# Introductory Python Notes for Scientists and Engineers.

## Rahul Jaichund Maharaj

**Abstract:**

In this set of notes, the basics of python are introduced to students in science and engineering fields.Coding is an essential tool for scientific research. This is as it is used to store, manipulate and visualise data. These notes cover storing data into variables, using loops for repetitive tasks, using functions to transform input data into output, storing data into lists and using ETL. I trust that these notes are of benefit to anyone who requires the basic grip of python as fast as possible.

# Table of Contents:

ETL

# Introduction to Python

Python is used in machine learning, artificial intelligence, in business apps and many other awesome technologies. Python is used as it is easy to learn. In this set of notes we will learn how to use python.

These notes are designed with great assistance from various lecturers lecturing online Python such as Telusko. Python is an easy to learn programming language that we use to tell computers as well as other digitised machinery to do in seconds what it would take the common man to do in days, weeks, months or even years. It completes repetitive mundane tasks in one shot.

I was fortunate to be able to participate in the CHPS/NiTheCS  Coding Summer School  in 2024. I was one out of 817 students. In the summer school, I learned basic python programming, data computation and analysis in the first week.In the second week, I was introduced to the applications of python. It was gruelling and intense but worth it.

In scientific research such as astronomy/physics/chemistry/biology, python is used as a tool to simulate events that are impossible to observe beyond earth. One of the examples would be how clouds collide  with each other and can form stars.

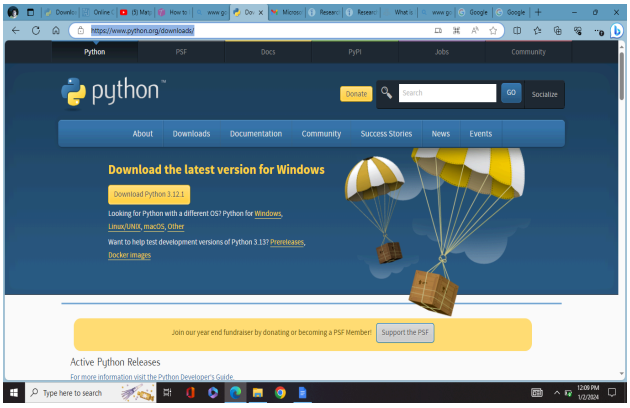
In scientific research, python is used to manipulate a large set of data i.e the radio emission of a star during the course of a year into useful data that can be visualised and analysed. The aim of the notes is to introduce the reader who is not familiar with python.

I trust that these notes will be of great assistance to those who wish to study python in order to advance in scientific research as well as other fields of research. The reader is urged to be as resourceful as possible i.e visit the library and read as many books on python, watch online tutorials, and practice python problems on his/her own. Only reading these notes is not sufficient for understanding Python.

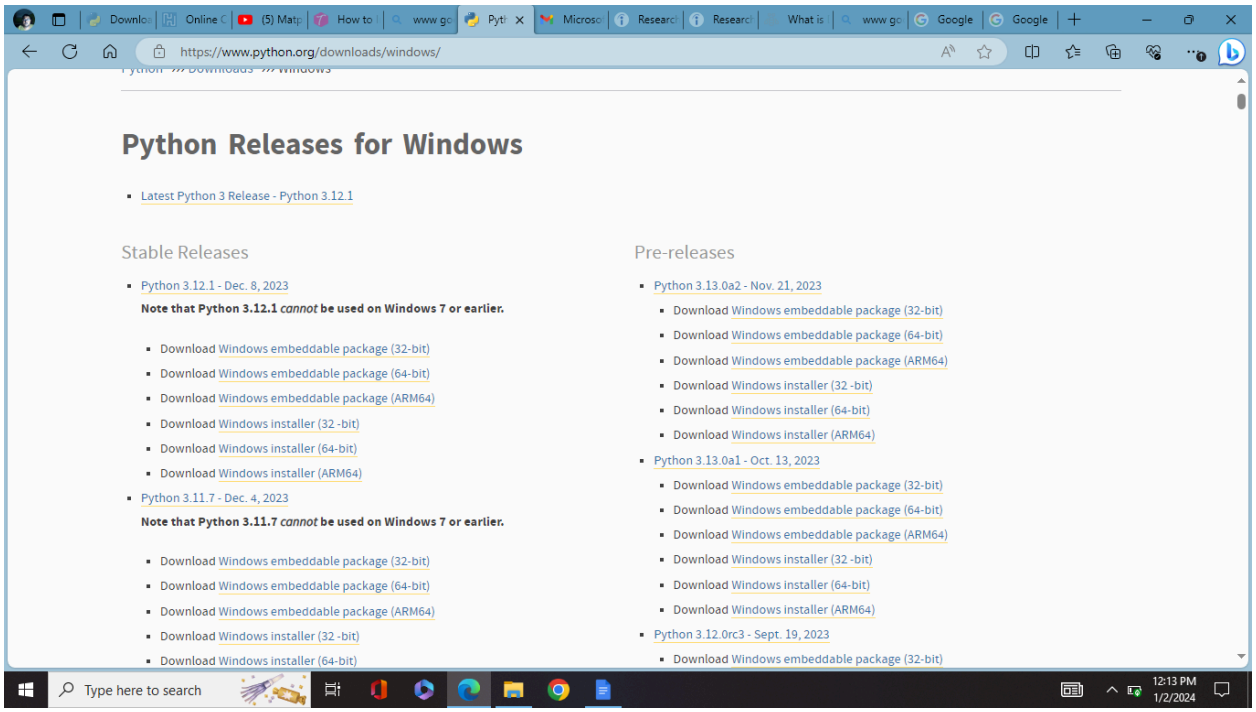
## Installing Python on Windows

If the reader uses a Mac or a Linux operating system, he/she is urged to seek information on the internet or an expert operating on that system to understand the procedure as to how to install python on the system you are operating on.

Step 1: Visit Python.org.com Download Python | Python.org



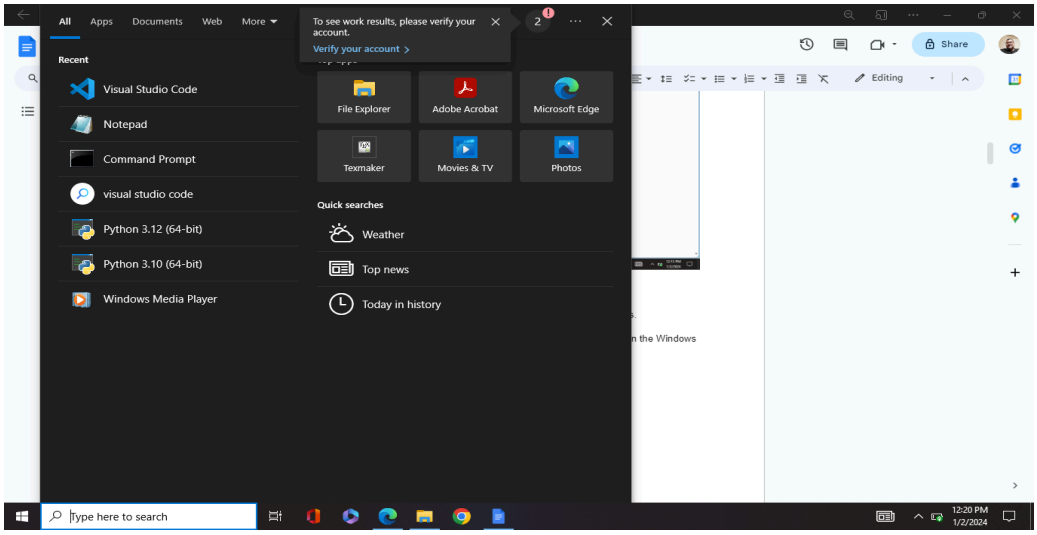

Step 2 Click on the Windows link on the lower right below the ,"*Download Python 3.12.1"* box.



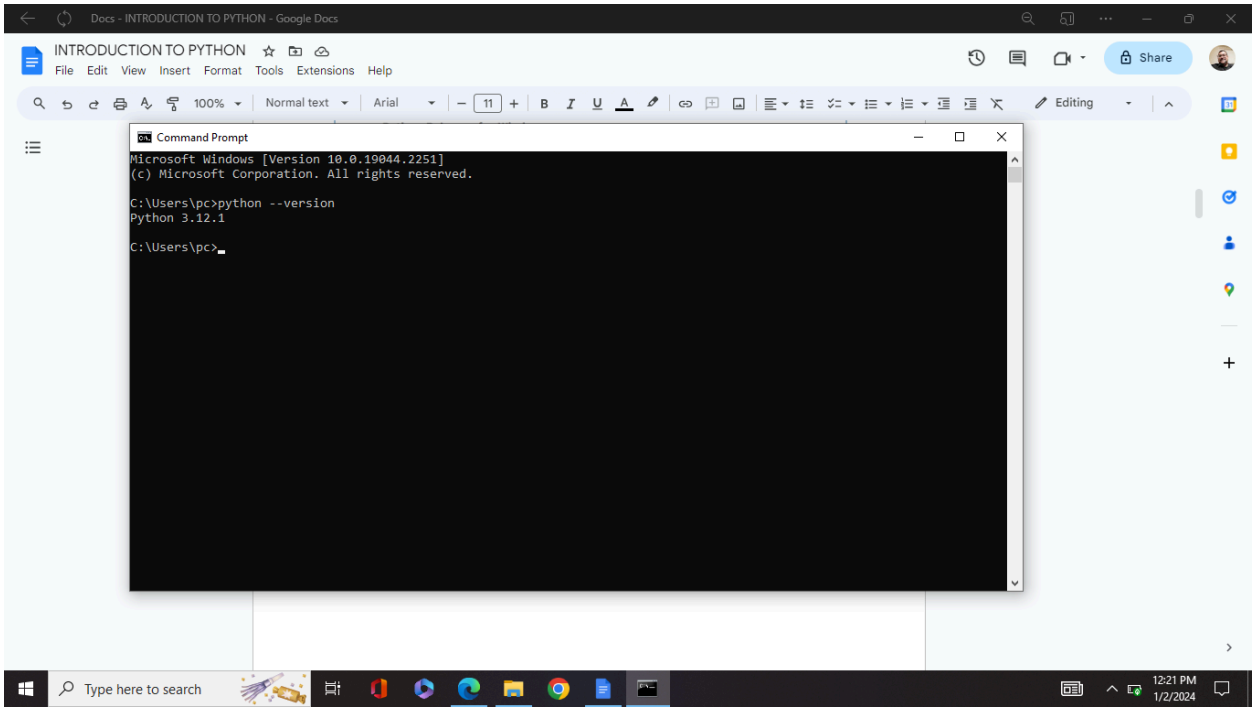

Step 3 Click on the link that is most suitable to you.

Step 4 Go to your Downloads folder, you should see your package.Extract all files.

Step 5 To check if python has been installed successfully on your system. Click on the Windows icon on the lower left of your screen. Search Command Prompt.




Step 6 Type the command *python - - version*.You should see that :




## Integrated Development Environment.

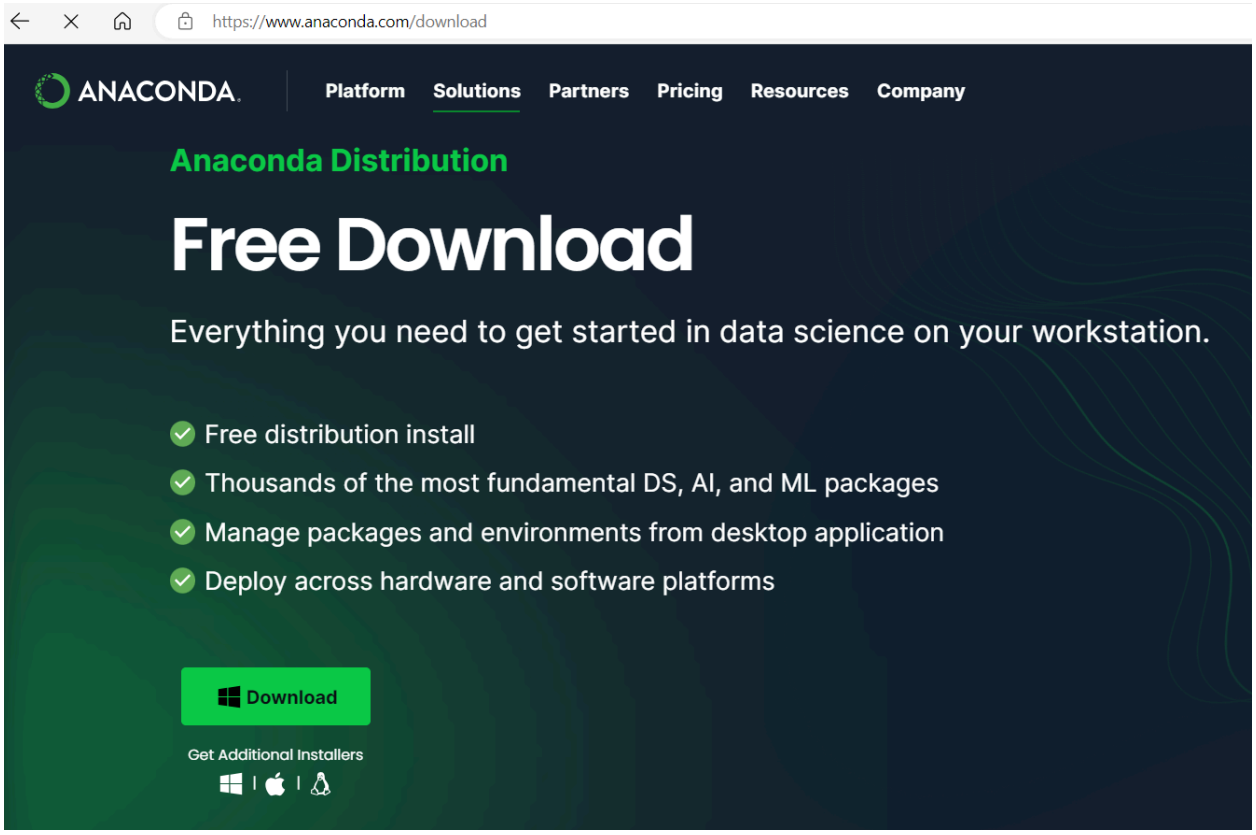
There are many integrated software development environments to run Python such as Pycharm, Visual Studio Code e.c.t. I use Visual Studio Code which I installed with aid of a video tutorial

from YouTube, The reader can consult the Internet to find out how to install a IDE to run python codes.

Here, is the procedure to install Spyder which was the IDE used in the 2024 CHPC/NiTheCS Summer School in Coding.

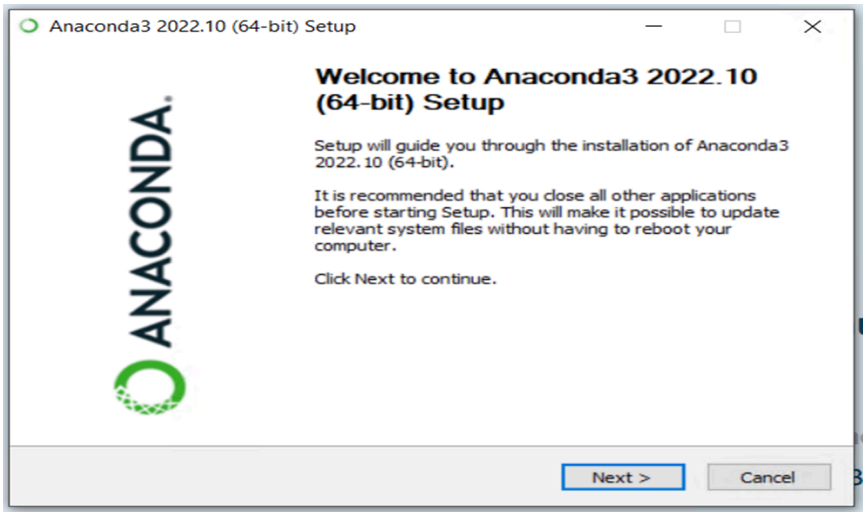
## Installing Anaconda

**Step 1:** Visit the link [Free Download | Anaconda]

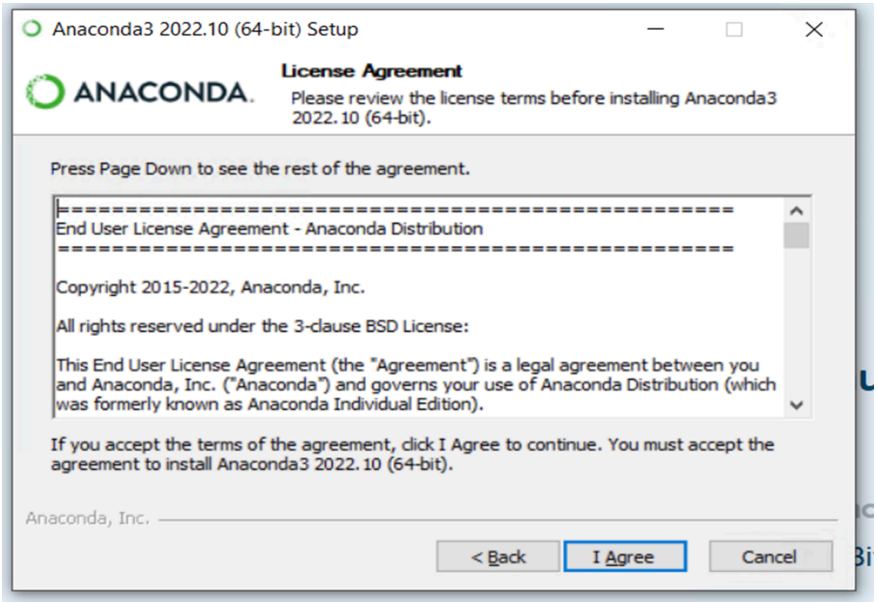



**Step 2:** Click on the Download option above.

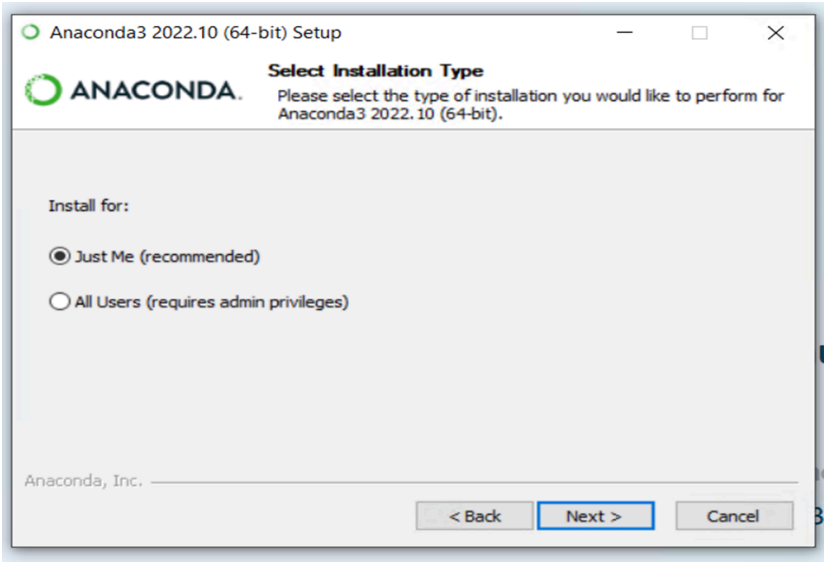
**Step 3:** After downloading and installation, select **Next** as shown below:



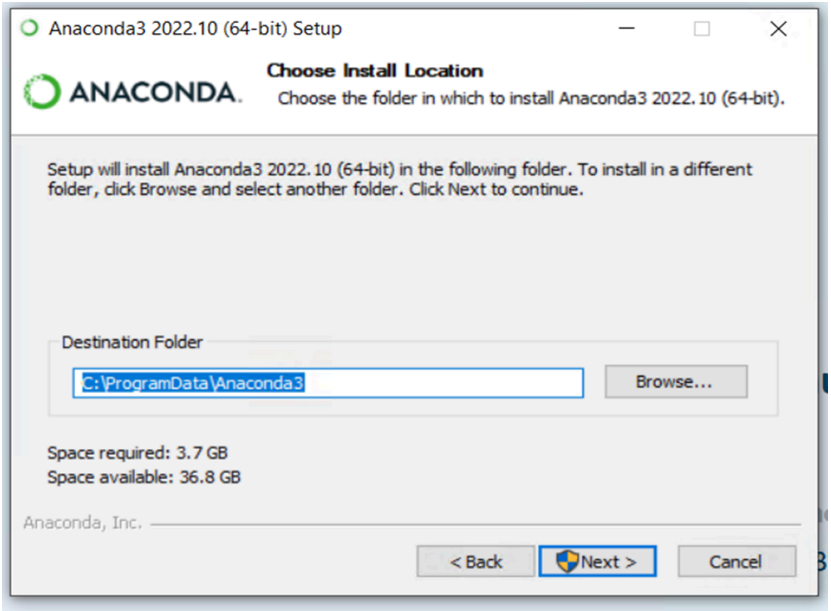

**Step 4:** Select I Agree.



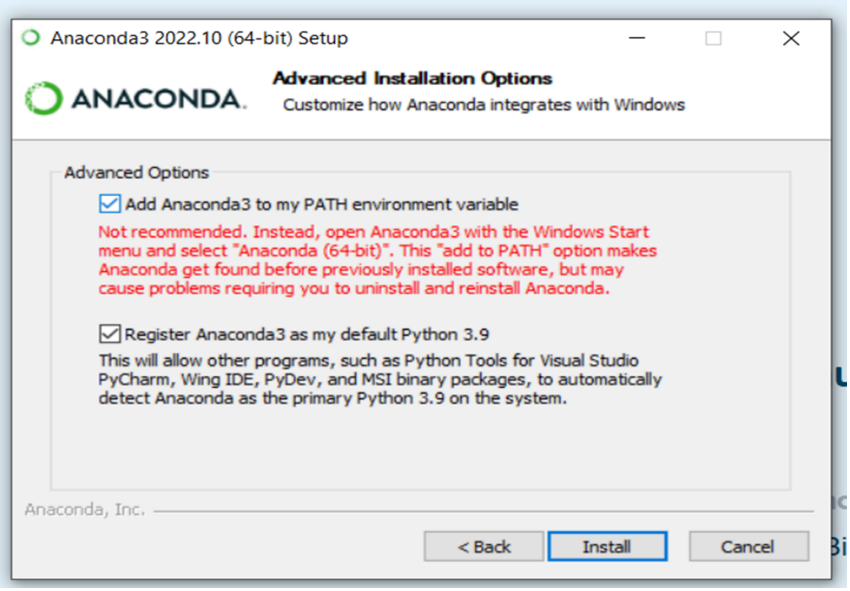

**Step 5:** Select **Just Me**.



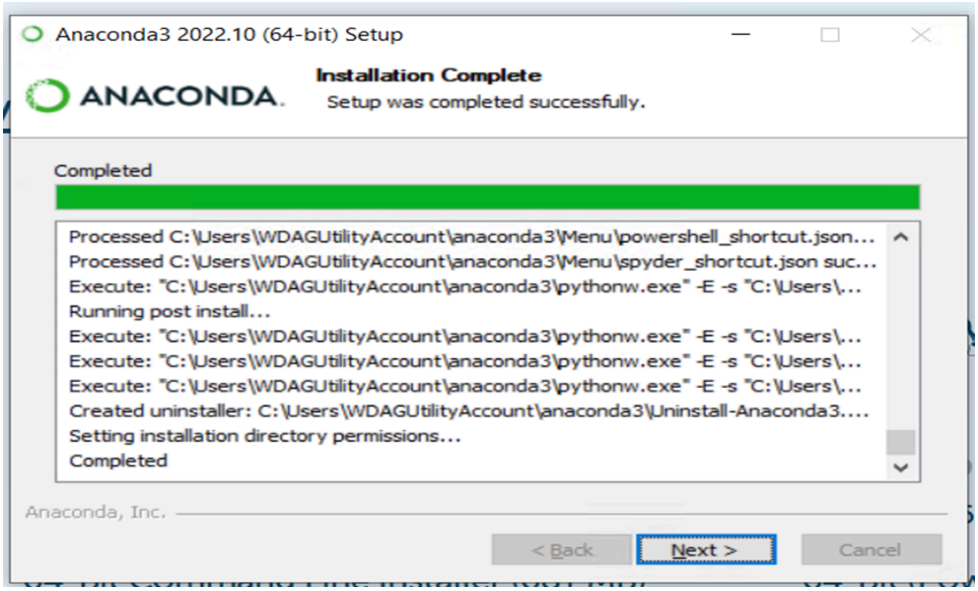

**Step 6:** Select **Next** unless the reader wants to use another directory.



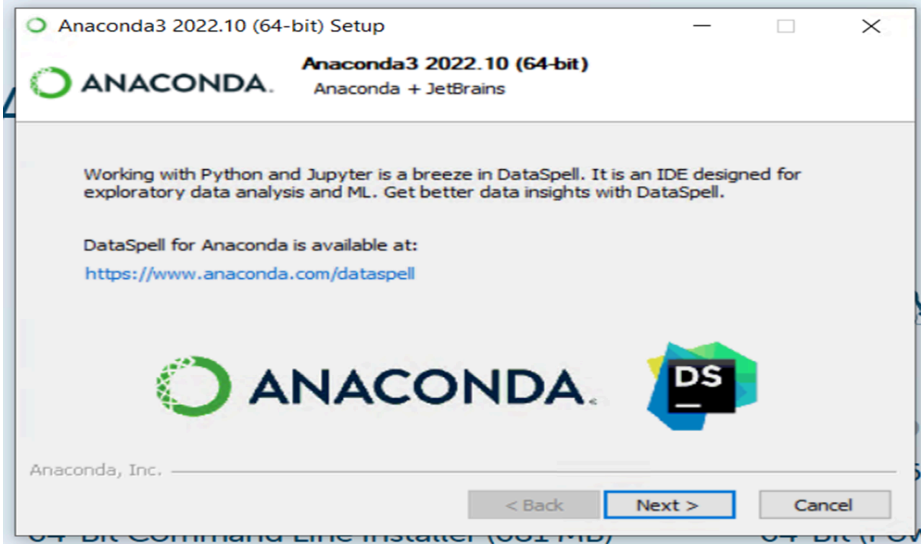

**Step 7:** Select **Add Anaconda to my PATH environment variable** and then select **Install**.



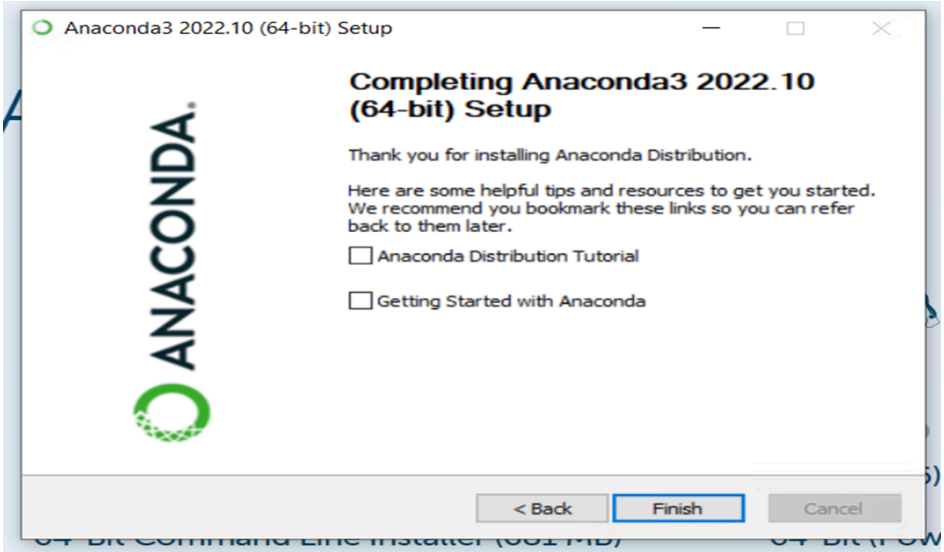

**Step 8:** Once installation is complete, select **Next**.



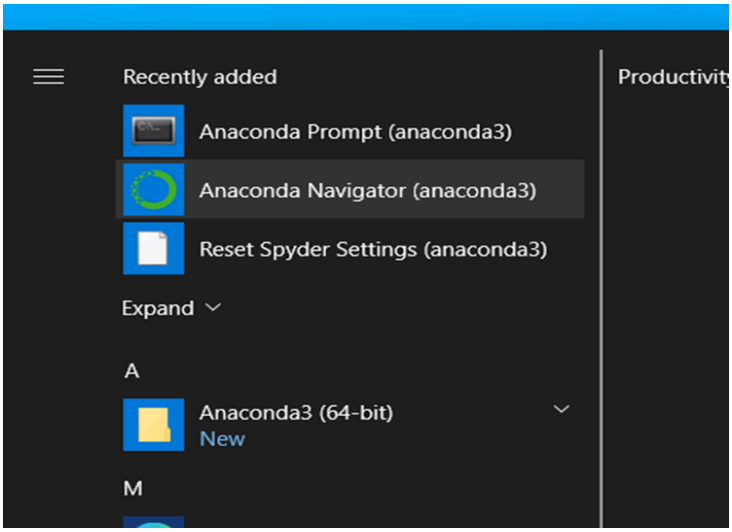

**Step 9:** Select **Next**.



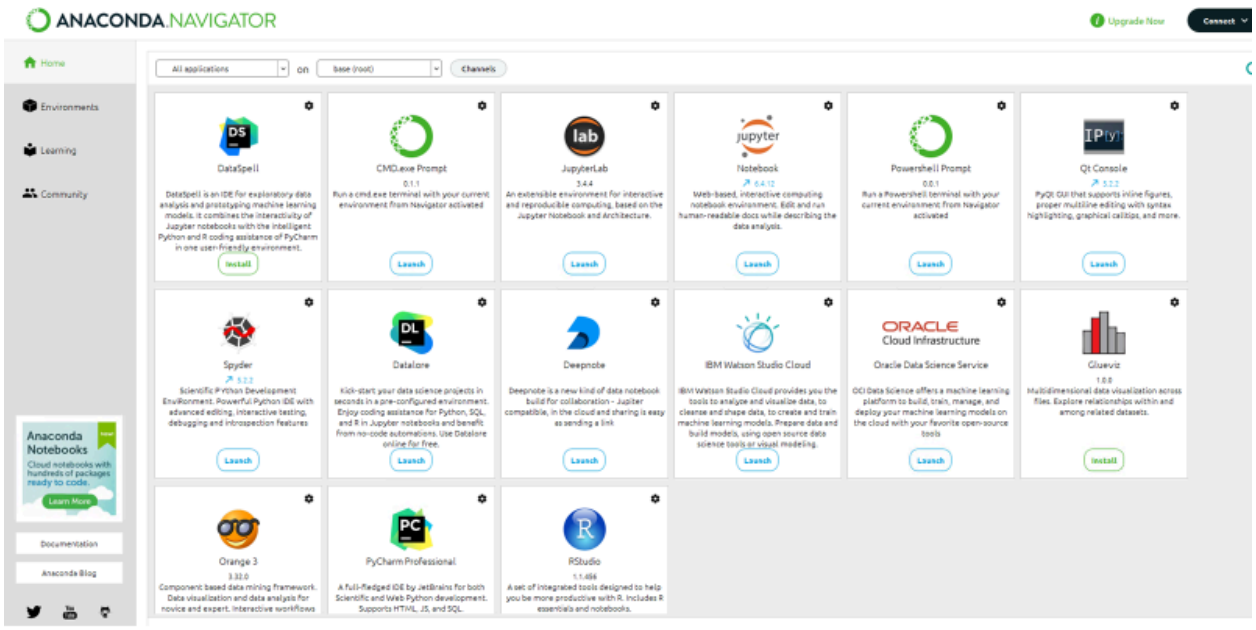

**Step 10:** Uncheck the boxes and then select **Finish**.



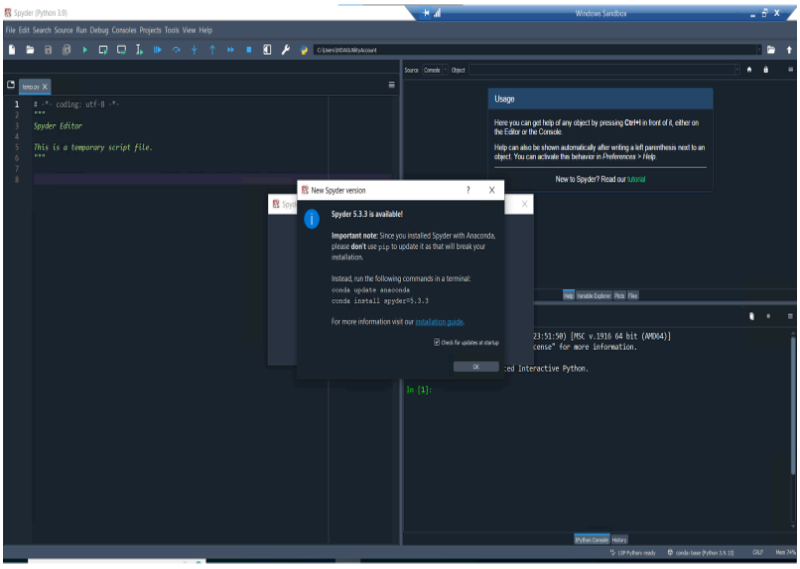

**Step 11:** Click on the Windows icon on the lower left of the screen. Search for the Anaconda Navigator App and then click on it.




**Step 12:** After loading, the reader should see the picture below:




**Step 13:** Select Spyder and wait for it to install.



All programs have been installed including Python. These programs cater towards data science.The Variable Xplorer feature which enables one to observe data as an Excel file. Spyder has built in libraries such as Pandas, Numpy, Matplotlib, Sklearn and many others.

There are many ways to install python. Consult as many YouTube Tutorial Videos as you can.

If Python does not have these libraries use the `pip install` command on your Command Prompt if Anaconda was not installed.

`pip install numpy` which is used to install Numpy

Otherwise use the `conda install` command in the Anaconda prompt if Anaconda is installed

`conda install numpy` installs Numpy.

# Basic Python Programming

## Storing Data in the Form of Variables

A variable is a container that stores information. Information is stored in the form of whole numbers, integers, string and boolean (values such as ,"TRUE" or "FALSE"and even arrays which you should see immensely in the Numpy handout.

Examples:

Write a code that asks the user to type his name and once the user has type his/her name , the output is ," *Welcome to python" + stored information*

## Input

The input command is used to read information stored by the user.

```
name = Input(" Enter your name" )
Enter your name.    Rahul

print("Welcome to Python" + name)
```

and you should see on your screen :

```
Enter your name. Rahul
Welcome to PythonRahul
```

Write a program that asks the user to enter his details such as his name , date of birth , marital status and driver;s licence

```
name = input("Enter your name.")
date_of_birth = input("Enter the date of your birth.")
marital_status = input("Please enter your marital status.")
drivers_liscence = input("Please enter your drivers liscence
code")

print("Thank you!")
print("Confirm these details:")
print(name + date_of_birth + marital_status)
print(drivers_liscence)
```

Output:

```
Enter your name. Rahul Maharaj
```

```
Enter the date of your birth. 27 June 1997
Please enter your marital status. Single
Please enter your drivers liscence code 8
Thank you!
Confirm these details:
Rahul Maharaj 27 June 1997Single
8
```

A variable : A variable can have a short name like x or y or more descriptive ones like age, my*age, country*list.

- A variable must start with a letter or underscore
- A variable cannot start with a number
- A variable can only contain alpha-numeric characters and underscores
- A variable name is case-sensitive: age, Age, and AGE are all different variable names

## Casting Variables from One Data Type to Another Data Type

A variable's data type can be changed. In other words, a variable can be changed from a string to an integer or from an integer to a float number.

Use the print(type()) to output the type of data that is stored in your variable for example:

```
x = 54
print(type(x))
```

Which is output as:

<class 'int'>

54 is classified as an integer as it is a whole number.

*Exercise: Please write a program that asks a user to input a number and outputs a number that is ten times the input number.*

```
number = input("Please enter a number which will be mulitplied
by 10.")
answer = number*10
print("The answer is" + answer.)
```

The output is `SyntaxError: bad input on line 12 in main.py`

The reason being is that the number input is a string data type. The calculation involves the string input. Calculations are done using integer and float data types and not string data types.

To solve this issue we need to convert the string input into an integer input. We use the command int(input("Please enter a number.")). The int() converts the data type into an integer.

The commands below illustrates the conversions of data types:

int() : converts data type to integer.
float() : converts data type to float.
str() : converts data type to string.

The correct code is shown as follows:

```
number = int(input("Please enter a number which will be
multiplied by 10."))
answer = number*10
print("The answer is" + str(answer))
```

which gives the output we want:

```
Please enter a number which will be multiplied by 10. 3
The answer is30
```

## Arithmetic in Python

Python uses this notation if one wishes to perform simple arithmetic :

```
+  for Addition
   -  for subtraction
      * for multiplication
      ** for exponentiation
      10e6 means 10^6
```

Example: Compute the operations :

5 + 5
3-2-5-6

```
v = 5 + 5
print("5 + 5 =",v)
```

Which affords:

```
5 + 5 = 10
```

```
v = 3-2-5-6
print("v =",v)
```

Which affords:

```
v = -10
```

Practice Exercise:

Compute the following operation in Python:

$$a = 3.4,\ b = 4.8$$
$$\sqrt{2ab} \times 3a^{8\,\frac{4.5b^2}{9.8c^8}}$$

*Write a Python code that calculates the area of a rectangle from the input values of the length and the width.*

```
length = float(input("Please input the length of the
rectangle."))
width = float(input("Please input the width of the rectangle."))

area = length*width

print("The area of the rectangle is" + str(area) + "square
units")
```

Which outputs the result:

.

```
Please input the length of the rectangle. 234
Please input the width of the rectangle. 123
The area of the rectangle is28782.0square units
```

*Exercise: Write a program that calculates a person's BMI using the formula:*

$$BMI = \frac{Weight}{Height^2}$$

```
print("Welcome to the BMI program")
weight = float(input("Please enter your weight in kilograms."))
height = float(input("Please enter your height in meters."))

BMI = weight/(height*height)

print("Your BMI is" + str(BMI) + "kilograms per square meter.")
```

*Welcome to the BMI program*
*Please enter your weight in kilograms. 140*
*Please enter your height in meters. 2.08*
*Your BMI is32.3594674556kilograms per square meter.*

**PRACTICE EXERCISES:**

1.1 Write a program that inputs a number and multiplies it by 1000 and outputs the answer.

1.2 Write a program that calculates the gravitational attraction :

$$F = \frac{GMm}{r^2}$$

Between earth and the sun.

1.3 Take the input from the user for a given distance in kilometres and converts it to miles.

## If Statements

If statements are used in python to make decisions based on a condition. The input makes the computer decide which decision is made based on a condition i.e if a number is to be decided whether it is even or odd , the computer decides based on the condition whether or not the number is divisible by 2. If the number is even , it is divisible by 2 if it is odd it is not divisible by two.

Example: Write a program that asks the user to input a number and check if it is even or odd.

```
# Check whether a number is even or odd:

x = int(input("Please Enter a Number :")) #The Input Comm-9and asks the
                        User to Enter a Value
                                # The input number is stored in a
                          variable called x
                                  # The input command outputs the read
                            in value as a
                                          # string data type.
                                # The output string is cast as an
                             integer

    # if a number is divided by 2 and has a zero remainder - Even
  # If a number is divided by 2 and has a non-zero remainder - Odd

 if  x % 2 == 0 :      # the % sign stands for mod which checks the
                       remainder of the number
          print("The number entered is an even number.")
                              else :
           print("The number entered is an odd number")
```

Write a program that checks to see whether or not a number input $x$ follows the conditions

$$-10 \leq x \leq 20$$

```
# Check if a number is greater than and equal to -10 but less than or
                            equal to 20

number = int(input("Please Enter a Number:")) # The Program asks the user
                          to enter a number
                                    # The program then stores the
              number into a variable called ,"number"
                                  # The number read in is stored as
                            a string
                                    #Python casts the read in number
                          as an integer

                        if -10 <= number <= 20 :
```

```
        print("The number is between -10 and 20")
                elif number < -10 :
          print("This number is less than -10")
                elif number > 20 :
         print("This number is greater than 20")
```

Please Enter a Number:-15
This number is less than -10

Please Enter a Number:5
The number is between -10 and 20

Please Enter a Number:35
This number is greater than 20

**PRACTICE EXAMPLES:**

1. Write a program that asks the user to input a number and checks whether it is positive , negative or zero.
2. Write a program that checks whether a number is divisible by 5 or not.

Write a program that checks whether a number is divisible by 5 and then checks to see whether it is divisible by 3. The code should output this ,"*This number is divisible by 3 and 5"* otherwise it should print ,"*This number is not divisible by 3 and 5"*.

```
# Check if a number is divisible by both 3 and 5

number = int(input("Please Enter a Number:")) # The Program asks the user
to enter a number
                                              # The program then stores the
number into a variable called ,"number"
                                              # The number read in is stored as
a string
                                              #Python casts the read in number
as an integer

# number% 3 = 0 is divisible by 3 (zero remainder)
```

```
# number% 5 = 0 is divisible by 5 (zero remainder)

if number %3 == 0 and number %5 == 0 :
    print("This number is divisible by 3 and 5")
else:
    print("This number is not divisible by 3 and 5 ")
```

Please Enter a Number:21
This number is not divisible by 3 and 5

Please Enter a Number:15
This number is divisible by 3 and 5

## While and For Loops

Loops are cycles in python that a program follows.Similar to what you do in your day. Python uses loops to output a list of information. It is used to automate repetive tasks i.e printing a text of string a million times as shown in the forthcoming examples.

While Loop: is a loop that runs based on a condition.

Example: Write a program that outputs 1-10.

```
#Write a program that outputs 1-10 onscreen

i = 1  #Loop Counter
while i <=10 : #Condition: Loop Counter Runs from i = 1-10
    print(i)     # i is printed out to screen
    i = i + 1    # i increases by 1 from 1 all the way to 10
```

which gives the output on screen:

1
2
3
4
5
6
7
8
9

10

Example: Write a program that calculates the sum of 20 numbers from 1-20.

```
sum = 0 # Initialising sum to be zero
        i = 1   #Loop Counter

while i<=20: #Condition i runs from 0 to 20
                sum = sum + i
                  i= i + 1
            print(sum)
```

which yields the output:

210

Example: Write a program that calculates the sum of the square of 1000 numbers.

```
    sum = 0 # Initialising sum to be zero
i = 0   #Loop Counter is initialised to zero

while i<=1000: #Condition i runs from 0 to 20
                sum = sum + i**2
                  i= i + 1
print("The sum of the squares of 1-1000 is", sum)
```

which yields the output:

The sum of the squares of 1-1000 is 333833500

For loops: Run based on a predetermined sequence.

Example: Write a program that prints *,"Hello World"*, 10 times.

```
# The aim of python program is to print ,"Hello World" ten times

for i in range(10):  #for each element i in the list of 10 numbers.
                        print("Hello World")
```

Output:

Hello World
Hello World
Hello World
Hello World
Hello World
Hello World

Hello World
Hello World
Hello World
Hello World

Example: Write a program that asks the user to input the initial value of a list, the final value of the list and the increment of the list. The output to screen is the sum of the entire list.

```
a = int(input("Please Enter the Initial Value of the List:")) # a is the
                        intial value of the list.
b = int(input("Please Enter the Final Value of the List:")) # b is the
                        final value of the list.
c = int(input("Please Enter the Increment of the Loop:"))  #c is the
                        increment of the list.

                    sum = 0  # sum is initialised to zero.


                for j in range(a,b,c) :  # j is counter
                                sum = sum + j
                                print(j,sum)
```

Please Enter the Initial Value of the List:20
Please Enter the Final Value of the List:60
Please Enter the Increment of the Loop:4
24 44
28 72
32 104
36 140
40 180
44 224
48 272
52 324
56 380

```
#Compute the sum of all multiples of 3 and 5 that are less than 100 ?

y = list(range(100))
sum = 0
for i in y:
    if i % 3 ==0 or i % 5 == 0:
     sum = sum + i
print(sum)
```

2318

Example : Print out a list :

(a) of numbers 1-10
(b) Of numbers 1-20 with a step size of 5

using a for loop in (a) and (b).

(a) The procedure is simple: Define an empty list using []
Use a for loop
.append() to add elements into the list.

as shown in the programs below:

```
#Defining an Empty List Using []
x = []

#Using a for loop.
for i in range(1,11) :
     x.append(i)    #Adds every number of the loop to the list

print(x)
```

Output:
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

b.)
```
#Defining an Empty List Using []
x = []

#Using a for loop.
for i in range(1,20,5) :    #Range(Intial Value, Final Value, Step Size)
     x.append(i)    #Adds every number of the loop to the list

print(x)
```

Output:
[1, 6, 11, 16]

c.) Suppose we want to perform arithmetic opperations to each and every element as shown in the code below:

```
y = []

for i in range (21):
    y.append(2*i + 4)

print(y)
```

which outputs :
[4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44]

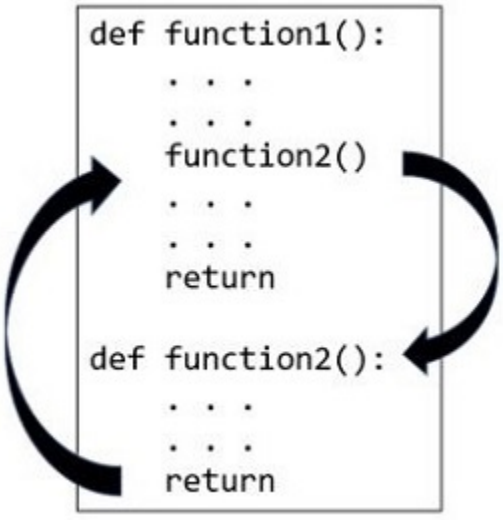
## Functions

A function is a procedure which outputs information from a given input. A function is structured as follows :

Defining the Function: the function is defined according to the parameters.

Body of the Function: All algorithms and procedures are written as to what the function is supposed to do.

Return : The function returns the value output from the input.




Source: Tutorialspoint
https://www.tutorialspoint.com/python/python_functions.htm

accessed on 2024/10/02 6:28 PM

Example: Using a function, print out ,"*Hello World*", ten times.

```
def func(i) : #Defining a function with parameter i

#Body of Function
  for j in range(i):
       print("Hello World")

func(10) #returns the output of the function i =10 is the input and
func(10) is the output
```

which outputs:

Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World
Hello World

Example: Use a function that inputs a value , doubles it and outputs the value.

```
#Defining a Function

def double(r) : #Defining a function with parameter/dummy variable i

#Body of Function
    dbl = 2*r    #doubles input value r
    return dbl   # returns output value dbl
```

```
print(double(40))
```

which outputs:

80

*Example:* Write a program that inputs two numbers, has a function that computes the sum and difference of those two numbers and outputs those results.

```
a = int(input("Please Enter an Arbitrary Number:"))
b = int(input("Please Enter an Another Arbitrary Number:"))

def y(c,d) :
    add = c + d
    sub = c -d
    return add , sub

print("The addition and subtraction of those Numbers are",y(a,b))
```

Output:

Please Enter an Arbitrary Number:10
Please Enter an Another Arbitrary Number:5
The addition and subtraction of those Numbers are (15, 5)

These are what functions do. I trust that the examples above have given the reader an insight of what functions are and what they do.

Plotting the Lorentz Factor:

The Lorentz Factor is important to relativistic physics. I'm not going into details with it. If the reader wishes to know more about it, he/she is welcome to consult a relativistic physics textbook.

The Lorentz Factor is defined as :

$$\gamma = \frac{1}{\sqrt{1-\left(\frac{v}{c}\right)^2}}$$

where γ is the Lorentz Factor, v is the speed of the object approaching the speed of light c which we all know as $c = 3.00 \times 10^8$ m/s.
In this program , I import two modules , Numpy and Matplotlib. Numpy is used to create and manipulate arrays (consult the second handout for more detail). Matplotlib is used to plot visuals which are informative.

In the program , the v values are stored as an array which consists of linearly spaced values of v that obey the restriction $0 \leq v \leq c$.

The Lorentz Factor is defined as a function. It is tedious to insert $v/c$ directly. So I let $\beta = v/c$ in the body of the function.

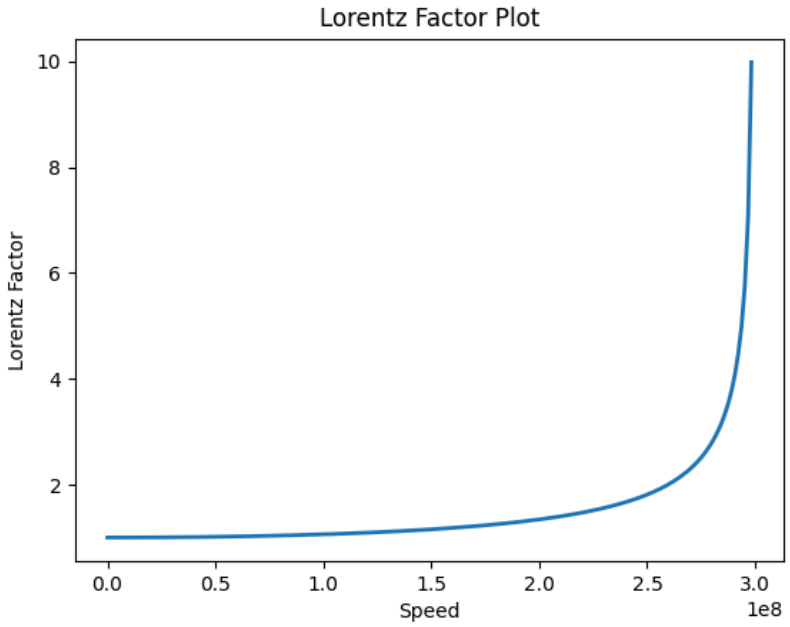
This enables us to obtain the γ values corresponding to the v values.

So long story short: define $v$ as a linearly spaced array where $0 \leq v \leq c$.
define γ as a function
plot γ vs $v$ using the Matplotlib library.

```
# Plotting the Lorentz Factor

# Importing all Relevant Modules.
import numpy as np
import matplotlib.pyplot as plt
from scipy import constants

#Defining c as a constant.
c = constants.speed_of_light # meters per second

#Defining v as a linearly spaced array from 0 -c
v = np.linspace(0,c,200)

#Defining Gamma as a Function
def Lorentz_Factor(v) :
```

```
 beta = v/c
 gamma = 1/np.sqrt(1 - beta**2)
 return gamma

#Plotting the Lorentz Factor
plt.plot(v,Lorentz_Factor(v), linewidth = 2.0)

#Labeling the x axis
plt.xlabel("Speed")

#Labeling the y axis
plt.ylabel("Lorentz Factor")

#Title
plt.title("Lorentz Factor Plot")

#Show plot
plt.show()
```

which gives us the plot below:



## Lists

Lists are one of the ways to store data in python. [] are used to store data. Commas are used to separate the items on the list. The items on the list are called elements.

We can do many things with lists:

- print all the items in the list using [:]
- we can add items to the end of the list using append()
- add items at specific positions using insert()
- remove an item from the list using the remove() function
- check how many variables are in the list using the len()
- print a range of values using [start:end] – just like in excel!

```
#Lists

nums = [25,36,95,14]  # list of integers
smackdown = ["Batista", "Undertaker", "John Cena", "Eddie
Guerrero" , "Chris Benoit"]   # list of strings

a_list_a_tingz = [45.0, "Drake" , True , 34.567] # a list with
various data types

print(len(smackdown))  # len((smackdown)) is the length of the
list smackdown
                       # print(len(smackdown)) outputs the
length of the list smackdown out to screen

nums.append(45)  #append adds an element to a list.
                 # 45 is added as an extra element to the list.

print(nums)

nums.insert(2,77) # inserts the element 77 in place 3 of list
nums.

print(nums)
```

```
nums.remove(77) # removes the number 77 of the list nums.
print(nums)

del smackdown[3:4]  # deletes elements 3 and 4 of smackdown
print(smackdown)

min(nums)   # finds the minimum value of nums
print(min(nums))

max(nums) # finds the maximum value of nums
print(max(nums))

range = max(nums) - min(nums)
print("range =", range)

sum(nums)  # finds the sum of the list
print("sum =", sum(nums))

nums.sort() #sorts out the list nums. It arranges list from
lowest to highest value
print(nums)
```

Output:

```
5
[25, 36, 95, 14, 45]
[25, 36, 77, 95, 14, 45]
[25, 36, 95, 14, 45]
['Batista', 'Undertaker', 'John Cena', 'Chris Benoit']
14
95
range = 81
sum = 215
[14, 25, 36, 45, 95]
```

Example: Consider the list below:
$x = 3409, 2356. 4234, 5642, 2345, 2123, 4566, 5008, 7643, 8903, 9873, 4532, 6422, 5632, 1234, 4321, 4567$
„ 3456, 5213, 5324, 5678, 4325, 5432, 5632

Calculate the sum, mean, variance and standard deviation of the above list:

```
#Aim of Program is to Calculate Mean, Variance and Standard
Deviation of:
# x = 3409, 2356. 4234, 5642,2345,2123,4566, 5008,
7643,8903,9873,4532,6422,5632,1234,4321,4567,,3456,5213,5324,567
8,4325,5432,5632

x =[3409, 2356, 4234, 5642,2345,2123,4566, 5008, 7643,8903,9873,

4532,6422,5632,1234,4321,4567,3456,5213,5324,5678,4325,5432,5632
]

print(x) # Done to print

sum(x)  #Calculates the sum of list x
print("sum =" , sum(x))

n = len(x)
mean = sum(x)/n #Calculates mean of list x
print("Mean =", mean)

sum_sq_distance = 0 # Initialising sum to be zero

for i in x:  # for loop sums up all square distances.
   sq_distance= (i - mean)**2  # Square Distance of Each Element to Mean.
   sum_sq_distance = sum_sq_distance + sq_distance  # Each Square Distance is Summed Up

var = sum_sq_distance/n  # var is short for variance
standard_dev = (var)**(1/2)  #standard_dev is the standard deviation
```

print("Variance =",var)
print("Standard Deviation =", standard_dev)

[3409, 2356, 4234, 5642, 2345, 2123, 4566, 5008, 7643, 8903, 9873, 4532, 6422, 5632, 1234, 4321, 4567, 3456, 5213, 5324, 5678, 4325, 5432, 5632]
sum = 117870
Mean = 4911.25
Variance = 3888288.1875
Standard Deviation = 1971.8742828841803

Perform these operations on the list above:
(i) Add these numbers to the list : 3456, 5432, 4563
(ii) Arrange the list from lowest to highest
(iii) Arrange the list from highest to lowest.

## Dictionaries.

A dictionary in Python is a collection of key-value pairs, where each key is unique. Dictionaries are also known as associative arrays or hash maps. They are similar to lists, but instead of using integer indices to access elements, you use keys. It is an unordered data type that is structured using keys and values and is defined with curly brackets {}. Where keys are like the column names and values are the lists of data.

The syntax for creating a dictionary is as follows:

```
d = {'key1': 'value1', 'key2': 'value2'}
```

Now, lets index information from the dictionary d:

```
print(d["key1"])

print(d["key2"])
```

value1

value2

## Data Frames

When dealing with tabular data or datasets, the Pandas library provides a powerful and versatile data structure called a DataFrame. A  data frame is a two-dimensional, labelled data structure with columns that can be of different types, similar to a spreadsheet or SQL table.

Let's convert the previous lists into a Pandas DataFrame to demonstrate its advantages over lists and dictionaries:

```
import pandas as pd

# Creating a DataFrame
data = {
    'age': [30, 40, 30, 49, 22, 35, 22, 46, 29, 25, 39],
    'gender': ["M", "M", "F", "M", "F", "F", "F", "M", "M", "F", "M"],
    'country': ["South Africa", "Botswana", "South Africa", "South
Africa", "Kenya", "Mozambique", "Lesotho", "Kenya", "Kenya", "Egypt",
"Sudan"]
}

#df = data frame
df = pd.DataFrame(data)

# Displaying the DataFrame
print(df)
```

| | age | gender | country |
|---|---|---|---|
| 0 | 30 | M | South Africa |
| 1 | 40 | M | Botswana |
| 2 | 30 | F | South Africa |
| 3 | 49 | M | South Africa |
| 4 | 22 | F | Kenya |
| 5 | 35 | F | Mozambique |
| 6 | 22 | F | Lesotho |
| 7 | 46 | M | Kenya |
| 8 | 29 | M | Kenya |
| 9 | 25 | F | Egypt |
| 10 | 39 | M | Sudan |

Accessing columns in a Data Frame is straightforward and intuitive.

Suppose that the reader is interested in a single column.

```
#Accessing specific columns

print(df["country"])    # Outputs the country column
```

Which is output as:

| | |
|---|---|
| 1 | Botswana |
| 2 | South Africa |
| 3 | South Africa |
| 4 | Kenya |
| 5 | Mozambique |
| 6 | Lesotho |
| 7 | Kenya |
| 8 | Kenya |
| 9 | Egypt |
| 10 | Sudan |

The advantage of a data frame is that we can store large sets of data into data frames. As mentioned above the scientific research done daily uses data frames.

Example : In an upcoming global athletic competition, the data of the athletes is shown below::

| Name | Age | Gender | Weight(kg) | Height (cm) | Nationality |
|---|---|---|---|---|---|
| Nelson | 22 | Male | 85 | 160 | American |
| Tia | 25 | Female | 60 | 145 | English |
| Priyanka | 24 | Female | 67 | 133 | Indian |
| Duncan | 25 | Male | 73 | 123 | Swedish |

| | | | | | |
|---|---|---|---|---|---|
| Jenna | 23 | Female | 62 | 164 | German |
| Eliot | 21 | Male | 65 | 232 | Ukrainian |

a.) Create a Data Frame for the following data above.

```
import pandas as pd #Importing Pandas as it is used to manipulate data


data = {
          "Names":["Nelson", "Tia", "Priyanka", "Duncan", "Jenna","Eliot"],
          "Age": [22, 25, 24, 25, 23, 21],
          "Gender" : ["Male", "Female", "Female", "Male", "Female",
"Male"],
          "Weight(kg)":[85, 60, 67,73,62,65],
          "Height(cm)" : [160, 145, 133, 123, 164, 232],
          "Nationality": ["American", "English", "Indian", "Swedish",
"German", "Ukrainian"]
       } #All data is stored into a dictionary.
          #The Data is stored into a variable called data


#Converting ,"data", into a Data Frame called dat_frame.


dat_frame = pd.DataFrame(data)


print(dat_frame)  # Output Data Frame
```

Which output as:

| | Names | Age | Gender | Weight(kg) | Height(cm) | Nationality |
|---|---|---|---|---|---|---|
| 0 | Nelson | 22 | Male | 85 | 160 | American |
| 1 | Tia | 25 | Female | 60 | 145 | English |
| 2 | Priyanka | 24 | Female | 67 | 133 | Indian |
| 3 | Duncan | 25 | Male | 73 | 123 | Swedish |
| 4 | Jenna | 23 | Female | 62 | 164 | German |
| 5 | Eliot | 21 | Male | 65 | 232 | Ukrainian |

b.) Drop the Names Column and the Nationality Column,

```
#Drops the column names.

dat_frame.drop(columns=["Names"], inplace = True)


#Drops the Nationality Column Names

dat_frame.drop(columns=["Nationality"], inplace=True)
```

| | Age | Gender | Weight(kg) | Height(cm) |
|---|---|---|---|---|
| 0 | 22 | Male | 85 | 160 |
| 1 | 25 | Female | 60 | 145 |
| 2 | 24 | Female | 67 | 133 |
| 3 | 25 | Male | 73 | 123 |
| 4 | 23 | Female | 62 | 164 |
| 5 | 21 | Male | 65 | 232 |

c.) Give information about the data

```
x = dat_frame.info()
print(x)
```

0 Age 6 non-null int64

1 Gender 6 non-null object

2 Weight(kg) 6 non-null int64

3 Height(cm) 6 non-null int64

d.) Give the descriptive statistics of the data.

```
x = dat_frame.describe()
print(x)
```

| | Age | Weight(kg) | Height(cm) |
|---|---|---|---|
| count | 6.000000 | 6.000000 | 6.000000 |
| mean | 23.333333 | 68.666667 | 159.500000 |
| std | 1.632993 | 9.179688 | 38.785306 |
| min | 21.000000 | 60.000000 | 123.000000 |
| 25% | 22.250000 | 62.750000 | 136.000000 |
| 50% | 23.500000 | 66.000000 | 152.500000 |
| 75% | 24.750000 | 71.500000 | 163.000000 |
| max | 25.000000 | 85.000000 | 232.000000 |

Basic Statistics:

```
# Basic statistics
print(df['age'].min())
print(df['age'].max())
print(df['age'].mean())
```

```
# Filtering data
print(df[df['age'] > 30])


# Slicing rows and columns
print(df[1:4])  # Select rows 1 to 3 and all columns


# Adding a new column
df['new_column'] = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
print(df)


# Removing a column
df.drop(columns=['new_column'], inplace=True)
print(df)
```

## Reading from an Excel File in Python

Consider an excel file ,"county_data" :

| Age | Gender | Country |
|---|---|---|
| 39 | M | South Africa |
| 25 | M | Botswana |
| 29 | F | South Africa |
| 46 | M | South Africa |
| 22 | F | Kenya |
| 35 | F | Mozambique |
| 22 | F | Lesotho |
| 49 | M | Kenya |

| | | |
|---|---|---|
| 30 | M | Kenya |
| 40 | F | Egypt |
| 30 | M | Sudan |

```
import pandas

file = pandas.read_csv("country_data.csv") #pandas is used to read all
contents of the excel file
                                           # it sends the information to a
variable called file

print(file) # file is output onto the sceen
```

```
0   39    M  South Africa
1   25    M      Botswana
2   29    F  South Africa
3   46    M  South Africa
4   22    F         Kenya
5   35    F    Mozambique
6   22    F       Lesotho
7   49    M         Kenya
8   30    M         Kenya
9   40    F         Egypt
10  30    M         Sudan
```

## Information about an Excel Spreadsheet

```
import pandas

file = pandas.read_csv("country_data.csv") #pandas is used to read all
contents of the excel file
                                           # it sends the information to a
variable called file

print(file.info()) # useful information about the excel file such as:
                   #Number of Rows
                   #Data Type of Each Element
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 11 entries, 0 to 10
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Age      11 non-null     int64
 1   Gender   11 non-null     object
 2   Country  11 non-null     object
dtypes: int64(1), object(2)
memory usage: 396.0+ bytes
None
```

## Descriptive Statistics of an Excel Spreadsheet

```
import pandas

file = pandas.read_csv("country_data.csv") #pandas is used to read all
contents of the excel file
                                           # it sends the information to a
variable called file

print(file.describe())  # Gives the descriptive data of the variable file
```

```
Age
count  11.000000
mean   33.363636
std     9.233339
min    22.000000
25%    27.000000
50%    30.000000
75%    39.500000
max    49.000000
```

Some Practice

| sepal_length | sepal_width | petal_length | petal_width | species |
|---|---|---|---|---|
| 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |

| | | | | |
|---|---|---|---|---|
| 4.9 | 3 | 1.4 | 0.2 | Iris-setosa |
| 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| 5 | 3.6 | 1.4 | 0.2 | Iris-setosa |
| 5.4 | 3.9 | 1.7 | 0.4 | Iris-setosa |
| 4.6 | 3.4 | 1.4 | 0.3 | Iris-setosa |
| 5 | 3.4 | 1.5 | 0.2 | Iris-setosa |
| 4.4 | 2.9 | 1.4 | 0.2 | Iris-setosa |
| 4.9 | 3.1 | 1.5 | 0.1 | Iris-setosa |
| 5.4 | 3.7 | 1.5 | 0.2 | Iris-setosa |
| 4.8 | 3.4 | 1.6 | 0.2 | Iris-setosa |
| 4.8 | 3 | 1.4 | 0.1 | Iris-setosa |
| 4.3 | 3 | 1.1 | 0.1 | Iris-setosa |
| 5.8 | 4 | 1.2 | 0.2 | Iris-setosa |
| 5.7 | 4.4 | 1.5 | 0.4 | Iris-setosa |
| 5.4 | 3.9 | 1.3 | 0.4 | Iris-setosa |
| 5.1 | 3.5 | 1.4 | 0.3 | Iris-setosa |
| 5.7 | 3.8 | 1.7 | 0.3 | Iris-setosa |
| 5.1 | 3.8 | 1.5 | 0.3 | Iris-setosa |
| 5.4 | 3.4 | 1.7 | 0.2 | Iris-setosa |
| 5.1 | 3.7 | 1.5 | 0.4 | Iris-setosa |
| 4.6 | 3.6 | 1 | 0.2 | Iris-setosa |
| 5.1 | 3.3 | 1.7 | 0.5 | Iris-setosa |
| sepal_length | sepal_width | petal_length | petal_width | species |
| 5.25326087 | 3.713768116 | 1.459057971 | 0.327173913 | Iris-setosa |
| 5.270521739 | 3.732202899 | 1.460449275 | 0.3333478261 | Iris-setosa |
| 5.287782609 | 3.750637681 | 1.46184058 | 0.3395217391 | Iris-setosa |
| 5.305043478 | 3.769072464 | 1.463231884 | 0.3456956522 | Iris-setosa |
| 5.322304348 | 3.787507246 | 1.464623188 | 0.3518695652 | Iris-setosa |
| 5.339565217 | 3.805942029 | 1.466014493 | 0.3580434783 | Iris-setosa |
| 5.356826087 | 3.824376812 | 1.467405797 | 0.3642173913 | Iris-setosa |
| 5.374086957 | 3.842811594 | 1.468797101 | 0.3703913043 | Iris-setosa |
| 5.391347826 | 3.861246377 | 1.470188406 | 0.3765652174 | Iris-setosa |
| 5.408608696 | 3.879681159 | 1.47157971 | 0.3827391304 | Iris-setosa |
| 5.425869565 | 3.898115942 | 1.472971014 | 0.3889130435 | Iris-setosa |

| | | | | |
|---|---|---|---|---|
| 5.443130435 | 3.916550725 | 1.474362319 | 0.3950869565 | Iris-setosa |
| 5.460391304 | 3.934985507 | 1.475753623 | 0.4012608696 | Iris-setosa |
| 5.477652174 | 3.95342029 | 1.477144928 | 0.4074347826 | Iris-setosa |
| 5.494913043 | 3.971855072 | 1.478536232 | 0.4136086957 | Iris-setosa |
| 5.512173913 | 3.990289855 | 1.479927536 | 0.4197826087 | Iris-setosa |
| 5.529434783 | 4.008724638 | 1.481318841 | 0.4259565217 | Iris-setosa |
| 5.546695652 | 4.02715942 | 1.482710145 | 0.4321304348 | Iris-setosa |
| 5.563956522 | 4.045594203 | 1.484101449 | 0.4383043478 | Iris-setosa |
| 5.581217391 | 4.064028986 | 1.485492754 | 0.4444782609 | Iris-setosa |
| 5.598478261 | 4.082463768 | 1.486884058 | 0.4506521739 | Iris-setosa |
| 5.61573913 | 4.100898551 | 1.488275362 | 0.456826087 | Iris-setosa |
| 5.633 | 4.119333333 | 1.489666667 | 0.463 | Iris-setosa |
| 5.65026087 | 4.137768116 | 1.491057971 | 0.469173913 | Iris-setosa |
| sepal_length | sepal_width | petal_length | petal_width | species |
| 5.667521739 | 4.156202899 | 1.492449275 | 0.4753478261 | Iris-setosa |
| 5.684782609 | 4.174637681 | 1.49384058 | 0.4815217391 | Iris-setosa |
| 5.702043478 | 4.193072464 | 1.495231884 | 0.4876956522 | Iris-setosa |
| 5.719304348 | 4.211507246 | 1.496623188 | 0.4938695652 | Iris-setosa |
| 5.736565217 | 4.229942029 | 1.498014493 | 0.5000434783 | Iris-setosa |
| 5.753826087 | 4.248376812 | 1.499405797 | 0.5062173913 | Iris-setosa |
| 5.771086957 | 4.266811594 | 1.500797101 | 0.5123913043 | Iris-setosa |
| 5.788347826 | 4.285246377 | 1.502188406 | 0.5185652174 | Iris-setosa |
| 5.805608696 | 4.303681159 | 1.50357971 | 0.5247391304 | Iris-setosa |
| 5.822869565 | 4.322115942 | 1.504971014 | 0.5309130435 | Iris-setosa |
| 5.840130435 | 4.340550725 | 1.506362319 | 0.5370869565 | Iris-setosa |
| 5.857391304 | 4.358985507 | 1.507753623 | 0.5432608696 | Iris-setosa |
| 5.874652174 | 4.37742029 | 1.509144928 | 0.5494347826 | Iris-setosa |
| 5.891913043 | 4.395855072 | 1.510536232 | 0.5556086957 | Iris-setosa |
| 5.909173913 | 4.414289855 | 1.511927536 | 0.5617826087 | Iris-setosa |
| 5.926434783 | 4.432724638 | 1.513318841 | 0.5679565217 | Iris-setosa |
| 5.943695652 | 4.45115942 | 1.514710145 | 0.5741304348 | Iris-setosa |
| 5.960956522 | 4.469594203 | 1.516101449 | 0.5803043478 | Iris-setosa |
| 5.978217391 | 4.488028986 | 1.517492754 | 0.5864782609 | Iris-setosa |
| 5.995478261 | 4.506463768 | 1.518884058 | 0.5926521739 | Iris-setosa |
| 6.01273913 | 4.524898551 | 1.520275362 | 0.598826087 | Iris-setosa |

| | | | | |
|---|---|---|---|---|
| 6.03 | 4.543333333 | 1.521666667 | 0.605 | Iris-setosa |
| 6.04726087 | 4.561768116 | 1.523057971 | 0.611173913 | Iris-setosa |
| 6.064521739 | 4.580202899 | 1.524449275 | 0.6173478261 | Iris-setosa |
| sepal_length | sepal_width | petal_length | petal_width | species |
| sepal_length | sepal_width | petal_length | petal_width | species |
| 5.25326087 | 3.713768116 | 1.459057971 | 0.327173913 | Iris-setosa |
| 5.270521739 | 3.732202899 | 1.460449275 | 0.3333478261 | Iris-setosa |
| 5.287782609 | 3.750637681 | 1.46184058 | 0.3395217391 | Iris-setosa |
| 5.305043478 | 3.769072464 | 1.463231884 | 0.3456956522 | Iris-setosa |
| 5.322304348 | 3.787507246 | 1.464623188 | 0.3518695652 | Iris-setosa |
| 5.339565217 | 3.805942029 | 1.466014493 | 0.3580434783 | Iris-setosa |
| 5.356826087 | 3.824376812 | 1.467405797 | 0.3642173913 | Iris-setosa |
| 5.374086957 | 3.842811594 | 1.468797101 | 0.3703913043 | Iris-setosa |
| 5.391347826 | 3.861246377 | 1.470188406 | 0.3765652174 | Iris-setosa |
| 5.408608696 | 3.879681159 | 1.47157971 | 0.3827391304 | Iris-setosa |
| 5.425869565 | 3.898115942 | 1.472971014 | 0.3889130435 | Iris-setosa |
| 5.443130435 | 3.916550725 | 1.474362319 | 0.3950869565 | Iris-setosa |
| 5.460391304 | 3.934985507 | 1.475753623 | 0.4012608696 | Iris-setosa |
| 5.477652174 | 3.95342029 | 1.477144928 | 0.4074347826 | Iris-setosa |
| 5.494913043 | 3.971855072 | 1.478536232 | 0.4136086957 | Iris-setosa |
| 5.512173913 | 3.990289855 | 1.479927536 | 0.4197826087 | Iris-setosa |
| 5.529434783 | 4.008724638 | 1.481318841 | 0.4259565217 | Iris-setosa |
| 5.546695652 | 4.02715942 | 1.482710145 | 0.4321304348 | Iris-setosa |
| 5.563956522 | 4.045594203 | 1.484101449 | 0.4383043478 | Iris-setosa |
| 5.581217391 | 4.064028986 | 1.485492754 | 0.4444782609 | Iris-setosa |
| 5.598478261 | 4.082463768 | 1.486884058 | 0.4506521739 | Iris-setosa |
| 5.61573913 | 4.100898551 | 1.488275362 | 0.456826087 | Iris-setosa |
| sepal_length | sepal_width | petal_length | petal_width | species |
| 5.25326087 | 3.713768116 | 1.459057971 | 0.327173913 | Iris-setosa |
| 5.270521739 | 3.732202899 | 1.460449275 | 0.3333478261 | Iris-setosa |
| 5.287782609 | 3.750637681 | 1.46184058 | 0.3395217391 | Iris-setosa |
| 5.305043478 | 3.769072464 | 1.463231884 | 0.3456956522 | Iris-setosa |
| 5.322304348 | 3.787507246 | 1.464623188 | 0.3518695652 | Iris-setosa |
| 5.339565217 | 3.805942029 | 1.466014493 | 0.3580434783 | Iris-setosa |
| 5.356826087 | 3.824376812 | 1.467405797 | 0.3642173913 | Iris-setosa |

| | | | | |
|---|---|---|---|---|
| 5.374086957 | 3.842811594 | 1.468797101 | 0.3703913043 | Iris-setosa |
| 5.391347826 | 3.861246377 | 1.470188406 | 0.3765652174 | Iris-setosa |
| 5.408608696 | 3.879681159 | 1.47157971 | 0.3827391304 | Iris-setosa |
| 5.425869565 | 3.898115942 | 1.472971014 | 0.3889130435 | Iris-setosa |
| 5.443130435 | 3.916550725 | 1.474362319 | 0.3950869565 | Iris-setosa |
| 5.460391304 | 3.934985507 | 1.475753623 | 0.4012608696 | Iris-setosa |
| 5.477652174 | 3.95342029 | 1.477144928 | 0.4074347826 | Iris-setosa |
| 5.494913043 | 3.971855072 | 1.478536232 | 0.4136086957 | Iris-setosa |
| 5.512173913 | 3.990289855 | 1.479927536 | 0.4197826087 | Iris-setosa |
| 5.529434783 | 4.008724638 | 1.481318841 | 0.4259565217 | Iris-setosa |
| 5.546695652 | 4.02715942 | 1.482710145 | 0.4321304348 | Iris-setosa |
| 5.563956522 | 4.045594203 | 1.484101449 | 0.4383043478 | Iris-setosa |
| 5.581217391 | 4.064028986 | 1.485492754 | 0.4444782609 | Iris-setosa |
| 5.598478261 | 4.082463768 | 1.486884058 | 0.4506521739 | Iris-setosa |
| 5.61573913 | 4.100898551 | 1.488275362 | 0.456826087 | Iris-setosa |
| sepal_length | sepal_width | petal_length | petal_width | species |
| 5.25326087 | 3.713768116 | 1.459057971 | 0.327173913 | Iris-setosa |
| 5.270521739 | 3.732202899 | 1.460449275 | 0.3333478261 | Iris-setosa |
| 5.287782609 | 3.750637681 | 1.46184058 | 0.3395217391 | Iris-setosa |
| 5.305043478 | 3.769072464 | 1.463231884 | 0.3456956522 | Iris-setosa |
| 5.322304348 | 3.787507246 | 1.464623188 | 0.3518695652 | Iris-setosa |
| 5.339565217 | 3.805942029 | 1.466014493 | 0.3580434783 | Iris-setosa |
| 5.356826087 | 3.824376812 | 1.467405797 | 0.3642173913 | Iris-setosa |
| 5.374086957 | 3.842811594 | 1.468797101 | 0.3703913043 | Iris-setosa |
| 5.391347826 | 3.861246377 | 1.470188406 | 0.3765652174 | Iris-setosa |
| 5.408608696 | 3.879681159 | 1.47157971 | 0.3827391304 | Iris-setosa |
| 5.425869565 | 3.898115942 | 1.472971014 | 0.3889130435 | Iris-setosa |
| 5.443130435 | 3.916550725 | 1.474362319 | 0.3950869565 | Iris-setosa |
| 5.460391304 | 3.934985507 | 1.475753623 | 0.4012608696 | Iris-setosa |
| 5.477652174 | 3.95342029 | 1.477144928 | 0.4074347826 | Iris-setosa |
| 5.494913043 | 3.971855072 | 1.478536232 | 0.4136086957 | Iris-setosa |
| 5.512173913 | 3.990289855 | 1.479927536 | 0.4197826087 | Iris-setosa |
| 5.529434783 | 4.008724638 | 1.481318841 | 0.4259565217 | Iris-setosa |
| 5.546695652 | 4.02715942 | 1.482710145 | 0.4321304348 | Iris-setosa |
| sepal_length | sepal_width | petal_length | petal_width | species |

| | | | | |
|---|---|---|---|---|
| 5.25326087 | 3.713768116 | 1.459057971 | 0.327173913 | Iris-setosa |
| 5.270521739 | 3.732202899 | 1.460449275 | 0.3333478261 | Iris-setosa |
| 5.287782609 | 3.750637681 | 1.46184058 | 0.3395217391 | Iris-setosa |
| 5.305043478 | 3.769072464 | 1.463231884 | 0.3456956522 | Iris-setosa |
| 5.322304348 | 3.787507246 | 1.464623188 | 0.3518695652 | Iris-setosa |
| 5.339565217 | 3.805942029 | 1.466014493 | 0.3580434783 | Iris-setosa |
| 5.356826087 | 3.824376812 | 1.467405797 | 0.3642173913 | Iris-setosa |
| 5.374086957 | 3.842811594 | 1.468797101 | 0.3703913043 | Iris-setosa |
| 5.391347826 | 3.861246377 | 1.470188406 | 0.3765652174 | Iris-setosa |

```
import pandas

file = pandas.read_csv("iris.csv") #pandas is used to read all contents of
the excel file
                                                # it sends the information to a
variable called file

print(file)  # Gives the descriptive data of the variable file
print(file.info())
print(file.describe())
```

```
sepal_length  sepal_width  petal_length  petal_width        species
0          5.1          3.5           1.4          0.2    Iris-setosa
1          4.9          3.0           1.4          0.2    Iris-setosa
2          4.7          3.2           1.3          0.2    Iris-setosa
3          4.6          3.1           1.5          0.2    Iris-setosa
4          5.0          3.6           1.4          0.2    Iris-setosa
..         ...          ...           ...          ...            ...
145        6.7          3.0           5.2          2.3  Iris-virginica
146        6.3          2.5           5.0          1.9  Iris-virginica
147        6.5          3.0           5.2          2.0  Iris-virginica
148        6.2          3.4           5.4          2.3  Iris-virginica
149        5.9          3.0           5.1          1.8  Iris-virginica

[150 rows x 5 columns]
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
```

```
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
None
       sepal_length  sepal_width  petal_length  petal_width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.054000      3.758667     1.198667
std        0.828066     0.433594      1.764420     0.763161
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000
```

```
import pandas

file = pandas.read_csv("diab_data.csv")
print(file)
print(file.info())
print(file.describe())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   preg_count  768 non-null    int64
 1   glucose     768 non-null    int64
 2   bp          768 non-null    int64
 3   skinfold    768 non-null    int64
 4   insulin     768 non-null    int64
 5   BMI         768 non-null    float64
 6   predigree   768 non-null    float64
 7   age         768 non-null    int64
 8   class       768 non-null    int64
dtypes: float64(2), int64(7)
```

```
memory usage: 54.1 KB
None
       preg_count     glucose          bp    skinfold     insulin         BMI  predigree         age       class
count  768.000000  768.000000  768.000000  768.000000  768.000000  768.000000  768.000000  768.000000  768.000000
mean     3.845052  120.894531   69.105469   20.536458   79.799479   31.992578    0.471876   33.240885    0.348958
std      3.369578   31.972618   19.355807   15.952218  115.244002    7.884160    0.331329   11.760232    0.476951
min      0.000000    0.000000    0.000000    0.000000    0.000000    0.000000    0.078000   21.000000    0.000000
25%      1.000000   99.000000   62.000000    0.000000    0.000000   27.300000    0.243750   24.000000    0.000000
50%      3.000000  117.000000   72.000000   23.000000   30.500000   32.000000    0.372500   29.000000    0.000000
75%      6.000000  140.250000   80.000000   32.000000  127.250000   36.600000    0.626250   41.000000    1.000000
max     17.000000  199.000000  122.000000   99.000000  846.000000   67.100000    2.420000   81.000000    1.000000
```

## ETL

ETL stands for Extract, Transform and Load. It is a method that is used by data scientists as well as any other python programer to organise data that is easier to read and understand by fellow python programmers.

The above section teaches one how to read data in python from the folder that has both the data and the python file.

Now suppose that we download a data file that comes in a folder and is extracted as a folder into the same directory that python is in. How do we access the data file ,"country_data_index.csv" that is stored in a folder called "data_02".

We write this code below. In the code below pandas is imported as pd. The pd function uses the pandas library to read the contents of the data file ,"country_data_index.csv". Long story short, we use the pandas library to read all the contents of the data file and we store it as a variable called df. We then print out the variable df which outputs everything about the file to screen.

```
import pandas as pd
```

```
df = pd.read_csv("data_02/country_data_index.csv")
print(df)
```

which is output as :

```
Unnamed: 0  Age Gender       Country
0           0   39     M  South Africa
1           1   25     M      Botswana
2           2   29     F  South Africa
3           3   46     M  South Africa
4           4   22     F         Kenya
5           5   35     F    Mozambique
6           6   22     F       Lesotho
7           7   49     M         Kenya
8           8   30     M         Kenya
9           9   40     F         Egypt
10         10   30     M         Sudan
```

which is everything the file “country_data_index” has.

Pandas reads data via link. Here is a link to some data:

https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data

Here is the code below:

```
import pandas as pd


df =
pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data")

print(df)
```

Which gives us:

```
0    4.9  3.0  1.4  0.2     Iris-setosa

1    4.7  3.2  1.3  0.2     Iris-setosa

2    4.6  3.1  1.5  0.2     Iris-setosa
```

```
3    5.0  3.6  1.4  0.2     Iris-setosa
4    5.4  3.9  1.7  0.4     Iris-setosa
..   ...  ...  ...  ...          ...
144  6.7  3.0  5.2  2.3  Iris-virginica
145  6.3  2.5  5.0  1.9  Iris-virginica
146  6.5  3.0  5.2  2.0  Iris-virginica
147  6.2  3.4  5.4  2.3  Iris-virginica
148  5.9  3.0  5.1  1.8  Iris-virginica

[149 rows x 5 columns]
```

Which is an enormous amount of data. As a researcher/data scientist you will be exposed to an enormous amount of data.

We've completed the extraction part. Now we come to the transformation part.

Above we can see that the data does not have columns. We want to add the columns , "sepal length", "sepal width", petal length", "petal width" and "class". We do this by using the following code:

```
import pandas as pd

columns = ["sepal length", "sepal width", "petal length", "petal width",
"class"]

df =
pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/iri
s/iris.data", names = columns)

print(df)
```

In the above code, we defined a list called columns. We enter the missing headings ,"sepal length", "sepal width", "petal length" , "petal width" and 'class" as elements to the list called columns.

This results in the output:

| | sepal length | sepal width | petal length | petal width | class |
|---|---|---|---|---|---|
| 0 | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| 1 | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| 2 | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| 3 | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| 4 | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |
| .. | ... | ... | ... | ... | ... |
| 145 | 6.7 | 3.0 | 5.2 | 2.3 | Iris-virginica |
| 146 | 6.3 | 2.5 | 5.0 | 1.9 | Iris-virginica |
| 147 | 6.5 | 3.0 | 5.2 | 2.0 | Iris-virginica |
| 148 | 6.2 | 3.4 | 5.4 | 2.3 | Iris-virginica |
| 149 | 5.9 | 3.0 | 5.1 | 1.8 | Iris-virginica |

150 rows x 5 columns

Reading different files in python:

*Text File:*

Don’t forget to include sep = ;

```
df = pd.read_csv("data_02/Geospatial Data.txt",sep=";")
```

*Microsoft Excel :*

```
df = pd.read_excel("data_02/residentdoctors.xlsx")
```

*JSON File:*

```
df = pd.read_json("data_02/student_data.json")
```

Loading a data file:

*CSV File:*

```
df.to_csv("data_02/output/iris_data_cleaned.csv")
```

*CSV File without Index*

```
df.to_csv("data_02/output/iris_data_cleaned.csv", index=False)
```

*Excel File*

```
df.to_excel("data_02/output/iris_data_cleaned.xlsx",
index=False, sheet_name='Sheet1')
```

*JSON File*

```
df.to_json("data_02/output/iris_data_cleaned.json",
orient='records')
```