Introduction, Setup, Coding & Statistics : Python

Prof Love Arora

BITS Pilani
Pilani|Dubai|Goa|Hyderabad

Introduction, Setup, Coding & Statistics : Python

Overview

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming.

Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL)

Overview

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Features

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
- **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

Features

- **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** − Python provides interfaces to all major commercial databases.
- **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable** − Python provides a better structure and support for large programs than shell scripting.

R vs Python

Parameter	R	Python
Objective	Data analysis and statistics	Deployment and production
Primary Users	Scholar and R&D	Programmers and developers
Flexibility	Easy to use available library	Easy to construct new models from scratch. I.e., matrix computation and optimization
Learning curve	Difficult at the beginning	Linear and smooth
Popularity of Programming Language. Percentage change	4.23% in 2018	21.69% in 2018
Average Salary	$99.000	$100.000
Integration	Run locally	Well-integrated with app
Task	Easy to get primary results	Good to deploy algorithm
Database size	Handle huge size	Handle huge size
IDE	Rstudio	Spyder, Ipthon Notebook
Important Packages and library	tydiverse, ggplot2, caret, zoo	pandas, scipy, scikit-learn, TensorFlow, caret

R vs Python

The choice between R or Python depends on various factors:

- The objectives of your mission: Statistical analysis or deployment
- Use Case considered for analysis.
- The amount of time you can invest.

Setup & Install Python

Unix and Linux Installation

Here are the simple steps to install Python on Unix/Linux machine.

- Open a Web browser and go to https://www.python.org/downloads/.
- Follow the link to download zipped source code available for Unix/Linux.
- Download and extract files.
- Editing the *Modules/Setup* file if you want to customize some options.
- run ./configure script
- make
- make install

This installs Python at standard location */usr/local/bin* and its libraries at */usr/local/lib/pythonXX* where XX is the version of Python.

Setup & Install Python

innovate achieve lead

Windows Installation

Here are the steps to install Python on Windows machine.

- Open a Web browser and go to https://www.python.org/downloads/.
- Follow the link for the Windows installer *python-XYZ.msi* file where XYZ is the version you need to install.
- To use this installer *python-XYZ.msi*, the Windows system must support Microsoft Installer 2.0. Save the installer file to your local machine and then run it to find out if your machine supports MSI.
- Run the downloaded file. This brings up the Python install wizard, which is really easy to use. Just accept the default settings, wait until the install is finished, and you are done.

Setup & Install Python

innovate achieve lead

Macintosh Installation

- Recent Macs come with Python installed, but it may be several years out of date. See http://www.python.org/download/mac/ for instructions on getting the current version along with extra tools to support development on the Mac. For older Mac OS's before Mac OS X 10.3 (released in 2003), MacPython is available.
- You can have full access to the entire documentation at website − http://www.cwi.nl/~jack/macpython.html. You can find complete installation details for Mac OS installation.

Python IDE's

- **Spyder -** It's an open source cross-platform IDE for data science. If you have never worked with an IDE, Spyder could perfectly be your first approach.
- **Jupyter Notebook -** Jupyter Notebook was born out of IPython in 2014. It is a web application based on the server-client structure, and it allows you to create and manipulate notebook documents - or just "notebooks".
- **Atom -** An open source text editor developed by Github.
- **Thonny -** an IDE for learning and teaching programming. It's a software developed at The University of Tartu, which you can download for free on the Bitbucket repository for Windows, Linux, and Mac
- **PyCharm -** PyCharm is an IDE made by the folks at JetBrain, a team responsible for one of the most famous Java IDE, the IntelliJ IDEA

What is Anaconda for Python?

Anaconda is a an open source distribution of the Python and R programming languages and it is used in data science, machine learning, deep learning-related applications aiming at simplifying package management and deployment. Anaconda Distribution is used by over 7 million users, and it includes more than 300 data science packages suitable for Windows, Linux, and MacOS.

Does Anaconda come with Python?

Anaconda is one of several Python distributions. Python on it's own is not going to be useful unless an IDE is installed. This is where Anaconda comes into picture.

Python distributions provide the Python interpreter, together with a list of Python editors, tools and packages.

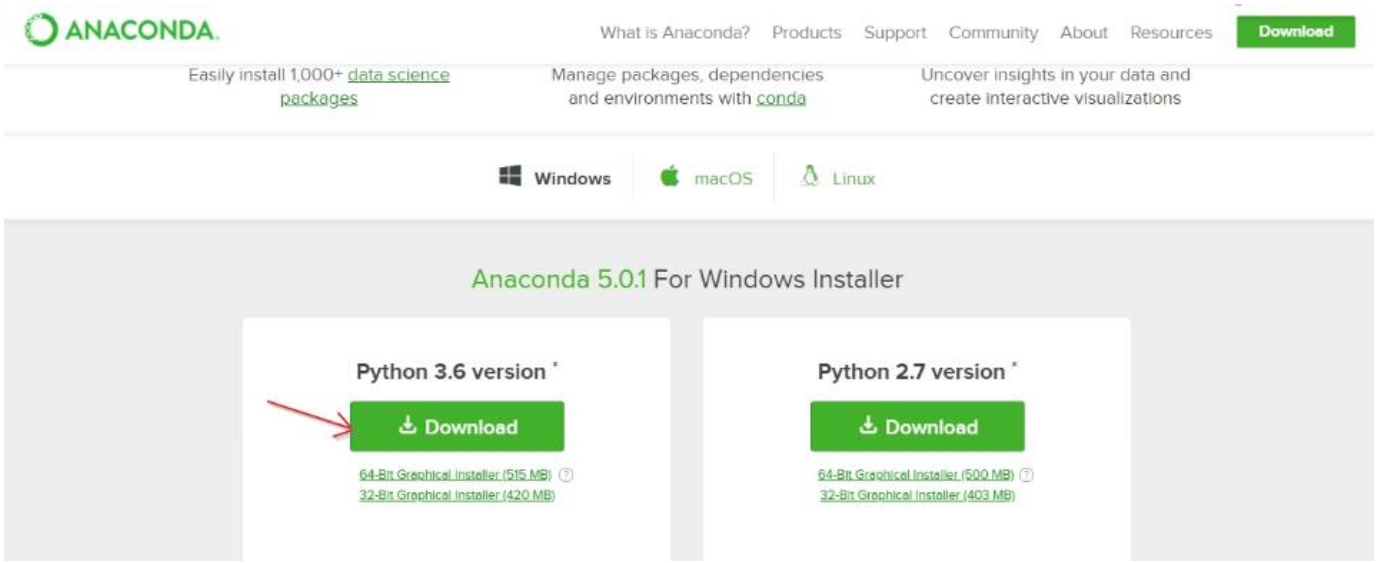
How to Install Anaconda 5.x in Windows

innovate achieve lead

Let's get started with Anaconda 5.x installation

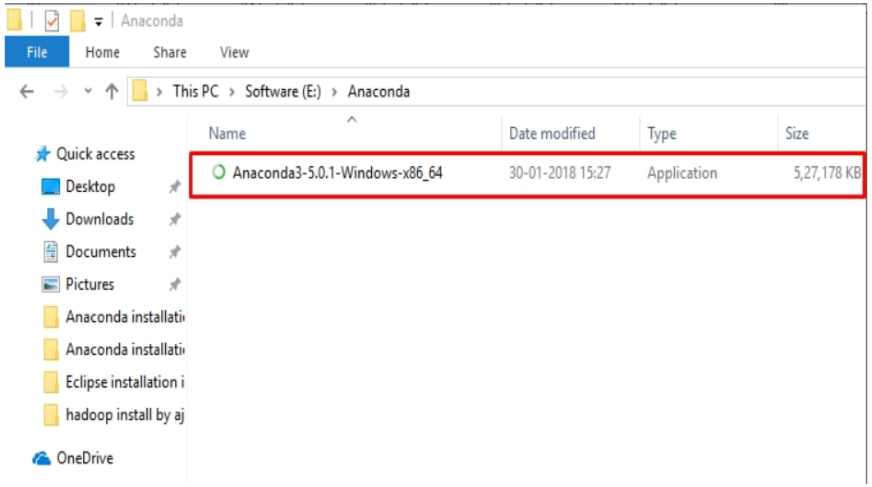
Step 1: Visit Anaconda website to download anaconda for windows. Click on Download

[Anaconda Python Download for windows]



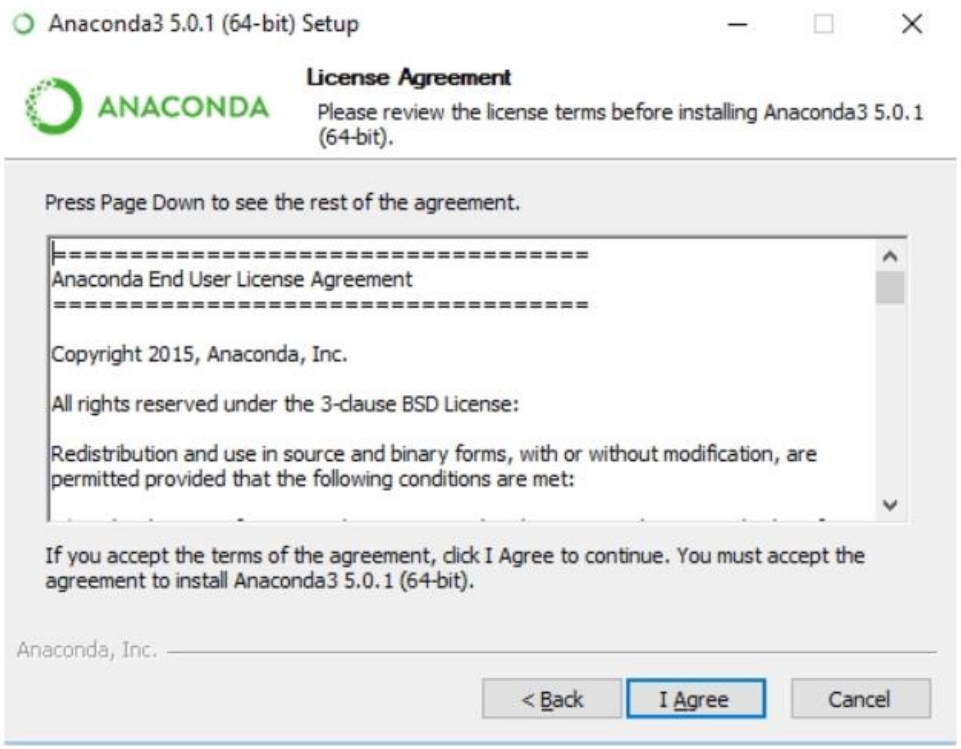
How to Install Anaconda 5.x in Windows

Step 2: Next, double click on setup and click on next.



How to Install Anaconda 5.x in Windows

Step 3: Click on “I agree” to Accept the agreement



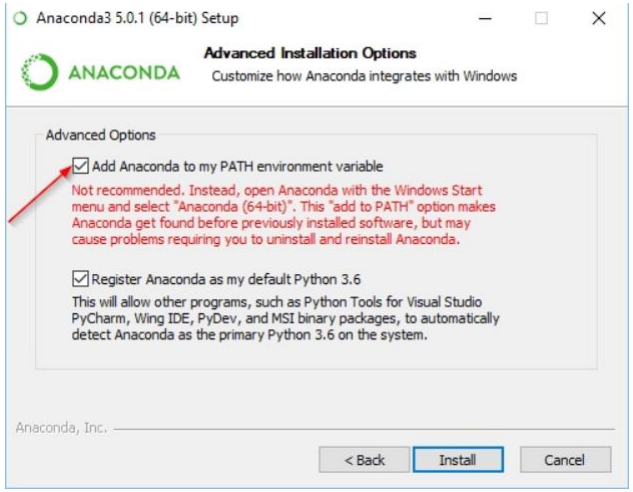
How to Install Anaconda 5.x in Windows

Step 4: Click on “Just Me (recommended)” and click on Next.

Step 5: Retain default location and click next.

Step 6: Click on “Add anaconda to my Path environment variable”.

NOTE: This step is important because jupyter notebook cannot be accessed without this command prompt.

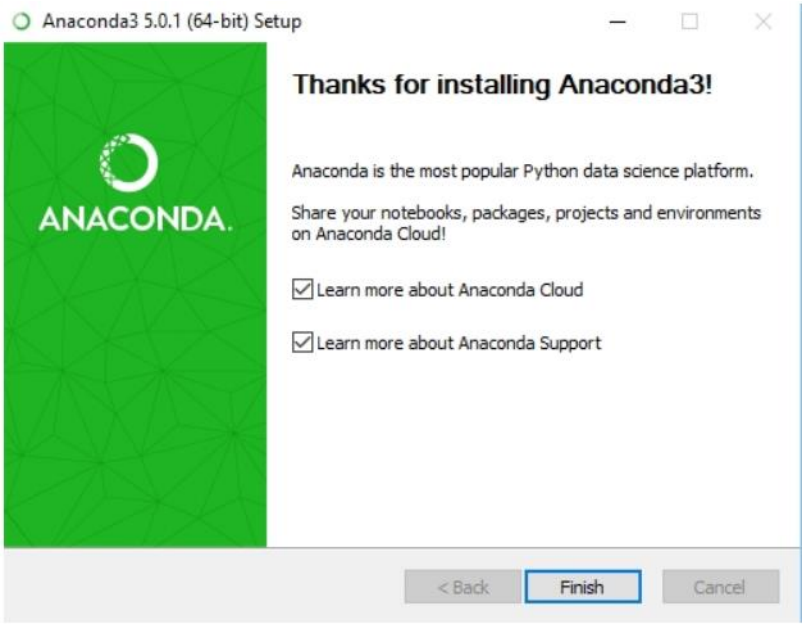


How to Install Anaconda 5.x in Windows

innovate achieve lead

Step 7: Click on the Install Button and wait till the installation is complete.

Step 8: Click on finish.



Preferred Versions for BITS Courses

innovate achieve lead

Python – Version 3.7

Anaconda Distribution for Python – 5.X.X

=> ***It is strongly recommend installing Python and Jupyter/Spyder using the Anaconda Distribution, which includes Python, the Jupyter Notebook, Spyder and other commonly used packages for scientific computing and data science.***

Python – Hello World!

innovate achieve lead

- Type the following text at the Python prompt and press the Enter:
  >>> print "Hello, Python!"
  >>> print ("Hello, Python!"); *New versions

- Create a python script and execute following command:
  >>> python test.py

Python – Comments

Comments in Python start with the hash character, #, and extend to the end of the physical line. A comment may appear at the start of a line or following whitespace or code, but not within a string literal. A hash character within a string literal is just a hash character. Since comments are to clarify code and are not interpreted by Python, they may be omitted when typing in examples.

```
# this is the first comment
spam = 1  # and this is the second comment
          # ... and now a third!
text = "# This is not a comment because it's inside quotes."
```

Python – Numbers

```
>>> 2 + 2
4
>>> 50 - 5*6
20
>>> (50 - 5*6) / 4
5.0
>>> 8 / 5  # division always returns a floating point number
1.6
>>> 5 ** 2  # 5 squared
25
```

Python – Strings (1/2)

innovate achieve lead

```
>>> 'spam eggs'  # single quotes
'spam eggs'
>>> 'doesn\'t'  # use \' to escape the single quote...
"doesn't"

>>> print('C:\some\name')  # here \n means newline!
C:\some
ame
>>> print(r'C:\some\name')  # note the r before the quote
C:\some\name

>>> 'Py' 'thon'
'Python'
```

Python – Strings (2/2)

innovate achieve lead

```
>>> word = 'Python'
>>> word[0]  # character in position 0
'P'
>>> word[5]  # character in position 5
'n’
>>> word[-1]  # last character
'n'
>>> word[-2]  # second-last character
'o’
>>> word[0:2]  # characters from position 0 (included) to 2 (excluded)
'Py'
>>> word[2:5]  # characters from position 2 (included) to 5 (excluded)
'tho'
```

Python – Lists

Python knows a number of *compound* data types, used to group together other values. The most versatile is the *list*, which can be written as a list of comma-separated values (items) between square brackets. Lists might contain items of different types, but usually the items all have the same type.

```
>>> squares = [1, 4, 9, 16, 25]
>>> squares
[1, 4, 9, 16, 25]
>>> squares[-3:]  # slicing returns a new list
[9, 16, 25]
```

**Unlike strings, which are immutable, lists are a mutable type, i.e. it is possible to change their content*

Type identification and conversion

innovate achieve lead

Type identification

type(var**)**

Example 1

```
>>> x = 2
>>> type (x)
<class 'int'>
```

Example 2

```
>>> x = 2.5
>>> type (x)
<class 'float'>
```

Type conversion

datatype(var**)**

Example 1

```
>>> x = 23.99
>>> int (x)
23
```

Example 2

```
>>> int ("25")
25
```

Basic Input and Output

Input can be taken from the user using the **input()** function.

Example 1: Input without a prompt

```
>>> name = input()
John
>>> lang = input()
Python
>>> print ( name, “is learning”, lang)
John is learning Python
```

Example 2: Input with a prompt

```
>>> name = input(“Enter your name:”)
Enter your name: John
>>> lang = input(“Enter language:”)
Enter language: Python
>>> print ( name, “is learning”, lang)
John is learning Python
```

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Control Structures

The first program

innovate achieve lead

Example: Performing basic math functions

```
import math

x = int(input("Enter an integer:"))

print ("x={}".format(x))
print ("x lies between {} and {}".format(x-1 , x+1))
print ("square(x)={} cube(x)={}".format(x**2 , pow(x,3)))
print ("sqrt(x)={}".format(math.sqrt(x)))
```

Import the math module that is required for functions like sqrt() etc.

The input is immediately typecast to type int as every input is by default taken in as a string input.

Square of an integer can be calculated using x**2 or pow(x,2)

Control Statements > if-else

innovate achieve lead

The if – else statement

if condition: statement block 1
else:
    statement block 2

```
#Program to check eligibility of voting

age= int(input("Enter your age:"))

if age >=18:    print ("You can vote!")
else:
  print ("Sorry, you can't vote.")
```

The if – elif – else statement

if condition 1: statement block 1
elif condition 2: statement block 2
else: statement block 3

```
#Program to check positive, negative, zero number
x= int(input("Enter your age:"))

if x >0: print ("{} is positive".format(x))
elif x<0: print ("{} is negative".format(x))
else: print ("You have entered zero.")
```

Note:
In situations where a statement is required, but there is nothing to do, you can use the **pass** keyword
Example: if x > 0: pass

Control Statements > while loop

The while loop syntax is as below

while condition:
    statement block

```
counter = int(input("Enter a number:"))

i = 1
while i <= counter:
    print (i)
    i = i+1
```

Output

```
Enter a number: 5
1
2
3
4
5
```

Note:

1. There is no increment operator (++) that we commonly use in C++ or Java programming.
2. The break statement can be used to exit from a loop.

Control Statements > for loop

The **for** loop syntax is as below. If the loop is to run through a sequence of numbers, then the range function can be used in conjunction with the for loop as shown below

for var in range(end):statement block

```
for i in range(5): print (i)
```

for var in range(start, end):statement block

```
for i in range(5,10): print (i)
```

for var in range(start, end, step):statement block

```
for i in range(10,50,5): print (i)
```

Note:

1. There is no need to specify the step in a for loop if the increment is by 1
2. The continue statement can be used to move to the next iteration value without doing anything for the current iteration

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Lists and Tuples

Lists

- A list in Python is similar but not limited to arrays in C++ and Java.
- A list is an ordered sequence of elements that can be dynamically changed and size altered on the fly

- Syntax
  - **list [ ]** is the syntax used if we have to define the type.

    Example: >>> L = **list([2,3,4])**

  - lists can also be directly defined using only the **[ ]** brackets.

    Example: >>> L = **[2,3,4]**

Lists - operations

The below operations are based on the list defined here
>>> L = [2,3,4,2,5]

Operation	Code Example
Accessing list elements	**>>>L[0]** #0 is the index to the element in the list 2
Counting list elements	**>>> len (L)** 5
Iterating through list elements	**>>> for i in L: print (i)** 2 3 4 2 5
Searching elements within the list	**>>> 3 in L** True
Counting occurrences	**>>> L.count(2)** 2

Lists - operations

The below operations are based on the list defined here
>>> L = [2,3,4,2,5]

Operation	Code Example		
List slices	**>>>L[1:3]** [3,4,2]	**>>>L[2:]** [4,2,5]	**>>>L[:3]** [2,3,4,2]
Replacing elements in a list slice	**>>> L[2:3]=[6,7]** >>>L [2,3,6,7,5]	**>>> L[1:3]=[9]** >>>L [2,9,5]	**>>> L[2:]=[]** >>>L [2,3]
Appending elements to a list	**>>>L.append(9)** [2,3,4,2,5,9]		
Merging two lists	>>> L2 = [1,2] **>>>L.extend(L2)** ; L [2,3,4,2,5,1,2]		
Deleting elements at position	**>>> del L[1]** #deletes element at position 1		
Remove specific elements from a list	>>>**L.remove(2)** ; L [3,4,2,5]	#used to remove the first occurrence of 2 #regardless of its position	

Lists - operations

The below operations are based on the list defined here
>>> L = [2,3,4,2,5]

Action	Code Example	
Delete and display the deleted element from a list	>>>**L.pop(2)** 4 >>>L [2,3,2,5]	
Finding the min element in a list	>>>**min(L)** 2	
Finding the max element in a list	>>>**max(L)** 5	
Reversing a list	>>>**L.reverse()** >>>L [5,2,4,3,2]	
Sorting a list	>>>**L.sort()** >>>L [2,2,3,4,5]	>>>**L.sort(reverse=True)** >>>L [5,4,3,2,2]

Tuples

- A Tuple is similar to a list except that the contents of a Tuple are fixed and cannot be changed once created

- Syntax
  - Tuple() is the syntax used if we have to define the type.

    Example: >>> T = **tuple(2,3,4)**

  - Tuples can also be directly defined using only the ( ) brackets.

    Example: >>> T = **(2,3,4)**

Note:

1. Most of the operations performed on lists can be performed of tuples as well.
2. The biggest difference between lists and tuples is that tuples are immutable while lists are dynamically alterable. Hence, if you want to use a list that is frozen, then switch to a tuple and vice versa.

Write to a file

Syntax to write to a text file

fileobject.write(string)

Example

```
f = open("output file.txt" , 'w')

f.write("Hello\n")
f.write("World\n\n")

f.close()
```

innovate achieve lead

BITS Pilani

Pilani|Dubai|Goa|Hyderabad

Functions

Simple Functions

Similar to C++ and Java, a function is a group of related statements that perform a specific task. You use def keyword to create functions in Python.

```
def function_name():
        function body
```

```
def print_lines():
        print("Hello")
        print("World")

# function call
print_lines()
```

Output

Hello
World

Functions with parameters

The parameters are by default sequential and do not have a data type associated.

def function_name(parameter1, parameter2,…):
function body

```
def print_lines(x,y):
    print(x)
    print(y)
```

```
#passing strings
print_lines("Hello","World")
```

```
Hello
World
```

```
#passing a combination of string and int
print_lines("Hello",5)
```

```
Hello
5
```

```
#passing with reference
print_lines(y="Hello",x="World")
```

```
World
Hello
```

Functions with return arguments

The return data type is not required in the definition of the function. You can simple use the return statement anywhere in the function with or without a value

Example of returning values

```
def sum(*x):
        s=0
        for i in x:
            s+=i
        return s
```

```
S = sum(2,3)
print (S)
```

5

```
S = sum(2,3,5,5,6,4)
print (S)
```

25

Statistics in Python

Mean

It is calculated by taking the sum of the values and dividing with the number of values in a data series.

Syntax : mean([value1, value2, value3,….])

Example:

```
import numpy as np
from scipy import stats

dataset= [1,1,2,3,4,6,18]
#mean value
mean= np.mean(dataset)
print("Mean: ", mean)
```

Median

The middle most value in a data series is called the median. The median() function is used in python using numpy to calculate this value.

Syntax : median([value1, value2, value3,….])

Example:

```
import numpy as np
from scipy import stats

dataset= [1,1,2,3,4,6,18]

#median value
median= np.median(dataset)
print("Median: ", median)
```

Mode

The mode is the value that has highest number of occurrences in a set of data. Unlike mean and median, mode can have both numeric and character data. The mode() function is used in python using numpy to calculate this value.

Syntax: mode([value1, value2, value3,….])

Example:

```
import numpy as np
from scipy import stats

dataset= [1,1,2,3,4,6,18]

#mode valuemode= stats.mode(dataset)
print("Mode: ", mode)
```

Variance

The **variance** is a numerical measure of how the data values is dispersed around the mean.

Syntax: Use **var()** function.

Example:
import numpy as np
from scipy import stats

dataset= [1,1,2,3,4,6,18]

Standard Deviation

The standard deviation of an observation variable is the square root of its variance.

Syntax: Use **std()** function.

Example:

```
import numpy as np
from scipy import stats

dataset= [1,1,2,3,4,6,18]

std_result = np.std(dataset)
```

Thank You!!