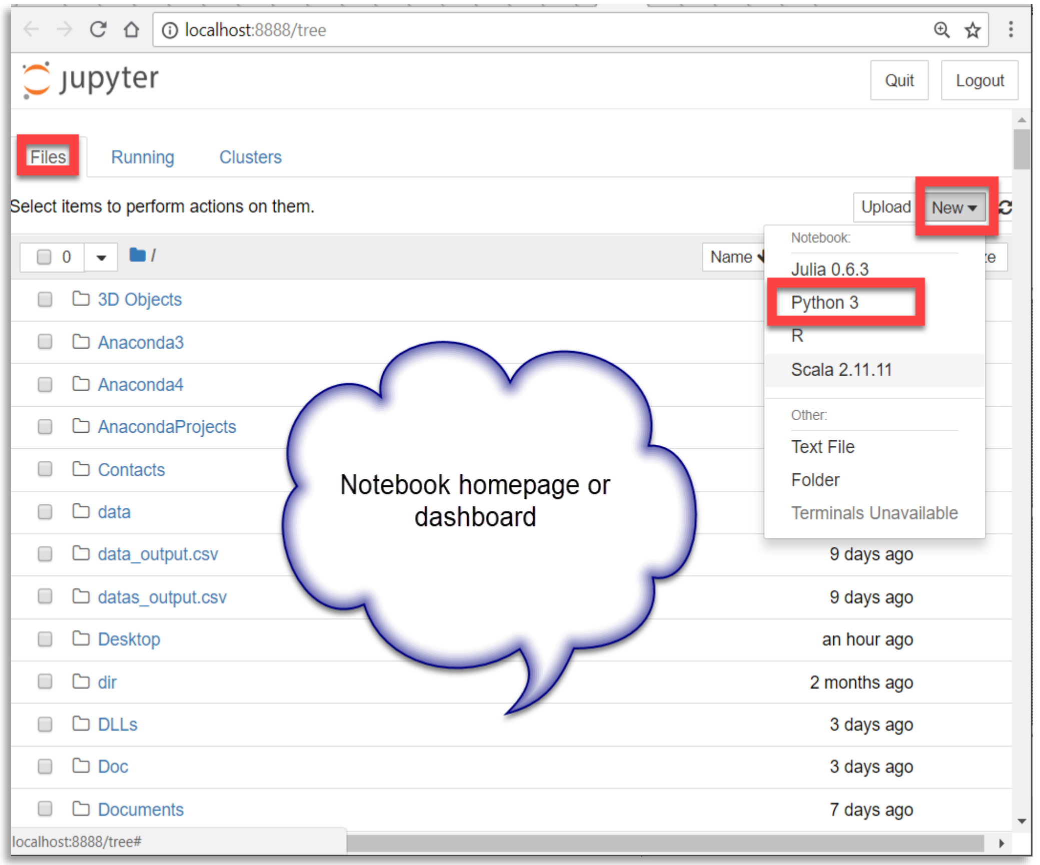


Select items to perform actions on them.

Name	Last Modified
3D Objects	10 days ago
Anaconda3	22 days ago
Anaconda4	a year ago

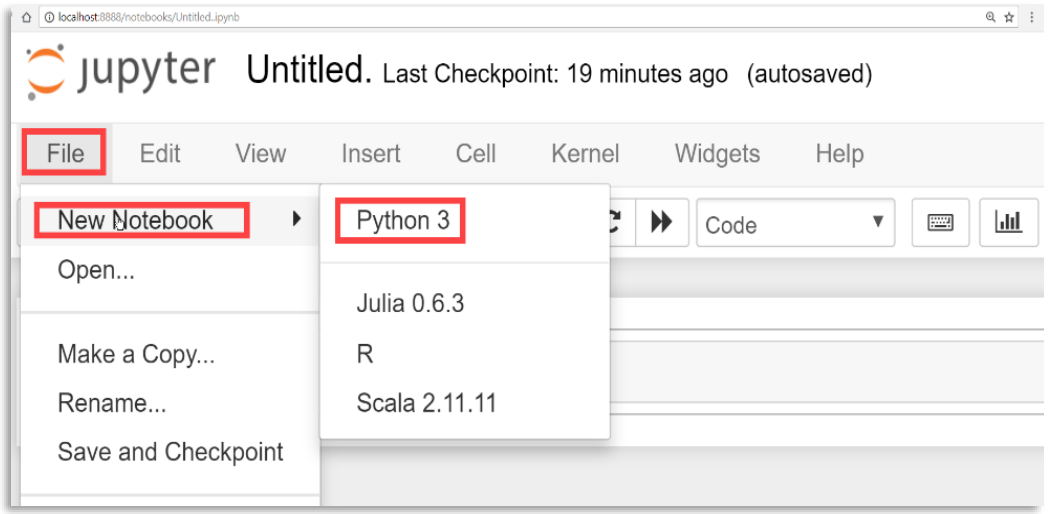
Jupyter Notebook Document

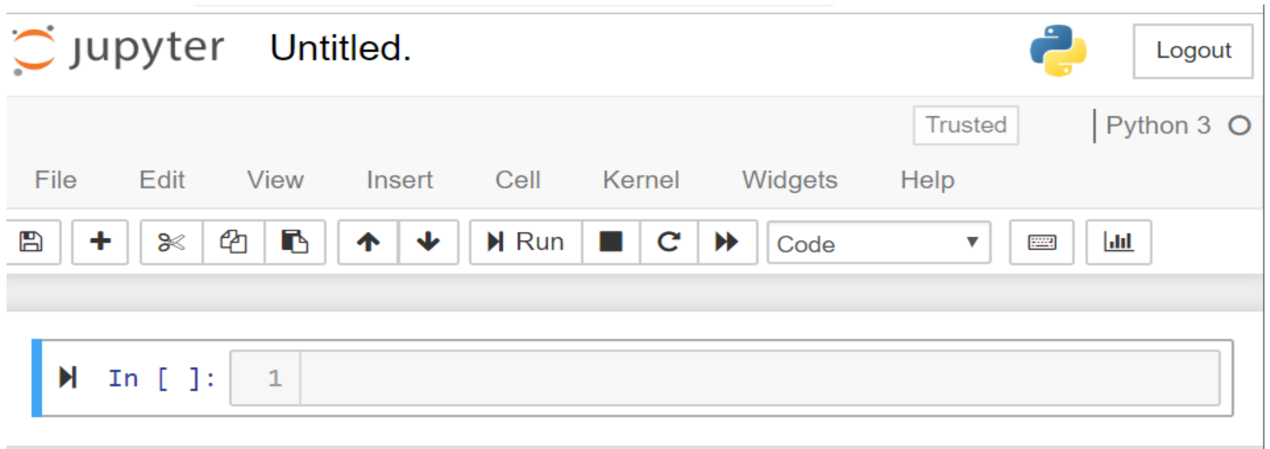
- The Notebook home page is used to open notebook documents.
- A Notebook document or "notebook" is a document created through the Notebook App, containing computer code such as Python code.
- To create a new notebook for writing and running Python code, click "**New**", then, "**Python 3**" on the dashboard.



Jupyter Notebook Document

- A new notebook could also be created within an active notebook as follows:

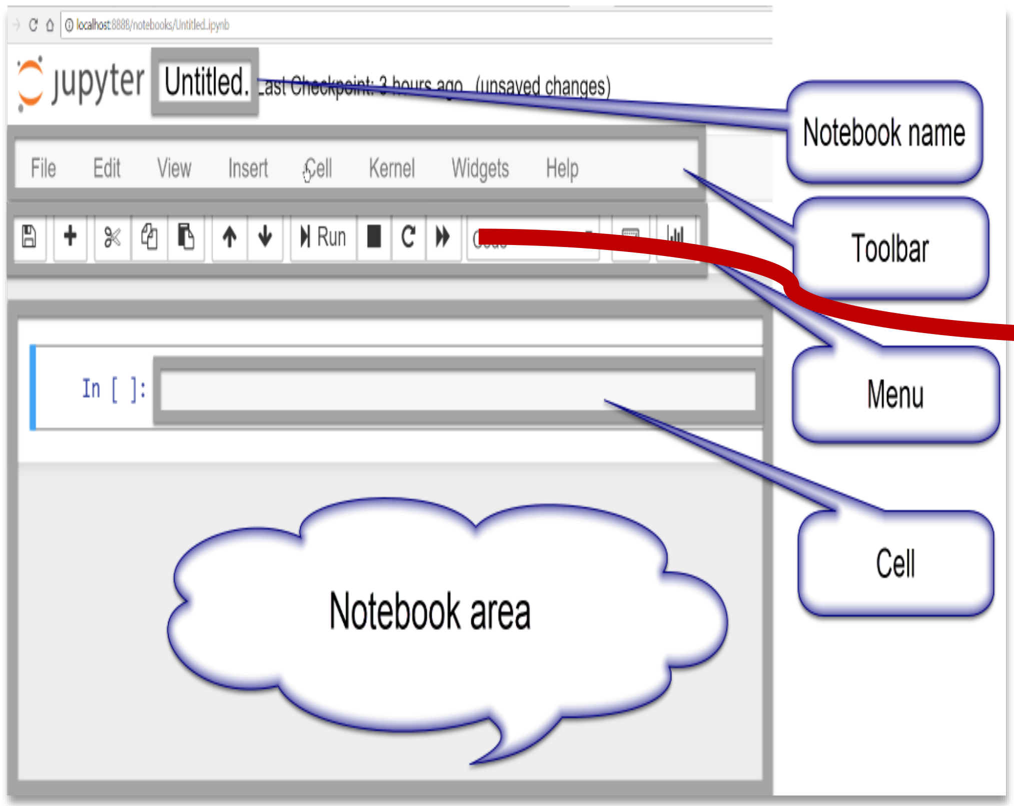

localhost:8888/notebooks/Untitled.ipynb
jupyter Untitled. Last Checkpoint: 19 minutes ago (autosaved)
File Edit View Insert Cell Kernel Widgets Help
New Notebook
Open...
Make a Copy...
Rename...
Save and Checkpoint
Python 3
Julia 0.6.3
R
Scala 2.11.11
Code

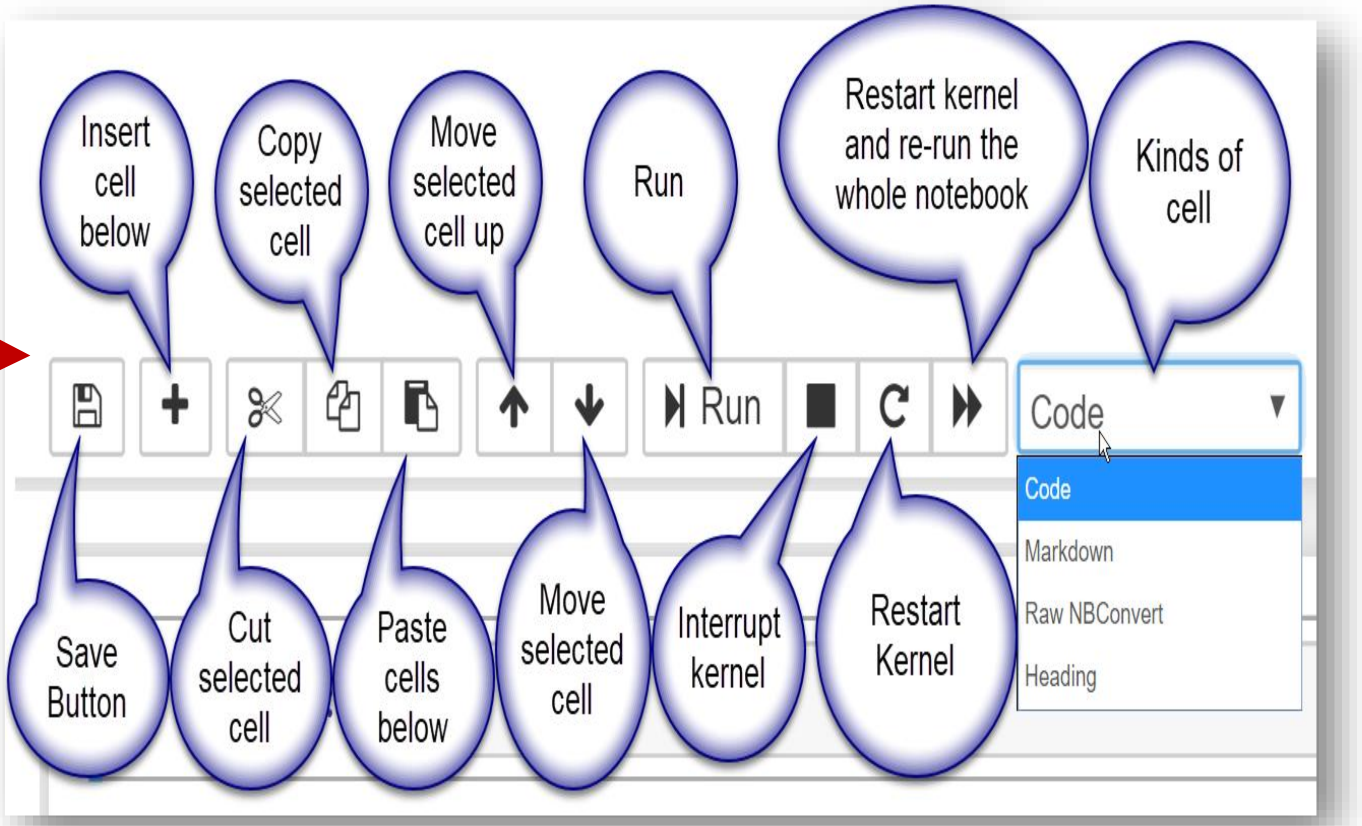


This figure illustrates a new notebook

An existing notebook can be opened from the dashboard by clicking on the name of the notebook.

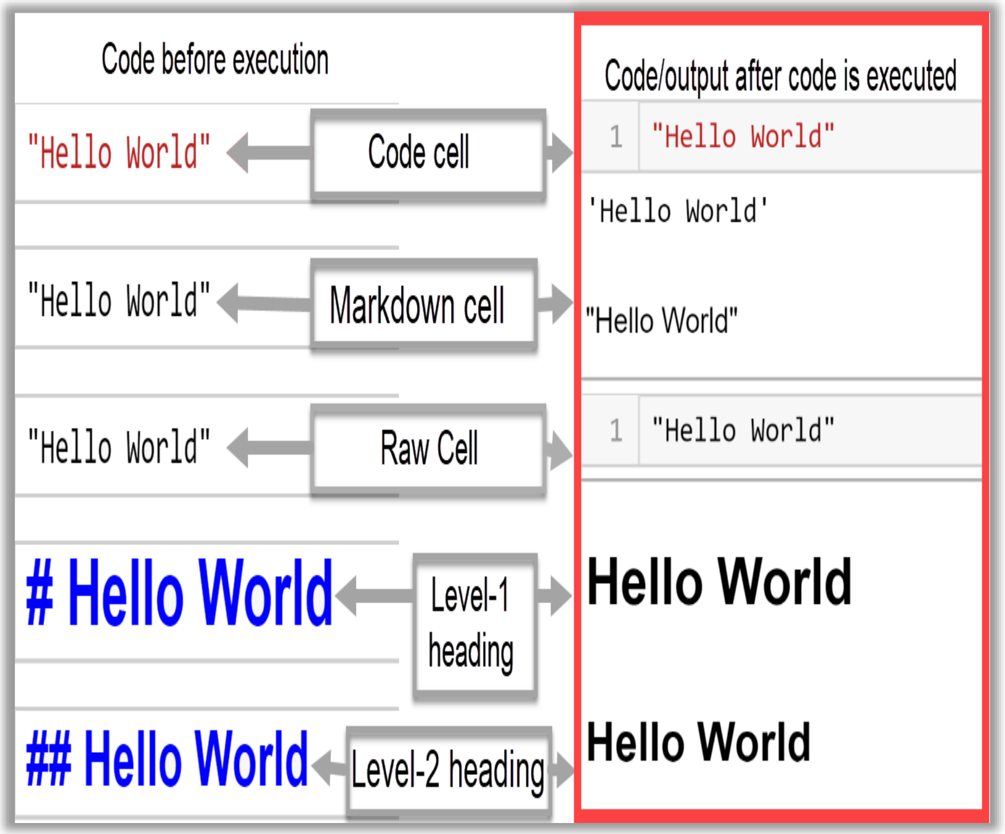
The Notebook User Interface/Tool Bar


localhost:8888/notebooks/Untitled.ipynb
jupyter Untitled Last Checkpoint: 3 hours ago (unsaved changes)
Notebook name
File Edit View Insert Cell Kernel Widgets Help
Toolbar
Run
Menu
In [ ]:
Cell
Notebook area



Jupyter Notebook Cells

- The Jupyter Notebook has different types of cells:
- **Code cells** allow us to write, edit and execute code.
- **Markdown cells** allow us to write and format rich descriptive text.
- **Header cells** allow us to write different levels of headings. The number of hash (#) indicate heading level



```
"Hello World"
```

```
1 "Hello World"
'Hello World'
```

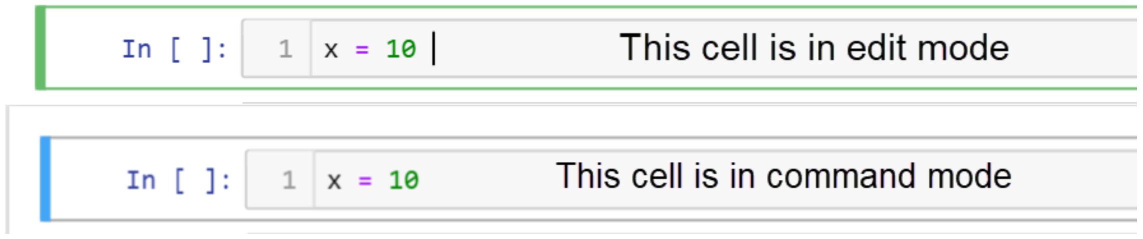
"Hello World"

Hello World

Hello World

Notebook Document Cell Mode

- A notebook cell could be in an edit or command mode
- The **edit mode** is activated when you click **inside** the cell.
- When you click **outside** the cell, you enter the **command mode.**
- In the edit mode, you can type inside the cell.
- In the command mode, you cannot type inside the cell but you can edit the entire notebook. The command mode allows you to use keyboard short cuts to delete cells, add cells, copy and paste cells, etc.



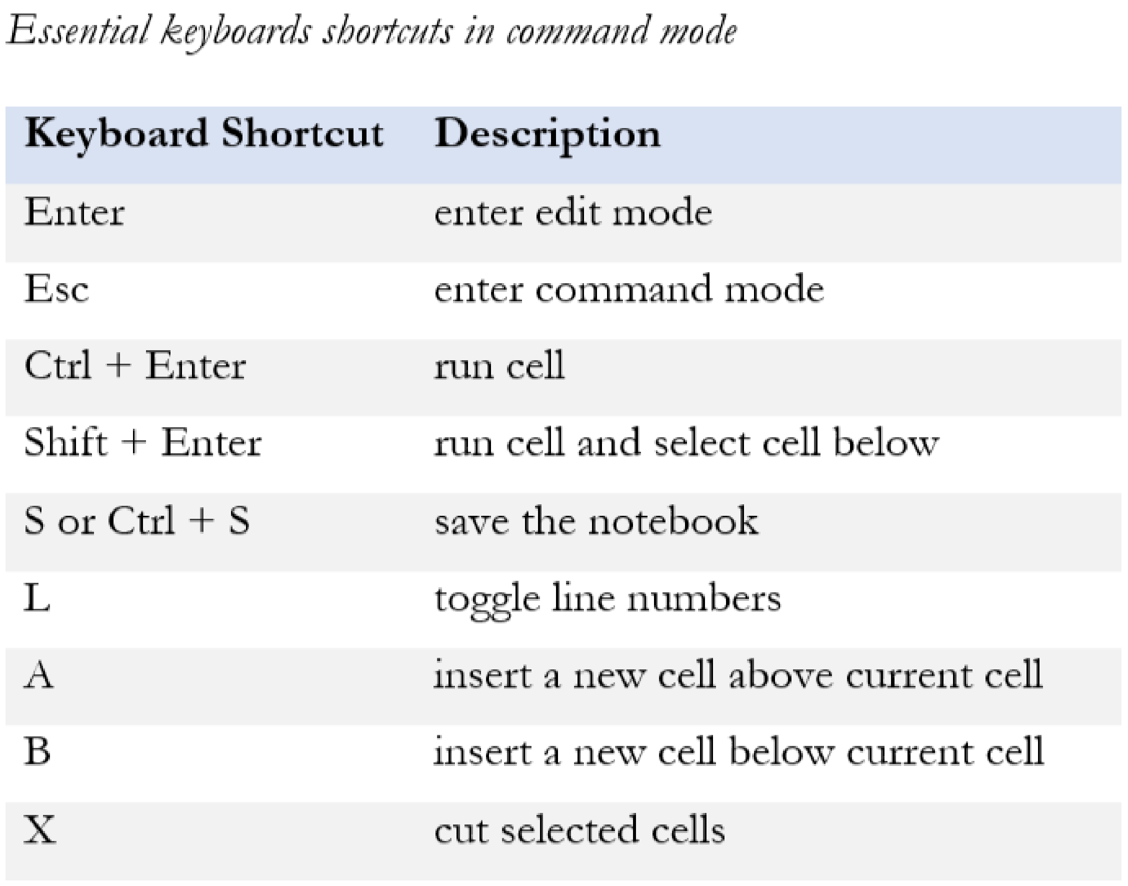
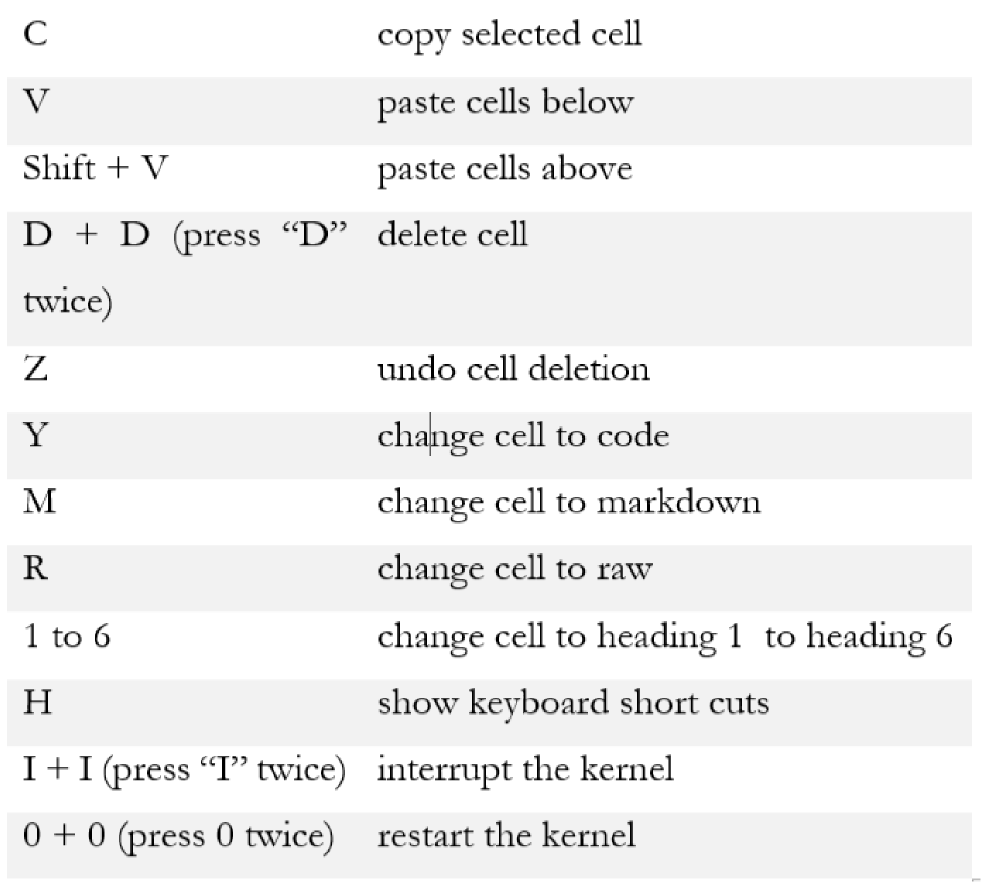
```
In [ ]:   x = 10 |
```

This cell is in edit mode

```
In [ ]:   x = 10
```

This cell is in command mode

Jupyter Notebook Keyboard Short Cuts

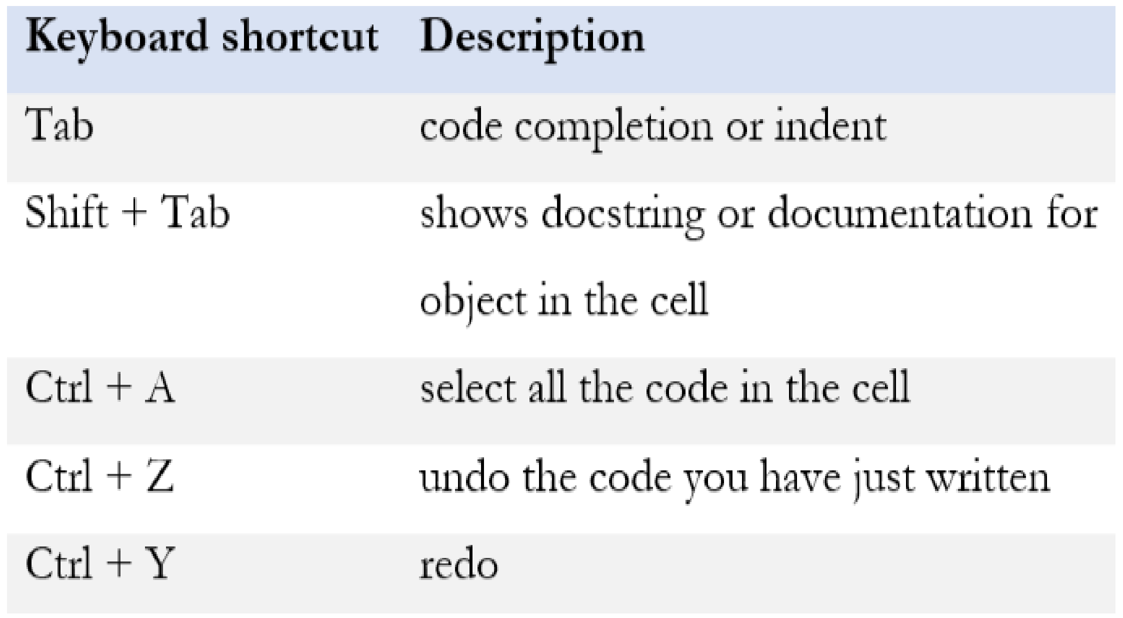



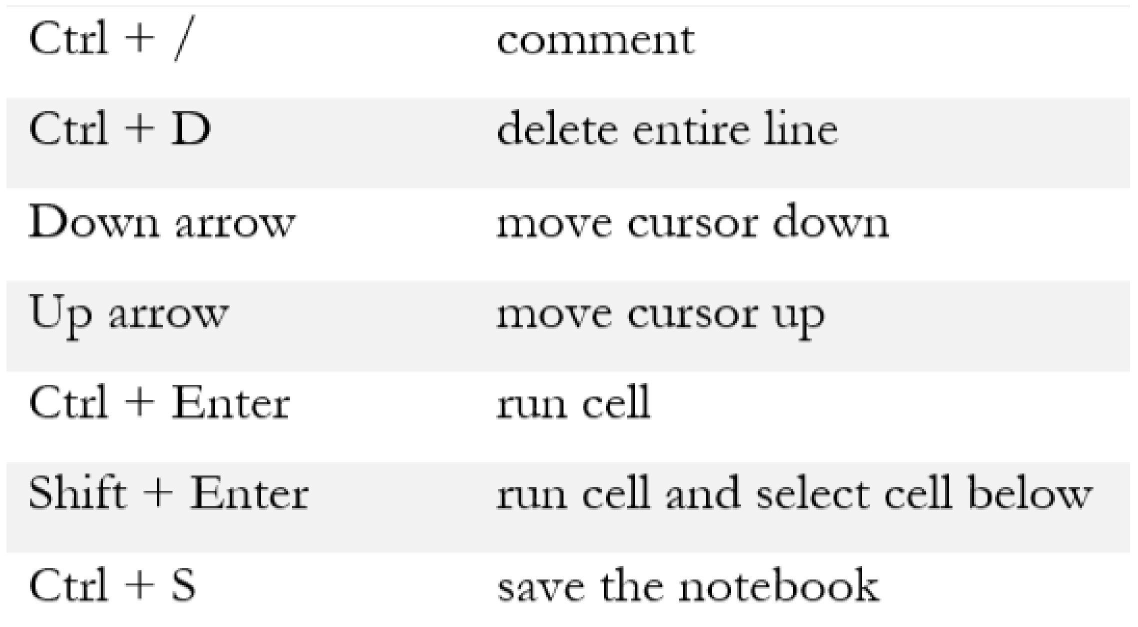
Essential keyboards shortcuts in command mode

Keyboard Shortcut	Description
Enter	enter edit mode
Esc	enter command mode
Ctrl + Enter	run cell
Shift + Enter	run cell and select cell below
S or Ctrl + S	save the notebook
L	toggle line numbers
A	insert a new cell above current cell
B	insert a new cell below current cell
X	cut selected cells
C	copy selected cell
V	paste cells below
Shift + V	paste cells above
D + D (press "D" twice)	delete cell
Z	undo cell deletion
Y	change cell to code
M	change cell to markdown
R	change cell to raw
1 to 6	change cell to heading 1 to heading 6
H	show keyboard short cuts
I + I (press "I" twice)	interrupt the kernel
0 + 0 (press 0 twice)	restart the kernel

Jupyter Notebook Keyboard Short Cuts

Essential keyboards shortcuts in edit mode

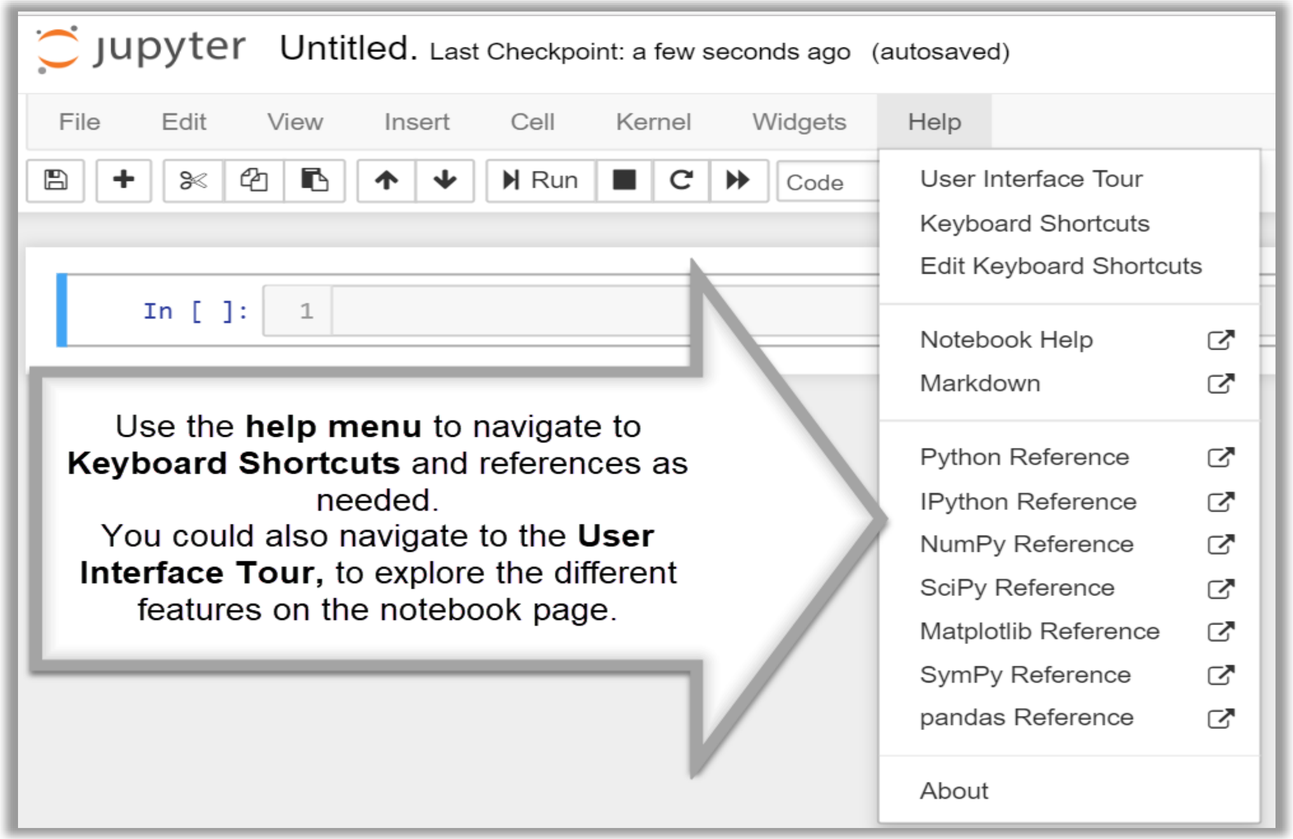




Keyboard shortcut	Description
Tab	code completion or indent
Shift + Tab	shows docstring or documentation for object in the cell
Ctrl + A	select all the code in the cell
Ctrl + Z	undo the code you have just written
Ctrl + Y	redo
Ctrl + /	comment
Ctrl + D	delete entire line
Down arrow	move cursor down
Up arrow	move cursor up
Ctrl + Enter	run cell
Shift + Enter	run cell and select cell below
Ctrl + S	save the notebook

Jupyter Notebook Help Menu

- You may visit the Jupyter notebook documentation for more information on how to use the Jupyter notebook (https://jupyter-notebook.readthedocs.io/en/stable/notebook.html).



Use the **help menu** to navigate to **Keyboard Shortcuts** and references as needed.
You could also navigate to the **User Interface Tour,** to explore the different features on the notebook page.

Now, Let’s Jump into Some Key Programming Concepts

Variables

- In Python or programming in general, a variable is a name that refers to, points to or references a value.
- A variable is used to give names to values or objects.

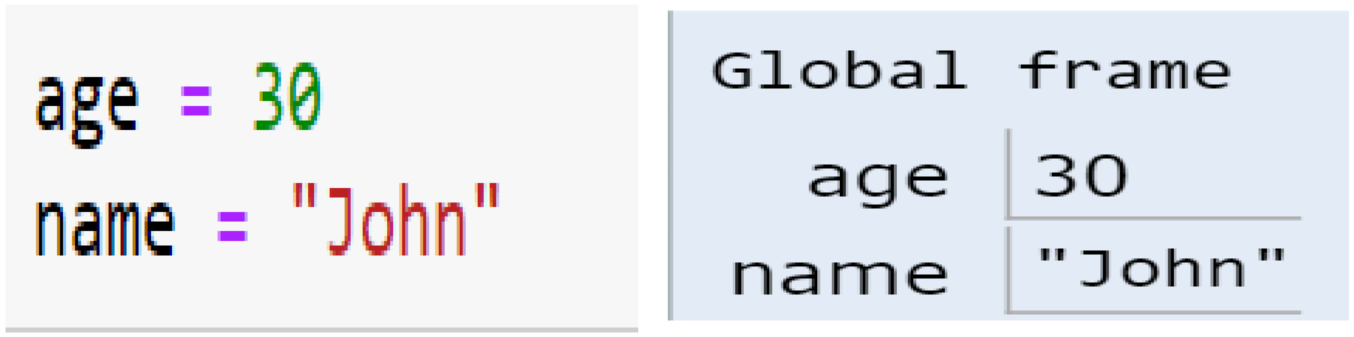
A Python variable has the following characteristics:

- Variables are created when they are first assigned a value and are therefore never declared ahead of time.
- A variable can not be used unless it references a value.
- The value of a variable is accessed or used when a variable is used in an expression.

Variables

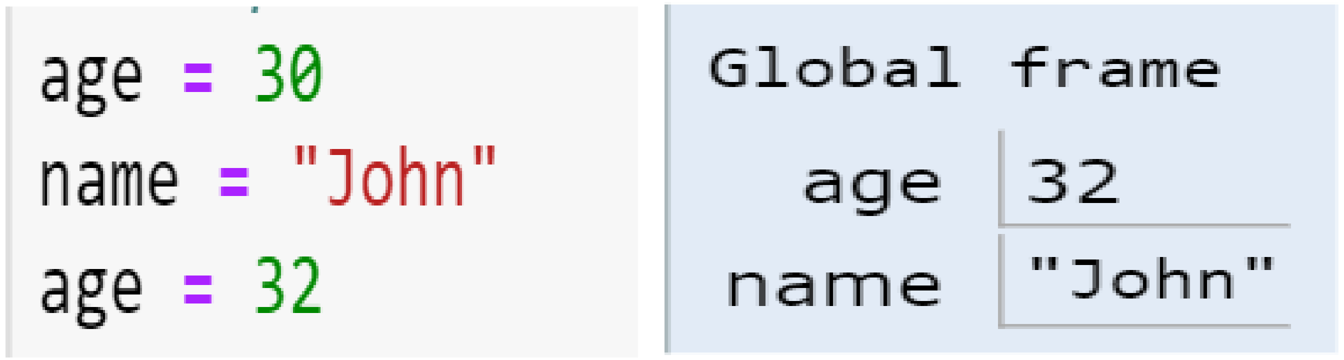
- Variables such as **age** or **first_name** are created through assignment statements.
- If a new value is assigned to the same variable name, the old value referenced by the variable is overwritten.

Memory Model for a Variable and its Associated Value



```
age = 30
name = "John"
```

Global frame	
age	30
name	"John"



```
age = 30
name = "John"
age = 32
```

Global frame	
age	32
name	"John"

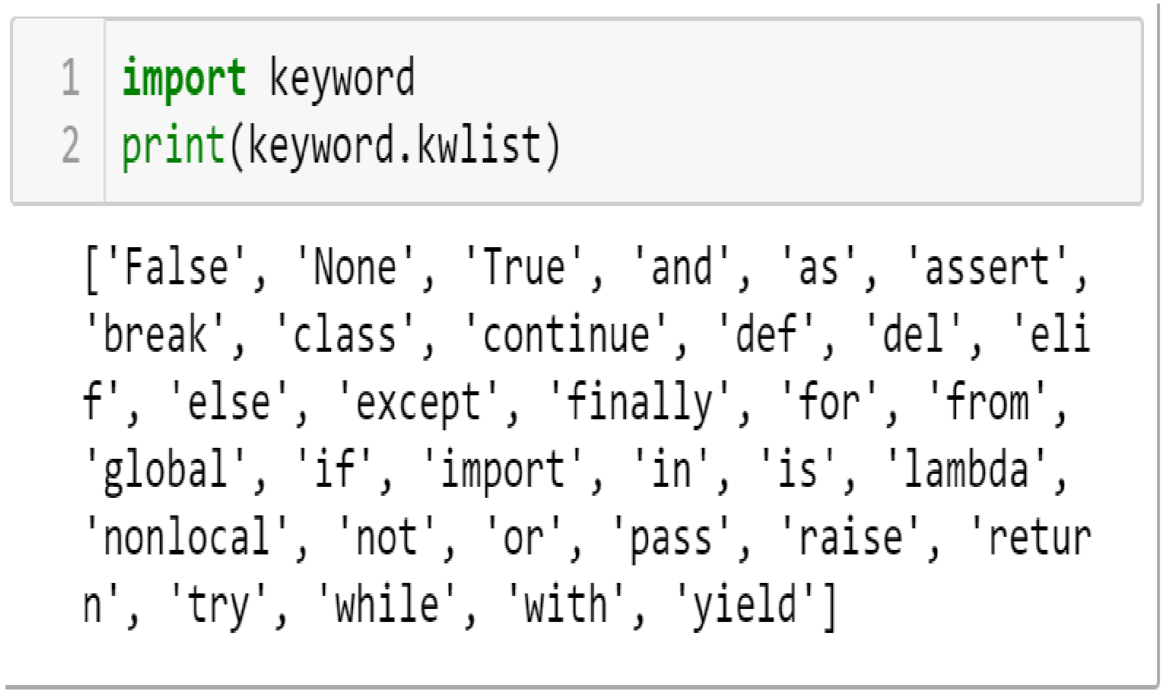
Best Practices for Naming Variables

- Variable names should be lower case and use underscores to separate words to increase readability.
- A variable name should be intuitive and short. Use a word such as **age** to refer to an age value instead using the letter **a**.
- Do not use spaces or special characters such as !, @, #, $, %, ^, &, *, etc.
- Do not start a variable name with a number or an underscore.
- Single letters could be used but avoid using l and o as that could be seen as one(1) and zero(0).

Best Practices for Naming Variables

- Avoid using Python keywords, reserved words or names of Python functions.
- Python key words can be output as shown
- See other best practices in PEP 8 – Style Guide for Python Code

https://www.python.org/dev/peps/pep-0008/#



```python
import keyword
print(keyword.kwlist)
```

```
['False', 'None', 'True', 'and', 'as', 'assert',
'break', 'class', 'continue', 'def', 'del', 'eli
f', 'else', 'except', 'finally', 'for', 'from',
'global', 'if', 'import', 'in', 'is', 'lambda',
'nonlocal', 'not', 'or', 'pass', 'raise', 'retur
n', 'try', 'while', 'with', 'yield']
```

Python Statements

It is important to understand the place of Python statements in a program structure. Generally:

- A program consist of modules (modules are simply Python files)
- Modules contain statements
- Statements are made up of expressions
- Expressions process objects

Examples of Python statements include:

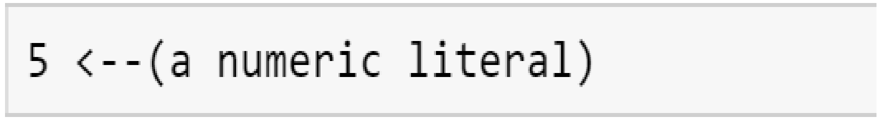
- Assignment, control flow, pass, continue, break, def, return, print, import, from, del, as/with, and try/except/finally statements.
- A statement is a unit of code that can be executed.

Expressions

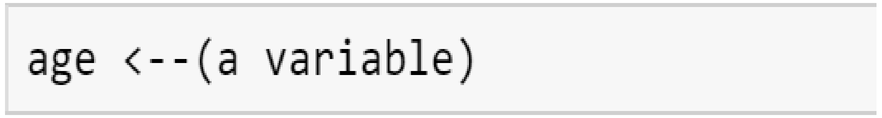
- Generally, an expression is a combination of values, variables and operators.
- Technically, an expression is a statement (an expression statement).
- Expressions are used to process data to generate new values.
- The simplest kind of expression is a literal. Literals are expressions that specify or indicate the value of an object.
- Therefore a value by itself is a literal expression, for example, 5 or “blue”. Note that an expression evaluates to a value but other statements do not.

Expressions

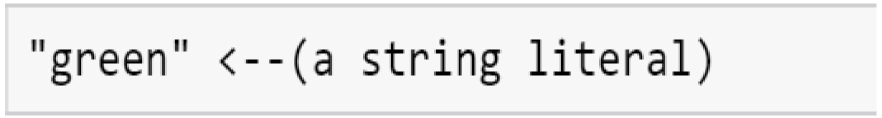
- Here are examples of expressions



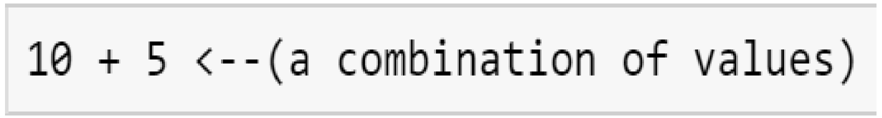
```
5 <--(a numeric literal)
```



```
age <--(a variable)
```



```
"green" <--(a string literal)
```



```
10 + 5 <--(a combination of values)
```

- For now, two basic types of data in Python are numeric and string types.
- Numeric types are numbers and string types are text, usually surrounded by commas.
- We will study more about data types when we cover data structures in Python.
- Function calls, method calls and print statements are also expressions.

Assignment Statements

- An assignment statement is a statement that has an equal sign (=) and is used to assign a value to a variable.
- age = 30 is an assignment statement. The value 30 is being assigned to the variable age.
- x = 10 + 5 is an assignment statement. Python first evaluates the expression, then assigns the value (15) to the the variable x.
- Again, note that variables are created only when an assignment statement is run.

Forms of Assignment Statements

```
first_name = "John"  <--(a basic assignment statement)
```

```
first_name, age = "John", 30   <--(tuple assignment - positional)
```

```
[first_name, age] = ["John", 30]   <--(list asssignment)
```

```
x, y, z = 4  <--(sequence assignment - generalized)
```

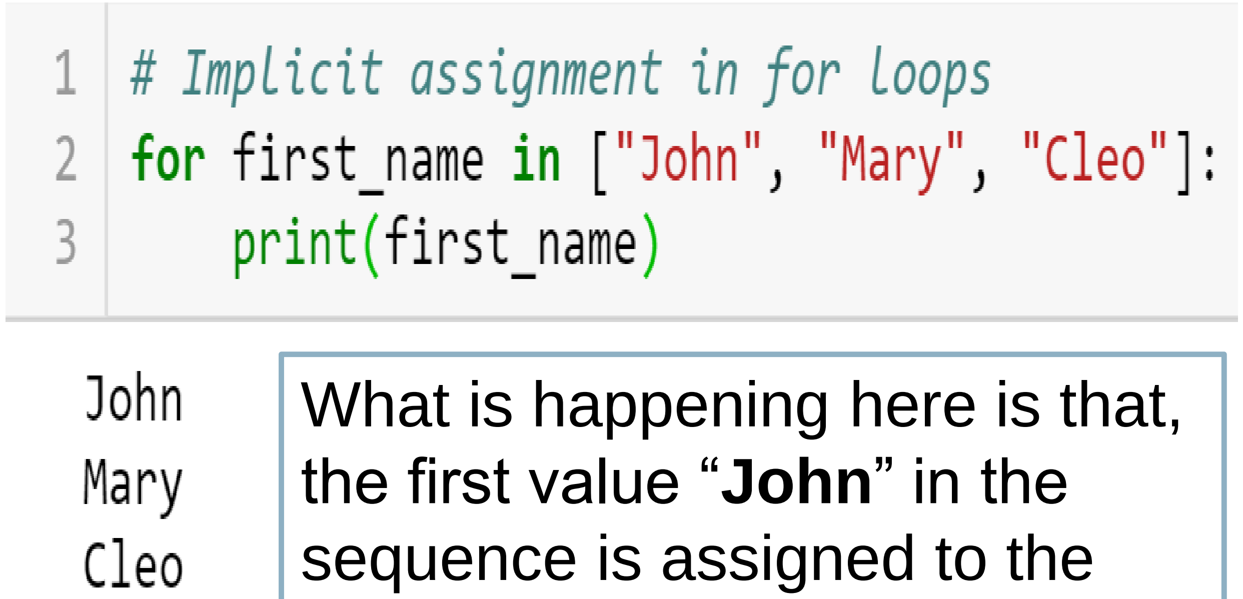
```
a, *b = [1, 2 ,3, 4]   <--(extended sequence unpacking)
1 will be assigned to a, and the rest of the sequence to b
```

```
a = b = 5   <--(multiple-target assignment)
```

```
x += 1 or x = x + 1   <--(augmented assignment)
```

Implicit Assignments

- Some assignments in Python are implicit and are used in different scenarios in Python
- Module imports, function and class definitions, for loops, function arguments all use assignments implicitly.



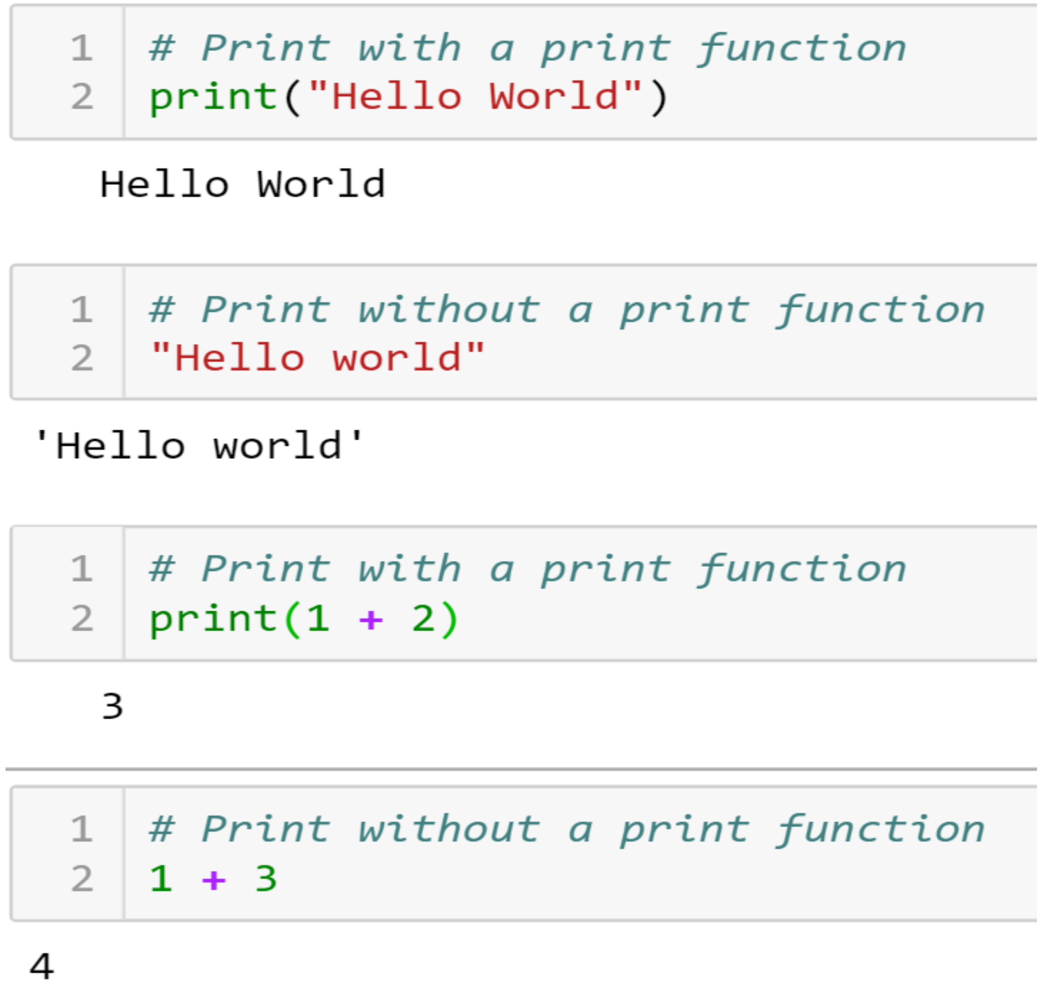
```python
# Implicit assignment in for loops
for first_name in ["John", "Mary", "Cleo"]:
    print(first_name)
```

```
John
Mary
Cleo
```

What is happening here is that, the first value “**John**” in the sequence is assigned to the variable **first_name,** then printed. This process repeats for the rest of the names (values) in the sequence.

Output and Print Statements

- When a Python statement such as an expression is run, the statement is executed and an output is generated
- You can generate output in Python by using a print statement or function, or by just running the expression without a print statement in an interactive interpreter.



```python
# Print with a print function
print("Hello World")
```

```
Hello World
```

```python
# Print without a print function
"Hello world"
```

```
'Hello world'
```

```python
# Print with a print function
print(1 + 2)
```

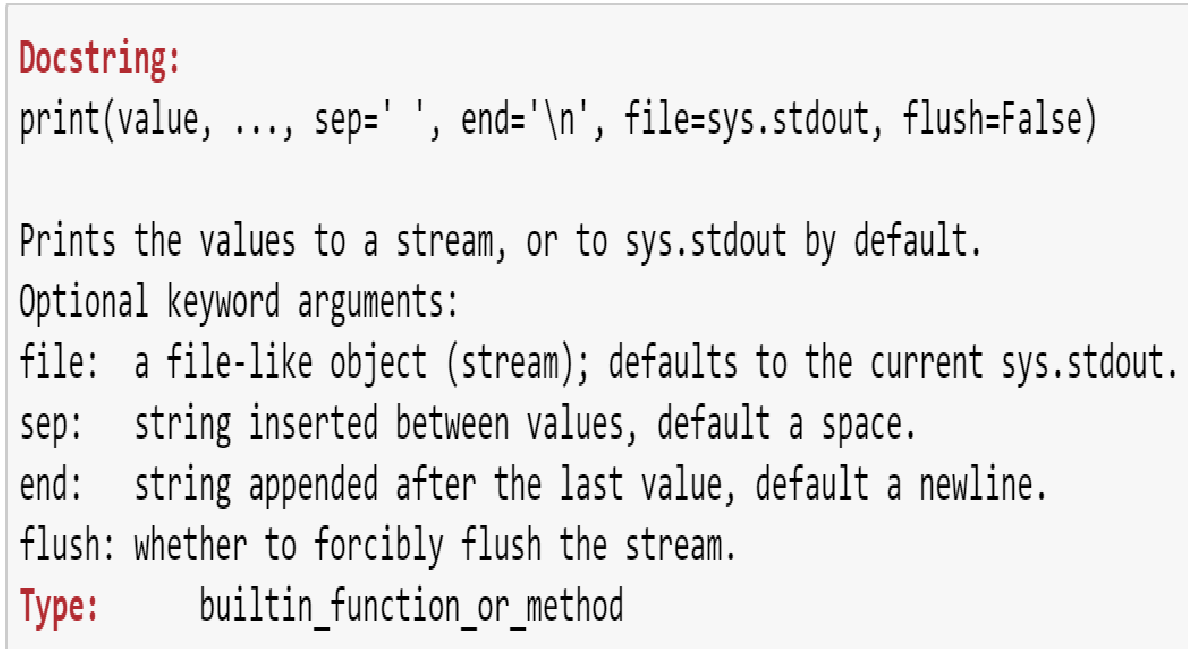
```
3
```

```python
# Print without a print function
1 + 3
```

```
4
```

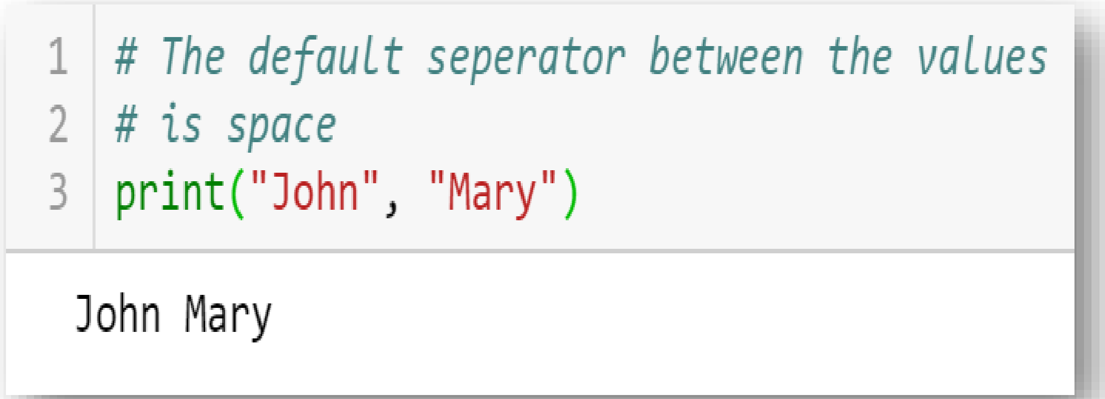
The **print()** Function

- The print() function is one of Python’s in-built function.
- The print() function prints values and has parameters as shown.
- The **sep** parameter specifies the string that should separate multiple values passed into the print function



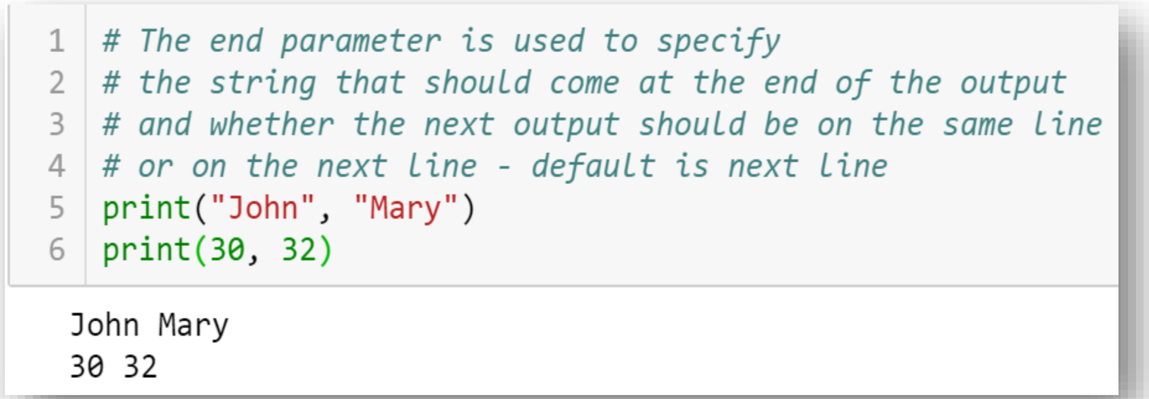
```
Docstring:
print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

Prints the values to a stream, or to sys.stdout by default.
Optional keyword arguments:
file:  a file-like object (stream); defaults to the current sys.stdout.
sep:   string inserted between values, default a space.
end:   string appended after the last value, default a newline.
flush: whether to forcibly flush the stream.
Type:      builtin_function_or_method
```

The **print()** Function



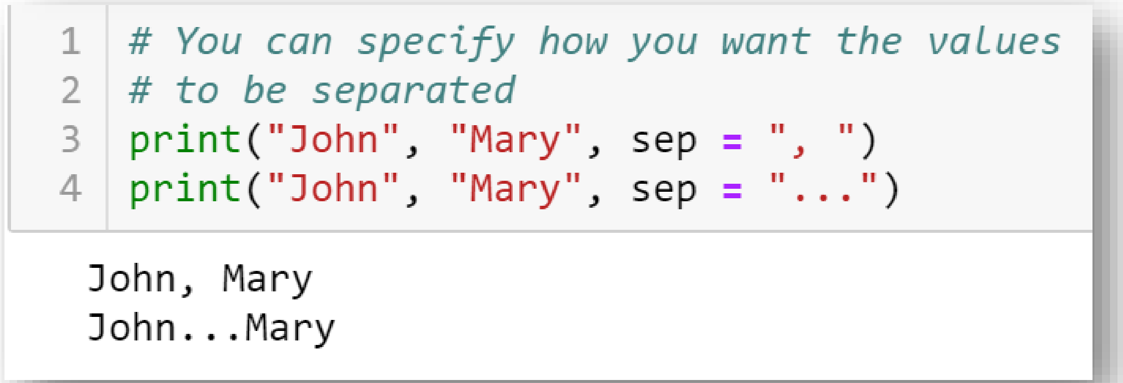
```python
# The default seperator between the values
# is space
print("John", "Mary")
```

```
John Mary
```



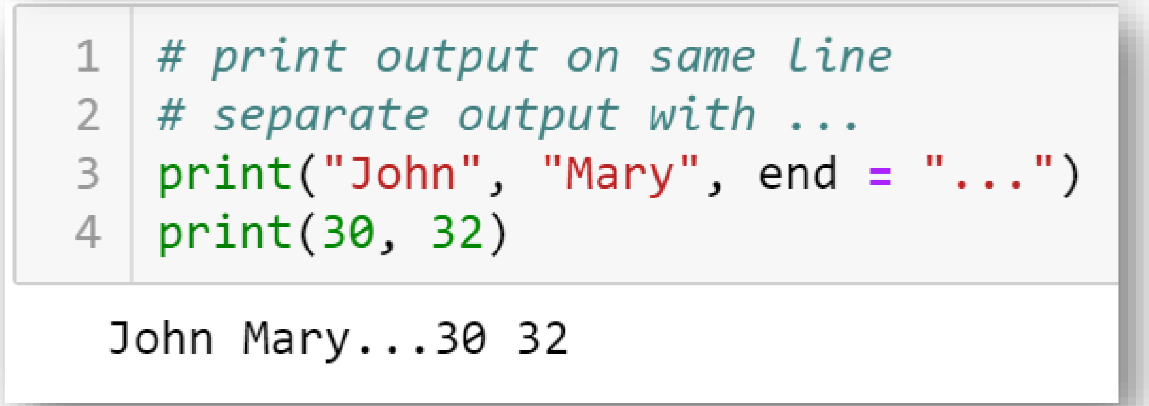
```python
# The end parameter is used to specify
# the string that should come at the end of the output
# and whether the next output should be on the same line
# or on the next line - default is next line
print("John", "Mary")
print(30, 32)
```

```
John Mary
30 32
```



```python
# You can specify how you want the values
# to be separated
print("John", "Mary", sep = ", ")
print("John", "Mary", sep = "...")
```

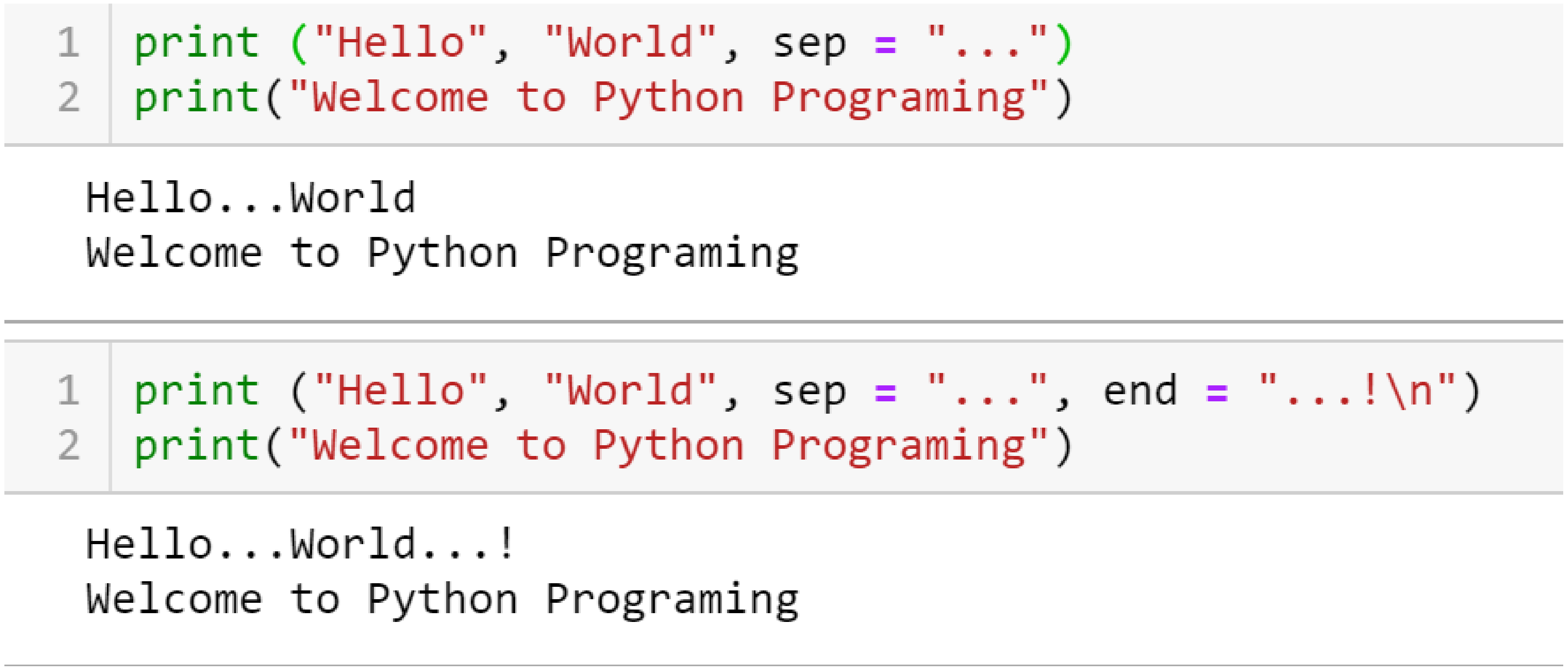
```
John, Mary
John...Mary
```



```python
# print output on same line
# separate output with ...
print("John", "Mary", end = "...")
print(30, 32)
```

```
John Mary...30 32
```

More Examples on the **print()** Function



```python
print ("Hello", "World", sep = "...")
print("Welcome to Python Programing")
```

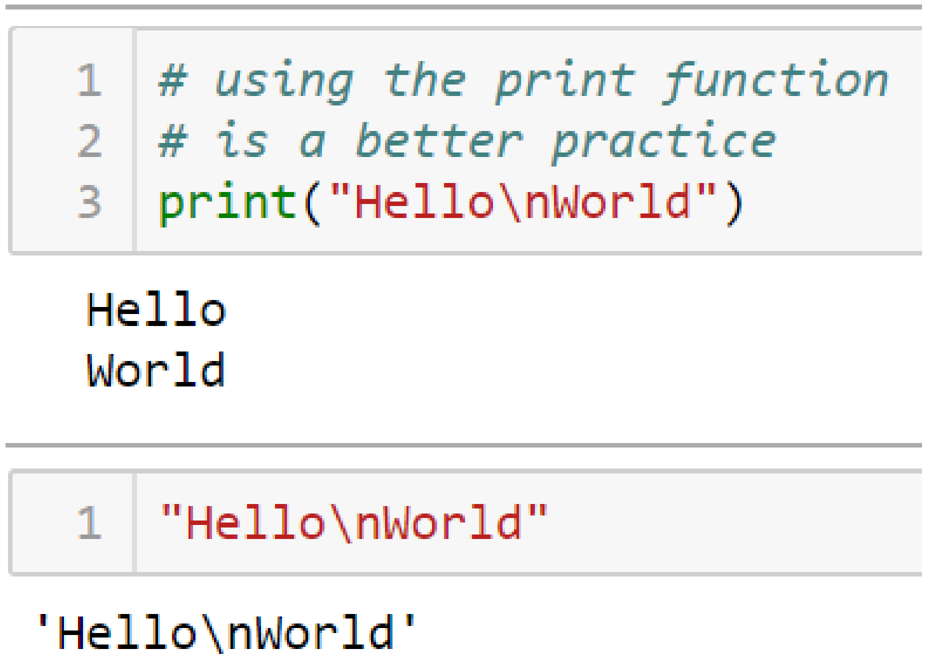
```
Hello...World
Welcome to Python Programing
```

```python
print ("Hello", "World", sep = "...", end = "...!\n")
print("Welcome to Python Programing")
```

```
Hello...World...!
Welcome to Python Programing
```

Using the print() Function is Better!

- Using the print() function instead of just running your code through the interactive interpreter is a better practice.
- Sometimes, generating output without the print function produces weird results.



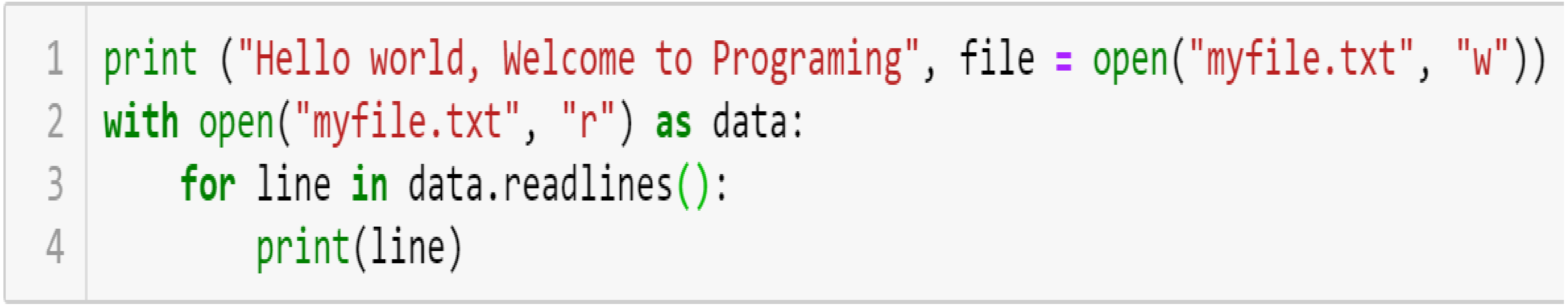
```
# using the print function
# is a better practice
print("Hello\nWorld")
```

```
Hello
World
```

```
"Hello\nWorld"
```

```
'Hello\nWorld'
```

Print into a File



```python
print ("Hello world, Welcome to Programing", file = open("myfile.txt", "w"))
with open("myfile.txt", "r") as data:
    for line in data.readlines():
        print(line)
```

```
Hello world, Welcome to Programing
```

Here, we have written some text to **myfile.txt** file object and now we can loop through the lines in that file to read and print out the lines into stdout (standard stream output) file object.

Printing the Hard Way

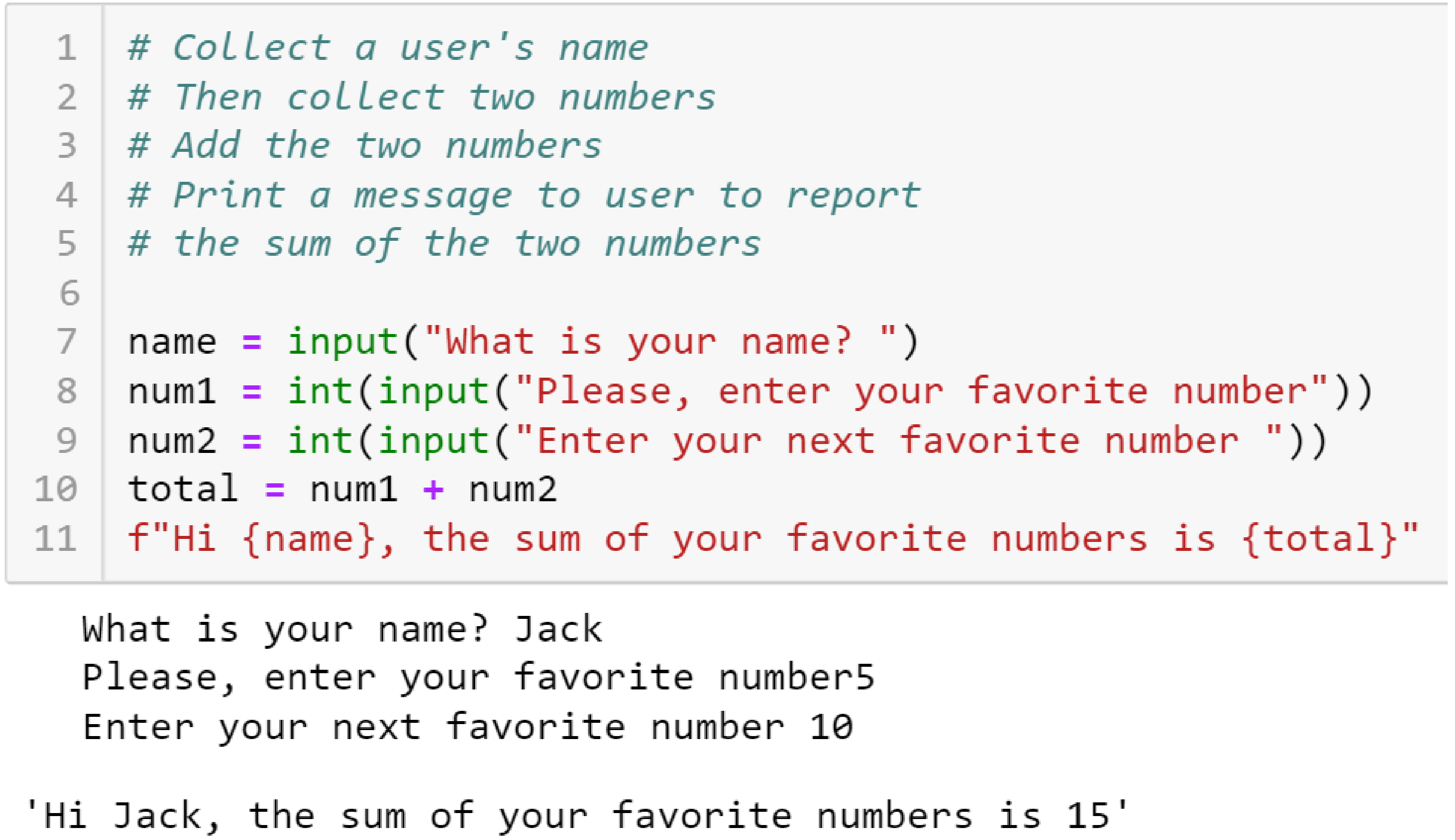
```python
# This is actually what happens
# behind the scene when you print
# note that the default file that  the print
# function sends the output to, is sys.stdout
import sys
sys.stdout.write("Hello World\n")
```

```
Hello World
```

Input or User Input

- Programs can be written to accept input from user using the input() function (called raw_input() in Python 2.X)
- Note that the input function returns what the user typed as a string
- So, if you need to process collected numbers mathematically, you will need to convert the numbers (str) to numeric type (int, float etc)
- Apart from using int, float, etc, you can also use the eval () function to remove quotes from a value.

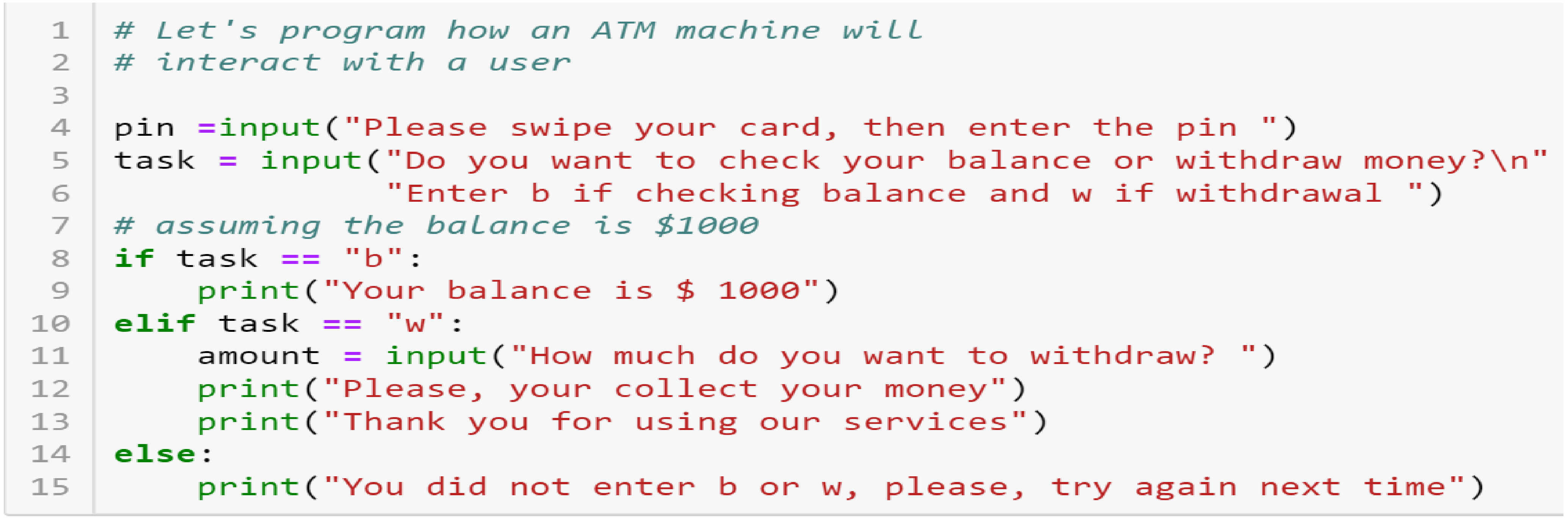
User Input in Action



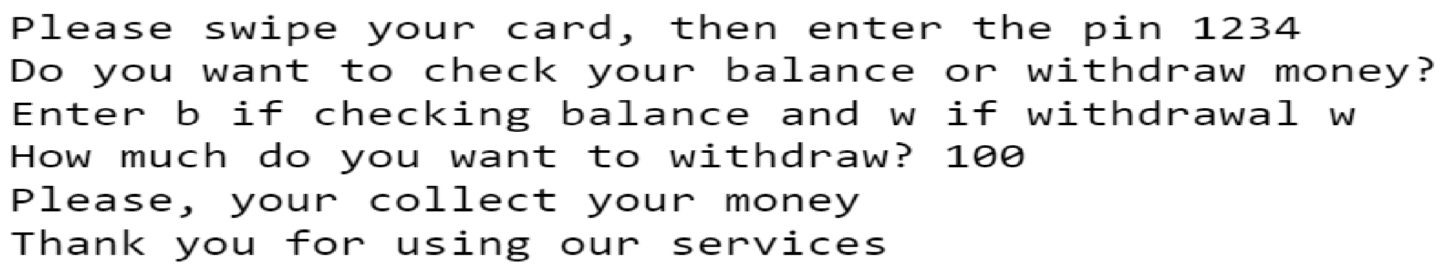
```python
# Collect a user's name
# Then collect two numbers
# Add the two numbers
# Print a message to user to report
# the sum of the two numbers

name = input("What is your name? ")
num1 = int(input("Please, enter your favorite number"))
num2 = int(input("Enter your next favorite number "))
total = num1 + num2
f"Hi {name}, the sum of your favorite numbers is {total}"
```

```
What is your name? Jack
Please, enter your favorite number5
Enter your next favorite number 10

'Hi Jack, the sum of your favorite numbers is 15'
```

User Input in Action



```
# Let's program how an ATM machine will
# interact with a user

pin =input("Please swipe your card, then enter the pin ")
task = input("Do you want to check your balance or withdraw money?\n"
             "Enter b if checking balance and w if withdrawal ")
# assuming the balance is $1000
if task == "b":
    print("Your balance is $ 1000")
elif task == "w":
    amount = input("How much do you want to withdraw? ")
    print("Please, your collect your money")
    print("Thank you for using our services")
else:
    print("You did not enter b or w, please, try again next time")
```

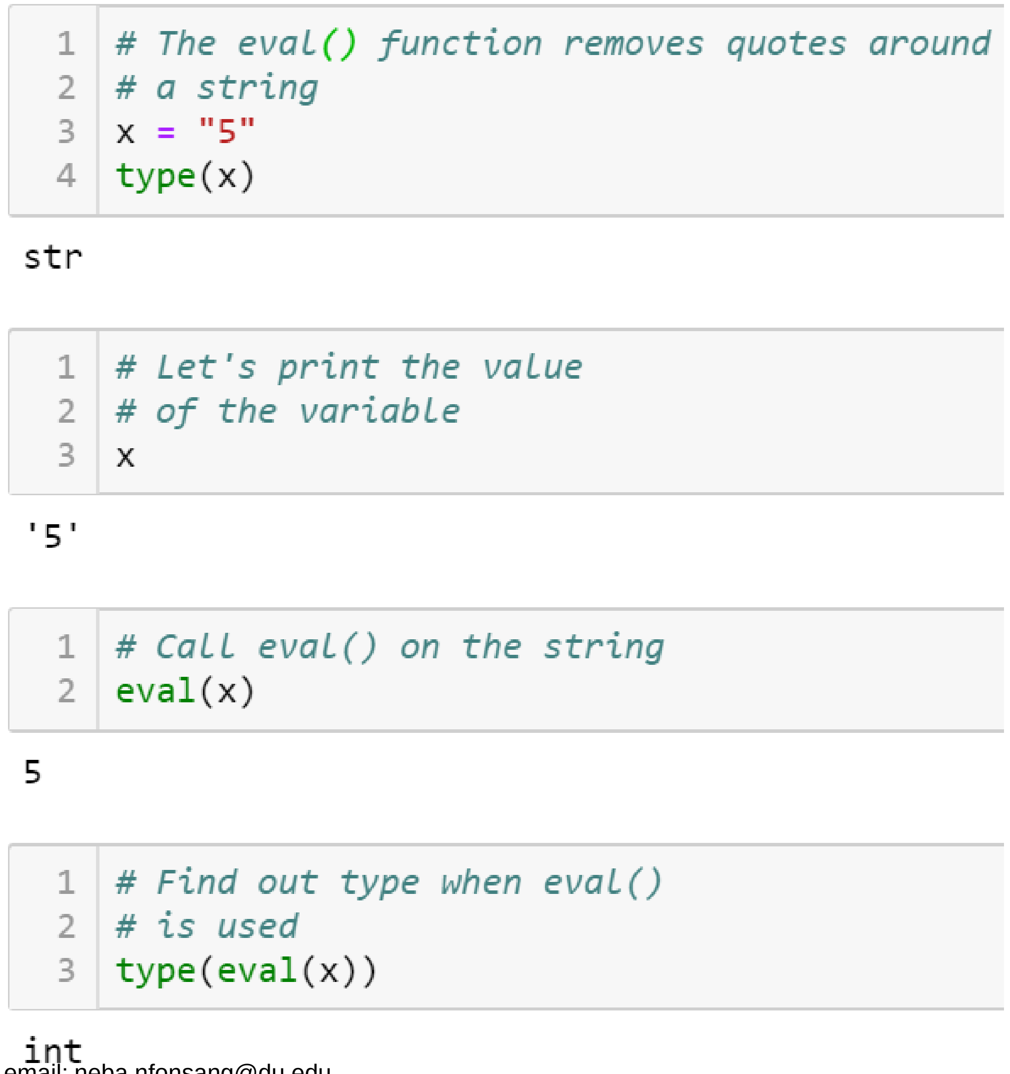


```
Please swipe your card, then enter the pin 1234
Do you want to check your balance or withdraw money?
Enter b if checking balance and w if withdrawal w
How much do you want to withdraw? 100
Please, your collect your money
Thank you for using our services
```

This program would accept text as amount for withdrawal, so it still needs a bit of work, but let’s make it better when we learn control flow statements

The **eval()** function

- You can technically use the eval function to convert a string to an integer.
- The eval() function removes the quotes around a string.



```
# The eval() function removes quotes around
# a string
x = "5"
type(x)
```

```
str
```

```
# Let's print the value
# of the variable
x
```

```
'5'
```

```
# Call eval() on the string
eval(x)
```

```
5
```

```
# Find out type when eval()
# is used
type(eval(x))
```

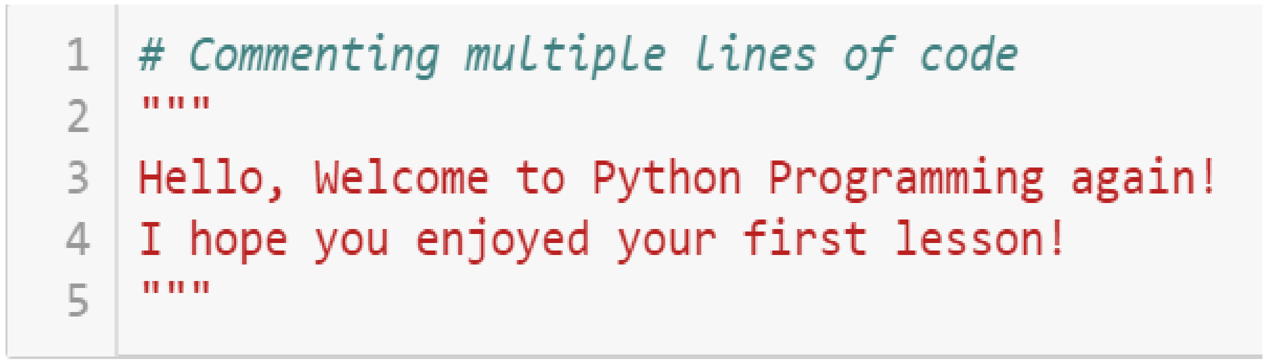
```
int
```

Comments

- As you can see, we have used comments to comment out some codes.
- A comment is a line of code that starts with the hash (#) sign.
- A comment is not executed when you run your code
- A comment is used to add description to what your code does.
- When programs become large or complex, it is good to provides comments where necessary.

Comments/Doc Strings

- You could use triple quotes to comment multiple lines of codes.
- Such triple quotes are usually used to create doc strings in functions.
- A doc string is used in a function to describe what a function does.
- A doc string should also describe what parameters a function takes as well as what the function returns



```
# Commenting multiple lines of code
"""
Hello, Welcome to Python Programming again!
I hope you enjoyed your first lesson!
"""
```

Key Packages Required for this Course

- This course will use the following packages:
- **Numpy:** used for scientific and numerical computing
- **Pandas:** great for data wrangling and manipulation. Pandas has a lot of data analysis tools
- **statsmodels**: provides tools for estimation of statistical models, conducting statistical tests and data exploration.
- **scikit-learn:** an efficient package for data mining and machine learning.
- **Matplotlib:** great for data visualization

Key Packages Required for this Course

- **searborn:** used for statistical data visualization. It is built on Matplotlib.
- **PyMongo:** provides tools for connecting and working with MongoDB database in Python.
- **FactorAnalyzer:** a package for exploratory factor analysis in Python.
- **Beutifulsoup:** a Python package for parsing and extracting data from the web or internet.

Python Objects

- Remember that everything in Python is an object.
- An object is a piece of memory allocated to a value.
- We will begin to explore object types (data types) in our next lesson.
- Happy Pythoning!

End of Lesson