|  |
| --- |
| **Lab 01: Introduction to Python IDEs and Python Data Types** |

Contents:

|  |
| --- |
| 1. Anaconda IDEs Jupyter Notebook 2. Python Data Types 3. Exercise |

|  |
| --- |
| **1. Anaconda IDE** |

**What is ANACONDA?**

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment.

**Languages used**: Python

**Software genre**: Programming language

**Source of definition:** <https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)#:~:text=Anaconda%20is%20a%20distribution%20of,simplify%20package%20management%20and%20deployment>.

**How to start and set up the project in ANACONDA?**

Install Anaconda, as any other software, by one click installation after downloading from the following link

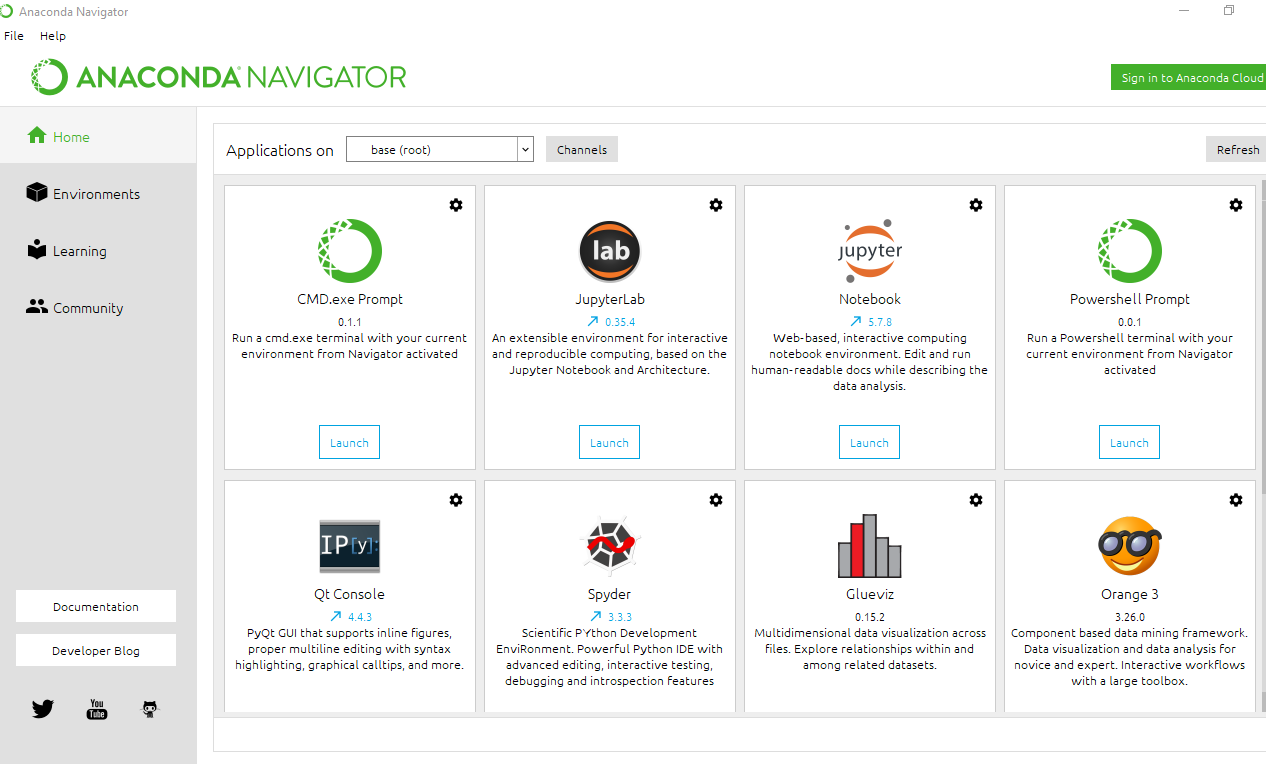
<https://www.anaconda.com/products/individual>

**Start Anaconda :**

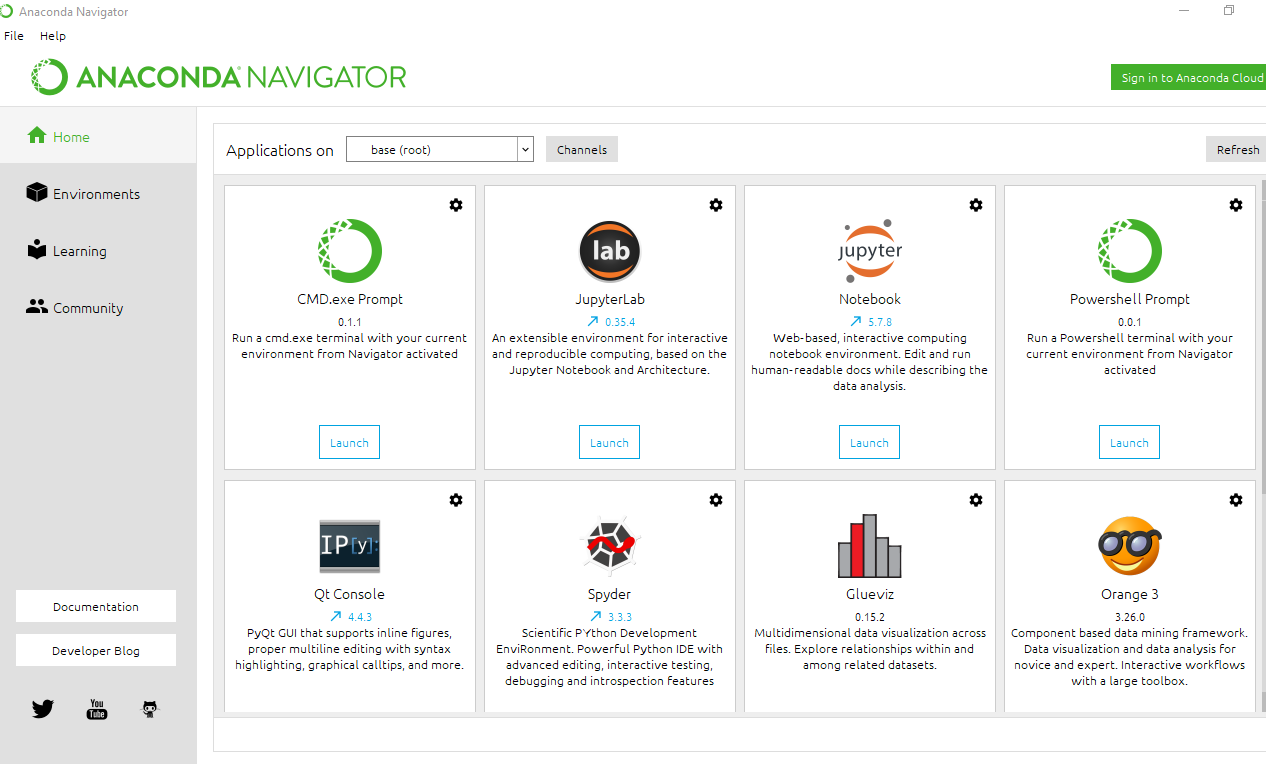
* Start Anaconda from Start Menu and it will look like this:



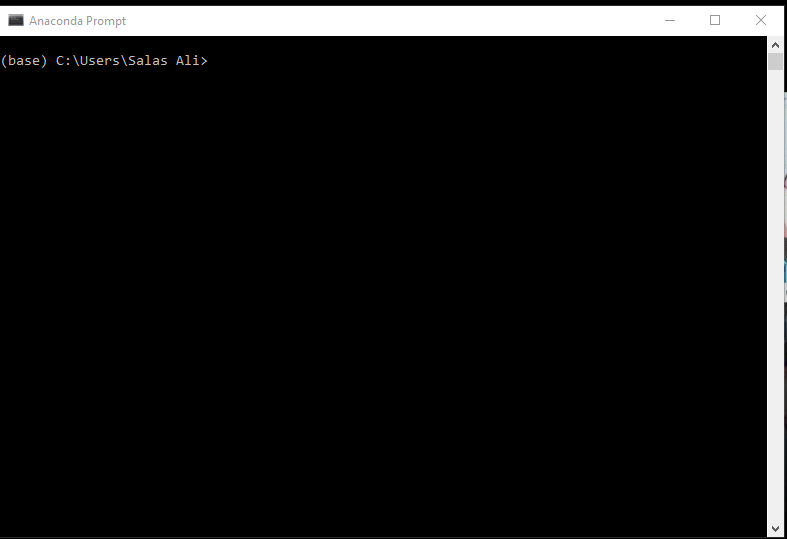
* Once it opens, the interface looks like this:



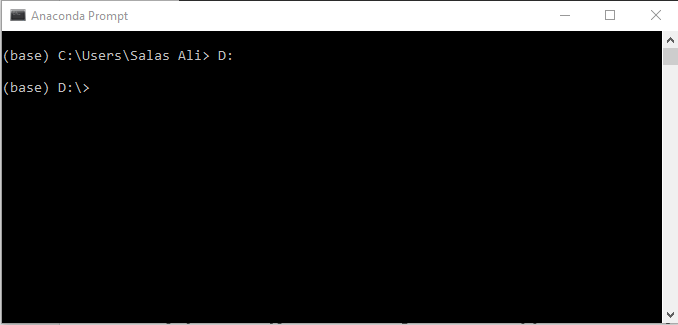
* As we are going to use, for now, jupyter notebook:



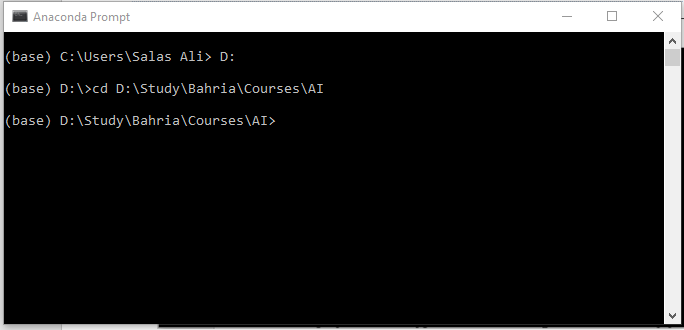
* We are supposed to open it in specific directory
* Go to start menu, type anaconda prompt
* Open it



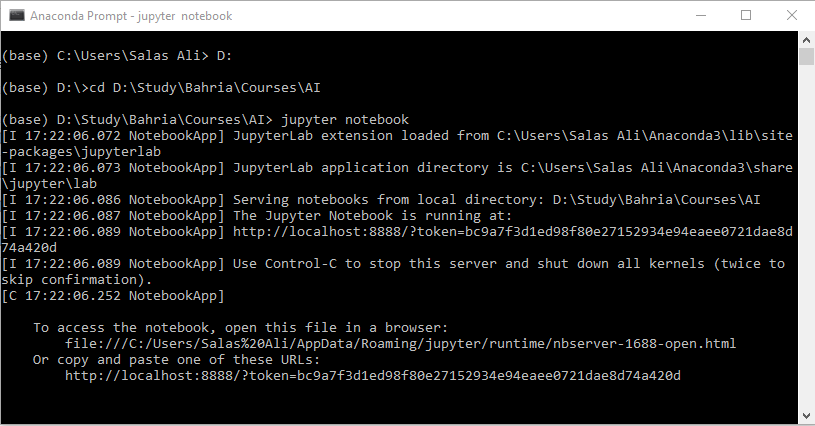
* Type commnd to navigate to the directory you want ot create a project



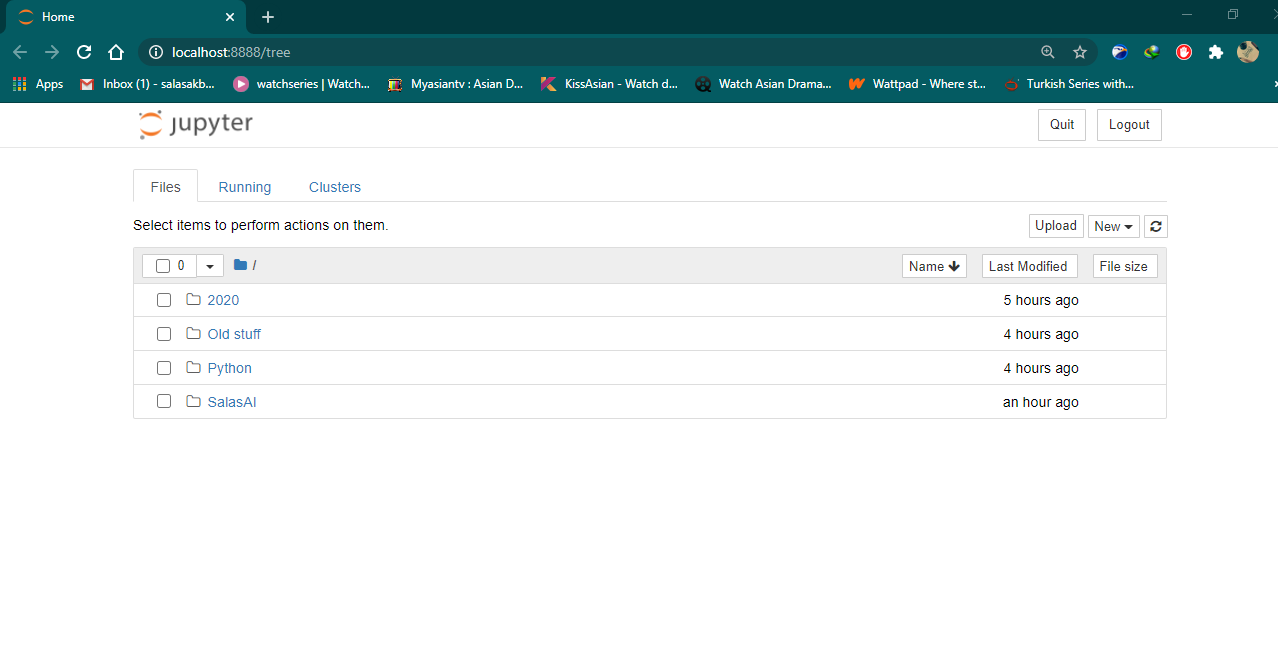
* Use cd command to go to the specific folder where your project will be created



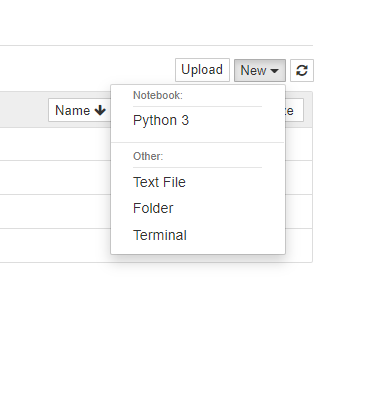
* After jumping to the specific folder (e.g. “AI”), type “Jupyter notebook”. It will start Jupyter Notebook and prompt for selecting the browser of your choice (Chrome is recommended).



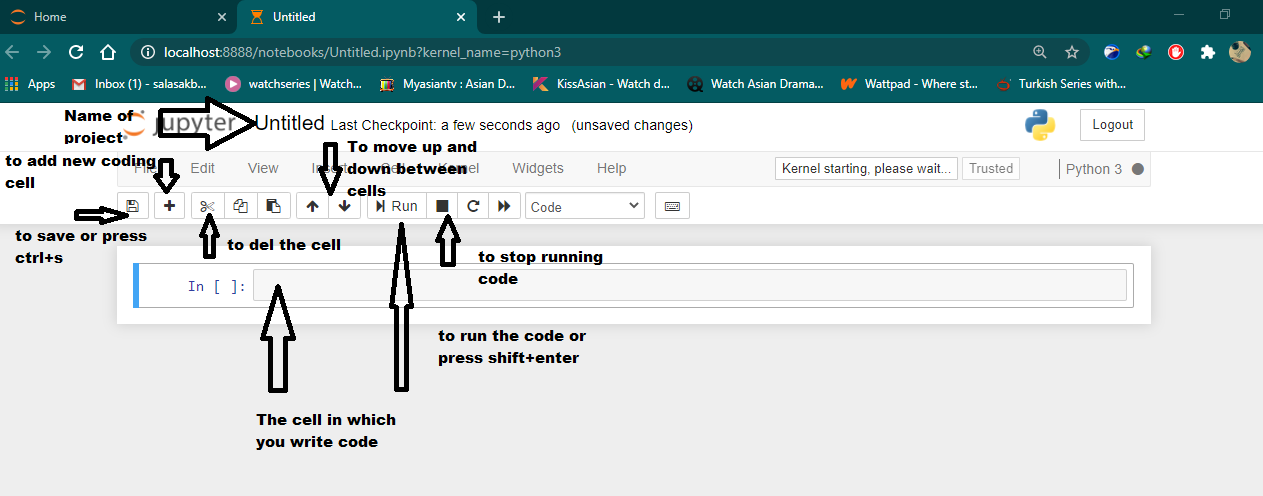
* Following tab appears once the browser application is chosen:



* Create new project by clicking on New, and select Python 3:

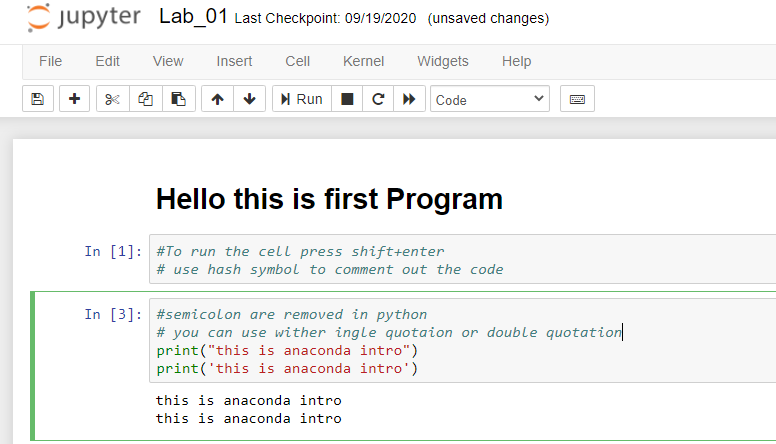


* You are now ready to code in the following window:



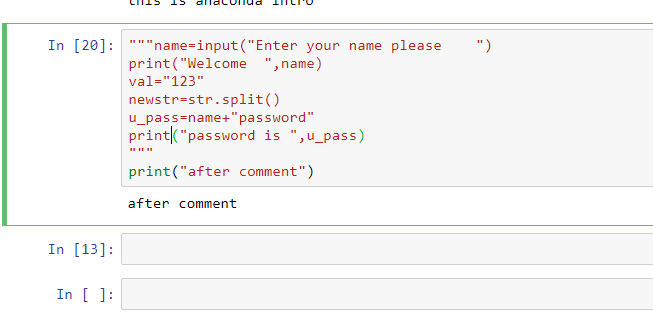
|  |
| --- |
| **4. Basic Python Program** |

This section explains further on how to write a python program. Read comments carefully in the given screen shot. print() is used to output something on the console:



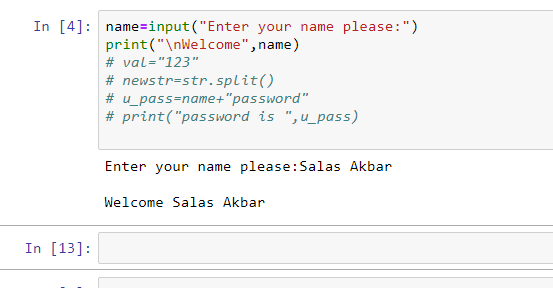
1. # is used for single line comment
2. To print output, print() function which accepted ‘String’ or “ String” in single or double quotes.
3. To run individual cell, press Shift+Enter or run button

* To comment on the code, use **“”” Your comments go here“””**



* To retrieve input from user, use input() function

**Note**. Python does not need to define data type for input variable, it assigns the type based on data provided by the user. For example, in the program below, Python assigns string type to variable **name**, since user entered a string data.

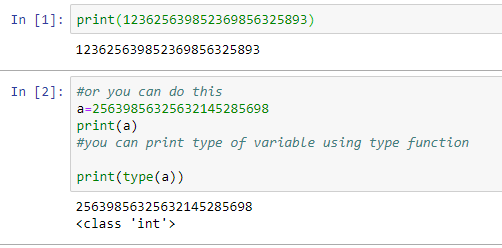


|  |
| --- |
| **2. Basic Data Types In Python** |

|  |
| --- |
| **2.1 Integer** |

In Python 3, there is effectively no limit to how long an integer value can be. Of course, it is constrained by the amount of memory your system has, as are all other computing resources.

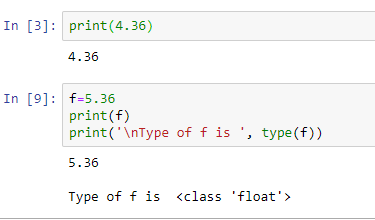
:



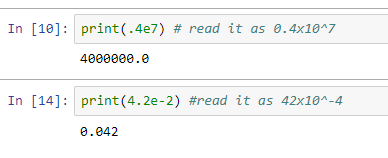
**NOTE: type()** is a built-in function that returns the class of the datatype, that the variable belongs to. It is used as : **type(variable\_name) or type(some\_value).**

|  |
| --- |
| **2.2 Float** |

The **float** type in Python designates a floating-point number. **float** values are specified with a decimal point.

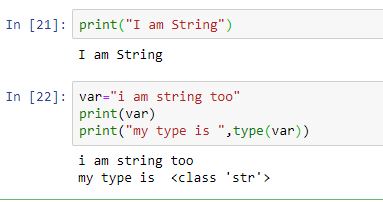


Optionally, the character **e**or**E** followed by a positive or negative integer may be appended to specify [scientific notation](https://en.wikipedia.org/wiki/Scientific_notation) also known as exponential notations.



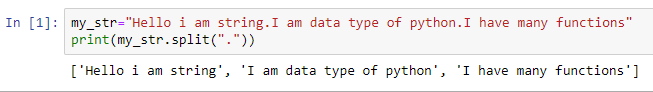
|  |
| --- |
| **2.4 String** |

Strings are sequences of character data. The [string type](https://realpython.com/python-strings/) in Python is called **str**. String literals may be delimited using either single or double quotes. All the characters between the opening delimiter and matching closing delimiter are part of the string.



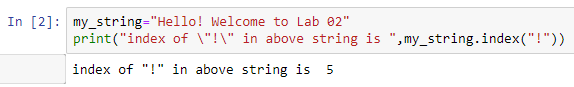
There are many built-in functions that can be applied on string. Some are discussed below.

**split():** this splits string on a certain pattern or character provided, take following example: if a string is supposed to be split on every **“.”,** following command should be used:



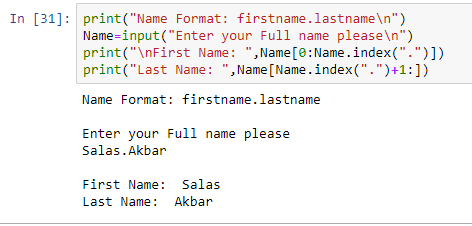
**Syntax: string\_variable.split(“pattern or character”)**

**index():** this returns index of specific character of string. For example:



Starting and ending index can also be provided to get substring from the main string.

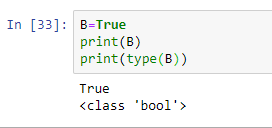
Syntax: **string\_variable[start\_index : end\_index].** For example:



***Note****: To get string from specific index to last index, do not write anything after :(colon); it will automatically take whole string, e.g. string\_var[start\_index : ]*

|  |
| --- |
| **2.5 Boolean** |

Boolean means True or False. If the condition is satisfied then **True** is generated else **False** is generated.

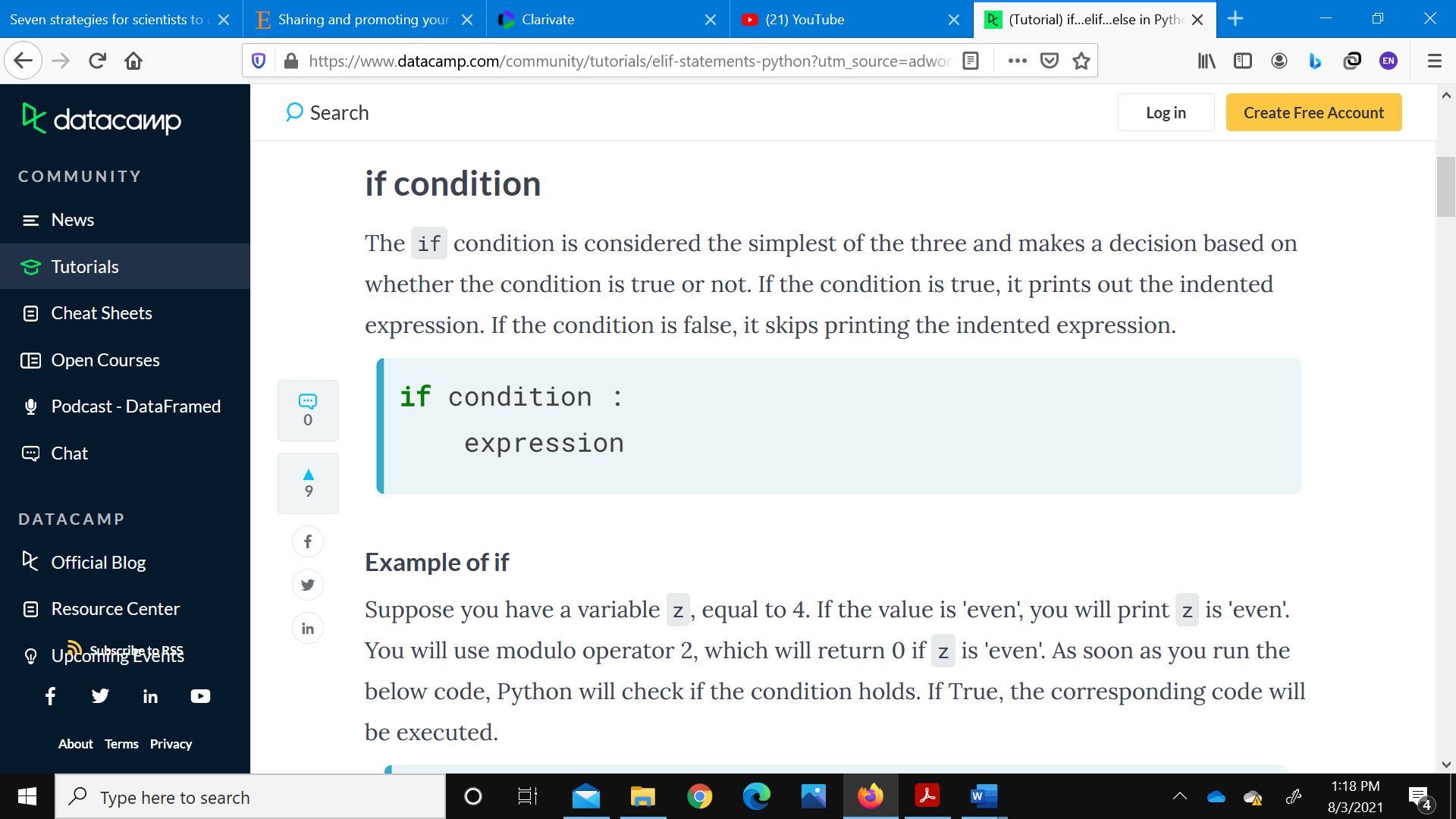


|  |
| --- |
| **1. IF Statement** |

**Conditional Statement in Python performs different computations or actions depending on whether a specific Boolean constraint evaluates to true or false. Conditional statements are handled by IF statements in Python.**

OR

**Python If statement is used for decision-making operations. It contains a body of code which runs only when the condition given in the “if” statement is true. If the condition is false, then the optional else statement runs, which contains some code for the else condition. When you want to justify one condition while the other condition is not true, then you use Python if else statement.**



|  |
| --- |
| **2. Python Conditions and If Statements** |

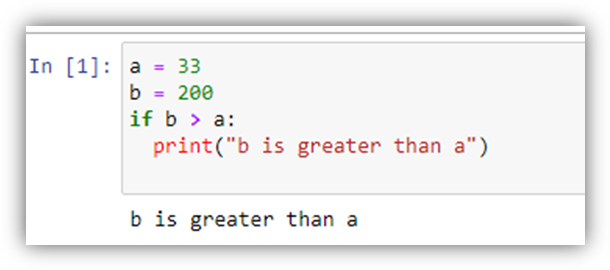
Python supports the usual logical conditions from mathematics:

* Equals: a == b
* Not Equals: a != b
* Less than: a < b
* Less than or equal to: a <= b
* Greater than: a > b
* Greater than or equal to: a >= b

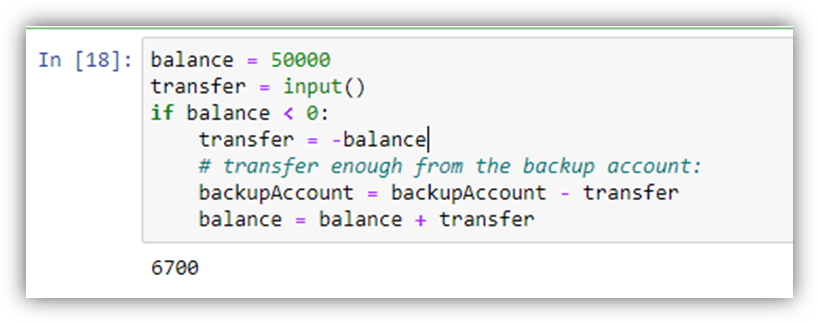
These conditions can be used in several ways, most commonly in "if statements" and loops.

An "if statement" is written by using the if keyword.

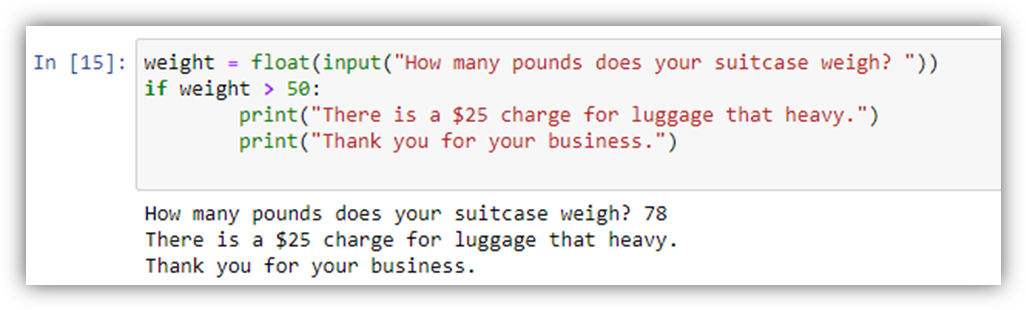
|  |
| --- |
| **Example (if statement)** |



|  |
| --- |
| **Example (if = True Statement)** |



|  |
| --- |
| **Example 2 (if = True Statement)** |

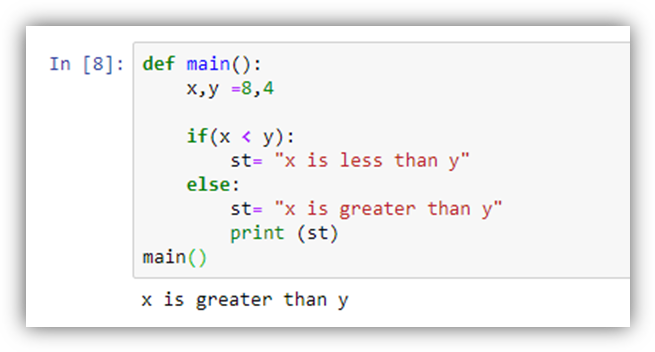


|  |
| --- |
| **Note :**  **Python relies on indentation (whitespace at the beginning of line) to define scope in the code. Other programming languages often use curly brackets for this purpose.** |

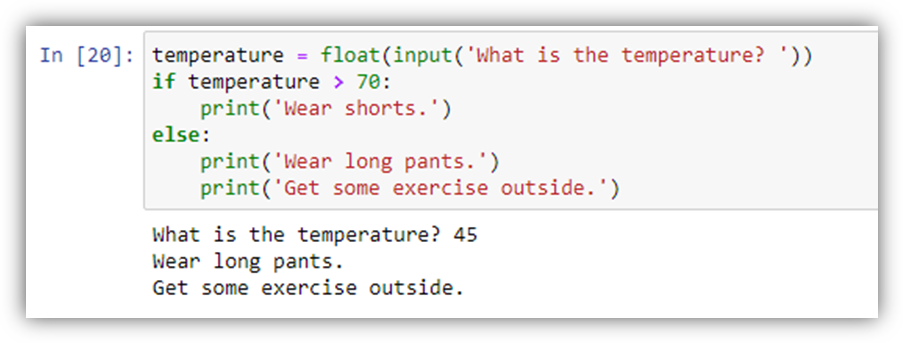
|  |
| --- |
| 1. **Python Basic Operations (if - else condition)** |

The "else condition" is usually used when you have to judge one statement on the basis of other. If one condition goes wrong, then there should be another condition that should justify the statement or logic.

|  |
| --- |
| **Example** |



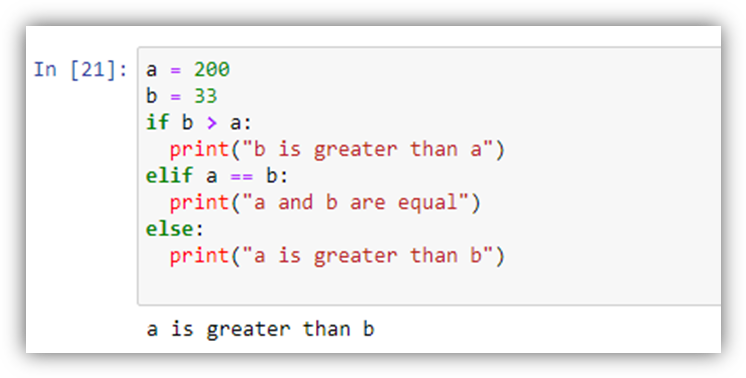
|  |
| --- |
| **Example 2** |



|  |
| --- |
| 1. **Python Basic Operations (elif condition)** |

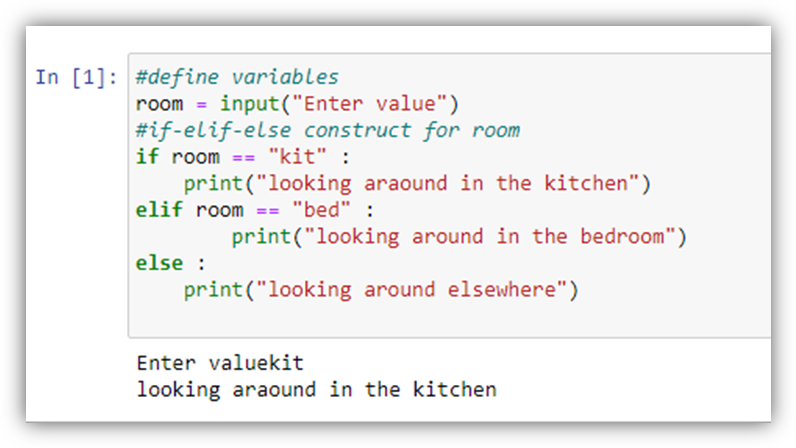
The most complex of these conditions is the if-elif-else condition. When you run into a situation where you have several conditions, you can place as many elif conditions as necessary between the if condition and the else condition. By using "**elif**" condition, you are telling the program to print out the third condition or possibility when the other condition goes wrong or incorrect.

|  |
| --- |
| **Example 1** |



In this example a is greater than b, so the first condition is not true, also the elif condition is not true, so we go to the else condition and print to screen that "a is greater than b".

|  |
| --- |
| **Example 2** |



|  |
| --- |
| **Explanation** |

In the Above example, we define two variables room and area and then construct if-elif-else and if-else conditions each for room and area, respectively.

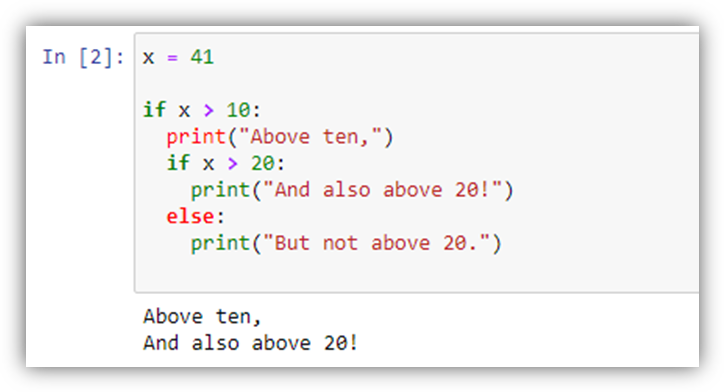
In the first condition, we check if we are looking in the kitchen, elif we are looking in the bedroom, else we are looking around elsewhere. Depending on the value of the room variable, the satisfied condition is executed.

|  |
| --- |
| **Python Basic Operations (nested if statement)** |

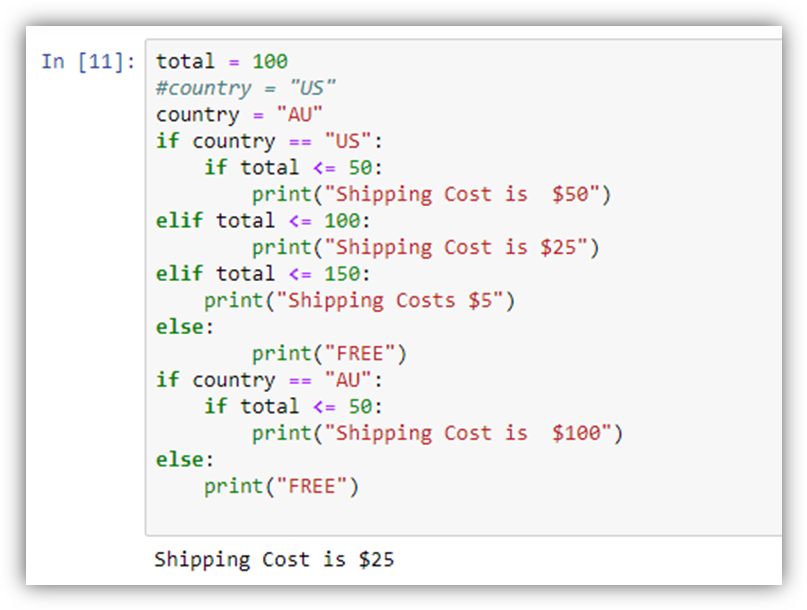
**Nested IF** functions, meaning one **IF** condition inside of another, allows you to test multiple criteria and increases the number of possible outcomes. So that  **nest** an **IF** condition by setting value to evaluate True or False in the inner conditional block of code.

Following examples demonstrates nested if Statement Python

|  |
| --- |
| **Example 1** |



|  |
| --- |
| **Example 2** |



|  |
| --- |
| **2.7. Types of Data under Big Data (Data Science) Domain** |

**DATA**

**STRUCTURED**

**SEMI-STRUCTURED**

**UN-STRUCTURED**

**Big Data are cetgorised into three diffreent types:**

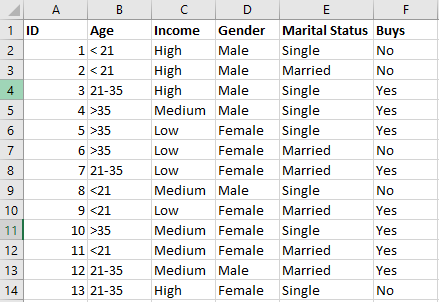
1. Structured

2. Semi-structured

3. Unstructured

|  |
| --- |
| **2.7.1 Structured Data** |

Structured data are the type of data which are sorted already in an order; like dictionaries, lists, arrays etc. lie under the category of structured data. Data presented in excel sheets (or in .CSV files) are also in the form of structured data. Another example can be considered as Table in the Relational Database.



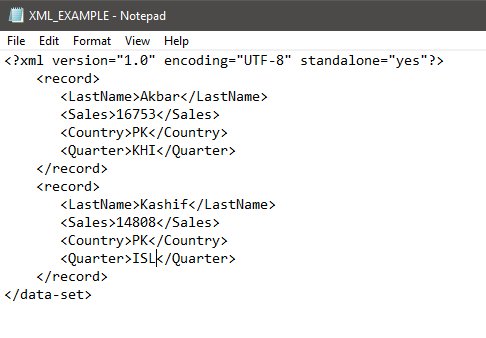
|  |
| --- |
| **2.7.2 Semi-Structured Data** |

They are intermediate state between structured and unstructured data; hence called called as *semi structured data*. Such data cannot be used with traditional database formats, however it contains some important orgainsational poperties.

The examples of Semi-Structured data are Spread-sheets, JSON or XML documents, NoSql Databases, etc.



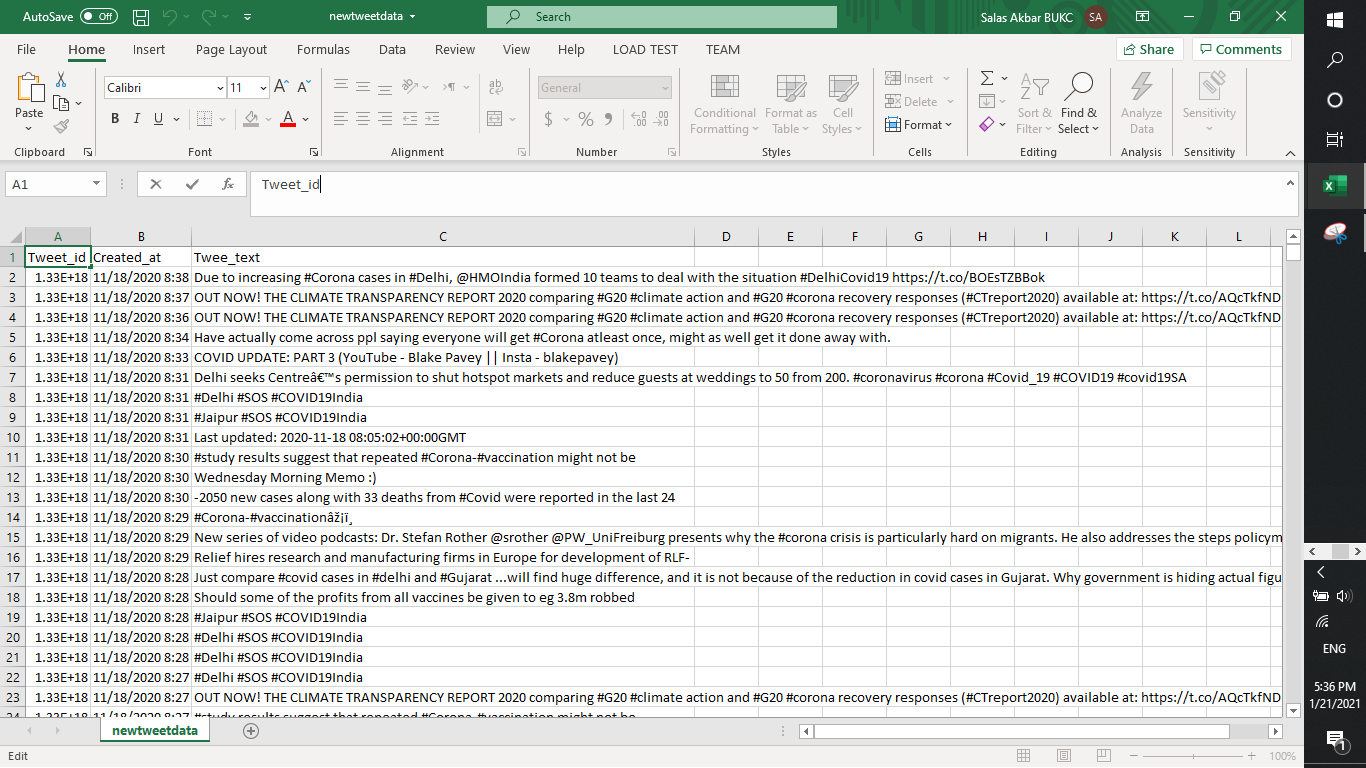
Above data is extracted from Scoial media platform **Twitter** in JSON format, and this is a samlpe of XML format.



|  |
| --- |
| **2.7.3 Unstructured Data** |

Unstructured data have no clear format in storage. Structured data can be stored in rows and column database, but unstructured cannot be presented in such format. A video is an example of unctructured data or a textual data is categorised as unstructured data. Many human geerated data can be categorised as unstrcutured data such as images, videos, social media data, articles, news papers, etc.

Example of unstructured data are text documents, PDFs, images, videos.



This is the screen shot of data extracted from Twitter. There is no proper format applied when data is extract from social media. It has to be pre-processed so that it becomes useful to apply data science tools.

|  |
| --- |
| **3. Exercise** |

1. Write a Python program that prompts user to enter the birth year and returns age in year
2. Write a python program that asks the user to input the cost of their meal as a float value. It then calculates the tip amount based on the meal cost. If the meal cost is less than 5000, the tip is calculated as 15% of the meal cost. If the meal cost is 5000 or more, the tip is calculated as 20% of the meal cost. The program then calculates the sales tax amount as 15% of the meal cost and calculates the total cost of the meal by adding the meal cost, tip amount, and sales tax amount together
3. Write Python program that asks the user for their age, education status, and years of work experience. The program first checks if the user is at least 18 years old, and then checks if they have a Bachelor's degree. If the user has a Bachelor's degree, the program checks if they have at least 2 years of work experience. If the user meets all of these conditions, the program prints a message congratulating them on their eligibility for the job. If the user does not meet any of these conditions, the program prints a message explaining why they are not eligible for the job.
4. Write a python program to calculate factorial of a given number
5. Write a program to check if the number is 0 , positive even/odd , or the number is negative. After that iterate over the numbers from 0 up to the entered number and check if each number is even or odd and how many numbers in total were even and odd.