

```

>>> print('welcome to IDLE')
welcome to IDLE
>>> print('Hello World')
Hello World
>>> str = input('Enter your name')
Enter your name Nirmal
>>> str = input('Enter your name');
Enter your name nirmal
>>> str = input('Enter your name'); print('Entered Name:', str)
Enter your namehello
Entered Name: hello
>>> str = input('Enter your name: '); print('Entered Name:', str)
Enter your name: nirmal
Entered Name: nirmal
>>> print(2 ** 8)
256
>>> lumberjack = 'Okay'
>>> lumberjack
'Okay'

```

```

>>> 'Hello! ' * 8
'Hello! Hello! Hello! Hello! Hello! Hello! Hello! '
>>> import os
>>> os.getcwd()
'C:\\Users\\USER\\AppData\\Local\\Programs\\Python\\Python39'
>>> print('datacamp','tutorial','on','python','print','function',sep='\n')
datacamp
tutorial
on
python
print
function
>>> print('datacamp','tutorial','on','python','print','function',sep=',')
datacamp,tutorial,on,python,print,function

```

```

>>> print('datacamp','tutorial','on','python','print','function',sep='\n\n')
datacamp

tutorial

on

python

print

function

```

```
>>> for x in 'spam':  
...     print(x)  
...  
s  
p  
a  
m
```

- `end`: It is a string appended after the last value, defaults to a newline. It allows the programmer to define a custom ending character for each print call other than the default newline or `\n`.

```
In [5]: def value(items):  
        for item in items:  
            print(item, end='')
```

```
In [6]: value([1, 2, 3, 4])  
  
1234
```

```
In [7]: def value1(items):  
        for item in items:  
            print(item, end='\n \n')
```

```
In [8]: value1([1, 2, 3, 4])  
  
1  
  
2  
  
3  
  
4
```

Running a Script file in Jupyter Notebook:

- 1) Save the file in whatever directory you want with a `<<FILENAME>>.py` extension
- 2) Open the Jupyter Notebook
- 3) Type the following command in the cell:

```
%run the file indicating the path
```

Example:

```
%run C:\Users\USER\Desktop\MLSLIDES-01SEP2021\MLLAB-01SEP2021\script01.py
```

The **OS module in Python** provides functions for interacting with the operating system. OS comes under Python's standard utility modules. This module provides a portable way of using operating system-dependent functionality.

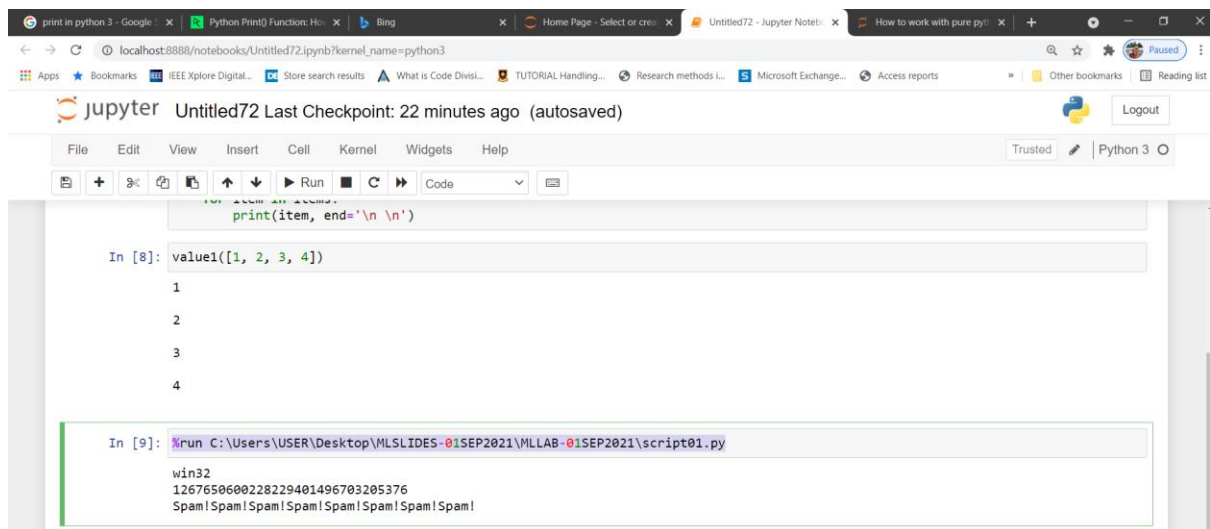
```
import os
```

Command-Line Usage Variations

Because this scheme uses shell command lines to start Python programs, all the usual shell syntax applies. For instance, you can route the printed output of a Python script to a file to save it for later use or inspection by using special shell syntax:

```
% python script1.py > saveit.txt
```

In this case, the three output lines shown in the prior run are stored in the file *saveit.txt* instead of being printed. This is generally known as *stream redirection*; it works



```
print(item, end='\n\n')
```

```
In [8]: value1([1, 2, 3, 4])
```

```
1
2
3
4
```

```
In [9]: %run C:\Users\USER\Desktop\MLSLIDES-01SEP2021\MLLAB-01SEP2021\script01.py
```

```
win32
1267650600228229401496703205376
Spam!Spam!Spam!Spam!Spam!Spam!Spam!Spam!
```

Python has ability to solve the mathematical expression, statistical data by importing **statistic** keyword. Python can do various types of statistical and mathematical operations.

These functions calculate the average value from a sample or population.

mean ()	Arithmetic mean value (average) of data.
harmonic_mean ()	Harmonic mean value of data.
median ()	Median value (middle value) of data.
median__low()	Low median value of data.
median__high()	High median value of data.
median__grouped()	Median of the grouped data and also calculate the 50th percentile of the grouped data.
mode()	Maximum number of occurrence of data.

mean()

This function calculates the arithmetic mean or average value of sample data in sequence or iterator.

Example

```
list = [1, 2, 3,3,4,5,]  
print ("The mean values is : ",end="")  
print (statistics.mean(list))
```

Output

```
The mean value is : 3
```

harmonic_mean ()

This function calculates a sequential or iterative real- valued numbers (harmonic_ mean).

Example

```
list = [1,2,3]  
print ("The harmonic _mean values is : ",end="")  
print (statistics.harmonic_mean(list))
```

Output

```
The harmonic _mean values is :1.6
```

median ()

This function calculates middle value of the arithmetic data in iterative order.

Example

```
list= [1, 3,5,7]
print ("The median values is : ",end="")
print (statistics.median(list))
```

Output

```
The median values is :4.0
```

median__low()

This function calculates the median of data in case of odd number but in case of even number of elements it calculates the lower of two middle elements of the data.

Example

```
list = [1,2,2,3,3,3]
print ("The median_low values is : ",end="")
print (statistics.median_low(list))
```

Output

```
The median_low values is :2
```

median_high()

This function calculates the median of data in case of odd number, but in case of even number of elements, it calculates the higher of two middle elements of the data.

Example

```
list = [1,2,2,3,3,3]
print ("The median_high values is : ",end="")
print (statistics.median_high(list))
```

Output

```
The median_high values is :3
```

median_grouped()

This function is used to calculate median of the grouped data also calculate 50th percentile of the grouped data

Example

```
list = [2,2,3,4]
print ("The median_grouped values is : ",end="")
print (statistics.median_grouped(list))
```

Output

```
The median_grouped values is : 2.5
```

mode()

This function returns the most common data point from discrete or nominal data or number with maximum number of occurrences.

Example

```
list = [2,2,3,4,4,1,2]
print ("The mode values is : ",end="")
print (statistics.mode(list))
```

Output

```
The mode values is : 2
```

Numbers and Numeric Representation

Python provides different functions which are used to represent numbers in different forms, for example -

Function	Description
Ceil(x)	It returns the ceiling value which is the smallest value, greater or equal to the number x.
copysign(x, y)	Returns the number of x and copy the sign of y to x.
fabs(x)	Return absolute value of x.
factorial(x)	Returns factorial of x where $x \geq 0$
floor(x)	Returns the floor value which is the largest integer, less or equal to the number x.
fsum(iterable)	Returns sum of the elements in an iterable object
gcd(x,y)	Returns the greatest common divisor of x and y.
isfinite(x)	Checks whether x is neither an infinity nor nan.
isinf(x)	Checks if x is infinity
isnan(s)	Checks whether s is not a number
remainder(x,y)	Find remainder after dividing x by y.

Example program to demonstrate the use of above functions

```
import math
print("The value of sin(45 degree): " + str(math.sin(math.radians(45))))
print('The value of cos(pi): ' + str(math.cos(math.pi)))
print("The value of tan(45 degree): " + str(math.tan(math.pi/2)))
print("the angle of sin(0.95504050560):" + str(math.degrees(math.sin(0.95504050560))))
```

Result

```
The value of sin(45 degree): 0.7071067811865475
The value of cos(pi): -1.0
The value of tan(45 degree): 1.633123935319537e+16
the angle of sin(0.95504050560):46.77267256206895
```