National University of Computer and Emerging Sciences

CL 461 Artificial Intelligence

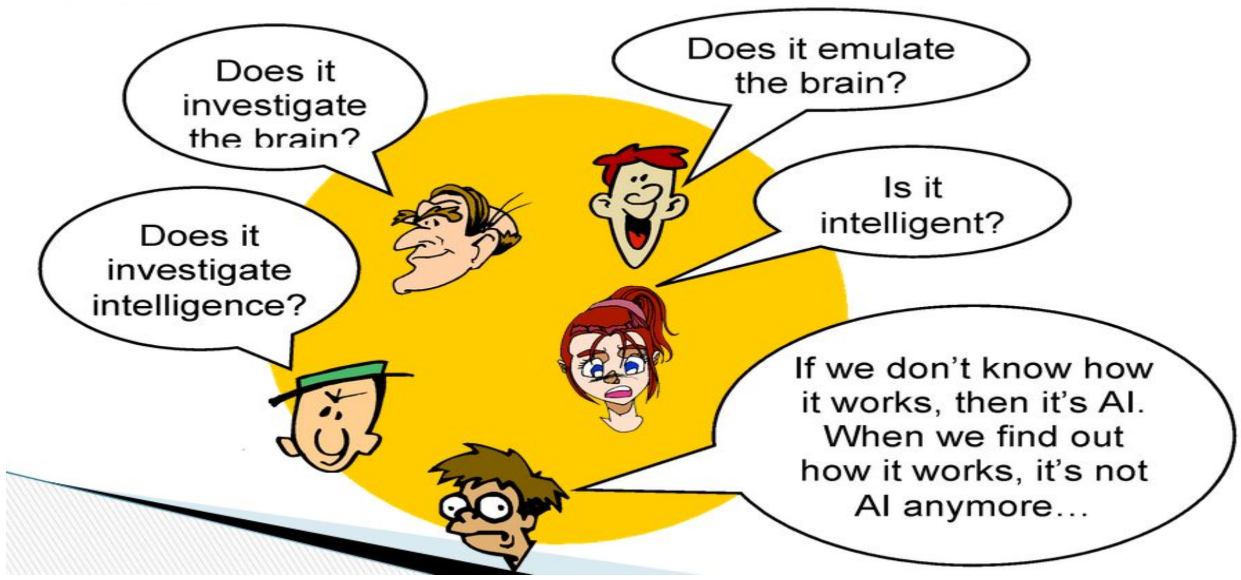
Lab Manual 01

Introduction to Python

Contents

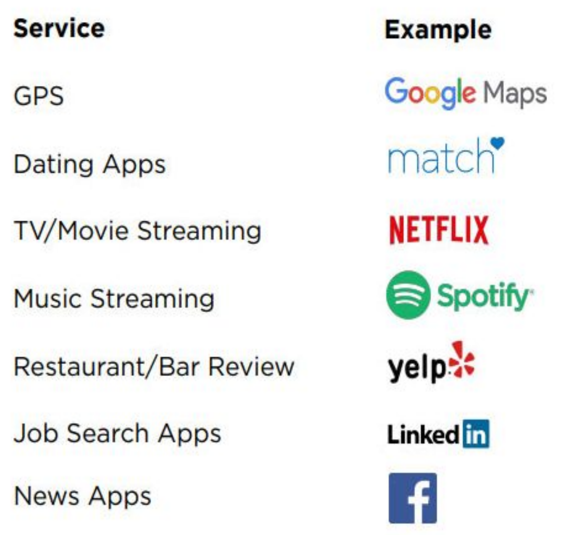
WHAT IS AI?

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving.

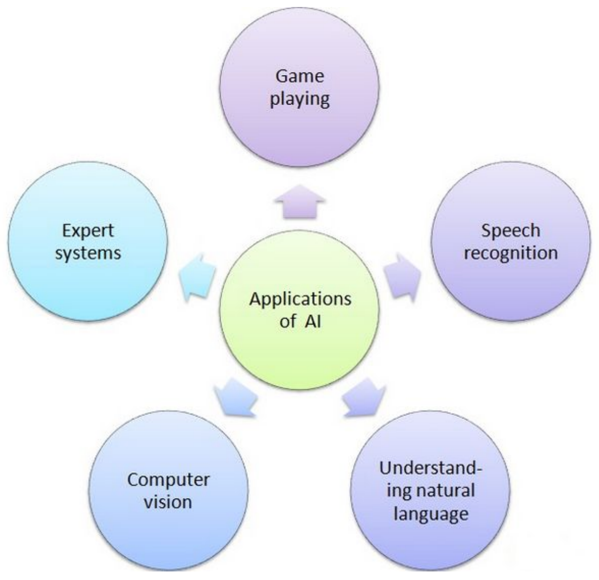

Does it investigate the brain?
Does it emulate the brain?
Is it intelligent?
Does it investigate intelligence?
If we don't know how it works, then it's AI. When we find out how it works, it's not AI anymore…

AI IN DAILY LIVES

With all the excitement and hype about AI that's "just around the corner" self-driving cars, instant machine translation, etc. it can be difficult to see how AI is affecting the lives of regular people from moment to moment. What are examples of artificial intelligence that you're already using right now?



Service	Example
GPS	Google Maps
Dating Apps	match
TV/Movie Streaming	NETFLIX
Music Streaming	Spotify
Restaurant/Bar Review	yelp
Job Search Apps	LinkedIn
News Apps	f

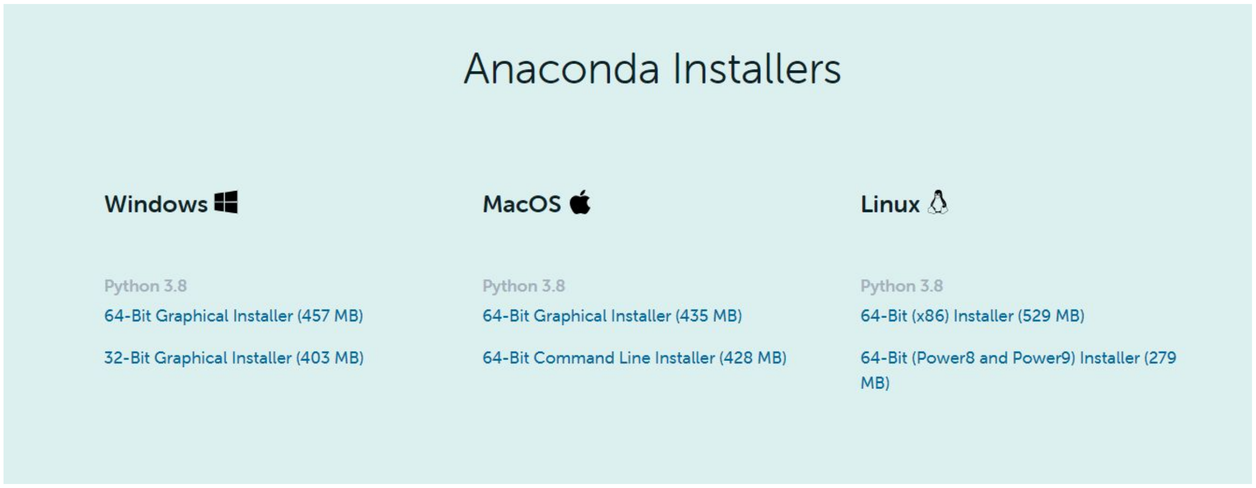


CONFIGURATION GUIDE

Link to download software

https://www.anaconda.com/products/individual

- After clicking the download option you will have the option to download any compatible software of your choice.



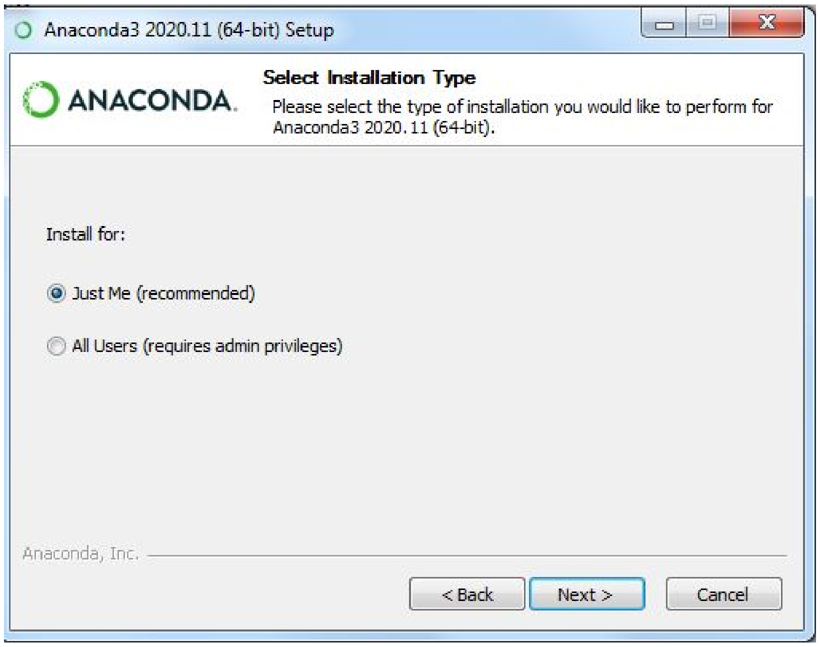
Anaconda Installers

Windows	MacOS	Linux
Python 3.8	Python 3.8	Python 3.8
64-Bit Graphical Installer (457 MB)	64-Bit Graphical Installer (435 MB)	64-Bit (x86) Installer (529 MB)
32-Bit Graphical Installer (403 MB)	64-Bit Command Line Installer (428 MB)	64-Bit (Power8 and Power9) Installer (279 MB)

INSTALLATION STEPS:

STEP#1

- Select the option, Install for Just Me which is also recommended one.



Anaconda3 2020.11 (64-bit) Setup

Select Installation Type

Please select the type of installation you would like to perform for Anaconda3 2020.11 (64-bit).

Install for:

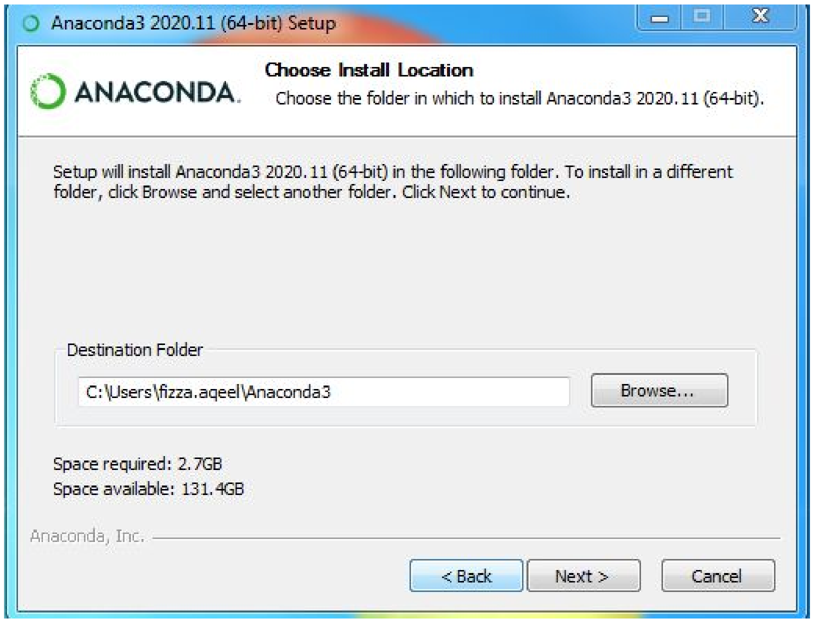
[✓ Just Me (recommended)]

All Users (requires admin privileges)

Anaconda, Inc.

STEP#2:

- You need at-least 3GB of space in the specified folder where you want to install Anaconda.



Anaconda3 2020.11 (64-bit) Setup

Choose Install Location

Choose the folder in which to install Anaconda3 2020.11 (64-bit).

Setup will install Anaconda3 2020.11 (64-bit) in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

Destination Folder

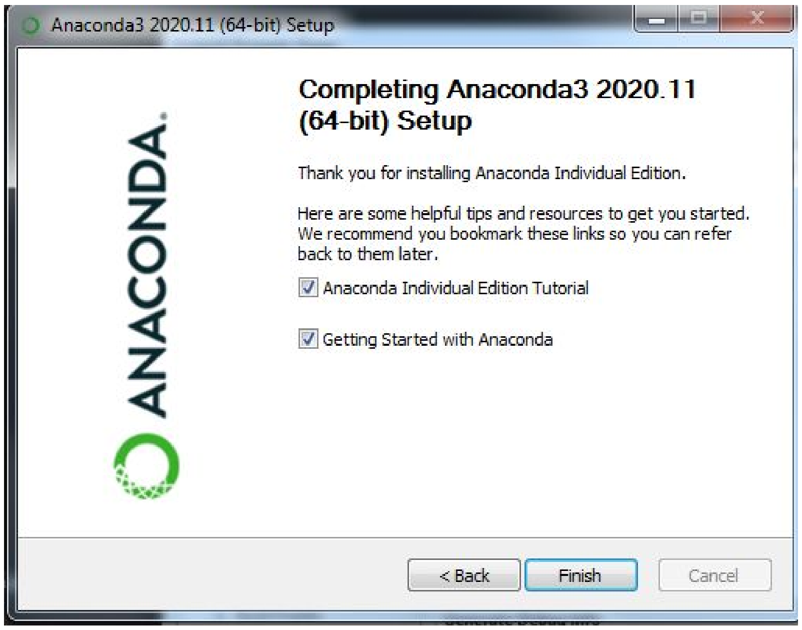
C:\Users\fizza.aqeel\Anaconda3

Space required: 2.7GB
Space available: 131.4GB

Anaconda, Inc.

STEP#3:

- Now, you can start working on your Anaconda.



Anaconda3 2020.11 (64-bit) Setup

Completing Anaconda3 2020.11 (64-bit) Setup

Thank you for installing Anaconda Individual Edition.

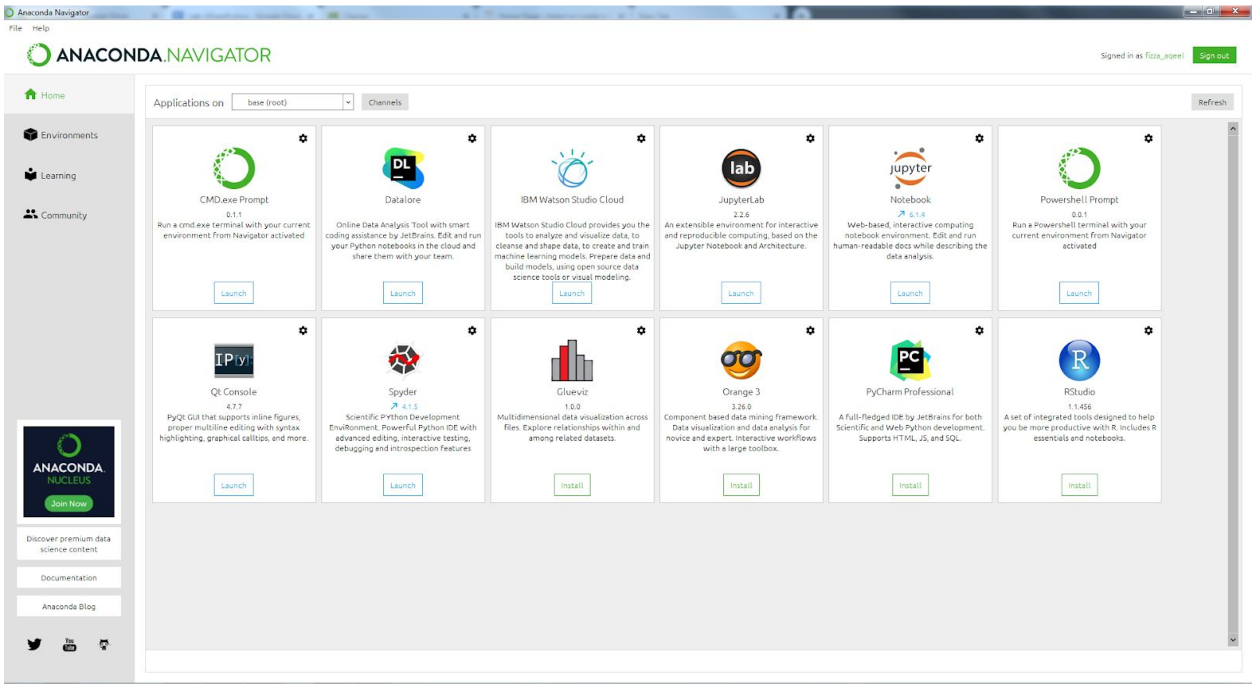
Here are some helpful tips and resources to get you started. We recommend you bookmark these links so you can refer back to them later.

☑ Anaconda Individual Edition Tutorial

☑ Getting Started with Anaconda

STEP#4:

- This is the interface of Anaconda, from here we can see different python IDE's. You are free to use either Jupyter Notebook or Spyder.



CMD.exe Prompt 0.1.1 — Run a cmd.exe terminal with your current environment from Navigator activated

Datalore — Online Data Analysis Tool with smart coding assistance by JetBrains. Edit and run your Python notebooks in the cloud and share them with your team.

IBM Watson Studio Cloud — IBM Watson Studio Cloud provides you the tools to analyze and visualize data, to cleanse and shape data, to create and train machine learning models. Prepare data and build models, using open source data science tools or visual modeling.

JupyterLab 2.2.6 — An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Notebook ↗ 6.1.4 — Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Powershell Prompt 0.0.1 — Run a Powershell terminal with your current environment from Navigator activated

Qt Console 4.7.7 — PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Spyder ↗ 4.1.5 — Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

Glueviz 1.0.0 — Multidimensional data visualization across files. Explore relationships within and among related datasets.

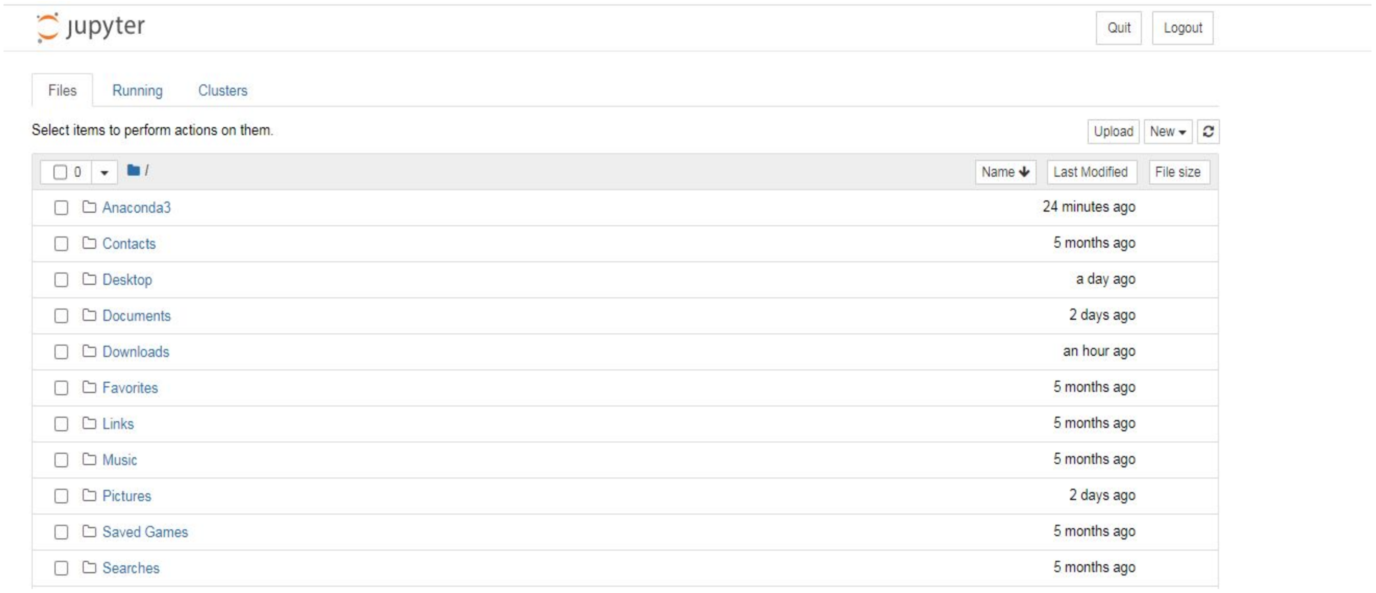
Orange 3 3.26.0 — Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

PyCharm Professional — A full-fledged IDE by JetBrains for both Scientific and Web Python development. Supports HTML, JS, and SQL.

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

STEP#5:

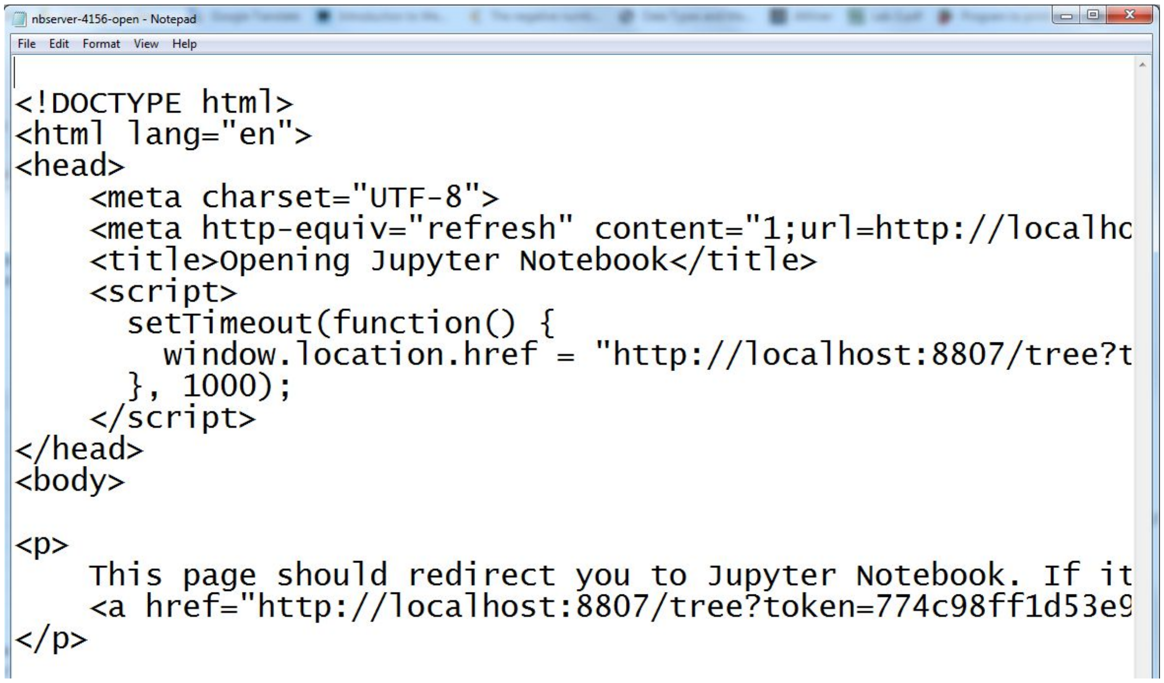
- If you are using a jupyter notebook just click the "Launch" option and you will see the screen below in one of your default browsers.



Select items to perform actions on them.

Name	Last Modified
Anaconda3	24 minutes ago
Contacts	5 months ago
Desktop	a day ago
Documents	2 days ago
Downloads	an hour ago
Favorites	5 months ago
Links	5 months ago
Music	5 months ago
Pictures	2 days ago
Saved Games	5 months ago
Searches	5 months ago

- But if sometimes it won't open in your browser and shows a note notepad file of your launching server like this:



nbserver-4156-open - Notepad

File Edit Format View Help

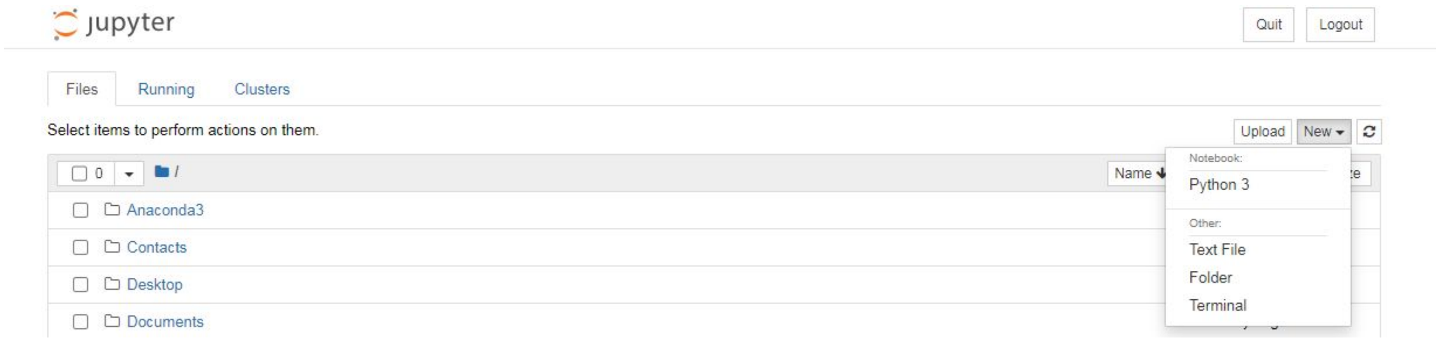
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="refresh" content="1;url=http://localhc
    <title>Opening Jupyter Notebook</title>
    <script>
      setTimeout(function() {
        window.location.href = "http://localhost:8807/tree?t
      }, 1000);
    </script>
</head>
<body>

<p>
    This page should redirect you to Jupyter Notebook. If it
    <a href="http://localhost:8807/tree?token=774c98ff1d53e9
</p>
```

- So in that case you have to select the "URL" value in the between the meta tags in your notepad file till the token id and copy paste that url value on your browser and same as above screen will be displayed:
- Below is the URL path of the notebook in my PC and in your notepad you have your individual paths, this is the alternate if the jupyter notebook cannot open directly.
- Copy the URL value from http till the full token value without collins.

URL=http://localhost:8807/tree?token=774c98ff1d53e94346efabc462c8bb030df654dd59d4d80c

<u>STEP#6:</u>

- Now, in the end you can select the New option from your Jupyter Notebook window under the new tab you have to select Python3 file to create your .py file and in the end you have created your first python file successfully and you are free to work there.



MODULES IN PYTHON:

- A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a file consisting of Python code. A module can define functions, classes and variables.
- In Python, modules are accessed by using the 'import' statement. When you do this, you execute the code of the module, keeping the scopes of the definitions so that your current file(s) can make use of these.
- Many build-in modules of python, some of above
    1. Math
    2. Random
    3. Fraction
    4. Decimal
    5. OS

Operator in Python:

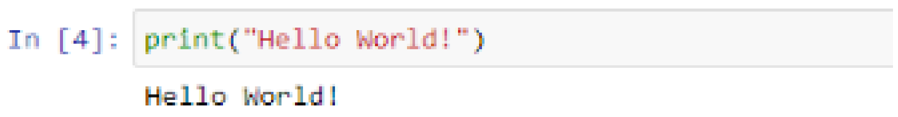
- Basic algebraic operations
    - ✔ Four arithmetic operations: a+b, a-b, a*b, a/b
    - ✔ Exponentiation: a**b
    - ✔ Other elementary functions are not part of standard Python, but included in packages like NumPy and SciPy
- Comparison operators
    - ✔ Greater than, less than, etc.: a < b, a > b, a <= b, a >= b
    - ✔ Identity tests: a == b, a != b
- Bitwise operators
    - ✔ Bitwise or: a | b
    - ✔ Bitwise exclusive or: a ^ b # Don't confuse this with exponentiation
    - ✔ Bitwise and: a & b
    - ✔ Shift a left or right by b bits: a << b, a >> b

Input( ) Function:

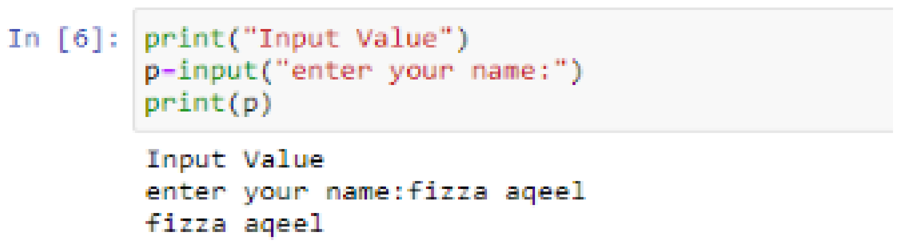
The input() function pauses your program and waits for the user to enter some text. Once Python receives the user's input, it stores it in a variable to make it convenient for you to work with.
The purpose of an input statement is to get some information from the user of a program and store it into a Variable.

- Syntax: <variable> = input (<prompt>)

Example:



```
In [4]: print("Hello World!")

Hello World!
```



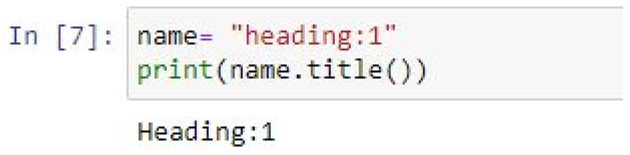
```
In [6]: print("Input Value")
        p-input("enter your name:")
        print(p)

Input Value
enter your name:fizza aqeel
fizza aqeel
```

STRINGS:

A *string* is simply a series of characters. Anything inside quotes is considered a string in Python, and you can use single or double quotes around your strings like this:
"This is a string."
'This is also a string.'

Changing Case in a String with Methods

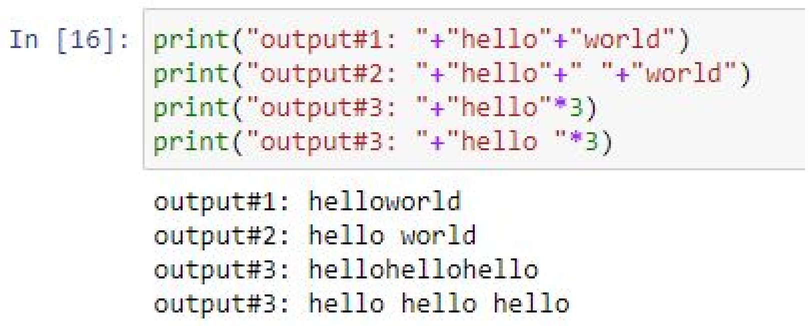
One of the simplest tasks you can do with strings is change the case of the words in a string. Look at the following code, and try to determine what's happening:



```
In [7]: name= "heading:1"
        print(name.title())

Heading:1
```

Combining or Concatenating Strings:

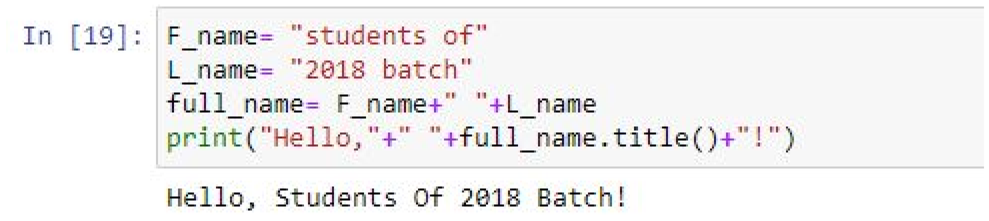
When applied to strings, the + operation is called concatenation. It produces a new string that is a copy of the two original strings pasted together end-to-end. Notice that concatenation doesn't do anything clever like insert a space between the words. The Python interpreter has no way of knowing that you want a space; it does exactly what it is told.



```
In [16]: print("output#1: "+"hello"+"world")
         print("output#2: "+"hello"+" "+"world")
         print("output#3: "+"hello"*3)
         print("output#3: "+"hello "*3)

output#1: helloworld
output#2: hello world
output#3: hellohellohello
output#3: hello hello hello
```

Strings can be concatenated with the '+' operator and repeated with '*'

Example:



```
In [19]: F_name= "students of"
         L_name= "2018 batch"
         full_name= F_name+" "+L_name
         print("Hello,"+" "+full_name.title()+"!")

         Hello, Students Of 2018 Batch!
```

INDEXING OF STRING:

Python starts indexing at 0. A string s will have indexes running from 0 to len(s)-1 (where len(s) is the length of s) in integer quantities.

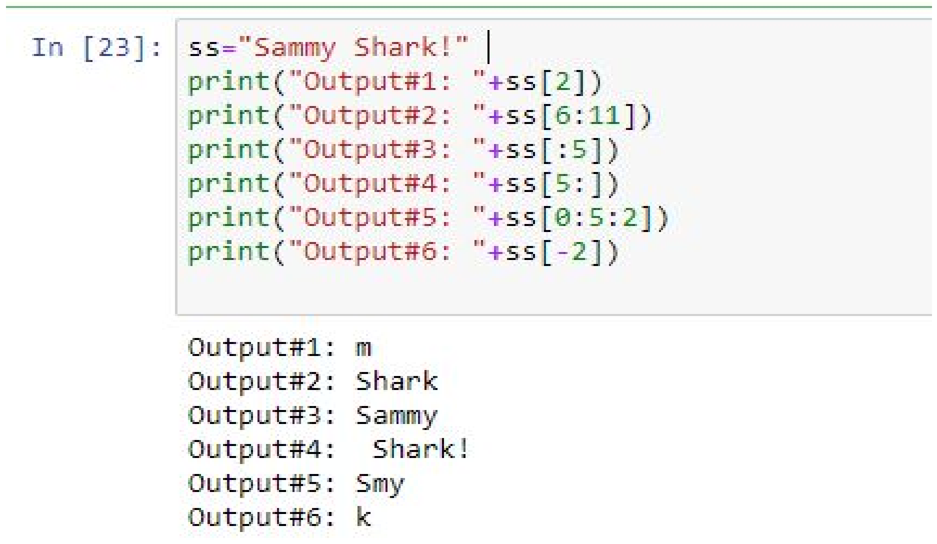
S	A	m	m	Y		S	h	A	r	k	!
0	1	2	3	4	5	6	7	8	9	10	11

Example:

- s[i] : fetches the ith element in s
- s[i:j] : fetches elements i (inclusive) through j (not inclusive)
- s[:j] fetches all elements up to, but not including j
- s[i:] fetches all elements from i onward (inclusive)
- s[i:j:k] extracts every kth element starting with index i (inlcusive) and ending with index j (not inclusive)
- Python also supports negative indexes. For example, s[-1] means extract the first element of s from the end (same as s[len(s)-1])

```
ss=”Sammy Shark! “
print(ss[2])
print(ss[6:11])
print(ss[:5])
print(ss[5:])
print(ss[0:5:2])
print(ss[-2])
```

Output:



```
In [23]: ss="Sammy Shark!"
         print("Output#1: "+ss[2])
         print("Output#2: "+ss[6:11])
         print("Output#3: "+ss[:5])
         print("Output#4: "+ss[5:])
         print("Output#5: "+ss[0:5:2])
         print("Output#6: "+ss[-2])
```

```
Output#1: m
Output#2: Shark
Output#3: Sammy
Output#4:  Shark!
Output#5: Smy
Output#6: k
```

<u>**Functions:**</u>

Basically, we can divide functions into the following two types:

1. Built-in functions - Functions that are built into Python.
2. User-defined functions - Functions defined by the users themselves.

A function is a block of organized, reusable code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusing.
As you already know, Python gives you many built-in functions like print(), etc. but you can also create your own functions. These functions are called user-defined functions.

<u>**DEFINING A FUNCTION:**</u>

You can define functions to provide the required functionality. Here are simple rules to define a function in Python.

- Function blocks begin with the keyword **def** followed by the function name and parentheses (( )).

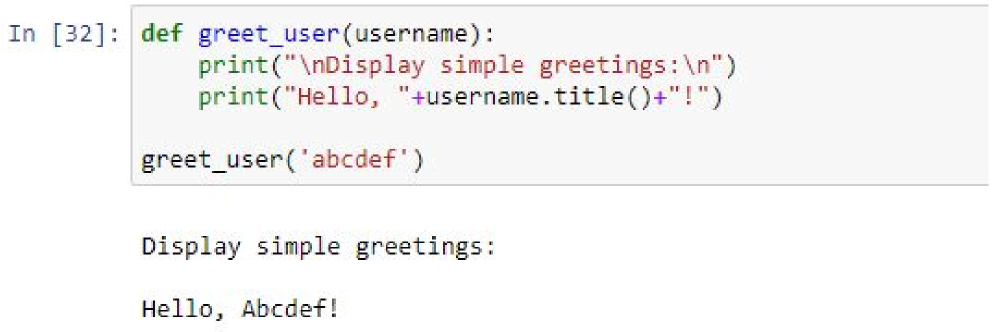
- Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.

- The code block within every function starts with a colon (:) and is indented.

Syntax:

```
def functioname( PARAMETERS ):
   statement
```

Example:



```
In [32]: def greet_user(username):
             print("\nDisplay simple greetings:\n")
             print("Hello, "+username.title()+"!")

         greet_user('abcdef')

         Display simple greetings:

         Hello, Abcdef!
```

For loop Statement:

Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable. It has the ability to iterate over the items of any sequence, such as a list or a string.

Syntax:

```
for iterating variable in sequence:
Statements(s)
```

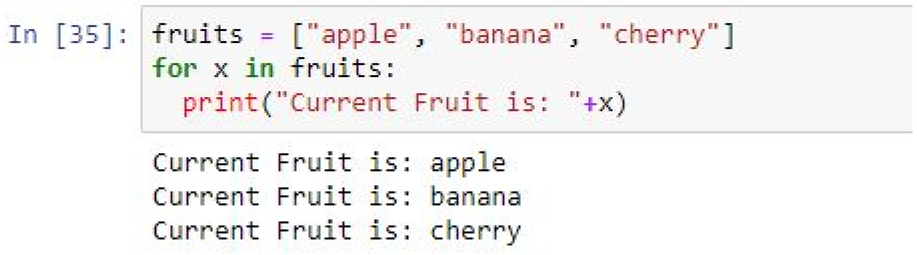
Example#01:

fruits = ["apple", "banana", "cherry"]

for x in fruits:

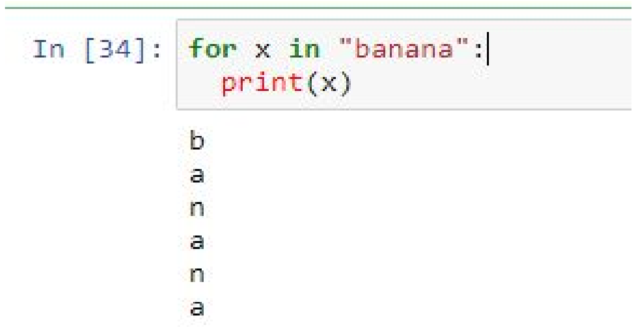
  print("Current Fruit is: "+x)

Output:



```
In [35]: fruits = ["apple", "banana", "cherry"]
         for x in fruits:
           print("Current Fruit is: "+x)

         Current Fruit is: apple
         Current Fruit is: banana
         Current Fruit is: cherry
```

Example#02:



```
In [34]: for x in "banana":
           print(x)

         b
         a
         n
         a
         n
         a
```

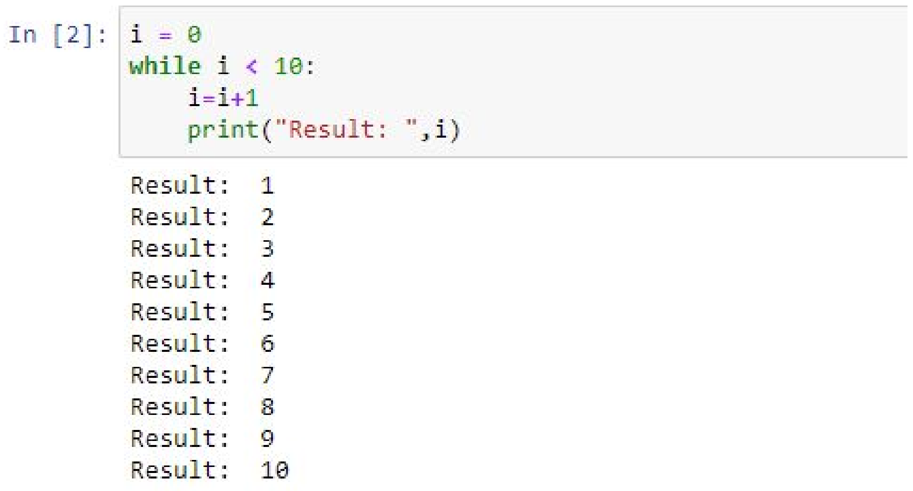
WHILE LOOP:

The for loop takes a collection of items and executes a block of code once for each item in the collection. In contrast, while loop statement in Python programming language repeatedly executes a target statement as long as a given condition is true.

Syntax:

The syntax of a while loop in Python programming language is –
while expression:
statement(s)

Example:



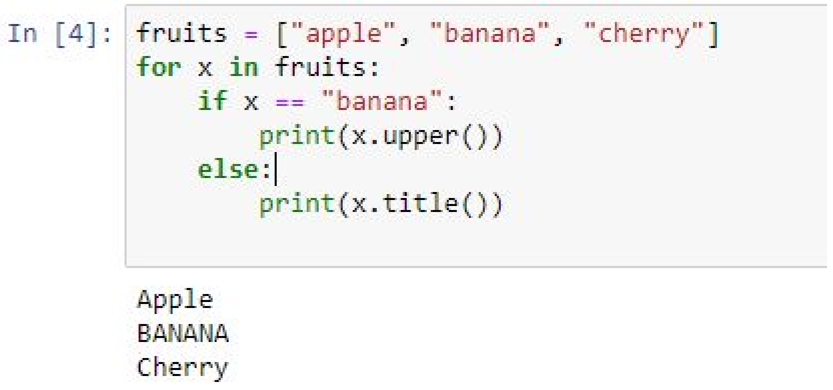
```
In [2]: i = 0
        while i < 10:
            i=i+1
            print("Result: ",i)

        Result:  1
        Result:  2
        Result:  3
        Result:  4
        Result:  5
        Result:  6
        Result:  7
        Result:  8
        Result:  9
        Result:  10
```

CONDITIONAL STATEMENTS:

IF-ELSE STATEMENT:

An *if-else* block is similar to a simple if statement, but the else statement allows you to define an action or set of actions that are executed when the conditional test fails.

Example:



```
In [4]: fruits = ["apple", "banana", "cherry"]
        for x in fruits:
            if x == "banana":
                print(x.upper())
            else:
                print(x.title())

        Apple
        BANANA
        Cherry
```

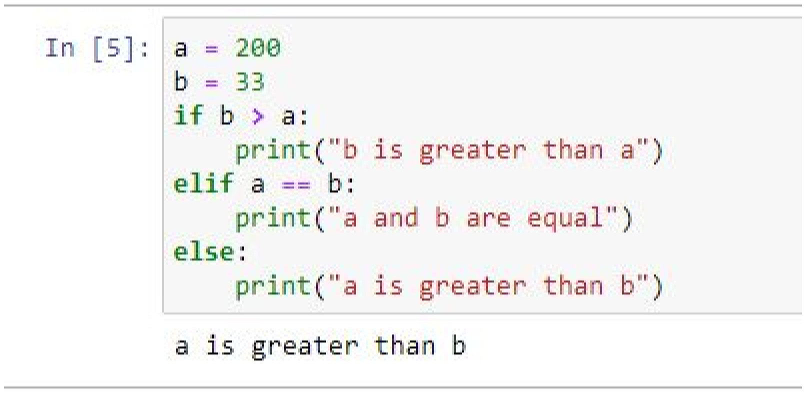
THE *IF-ELIF-ELSE* STATEMENT:

The elif statement allows you to check multiple expressions for TRUE and execute a block of code as soon as one of the conditions evaluates to TRUE.

Syntax:

```
if expression1:
  statement(s)
elif expression2:
  statement(s)
else:
  statement(s)
```

Example:

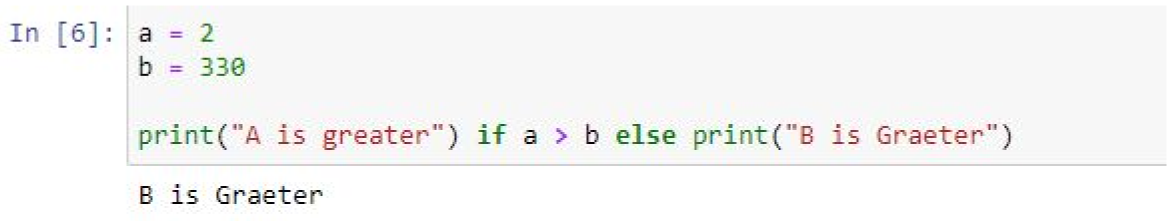


```
In [5]: a = 200
        b = 33
        if b > a:
            print("b is greater than a")
        elif a == b:
            print("a and b are equal")
        else:
            print("a is greater than b")

        a is greater than b
```

SHORTHAND *IF ... ELSE* STATEMENT:

If you have only one statement to execute, one for if, and one for else, you can put it all on the same line.

Example:



```
In [6]: a = 2
        b = 330

        print("A is greater") if a > b else print("B is Graeter")

        B is Graeter
```

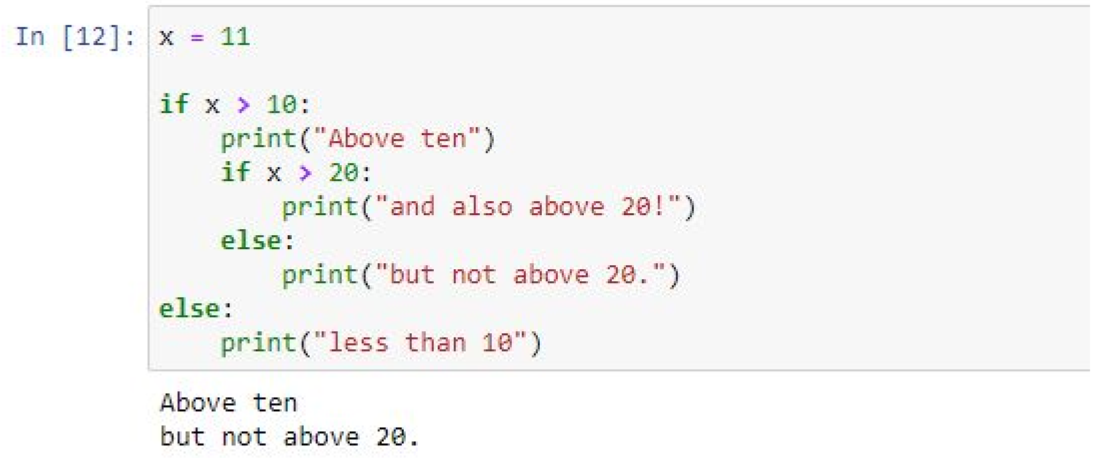
NESTED *IF ... ELSE* STATEMENT:

You can have if statements inside if statements, this is called *nested* if statements.

Syntax:

```
if expression1:
  statement(s)
        if expression2:
            statement(s)
        else:
            statement(s)
else:
  statement(s)
```

Example:

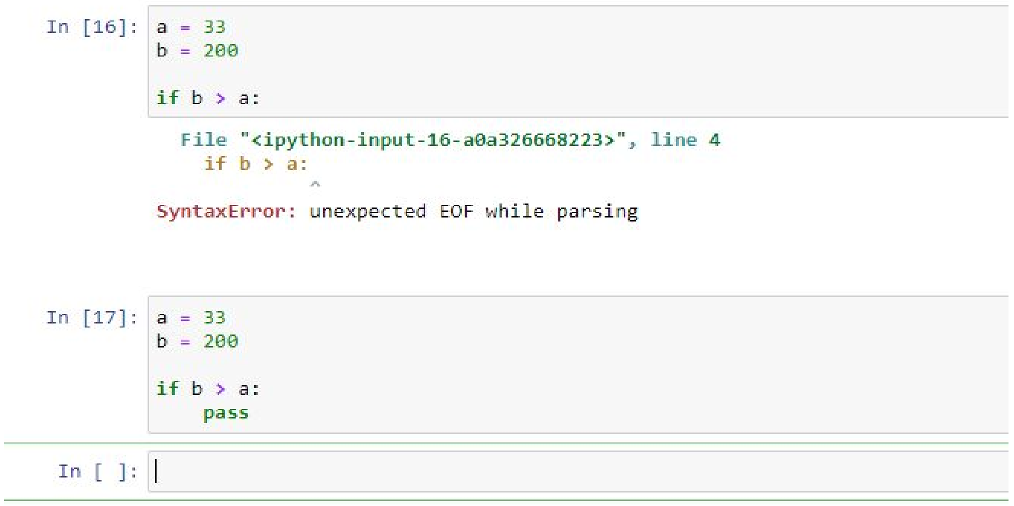


```
In [12]: x = 11

         if x > 10:
             print("Above ten")
             if x > 20:
                 print("and also above 20!")
             else:
                 print("but not above 20.")
         else:
             print("less than 10")

         Above ten
         but not above 20.
```

PASS STATEMENT:

if statements cannot be empty, but if you for some reason have an if statement with no content, put in the **pass** statement to avoid getting an error.

Example:

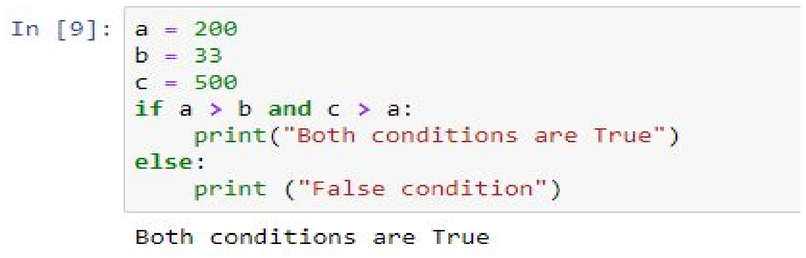


```
In [16]: a = 33
         b = 200

         if b > a:

           File "<ipython-input-16-a0a326668223>", line 4
             if b > a:
                     ^
         SyntaxError: unexpected EOF while parsing


In [17]: a = 33
         b = 200

         if b > a:
             pass

In [ ]: |
```

AND … OR STATEMENTS:

- The **and** keyword is a logical operator, and is used to combine conditional statements.

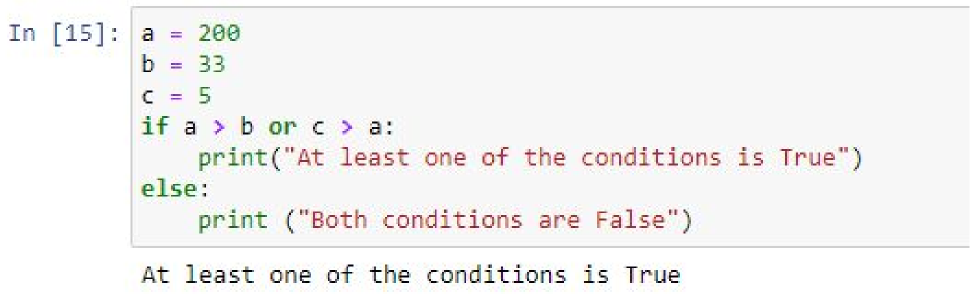
Example:



```
In [9]: a = 200
        b = 33
        c = 500
        if a > b and c > a:
            print("Both conditions are True")
        else:
            print ("False condition")

        Both conditions are True
```

- The **or** keyword is a logical operator, and is used to combine conditional statements.

Example:



```
In [15]: a = 200
         b = 33
         c = 5
         if a > b or c > a:
             print("At least one of the conditions is True")
         else:
             print ("Both conditions are False")

         At least one of the conditions is True
```

RECOMMENDED COURSES:

[CS50's Introduction to Artificial Intelligence with Python by Hardvard University on edX]

[Programming for Everybody (Getting Started with Python) by University of Michigan on Coursera]

LAB#1 EXERCISE:

1. Write a python program that takes user input for a number then adds 3 to that number.Then multiplies the result by 2, subtract 4, then again adds 3, then prints the result.
2. Write a python program that takes input as radius then calculates the area of the circle. (hint: $A = \pi r^2$).
3. Write a Python program that asks users for their favourite color. Create the following output (assuming blue is the chosen color) (hint: use '+' and '*')

```
blueblueblueblueblueblueblueblueblueblue
blue                                blue
blueblueblueblueblueblueblueblueblueblue
```

4. Store a person's name, and include some whitespace characters at the beginning and end of the name. Make sure you use each character combination, "\t" and "\n", at least once. Print the name once, so the whitespace around the name is displayed. Then print the name using each of the three stripping functions, lstrip(),rstrip(), and strip().
5. Write a function called absolute_num() that accepts one parameter, num. The function should return only positive value , and apply condition on it. This function returns the absolute value of the entered number.

6. Write a python program that takes a user input and to create the multiplication table (from 1 to 10) of that number.

7. Write a Python program that prints all the numbers from 0 to 6 except 3 and 6.
   Note : Use 'continue' statement.

8. Stages of Life: Write an if-elif-else chain that determines a person's stage of life. Set a value for the variable age, and then:
   - If the person is less than 2 years old, print a message that the person is a baby.
   - If the person is at least 4 years old but less than 13, print a message that the person is a kid.
   - If the person is at least 13 years old but less than 20, print a message that the person is a teenager.
   - If the person is at least 20 years old but less than 65, print a message that the person is an adult.
   - If the person is age 65 or older, print a message that the person is an elder.