INTRODUCTION TO PYTHON PROGRAMMING

Course Notes

Gaurav Singh

Department of Computer Science,

Microtek College of Management & Technology

12-Jan-2023

PYTHON

Python was created by **Guido van Rossum**, and first released on **February 20, 1991**. While you may know the python as a large snake, the name of the Python programming language comes from an old BBC television comedy sketch series called Monty Python's Flying Circus.

Guido van Rossum did not develop and evolve all the python components himself. Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc.).

• Python has a simple syntax similar to the English language.

• Python has syntax that allows developers to write programs with fewer lines than some other programming languages.

• Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.

• Python can be treated in a procedural way, an object-orientated way or a functional Way

Python programming is a general-purpose skill used in almost all fields, including:

Data science

Scientific and mathematical computing

Web development

Finance and trading

System automation and administration

Computer graphics

Basic game development

Security and penetration testing

General and application-specific scripting

Python Language : Character Set -> Tokens -> Instruction -> Program -> Software

There are five types of tokens :

- Keywords.
- Identifiers.
- Literals.
- Operators.
- Punctuators.

Literal / Constant

- Literal is a fixed data element or values of program like - 25, 3.5, "Hello"

Character Set

All the set of character that are used in Python language.

Digits, alphabets, Special Symbols.

A, a, & *, ^, $, #, @, (,), {, }, [,], \, /, |, <, >, ', ", 1, 2, .,

Python supports ASCII characters and Unicode

Identifier

Identifier is the name of anything in a program like variable name, function name, object name, class name etc.

Identifier is a collection of valid character that identifies something Python supports ASCII characters and Unicode

Python Installation

To check if you have python installed on a Windows PC, search in the start bar for Python or run the following on the Command Line (cmd.exe):

C:\Users\Your Name>python --version

To check if you have python installed on a Linux or Mac, then on linux open the command line or on Mac open the Terminal and type:

python --version

If you find that you do not have python installed on your computer, then you can download it for free from the following website:
https://www.python.org

Indentation refers to the spaces at the beginning of a code line.

Where in other programming languages the indentation in code is for readability only, the

indentation in Python is very important.

Python uses indentation to indicate a block of code.

Example

```
if 5 > 2:
    print(“Five is greater than two!")
```

Python will give you an error if you skip the indentation:

Python Data Types

Numeric Types: int, float, complex

Text Type: str

SEQUENCE TYPES: list, tuple, range

MAPPING TYPE: dict

SET TYPES: set, frozenset

BOOLEAN TYPE: bool

BINARY TYPES: bytes, bytearray, memoryview

Getting The Data Type :You can get the data type of any object By Using The type() Function:

Example

Print the data type of the variable x:

```
x = 5

print(type(x))
```

Python Collections (Arrays)

List is a collection which is ordered and changeable. Allows duplicate members.

• Tuple is a collection which is ordered and unchangeable. Allows duplicate members.

• Set is a collection which is unordered and unindexed. No duplicate members

• Dictionary is a collection which is unordered, changeable and indexed. No duplicate members

Variable declaration

Example

```
x=5

print(x)
```

Output

5

Here x = 5 i.e, is integer value and to confirm the integer variable, in python allows type function

```
print(type(x))
```

Output:

<class 'int'>

a=5.5

Here a = 5.5 i.e, is float value and to confirm the float variable, in python allows type function

print(type(a))

Output:

<class 'float'>

a=’2.6’

Here a = ‘2.6’ i.e, s String value because it is written with coat (‘ ‘). And to confirm the String variable, in python allows type function

print(type(a))

Output:

<class 'str'>

Example

a="17"

b="26"

c=a+b

print(c)

Output

1726

In above given variable a and b both are string data type which is assigned the value with double quotes (" ") when the value is added it into c=a+b then result comes as concatenation

Example

```
print(type(c))
```

Output:

```
<class 'str'>
```

int() Function

The int function in python is a built-in method that converts a string or a number into an integer

Example

```
a="17"

b="26"

c=int(a)+int(b)

print(c)
```

Output:

```
43
```

Python Indentation

Indentation refers to the spaces at the beginning of a code line. Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important

Example

```
 if 5 > 2:

print("Five is greater than two!")

if 5 > 2:

print("Five is greater than two!")
```

You have to use the same number of spaces in the same block of code, otherwise Python will give you an error:

Creating a Comment

Comments starts with a **#**, and Python will ignore them

Multi Line Comments

Python does not really have a syntax for multi line comments. To add a multiline comment you could insert a **#** for each line:

Example

#This is a comment

#written in

#more than just one line

print("Hello, World!" Or, not quite as intended, you can use a multiline string.

Variable Names A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume). Rules for Python variables:

• A variable name must start with a letter or the underscore character

• A variable name cannot start with a number

• A variable name can only contain alpha-numeric characters and underscores (A-z, 0- 9, and _)

• Variable names are case-sensitive (age, Age and AGE are three different variables)

Example

#Legal variable names:

myvar = "John"

my_var = "John"

_my_var = "John"

myVar = "John"

MYVAR = "John"

myvar2 = "John"

#Illegal variable names:

2myvar = "John"

my-var = "John"

my var = "John"

#Remember that variable names are case-sensitive

Assign Value to Multiple Variables

Python allows you to assign values to multiple variables in one line:

x, y, z = "Orange", "Banana", "Cherry"

print(x)

print(y)

print(z)

And you can assign the same value to multiple variables in one line:

Example

x = y = z = "Orange

print(x)

print(y)

print(z)

Output Variables

The Python print statement is often used to output variables. To combine both text and a variable, Python uses the + character:

Example

x = "awesome" print("Python is " + x)

You can also use the + character to add a variable to another variable:

Example

x = "Python is "

y = "awesome"

z = x + y

print(z)

Global Variables

that are created outside of a function (as in all of the examples above) are known as global variables. Global variables can be used by everyone, both inside of functions and outside.

Example Create a variable outside of a function, and use it inside the function

x = "awesome"

def myfunc():

print("Python is " + x)

myfunc()

Python List :

x = [3,1,4,2]

print(x)

output :

[3,1,4,2]

Here x = [3,1,4,2] i.e, is list value in integer format. And to confirm the integer variable into list data type, in python

print(type(x))

output :

<class 'list'>

In List first value of default index position is 0 to n-1

If list have four items example: [3,1,4,2] so index position is 0 1 2 3

List Methods

Python has a set of built-in methods that you can use on lists.

Lists are the simplest containers that are an integral

part of the Python language. Lists need not be homogeneous

always which makes it the most powerful tool in Python.

A single list may contain DataTypes like Integers, Strings, as well as Objects.

Lists are mutable, and hence, they can be altered even after their creation.

Method Description

append() Adds an element at the end of the list

clear() Removes all the elements from the list

copy() Returns a copy of the list

count() Returns the number of elements with the specified value

extend() Add the elements of a list (or any iterable), to the end of the current list

index() Returns the index of the first element with the specified value

insert() Adds an element at the specified position

pop() Removes the element at the specified position

remove() Removes the item with the specified value

reverse() Reverses the order of the list

sort() Sorts the list

Python Tuples

Tuple

A tuple is a collection, which is **ordered** and **unchangeable**. In Python tuples are written

with round brackets.

Example

Create a Tuple:

```
m = ("apple", "banana", "cherry")

print(m)
```

Access Tuple Items

You can access tuple items by referring to the index number, inside square brackets:

Example

Print the second item in the tuple:

```
m = ("apple", "banana", "cherry")

print(m[1])
```

output :

banana

Negative Indexing

Negative indexing means beginning from the end, -1 refers to the last item, -2 refers to the second last item etc.

Example

Print the last item of the tuple:

```
x = ("apple", "banana", "cherry")

print(x[-1])
```

output :

cherry

Range of Indexes

You can specify a range of indexes by specifying where to start and where to end the range. When specifying a range, the return value will be a new tuple with the specified items.

Example

Return the third, fourth, and fifth item:

```
x = ("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")

print(x[2:5])
```

Note: The search will start at index 2 (included) and end at index 5 (not included).

Remember that the first item has index 0.

Range of Negative Indexes

Specify negative indexes if you want to start the search from the end of the tuple:

Example

```
x = ("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")
```

print(x[-4:-1])

This example returns the items from index -4 (included) to index -1 (excluded)

=	"apple",	banana",	cherry",	orange",	kiwi",	melon",	mango")
	-7	-6	-5	-4	-3	-2	-1

print(x[-4:-1])

Here -4 is start position and -1 is stop position -1 (-2)

output : ("orange", "kiwi", "melon")

Set Declaration

set1 = {"a", "b" , "c"}

print(set1)

output :

{'c', 'a', 'b'}

Set Update Function

The update() method updates the current set, by adding items from another set (or any other iterable).

set1 = {"a", "b" , "c"}

set2 = {1, 2, 3}

set1.update(set2)

print(set1)

Output:

{'a', 'b', 1, 2, 3, 'c'}

Set Union Function

Return a set that contains all items from both sets, duplicates are excluded:

```
set1 = {"a", "b" , "c"}

set2 = {"z","y","x"}

set3=set1.union(set2)

print(set3)
```

Output :

{'b', 'z', 'y', 'a', 'c', 'x'}

issubset() method

The issubset() method returns True if set A is the subset of B, i.e. if all the elements of set A are present

in set B . Else, it returns False.

```
a={1,2,5,4,8}

b={2,4,3,6,9}

c=a.issubset(b)

print(c)
```

output:

False

issuperset() Method

The issuperset() method returns True if a set has every elements of another set (passed as an argument). If not, it returns False.

Set X is said to be the superset of set Y if all elements of Y are in X.

```
x = {"f", "e", "d", "c", "b", "a"}

y = {"a", "b", "c"}

z = x.issuperset(y)

print(z)
```

Output :

True

Set intersection() Method

Return a set that contains the items that exist in both set x, and set y:

```
x = {"apple", "banana", "cherry"}

y = {"google", "microsoft", "apple"}


z = x.intersection(y)

print(z)
```

Output :

{'apple'}

```
x = {"a", "b", "c"}

y = {"c", "d", "e"}

z = {"f", "g", "c"}

t = x.intersection(y, z)

print(t)
```

Output :

{‘c’}

Algorithm

Algorithm is a step-by-step procedure, which defines a set of instructions to be executed in a certain order to get the desired output.

Algorithms are generally created independent of underlying languages, i.e. an algorithm can be implemented in more than one programming language.

From the data structure point of view, following are some important categories of algorithms −

Search − Algorithm to search an item in a data structure.

Sort − Algorithm to sort items in a certain order.

Insert − Algorithm to insert item in a data structure.

Update − Algorithm to update an existing item in a data structure.

Delete − Algorithm to delete an existing item from a data structure.

Characteristics of an Algorithm

Not all procedures can be called an algorithm. An algorithm should have the following characteristics −

Unambiguous − Algorithm should be clear and unambiguous. Each of its steps (or phases), and their inputs/outputs should be clear and must lead to only one meaning.

Input − An algorithm should have 0 or more well-defined inputs.

Output − An algorithm should have 1 or more well-defined outputs, and should match the desired output.

Finiteness − Algorithms must terminate after a finite number of steps.

Feasibility − Should be feasible with the available resources.

Independent − An algorithm should have step-by-step directions, which should be independent of any programming code

Example 1

Algorithm 1: Add two numbers entered by the user

Step 1: Start

Step 2: Declare variables num1, num2 and sum.

Step 3: Read values num1 and num2.

Step 4: Add num1 and num2 and assign the result to sum.

sum←num1+num2

Step 5: Display sum

Step 6: Stop

Example 2

Step 1: Start

Step 2: Declare variables a,b and c.

Step 3: Read variables a,b and c.

Step 4: If a > b

If a > c

Display a is the largest number.

Else

Display c is the largest number.

Else

If b > c

Display b is the largest number.

Else

Display c is the greatest number.

Step 5: Stop

If condition using and

```
a = int(input('Enter first number  : '))
b = int(input('Enter second number : '))
c = int(input('Enter third number  : '))
largest = 0
if a > b and a > c:
    largest = a
if b > a and b > c:
    largest = b
if c > a and c > b:
    largest = c
print(largest, "is the largest of three numbers.")
```

Range Function

The range() function

We can generate a sequence of numbers using range() function. range(10) will generate numbers from 0 to 9 (10 numbers).

We can also define the start, stop and step size as range(start, stop,step_size). step_size defaults to 1 if not provided.

To force this function to output all the items, we can use the function list()

Example :

```
>>>print(range(10))

range(0,10)


>>>print(list(range(10)))

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

>>>print(list(range(2,10)))

[2, 3, 4, 5, 6, 7, 8, 9]

>>>print(list(range(2, 20, 3)))

[2, 5, 8, 11, 14, 17]
```

Len Function

Return the number of items in a list, tuple, set, and dictionary

```
x=[2, 5, 8, 11, 14, 17]

print(len(x))
```

Output :

6

List Slicing :

```
thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]

print(thislist[2:])
```

Output

```
['cherry', 'orange', 'kiwi', 'melon', 'mango']


>>>thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon",
"mango"]

print(thislist[-3:-1])
```

Output:

['kiwi', 'melon']

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]

print(thislist[-7:-1:2])

Output:

['apple', 'cherry', 'kiwi']

List slicing with double colon (::)

Example

m = range(10)

print(list(m[2::2]))

Note : here in range function no start value assign so it starts from 0

m =	0	1	2	3	4	5	6	7	8	9	10
	0	1	2	1	2	1	2	1	2		
			0		2		4		6		

So [2::2] would return every 2nd element of the list/string.

Output :

[2, 4, 6, 8]

Example

m=range(1,14,2)

print(list(m[3::-2]))

here in range function start value is 1 and stop value is 14 and step value is 2

so stop will be -1 hence elements will be from 1 to 13 with 2 step value.

m =	1	2	3	4	5	6	7	8	9	10	11	12	13	14
	0	1	2	3	4	5	6	7	8	9	10	11	12	13

1	3	5	7	9	11	13
0	1	2	3	4	5	6
	-2	-1	0			

So in question m[3::-2] which means it'll start from 3rd position and i.e, 7 and -2 which is negative value and start from reverse order

Output:

[7, 3]

Example

```
s="MICROTEK COLLEGE"

z=len(s)

m=range(3,z,2)

print(list(m[2::-2]))
```

Solution

S=

M	I	C	R	O	T	E	K		C	O	L	L	E	G	E
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Eq. 1 Now m=range(3,z,2)

Here z= 16 which is length of s

In eq1. Range starts from 3 position and z=16 which is stop-1 i.e., 15 and step value is 2 so m will be in Eq2.

Eq2.

m=

3	5	7	9	11	13	15
0	1	2	3	4	5	6

Eq3. list(m[2::-2])

Here in Eq. 3 start value 2 position of Eq2. and stop value is -2 position means reverser order.

m=

3	5	7	9	11	13	15
0	1	2	3	4	5	6
-2	-1	0				

Output :

[7,3]

For loop

A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

This is less like the for keyword in other programming languages, and works more like an iterator method as found in other object-orientated programming languages.

The for loop in Python is used to iterate over a sequence (list, tuple, string) or other Iterate objects. Iterating over a sequence is called traversal.

Example

```
x = 0
for i in range(6):
    x = x + i
    print(x,end=" ")
```

Solution

i = 0, 1 , 2, 3, 4, 5 #here i is will increment from 0 to 5

x is printing loop

1 : step > x = x + i = 0 + 0

0

2 : step > x = x + i = 0 + 1

0, 1

3 : step > x = x + i = 1 + 2

0, 1, 3

4 : step > x = x + i = 3 + 3

0, 1, 3, 6

5 : step > x = x + i = 6 + 4

0, 1, 3, 6, 10

6 : step > x = x + i = 10 + 5

0, 1, 3, 6, 10, 15

Output:

0, 1, 3, 6, 10, 15

Example

```
for i in range(10):
        print(i,end=" ")
#Here in above example i value starts from 0 to 9
```

Output:

0 1 2 3 4 5 6 7 8 9

Example

```
x=[4,5,1,7,9,12,8]
z=len(x)
for i in range(z):
        print(x[i],end=" ")
```

Output:

4 5 1 7 9 12 8

Example

```
num = int(input("Enter number:"))
for i in range(1,11):
    x = i*num
    print(x)
#in above program num will store input value suppose num = 2
```

1 : step > x = 1*2 = 2

2 : step > x = 2*2 = 4

3 : step > x = 3*2 = 6

4 : step > x = 4*2 = 8

.

.

.

10 : step > x = 10*2 = 20

Output:

```
Enter number:2
2
4
6
8
10
12
```

14

16

18

20

Example

Factorial program through for loop

```
fact = 1
for i in range(1,6):
    fact = fact * i
    print(fact,end=" ")
```

#Here fact variable storing the multiply with fact * I into the loopi = 1

1 : step > fact = fact * i = 1 * 1 = 1

2 : step > fact = fact * i = 1 * 2 = 2

3 : step > fact = fact * i = 2 * 3 = 6

4 : step > fact = fact * i = 6 * 4 = 24

5 : step > fact = fact * i = 24 * 5 = 120

Output:
1 2 6 24 120

Example Sum of List

```
x=[4,5,1,7,9,12,8]
z=len(x)
c=0
for i in range(z):
```

```
    c+=x[i]
print(c)
```

#Here in program is adding the value of x into c and value of c is printed after completion of loop.

Output :

46

Example for the display of Reverse order list value.

```
x=[4,5,1,7,9,12,8]
z=len(x)
for i in range(z-1,-1,-1):
        print(x[i],end=" ")
```

#Here z is stored the length of list x = 7 i.e., z=7 so in range(z-1,-1,0) which is range(7-1,-1,0) equaling range(6,-1,0) and we are printing x[i] so i is based on the range function which starts position from 6 to 0

Output:

8 12 9 7 1 5 4

Example for another option in reverse order into list value

```
x=[4,5,1,7,9,12,8]
z=len(x)
for i in range(z,0,-1):
        print(x[i-1],end=" ")
```

#Here z is stored the length of list x = 7 i.e., z=7 so in range(z,0,-1) which is range(7,0,-1) and we are printing x[i-1] so i is based on the range function which starts position from 7 to 1

In loop i value is changing from 7,6,5,4,3,2,1

x[i-1] -> x[7-1]=x[6] -> 8

x[i-1] -> x[6-1]=x[5] -> 12

x[i-1] -> x[5-1]=x[4] -> 9

x[i-1] -> x[4-1]=x[3] -> 7

x[i-1] -> x[3-1]=x[2] -> 1

x[i-1] -> x[2-1]=x[1] -> 5

x[i-1] -> x[1-1]=x[0] -> 4

Output

8 12 9 7 1 5 4

Example

```
genre = ['pop', 'rock', 'jazz']
for i in range(len(genre)):
        print("I like",genre[i])
```

Output:

I like pop

I like rock

I like jazz

Example

```
x = [4,1,7,9,3]
z=len(x)
```

```
for i in range(z-1,-1,-1):
    print(i,"=","x[",i,"]","=",x[i])
```

Output:

```
4 = x[ 4 ] = 3
3 = x[ 3 ] = 9
2 = x[ 2 ] = 7
1 = x[ 1 ] = 1
0 = x[ 0 ] = 4
```

Example

```
x = [4,1,7,9,3]
z=len(x)
for i in range(z+1-2,1-2,-1):
    print(i,"=","x[",i,"]","=",x[i])
```

Output:

```
4 = x[ 4 ] = 3
3 = x[ 3 ] = 9
2 = x[ 2 ] = 7
1 = x[ 1 ] = 1
0 = x[ 0 ] = 4
```

Add Function

To add an item to the end of the list, use the append() method:

Example

Using the append() method to append an item:

```
thislist = ["apple", "banana", "cherry"]
thislist.append("orange")
print(thislist)
```

Output:

['apple', 'banana', 'cherry', 'orange']

Example

```
b = [[9,6],[4,5],[7,7]]
x = b[:2]
x.append(10)
print(x)
```

Output:

[[9, 6], [4, 5], 10]

Example

```
b = [[9,6],[4,8],[7,7]]
x = b[:2]
x[1].append(10)
print(x)
```

Output

[[9, 6], [4, 8, 10]]

Insert Function

Example

```
thislist = ["apple", "banana", "cherry"]

thislist.insert(1, "orange")

print(thislist)
```

Output:

```
['apple', 'orange', 'banana', 'cherry', 'orange']
```

Remove Function

There are several methods to remove items from a list:

Example

The remove() method removes the specified item:

```
thislist = ["apple", "banana", "cherry"]

thislist.remove("banana")

print(thislist)
```

Output:

```
['apple', 'cherry']
```

pop function

The pop() method removes the specified index, (or the last item if index is not specified):

```
thislist = ["apple", "banana", "cherry"]

thislist.pop()

print(thislist)
```

Output:

['apple', 'banana']

The del keyword removes the specified index:

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

del Keyword

del thislist[0]

print(thislist)

Output:

['banana', 'cherry']

Copy a List

You cannot copy a list simply by typing list2 = list1, because: list2 will only be

a reference to list1, and changes made in list1 will automatically also be made in list2.

There are ways to make a copy, one way is to use the built-in List method copy().

Change Item Value

To change the value of a specific item, refer to the index number:

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

thislist[1] = "blackcurrant"

print(thislist)

Output:

['apple', 'blackcurrent', 'cherry']

Enumerate() Function

Enumerate method comes with an automatic counter/index

to each of the items present in the Enumerate list in Python.

Advantages of using Enumerate

Here, are pros/benefits of using Enumerate in Python:

Enumerate allows you to loop through a list, tuple, dictionary, string, and gives the values along with the index.

To get index value using for-loop, you can make use of list.index(n). However, list.index(n) is very expensive as

it will traverse the for-loop twice. Enumerate is very helpful in such a case as it gives the index and items at one go.

Example

```
>>>mylist = ["a","b","c","d"]

>>>urlist = enumerate(mylist)

>>>print(list(urlist))
```

Output:

```
[(0, 'a'), (1, 'b'), (2, 'c'), (3, 'd')]
```

Example

```
l1 = ["eat", "sleep", "repeat"]

for ele in enumerate(l1):

        print (ele,end=" ")
```

Output:

```
(0, 'eat') (1, 'sleep') (2, 'repeat')
```

Format function

The format() method formats the specified value(s) and insert them inside the string's placeholder.

The placeholder is defined using curly brackets: {}. Read more about the placeholders in the Placeholder section below.

The format() method returns the formatted string.

Example

```
txt = "For only {price:.2f} dollars!"

print(txt.format(price = 49))
```

Output:

For only 49.00 dollars!

Example

```
presidents = ["Washington", "Adams", "Jefferson", "Madison", "Monroe", "Adams", "Jackson"]

for num, name in enumerate(presidents, start=1):

    print("President {}: {}".format(num, name))
```

Output:

President 1: Washington

President 2: Adams

President 3: Jefferson

President 4: Madison

President 5: Monroe

President 6: Adams

President 7: Jackson

sleep function

The sleep() function suspends (waits) execution of the current thread for a given number of seconds.

Python has a module named time which provides several useful functions to handle time-related tasks. One of the popular functions among them is sleep().

Example

```
import time

print("Start : %s" % time.ctime())

time.sleep( 1 )

print("End : %s" % time.ctime())
```

Output:

```
Start : Thu Jan 12 14:45:46 2023

End : Thu Jan 12 14:45:47 2023
```

Example

```
import time

list1=[1,2,3,4,5,6,7,8,9,10]

for i in list1:

    time.sleep(1)#sleep for 1 second

    print(i,end=", ")

#in Output i value will display in every one second from 1 to 10
```

Output:

```
1, 2, 3, 4, 5, 6, 7, 8, 9, 10
```

Example

```
import time

x="Gaurav"

for i in x:

        time.sleep(1)

        print(i,end=" ")
```

#In above loop x will transfer each character to i i.e, and i value will display in every one second : G a u r a v

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:

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

```
x=5

if x > 0:

    print("Microtek")       #True

else:

    print("College")        #False
```

Output:

Microtek

Example
if condition with elif

```
x=5
if x < 0:
    print("Microtek")
elif x > 0:
    print("College")
else:
    print("Gaurav")
```

Output:

College

Note : The and keyword is a logical operator, and is used to combine conditional statements:

Example

```
a=5

b=17

c=11

if a > b and a>c:

        print("a is the largest number.")

elif b > a and b > c:

        print("b is the largest number.")

else:

    print("c is largest value :")
```

Output:

b is the largest number

Example

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

if (num % 2) == 0:

   print("{0} is Even".format(num))

else:

   print("{0} is Odd".format(num))
```

Output:

Enter a number: 15

15 is Odd

if condition with or keyword

Or

The or keyword is a logical operator, and is used to combine conditional statements:

Example

Test if a is greater than b, OR if a is greater than c:

```
a = 200

b = 33

c = 500

if a > b or a > c:

  print("At least one of the conditions is True")
```

Example
At least one of the conditions is True

extend() Function

Use the extend() method to add list2 at the end of list1:

```
list1 = ["a", "b" , "c"]
list2 = [1, 2, 3]
list1.extend(list2)

print(list1)
```

Output:
['a', 'b', 'c', 1, 2, 3]

split() Function
Split a string into a list where each word is a list item and converts into list:

Example
```
string = "one,two,three"
words = string.split(',')
print(words)
```

Output:
['one', 'two', 'three']

Example
```
web_addr = ["address1.com", "address2.org", None, "address3.in"]
for addr in web_addr:
    if addr is not None:
        print(addr.split("."))
```

Output:
['address1', 'com']
['address2', 'org']
['address3', 'in']

Example
if condition with conditional statement for Even and Odd Value into list.

```
x=[4,8,13,15,9,2]
z=len(x)
for m in x:
    if m%2==False:
        print("Even",m)
    elif m%2!=False:
        print("odd",m)
```

Output:
Even 4
Even 8
odd 13

odd 15
odd 9
Even 2

Example

Fibonacci series

```
a = 0
b = 1
print(a,b,end=" ")
num = int(input("Enter any No.:"))
for i in range(1,num):
    c = a + b
    a = b
    b = c
    print(c,end=" ")
```

Solution :

Suppose num = 6:

i start at 1
1 : step > c = 0 + 1 = 1
a = 1
b = 1
print c --> 1

i start at 2
2 : step > c = 1 + 1 = 2
a = 1
b = 2
print c --> 1 2

i start at 3
3 : step > c = 1 + 2 = 3
a = 2
b = 3
print c --> 1 2 3

i start at 4
4 : step > c = 2 + 3 = 5
a = 3
b = 5
i start at 5
5: step > c = 3 + 5 = 8
a = 5
b = 8

print c --> 1 2 3 5 8

Output:
0 1 Enter any No.:6
1 2 3 5 8

String Slicing
fruit = "banana"
print(fruit[:3])

print(fruit[3:])

Solution

b	a	n	a	n	a
0	1	2	3	4	5

When we write [:3] means starting value start is null means by default it'll start from index position 0 here stop-1 is 3-1=2 position value print
ban

When we write [3:] means starting value start is null means by default it'll start from index position 3 here stop value is not assign
ana

Output:
ban
ana

While Loop

A while loop will always first check the condition before running. in programming language some time needs to repeat statement
so the repetition will allow with loop or without loop means copy & paste
If the condition evaluates to True then the loop will run the code within the loop's body.
So, suppose u want to print "Hello world" 5 times

Example

```
i = 1
while i<=5:
    print("Hello World:")
    i +=1
```

Output:

```
Hello World:
Hello World:
Hello World:
Hello World:
Hello World:
```

Example

```
i = 1
while i < 6:
    print(i,end=" ")
    i = i+1
```

Output:

```
1 2 3 4 5
```

Example

```
n = 5
while n > 0:
    n -= 1
    if n == 2:
        break
    print(n)
print('Loop ended.')
```

Output:

4

3

Loop ended.

Example to display string through while loop

```
x="MICROTEK COLLEGE"

i=0
while i < len(x):
    print(x[i],end=" ")
    i += 1
```

Output:

M I C R O T E K C O L L E G E

Example

While loop using pop method

```
a = [1, 2, 3, 4]
while a:
    print(a.pop(),end=" ")
print(a)
```

Output:

4 3 2 1 []

Example

```
x="MICROTEK"
z=len(x)

while z > 0:
    print(x[z-1],end=" ")
    z-=1
```

Solution:

M	I	C	R	O	T	E	K
1	2	3	4	5	6	7	8
0	1	2	3	4	5	6	7

Here z is stored length of x i.e, z=8 again while condition check.
z > 0 which means 8 > 0 it is printing (x[z-1],end=" ") x[8-1] = x[7] first character will print ‘K’ again z-=1 or z=z-1 now z=7 again it check condition in while loop z > 0 which means 7 > 0 it is printing (x[z-1],end=" ") x[7-1] = x[6] second character will print ‘E’ again z-=1 or z=z-1 now z=5 again it check condition in while loop. repeat the same procedure when z=1 z > 0 printing (x[1-1]) x[0] character will print ‘M’ when z-=z-1 or z=z-1 z=0 again z check condition in loop z > 0 which means 0 > 0 and condition will become False loop will be exit..

Output:
K E T O R C I M

While loop using append method
Example

```
s=list()
i=1
while i < 5:
    val=int(input("enter the value :"))
    s.append(val)
    i +=1
print(s,end=" ")
```

Output:

```
enter the value :5
enter the value :2
enter the value :3
enter the value :7
[5, 2, 3, 7]
```

Example

Write a program to take input value and from input value display table.

```
a=int(input("enter table number"))
i=1
num = 0
while i<=10:
    num = a * i
    print(a,"x",i,"=",num)
    i=i+1
```

Output

```
enter table number2
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
2 x 10 = 20
```

Example

Write a program which takes three input and display into list and display total and average value

```
c=[]
i=0
m=0
sum=0
avg=0
while i < 3:
    val=int(input("Enter the Value.."))
    c.append(val)
    i +=1
print(c)
z=len(c)
```

```
i=0
while i < z:
    sum +=c[i]
    avg=sum/z
    i+=1
print("Total Value :",sum)
print("Average :",avg)
```

Output:

```
Enter the Value..4
Enter the Value..2
Enter the Value..6
[4, 2, 6]
Total Value : 12
Average : 4.0
```

Nested for loop
Example 1

```
for i in range(4):
    print("\n")
    for j in range(4):
        print("#",end=" ")
```

Output:

```
# # # #

# # # #

# # # #

# # # #
```

Example 2

```
for i in range(4):
    print("\n")
    for j in range(i+1):
        print("#",end=" ")
```

Output:

```
#

# #

# # #

# # # #
```

Example 3

```
for i in range(4):
    print("\n")
    for j in range(4-i):
        print("#",end=" ")
```

Output:

```
# # # #

# # #

# #

#
```

Example

```
n = int(input("Enter any Value:"))
for i in range(n):
    print("\n")
    for j in range(i+1):
        print(j+1,end=" ")
```

suppose n = 4

Output:

1

1 2

1 2 3

1 2 3 4

Example

```
for i in range(n):
    print()
    for j in range(n-i-1):
        print(" ",end=" ")

    for j in range(i+1):
        print(j+1,end=" ")
```

Output:

```
      1
    1 2
  1 2 3
1 2 3 4
```

Example

Write a program to check value number is Prime or Not Prime.

```
num=12
temp=0
for i in range(2,num):
    if num%i==0:
        temp=1
        break
```

```
if temp == 1:
    print("given number is not prime")
else:
    print("given number is prime")
```

Output:

given number is not prime

Nested While Loop
Example

```
i = 0
while i < 4 :
    j=0
    while  j < 4 :
        print("*", end=" ")
        j += 1
    print()
    i += 1
```

Output:

```
* * * *
* * * *
* * * *
* * * *
```

```
i = 0
```

Example

```
while i < 4 :
    j=0
    while  j < i+1 :
        print("*", end=" ")
        j += 1
    print()
    i += 1
```

Output:

```
*
* *
* * *
* * * *
```

Example

```
i = 1
while i <= 5:
    print()
    print("Microtek ",end=" ")
    i=i+1
    j = 1

    while j <= 4:
        print("College ",end=" ")
        j=j+1
```

Output:

```
Microtek  College  College  College  College
Microtek  College  College  College  College
Microtek  College  College  College  College
Microtek  College  College  College  College
Microtek  College  College  College  College
```

Dictionary :

Dictionary in Python is a collection of keys values, used to store data values like a map. Which, unlike other data types which hold only a single value as fan element.
Dictionary holds pairs of values, one being the Key and the other corresponding pair element being its Key **:** Value. Values in a dictionary can be of any data type and can be duplicated, whereas keys can't be repeated and must be immutable.

Note – Dictionary keys are case sensitive, the same name but different cases of Key will be treated distinctly.

```
student_age = {'Jack': 25, 'Ritika': 15, 'Jack' : 28}
print(student_age)
```

Output:
{'Jack': 28, 'Ritika': 15}

Example

```
x={"A":98,"B":5,"C":7,"D":4}
print(x)
```

Output:
{'A': 98, 'B': 5, 'C': 7, 'D': 4}

Example

```
print(x["C"])
```

Output:
7

get() Method in dictionary

The get() method returns the value of the item with the specified key.

```
x={"A":98,"B":5,"C":7,"D":4}
print(x.get("B"))
```

Output:
5

Example

```
my_dict = {'name': 'Jack', 'age': 26}
print(my_dict['name'])
```

Output:
Jack

zip function:
The zip() function returns a zip object, first item in each passed iterator is paired together,
and then the second item in each passed iterator are paired together etc.

Example

```
keys = ["Navin","Kiran","Harsh"]
values = ["Python","Java","JS"]

data = dict(zip(keys,values))
print(data)
```

Output:

```
{'Navin': 'Python', 'Kiran': 'Java', 'Harsh': 'JS'}
```

Example
Add element in Dictionary

```
data["manoj"]="CS"
print(data)
```

Output:

```
{'Navin': 'Python', 'Kiran': 'Java', 'Harsh': 'JS', 'manoj': 'CS'}
```

Example

```
prog =
{'JS':'manoj','CS':'VS','python':['pycharm','jupiter'],'java':{'JS':'VS','JSP':'Eclip
se'}}
print(prog)
```

Output:

```
{'JS': 'manoj', 'CS': 'VS', 'python': ['pycharm', 'jupiter'], 'java': {'JS': 'VS',
'JSP': 'Eclipse'}}
```

Example

```
prog =
{'JS':'manoj','CS':'VS','python':['pycharm','jupiter'],'java':{'JS':'VS','JSP':'Eclip
se'}}

print(prog['python'])
```

Output:

```
['pycharm', 'jupiter']
```

Example

Dictionary program with for loop

```
dicts = {}
keys = range(4)
values = ["A", "B", "C", "D"]
for i in keys:
    dicts[i] = values[i]
print(dicts)
```

Output:

```
{0: ‘A’, 1: ‘B’, 2: ‘C’, 3: ‘D’}
```

Values function in Dictionary

```
salary = {"raj" : 50000, "striver" : 60000, "vikram" : 5000}

# stores the salaries only

list1 = salary.values()
print(list1)
```

Output:

```
dict_values([50000, 60000, 5000])
```

keys Function with dictionary
Example

```
d = {'x': 1, 'y': 2, 'z': 3}
print(list(d))
print("\n")
print(d.keys())
```

Output:
['x', 'y', 'z']

Python User-defined Functions
In this tutorial, you will find the advantages of using user-defined functions and best practices to follow.

What are user-defined functions in Python?
Functions that we define ourselves to do certain specific task are referred as user-defined functions. The way in which we define and call functions in Python are already discussed.

Functions that readily come with Python are called built-in functions. If we use functions written by others in the form of library, it can be termed as library functions.

Creating a Function

We can create a Python function using the def keyword.

Example: Python Creating Function

```
# Declaring a function

def fun():
  print("Welcome to Microtek")
fun() #this is prototype name of function declaration and while it is calling function
```

Output:
Welcome to Microtek

Parameterized Function
The function may take arguments(s) also called parameters as input within the opening and closing parentheses, just after the function name followed by a colon.
Syntax:

```
def function_name(argument1, argument2, ...):
    statements
    .
    .
```

Example:

```
# Python program to
# demonstrate functions

# A simple Python function to check
# whether x is even or odd

def evenOdd( x ):
    if (x % 2 == 0):
        print("even")
    else:
        print("odd")

# Driver code
evenOdd(2)
evenOdd(3)
```

Output:
even
odd

Example

```
def add(a,b):
    c=a+b
    print(c)

add(10,23)
```

Actual Argument is Four Types :

1. Position
2. Keyword
3. Default
4. Variable Length

Position:

```
def persone(name,age):
    print(name)
    print(age)

persone("amit",23)
```

#Here in function is assign variable name and age and position of variable's value is calling in function: here position is maintained in sequence. but what is sequence is not allowed..

Output:

```
amit
23
```

Keyword Argument Function:

```
def persone(name,age):
    print(name)
    print(age)

persone(age=23,name="amit")
```

Output:
amit
23

Default argument:

```
def persone(name,age):
    print(name)
    print(age)

persone("manoj")
```

In above program the persone function is expectation two argument but it is calling only one argument which will be error.. So in this case we can declare next variable's value..

```
def persone(name,age=22):
    print(name)
    print(age)

persone("manoj")

#so in above program if we're not passing argument then actual function will assign default value in age variable.
```

Output:
manoj
22

Variable length :

It may need to process a function for more arguments than you specified while defining the function. These arguments are called variable-length arguments and are not named in the function definition, unlike required and default arguments. An asterisk (*) is placed before the variable name that holds the values of all nonkeyword variable arguments. This tuple remains empty if no additional arguments are specified during the function call.

```
def add(a,*b):
    c=a
    for i in b:
        c=c+i
    print(c)
add(5,6,34,78)
```

Output:

123

Example

```
def calculateTotalSum(*arguments):
    totalSum = 0
    for number in arguments:
        totalSum += number
    print(totalSum)

# function call
calculateTotalSum(5,4,3,2,1)
```

Output:

15

Example

```
def getstudent(**args):
    for key,value in args.items():
        print("%s=%s" %(key,value))

getstudent(naveen=54,suresh=62,manish=73)
```

Output:

naveen=54
suresh=62
manish=73

Example

Write a program in function, which return the value

```
def addsum(x, y):
    res = x + y
    return res
num1 = int(input("enter First No. :"))
num2 = int(input("enter Two No. :"))

m = addsum(num1, num2)
print("the sum of two ",m)
```

Output:

```
enter First No. :16
enter Two No. :25
the sum of two  41
```

Example

Function in Factorial

```
def fact(x):
    i = 1
    fact = 1
    while i <= x:
        fact = fact * i
        print(fact,end=" ")
        i +=1
m = int(input("enter any Number :"))
fact(m)
```

Output:

```
enter any Number :6
1 2 6 24 120 720
```

Example

Write a program which takes input from user and display the value is Prime or Not Prime

```
def prime(num):
    count=0
    for i in range(2,num):
        if num%i==0:
            print("not prime.")
            break
    else:
        print("Prime..")

x=int(input("Enter value:"))
prime(x)
```

Output:

```
Enter value:19
Prime..
```

Example

Write a program which takes input from user and display the series of prime number of input value.

```
def check(num):
    for i in range(2,num):
        if num%i==0:
            break
    else:
        print(num,end=" ")

x=int(input("Enter value:"))
for m in range(2,x):
    check(m)
```

Output:

```
Enter value:10
2 3 5 7
```

Array

An array is a collection of items stored at contiguous memory locations.
The idea is to store multiple items of the same type together.
In python the size of array is not fixed. Array can increase or decrease their size dynamically.
Array and List are not same. Python arrays store homogenous values only

Note : Python does not Multi Dimensional Array But we can create Multi Dimensional Array Using Third Party packages like numpy

```
# Python program to demonstrate
# Creation of Array

# importing "array" for array creations
import array as arr

# creating an array with integer type
a = arr.array('i', [1, 2, 3])

# printing original array
print("The new created array is : ", end =" ")
for i in range (0, 3):
    print (a[i], end =" ")
```

Array :

```
import array
std_roll = array.array('i',[101,102,103,104,105])
print(std_roll[0])
```

Output :

101

Backward Index :

```
import array as arr
std_roll = arr.array('i',[101,102,103,104,105])
```

```
print(std_roll[-2])
print(std_roll[-5])
```

Output:

```
104
101
```

Remove Array Element

The del keyword is used for removing the entire object from the memory location as well as delete
any specific element

```
import array as arr
std_roll = arr.array('i',[101,102,103,104,105])
print(std_roll)
del std_roll[1]
print("after delete array ",std_roll)
```

Output:

```
array('i', [101, 102, 103, 104, 105])
after delete array  array('i', [101, 103, 104, 105])
```

Using the pop() method:

```
import array as arr
std_roll = arr.array('i',[101,102,103,104,105])
print(std_roll)
std_roll.pop()
print("after delete array ",std_roll)
```

Output:

```
array('i', [101, 102, 103, 104, 105])
after delete array  array('i', [101, 102, 103, 105])
```

using the remove() method:

```
import array as arr
std_roll = arr.array('i',[101,102,103,104,105])
print(std_roll)
std_roll.remove(103)
print("after delete array ",std_roll)
```

Output:

```
array('i', [101, 102, 103, 104, 105])
after delete array  array('i', [101, 102, 104, 105])
```

Using setdiscard() method

The setdiscard() method is another useful method for removing elements from the array.
But, for using it, we have to convert the array object into a set object using the set() function.

```
import array as arr
std_roll = arr.array('i',[101,102,103,104,105])

s = set(std_roll)
print("original Array :",s)
s.discard(104)
print("After removing :",s)
```

Output:

```
original Array : {101, 102, 103, 104, 105}
After removing : {101, 102, 103, 105}
```

Using the Lambda function along with the filter()

```
import array as arr
std_roll = arr.array('i',[101,102,103,104,105])
print("Before Remove :" + str(std_roll))
std_roll = list(filter(lambda x: x!=102,std_roll))
print("after Remove array " + str(std_roll))
```

Output:
Before Remove :array('i', [101, 102, 103, 104, 105])
after Remove array [101, 103, 104, 105]

without Index

```
import array
std_roll = array.array('i',[101,102,103,104,105])
for x in std_roll:
        print(x,end=" ")
```

Output:
101 102 103 104 105

With Index

```
import array as arr
std_roll = arr.array('i',[101,102,103,104,105])
n = len(std_roll)
for i in range(n):
    print(std_roll[i],end=" ")
```

Output:
101 102 103 104 105

Slicing on Arrays
Slicing on arrays can be used to retrieve a piece of the array that can be group of elements. Slicing is useful to retrieve a range of element

```
import array as arr
std_roll = arr.array('i',[101,102,103,104,105])
n = len(std_roll)
for i in range(n):
    print(i,"=",std_roll[i])
```

```
print("--Slicing--")
a = std_roll[1:3]
for i in a:
    print(i,end=" ")
```

Output:

```
0 = 101
1 = 102
2 = 103
3 = 104
4 = 105
--Slicing--
102 103
```

Example

Program in User input the integer Value in Array :

```
import array as arr
std_roll = arr.array('i',[])
n = int(input("Enter any size of Array :"))
for i in range(n):
    val = int(input("\nEnter Value :"))
    std_roll.append(val)
    print(end=" ")

    for i in std_roll:
        print(i,end=" ")
```

Output:

```
Enter any size of Array :5

Enter Value :1
 1
Enter Value :2
 1 2
```

```
Enter Value :3
 1 2 3
Enter Value :4
 1 2 3 4
Enter Value :5
 1 2 3 4 5
```

#Array form data from user input and display into list

```
import array as arr
def funcarr(x):
    c=[]
    std_roll = arr.array('i',[])
    for i in range(x):
        val = int(input("Enter the Value."))
        std_roll.append(val)
        print(" ")
    for i in std_roll:
        c.append(i)
    print(c)

n = int(input("Enter any size of Array :"))
funcarr(n)
```

Output:

```
Enter any size of Array :4
Enter the Value.2

Enter the Value.6

Enter the Value.1

Enter the Value.7

[2, 6, 1, 7]
```

Program of sum of array.

```
import array as arr
def funcarr(x):
    sum = 0
    std_roll = arr.array('i',[])
    for i in range(x):
        val = int(input("\nEnter Value :"))
        std_roll.append(val)
        print(end=" ")
        for i in std_roll:
            print(i,end=" ")
    x= std_roll
    print("\n")
    m=len(x)
    print(m)
    for j in range(m):
        print("\n")
        print("Value are : ",std_roll[j])
        sum = sum + std_roll[j]
        print("\n")
        print("total value are :",sum)


n = int(input("Enter any size of Array :"))
funcarr(n)
```

Output:

```
Enter any size of Array :4

Enter Value :3
 3
Enter Value :2
 3 2
Enter Value :1
 3 2 1
```

```
Enter Value :4
 3 2 1 4
4


Value are :  3

total value are : 3

Value are :  2

total value are : 5

Value are :  1

total value are : 6

Value are :  4

total value are : 10
```

Program from Array to List in string type:

```
import array as arr

def check(n):
    dicts={}
    m = arr.array('i',[])
    c=[]
    for i in range(n):
        val=int(input("Enter the Value" ))
        m.append(val)
    #print(m)
    for t in m:
        c.append(t)
    print(c)
#    z=list(map(str,list(c)))
```

```
    g=[str(k) for k in list(c)]
    print(g)

x=int(input("Enter the Array Size "))
check(x)
```

Output:

```
Enter the Array Size 4
Enter the Value8
Enter the Value6
Enter the Value5
Enter the Value3
[8, 6, 5, 3]
['8', '6', '5', '3']
```

Numpy

Numpy is a Python package which stands for "Numerical Python". This was created by
Travis Oliphant.
Numpy is the core library for scientific computing in Python.
It provides a high performance multidimensional Array object, and tools for working with these Array.
NumPy is often called as an alternate for MatLab(a programming Plateform designed specifically for
engineers and scientists for Data Analysis etc. Developing Algorithms, create Model and Application
etc.

A[1,0], A[1,1], A[1,2] --> 4 5 6

A[2,0], A[2,1], A[2,2] --> 7 8 9

Working with NumPy :

You need to create a Folder (Create a Folder in Desktop named : "mynumpy") and create our all files inside it. now go to Anaconda, Framework and select your directory and Type the following
Command as your First Statement :

import numpy as np And run it. if run successfully without any Error, Numpy is successfully installed in your computer.
if any error occurs. then open Anaconda Prompt and write a command :
conda install numpy

Work with Jupyter Notebook

Jupyter Notebook has two Mode :
1- Command Mode - Blue Color
2- Edit Mode - Green Color

Shortcuts to Remember :
1- To Run program You have written Press : Cnt + Enter.
2- To Run Command as Well as Insert a Cell Below : Shift + Enter.
3- To Insert a Cell below Just Press "b"
To Delete a Cell - Press Esc button (to go to Command Mode as in Edit mode You can not delete the Cell) and then "d" two times

Why should we Use?
We use python NumPy Array instead of a list because of the below three reasons:

Less Memory usage
Fast performance
Convenient to Work

The first reason to prefer python NumPy arrays is that it takes less memory than the python list.
Then, it is fast in terms of execution, and at the same time, it is convenient and easy to work with it.

```
a = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
print(a)
```

Output:

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
```

Example

```
import numpy as np
arr = np.array([5,2,4,9,1])
arr = arr + 5
print(arr)
```

Output:

```
[10  7  9 14  6]
```

Example

Insert the list value into the position value

```
from array import *
x = [[11, 12, 5, 2], [15, 6,10], [10, 8, 12, 5], [12,15,8,6]]

x.insert(2, [0,5,11,13,6])

for r in x:
   for c in r:
      print(c,end = " ")
   print()
```

Output:

```
11 12 5 2
15 6 10
0 5 11 13 6
10 8 12 5
12 15 8 6
```

Example

Updating Values

We can update the entire inner array or some specific data elements of the inner array by reassigning the values using the array index.

```
from array import *

T = [[11, 12, 5, 2], [15, 6,10], [10, 8, 12, 5], [12,15,8,6]]

T[2] = [11,9]
T[0][3] = 7
for r in T:
   for c in r:
      print(c,end = " ")
   print()
```

Output:

```
11 12 5 7
15 6 10
11 9
12 15 8 6
```

Deleting the Values

We can delete the entire inner array or some specific data elements of the inner array by reassigning the values using the del() method with index. But in case you need to remove specific data elements in one of the inner arrays, then use the update process described above.

Example

```
from array import *
T = [[11, 12, 5, 2], [15, 6,10], [10, 8, 12, 5], [12,15,8,6]]
del T[3]
for r in T:
   for c in r:
      print(c,end = " ")
   print()
```

Output:
When the above code is executed, it produces the following result −

```
11 12 5 2
15 6 10
10 8 12 5
```

Example
Addition of two array

```
import numpy as np
arr1 = np.array([[1, 2, 3, 4]])
arr2 = np.array([[5, 6, 9, 8]])
arr3 = arr1 + arr2
print(arr3)
```

Output:

```
[[ 6  8 12 12]]
```

Example
Display the square root in array

```
from numpy import *
#import numpy as np
arr1 = np.array([5,2,4,7,1])
print(sqrt(arr1))
```

Output

```
[2.23606798 1.41421356 2.         2.64575131 1.        ]
```

Python – Itertools.Product()

itertools.product() is used to find the cartesian product from the given iterator, output is lexicographic ordered. The itertools.product() can used in two different ways:

itertools.product(*iterables, repeat=1):
It returns the cartesian product of the provided iterable with itself for the number of times specified by the optional keyword “repeat”. For example, product(arr, repeat=3) means the same as product(arr, arr, arr).
itertools.product(*iterables):
It returns the cartesian product of all the iterable provided as the argument. For example, product(arr1, arr2, arr3).

Example

```
from itertools import product
import numpy as np
np.array(list(product(range(3), range(3))))
```

Output:

```
array([[0, 0],
       [0, 1],
       [0, 2],
       [1, 0],
       [1, 1],
       [1, 2],
       [2, 0],
       [2, 1],
       [2, 2]])
```

Example

```
import numpy as np
np.array(list(product(range(1,3), range(1,3))))
```

Output:

```
array([[1, 1],
       [1, 2],
       [2, 1],
       [2, 2]])
```

Example

```
from itertools import product
def cartesian_product(arr1, arr2):

    # return the list of all the computed tuple
    # using the product() method
    return list(product(arr1, arr2))

# Driver Function
if __name__ == "__main__":
    arr1 = [1, 2, 3]
    arr2 = [5, 6, 7]
    print(cartesian_product(arr1, arr2))
```

Output:

```
[(1, 5), (1, 6), (1, 7), (2, 5), (2, 6), (2, 7), (3, 5), (3, 6), (3, 7)]
```

One of the most beneficial but straightforward indexing routines in NumPy is the npindex(). This routine provides us iterator that returns the indices of elements in an N-dimensional array.

Example

```
import numpy as np
for m in np.ndindex(3,2):
    print(m)
```

Output:

```
(0, 0)
(0, 1)
(1, 0)
(1, 1)
(2, 0)
(2, 1)
```

In this case, we use the ndindex function to get the index of the elements in an array of shapes (3,2).

Shape of an Array

NumPy arrays have an attribute called shape that returns a tuple with each index having the number of corresponding elements.

Example

```
import numpy as np
arr = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])
print(arr.shape)
```

Output:

```
(2, 4)
```

Example

```
import numpy as np
arr =  np.array([[1,2,3], [4,5,6]])
for m in np.ndindex((arr.shape)):
    print(m)
```

Output:

```
(0, 0)
(0, 1)
(0, 2)
(1, 0)
(1, 1)
(1, 2)
```

Reshaping arrays

Reshaping means changing the shape of an array.

The shape of an array is the number of elements in each dimension.

By reshaping we can add or remove dimensions or change number of elements in each dimension.

Reshape From 1-D to 2-D

Convert the following 1-D array with 12 elements into a 2-D array.

Example
The outermost dimension will have 4 arrays, each with 3 elements:

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])
x = arr.reshape(4, 3)
print(x)
```

Output:
```
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
```

ndmin
Specifies minimum dimensions of resultant array

Example
Create an array with 5 dimensions using ndmin using a vector with values 1,2,3,4 and verify that last dimension has value 4:

```
import numpy as np
arr = np.array([1, 2, 3, 4], ndmin=5)
print(arr)
print('shape of array :', arr.shape)
```

Output:
```
[[[[[1 2 3 4]]]]]
shape of array : (1, 1, 1, 1, 4)
```

Example
Insert the value for 2D Array in the form of Matrix through function

```
import numpy as np
def funcarr(row,col):
    matrix=[]
```

```
    for i in range(row):
        a=[]
        for j in range(col):
            val = int(input("Enter the Number: "))
            a.append(val)
        matrix.append(a)
        arr=np.array(matrix)
        #print(arr)
    m=[]
    for i in range(row):
        c=[]
        for j in range(col):
            #print(arr[i][j],end=" ")
            c.append(arr[i][j])
        m.append(c)
    print(m)

x = int(input("Enter the Row Number: "))
z = int(input("Enter the Column Number: "))
funcarr(x,z)
```

Output:

```
Enter the Row Number: 2
Enter the Column Number: 2
Enter the Number: 1
Enter the Number: 2
Enter the Number: 3
Enter the Number: 4
[[1, 2], [3, 4]]
```

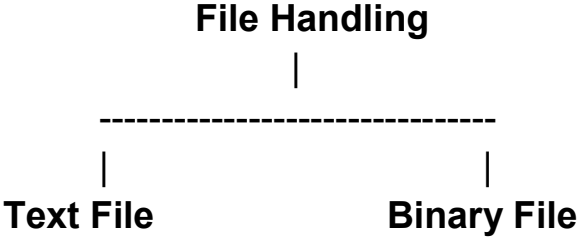
File Handling
till we've learn so many program in Python
whenever we execute any Program then output comes but when we close the Program

Output doesn't store any where. so for this reasone we need File handling for save the records as like output.

File Handling

So, Program's Output can be Saved in two Method :
1> File Handling
2> Database

Python too supports file handling and allows users to handle files i.e., to read and write files, along with
many other file handling options, to operate on files. The concept of file handling has stretched over various other languages,
but the implementation is either complicated or lengthy, but like other concepts of Python, this concept here is also easy and short.
Python treats files differently as text or binary and this is important.



Text File : Ram is a Good Boy. for new line print("\n)

* Text file is slower than Binary File because it takes command
so translation takes place.

Binary File : Ram is a Good Boy.("\n)
so it will not take print command for new line.

Translation doesn't happen, so this is faster execution.

Where the following mode is supported:
r: open an existing file for a read operation.
w: open an existing file for a write operation. If the file already contains some data then it will be overridden.
a: open an existing file for append operation. It will not override existing data.

r+: To read and write data into the file. The previous data in the file will be overridden.
w+: To write and read data. It will override existing data.
a+: To append and read data from the file. It won't override existing data.

```
#a file named "geek", will be opened with the reading mode.
file = open('geek.txt', 'r')
# This will print every line one by one in the file
for each in file:
    print (each)
```

The open command will open the file in the read mode and the for loop will print each line present in the file.

Working of open() function

Before performing any operation on the file like reading or writing, first, we have to open that file.
For this, we should use Python's inbuilt function open() but at the time of opening, we have to specify
the mode, which represents the purpose of the opening file.

```
f = open(filename, mode)
```

This will create a File file.txt [notepad file] in the root directory.

```
fobj = open("std.txt","w")
```

store any data in file

```
fobj.write("Hello World")
```

Save in IDLE, Run Program
so in std.txt file a data display : Hello World

again type any data in text file

```
fobj.write("I'm Gaurav")
```

```
fobj.close()
```

Save file and Run the Program

again add new line in a text File.

```
fobj = open("std.txt","w")
fobj.write("This is Microtek")
fobj.close()
```

Save file and Run the Program
Now This is Microtek will be added inplace of previous data.

Change the Path of File

```
fobj=open("D:\\aspnet\\std.txt","w")
fobj.write("This is Gaurav")
fobj.close()
```

std.txt will be create a copy in the path : D:\aspnet

for using single backward slash symbol then add "r" before path

```
fobj=open(r"D:\aspnet\mash.txt","w")
fobj.write("Varanasi Microtek")
fobj.close()
```

Type the content in output file and check file after type

```
fobj=open("chage.txt","w")
s=input("enter String= ")
fobj.write(s)
fobj.close()
```

Output:
enter String= Hi. this is from Microtek.

Press Enter

-Again Open change.txt >
Hi. this is from Microtek

Byte Read in Text File

```
fobj = open("std.txt","r")
val = fobj.read(12)
print(val)
```

Output:

This is Micr

in File : hello.txt stored data which is shown below :

```
kjl
ygf
119
shlok
120
naman
mahesh
```

```
fobj=open("D:\\CheckRes\\hello.txt","r")
x=fobj.read(5)
print(str(x))
fobj.close()
```

Output:
kjl
y

Example

```
def writefile():

    with open("D:\\CheckRes\\ayush.txt","w") as f1 :
        for i in range(5) :
            sqr = i*i
            f1.write(str(sqr)+" ")
        f1.close()

def myread():
    writefile()
    with open("D:\\CheckRes\\ayush.txt","r") as f1 :
        x=f1.read(3)
        print(x,end=" ")
    f1.close()

myread()
```

Output:

```
0 1
```

Example

```
fobj = open("std.txt","r")
val = fobj.read()
print(val)
```

Output:

```
This is Microtek Hi i'm going to Microtek
```

Read line at a time.

```
fobj=open("D:\\aspnet\\std.txt","r")
val=fobj.readline()
print(val)
val=fobj.readline()
print(val)
fobj.close()
```

Output:

This is Gaurav
This is from Microtek

Example

```
fobj=open("chage.txt","r")
while str:
   str = fobj.readline()
   print(str)
fobj.close()
```

Output:

101,ram,87.0

102,jai,65.0

103,manoj,81.0

104,alok,49.0

105,nishant,43.0

```
fobj=open("D:\\aspnet\\std.txt","w")

n = int(input("How many Line:="))
for i in range(n):
       s=input("Enter the String :")
       fobj.write(s+"\n")
fobj.close()
```

After file creation in Path. directory

```
fobj=open("C:\\Users\\Gaurav\\AppData\\Local\\Programs\\Python\\Python3
10\\chage.txt","w")
n = int(input("How many Line:="))
x=[]
for i in range(n):
    s=input("Enter the String :")
    x.append(s+"\n")
fobj.writelines(x)
fobj.close()
```

After file creation in Path. directory

```
fobj=open("chage.txt","w")
n = int(input("How many Line:="))
x=[]
for i in range(n):
    s=input("Enter the String :")
    x.append(s+"\n")
fobj.writelines(x)
fobj.close()
```

Output:

```
How many Line:=3
Enter the String :Gaurav
Enter the String :singh
Enter the String :Microtek
```

File Handling with function :

```
def newwrite():
    #global fobj

    fobj=open("D:\\CheckRes\\hello.txt","w")
    n=int(input("Enter The Value :"))
    for i in range(n):
        s=input("enter the String :")
```

```
        fobj.write(s+"\n")
    fobj.close()

def myread():
    fobj=open("D:\\CheckRes\\hello.txt","r")
    x=fobj.read()
    print(x)
    fobj.close()


newwrite()
myread()
```

Output:

```
Enter The Value :3
enter the String :manoj kumar
enter the String :ajay pandey
enter the String :harshit mishra
manoj kumar
ajay pandey
harshit mishra
```

CSV File format

```
101,Amit,55.5
102,Mayank,68
103,Harshit,82
```

Example

```
size = int(input("How many Student.? "))
fobj=open("chage.txt","w")
for i in range(size):
    roll=int(input("Enter the Roll Number :"))
    name=input("Enter the Name :")
    marks=float(input("Enter the Marks :"))
    rec=str(roll)+","+name+","+str(marks)+"\n"
    fobj.write(rec)
fobj.close()
```

Output:

```
How many Student.? 2
Enter the Roll Number 101
Enter the Name :suraj
Enter the Marks :74
Enter the Roll Number 102
Enter the Name :aman
Enter the Marks :63
```

after the input all data check chage.txt file.

Example

Write a program which increment the value of variable and multiplies of its index value of 5

```
def writefile():

    with open("D:\\CheckRes\\ayush.txt","w") as f1 :
        for i in range(5) :
            sqr = i*i
            f1.write(str(sqr) + "\n")
        f1.close()

def myread():
    writefile()
    with open("D:\\CheckRes\\ayush.txt","r") as f1 :
        x=f1.read()
        print(x)
    f1.close()

myread()
```

Output:

```
0
1
4
9
16
```

tell() Function

Definition and Usage

The tell() method returns the current file position in a file stream.

```
fobj=open("chage.txt","r")
str=fobj.read()
print(str)
print("innitally the position is : ", fobj.tell())
```

Output:

```
101,shukla,65.0
102,mohit,78.0
103,sanjay,64.0
104,Kamal,66.0
105,Radha,70.0

initially the position is :  82
```

Seek() Function

In Python, seek() function is used to change the position of the File Handle to a given specific position.
File handle is like a cursor, which defines from where the data has to be read or written in the file.

Pass

Pass statement is used to do nothing. it can be used inside a loop or if statement to represent
no operation. Pass is useful when we need statement syntactically correct we do any operation.

```
if (Condition):---> if this condition true
        pass -----> Do not perform any operation
else:
        statements -----> So only this will execute when condition is false.
```

Example:

```
a = 33
b = 200

if b > a:
  pass
```

Output:

No output

strip()

returns a copy of the string with both leading and trailing characters removed

```
string = "  Varanasi Kashi    "

# prints the string without stripping
print(string)

# prints the string by removing leading and trailing whitespaces
print(string.strip())
```

Output:

```
  Varanasi Kashi
Varanasi Kashi
```

Example

Read Text File and result comes with every word into list in string type.

```
fh=open("yes.txt","r")
for line in fh:
    word = line.split()
    print(word,end=" ")
```

Solution
Here in create a text/notepad file and saved in python default path write a something like I've written as given below.
First of all, what is a CSV ?

Output:
['First', 'of', 'all,', 'what', 'is', 'a', 'CSV', '?']

Binary File in File handling
When we create binary file then we need two term
1> **Pickling**
2> **Unpickling**

Pickling :> Data Structure : List/
Dictionary -> convert in Byte Stram
(Binary), then write it, then it'll save in file.

Pickling
Strauctur e----------->Byte Stream

Unpickling

Unpickling : this is opposite of pickling means if you've done write then
Byte Stram > then Data Strauctur e.

Unpickling
Byte Stream----------> Structure

So, in Binary File : we have to create Pickle Module

For storing
b = pickle.dumps(db)

 # For loading
myEntry = pickle.loads(b)

Dump()
Write the content into Binary file is used Dump()

pickle.dump(structure,fileobject)

load()
Reads the contents of Binary file is used load()

Structure Var. = pickle load(fileobject)

.dat File :

A DAT file is a generic data file.
Most can be opened with a text editor like Notepad++.
Use that program to convert one to CSV, HTML, or other text formats.

Sometimes you'll find them by themselves but often they're
with other configuration files like DLL files.

Example
Create a binary file and store the data into list format

```
import pickle
f=open("myfile.dat","wb")

def write():
        f=open("myfile.dat","wb")
        lst=['abc','mnp','pqr','xyz']
        pickle.dump(lst,f)
        f.close()

write()
print("Data Saved..")
```

Example

```
import pickle

f=open("myfile.dat","wb")

def write():
        f=open("myfile.dat","wb")
        lst=['abc','mnp','pqr','xyz']
        pickle.dump(lst,f)
        f.close()

def readfile():
    f=open("myfile.dat","rb")
    list1=pickle.load(f)
    print(list1)
    f.close()


write()
readfile()
```

Example

Create a binary file and store data and read and display the data from binary file.

```
def write():
    lst=""
    f=open("myfile.dat","wb")
    lst="microtek courses"
    pickle.dump(lst,f)
    f.close()
def readfile():
    f=open("myfile.dat","rb")
    list1=pickle.load(f)
    print(list1)
    f.close()
```

write()
readfile()

1. **try and except block with while Statement**

2. **While True: --> it'll become False when end of file is reached EOF exception)** except EOFError:

Using " **with** " Statement
The with statement is a concept statement which combines
the opening of file and processing
of file along with inbuilt exception handling.

Example
Program of list data input in Binary File.

```
import pickle
f=open("myfile.dat","wb")

def write():
    f=open
    c=[]
    for m in range(3):
        n = input("Enter the String: ")
        c.append(n)
        pickle.dump(c,f)
    f.close()

def readfile():
    m=[]
    f=open("myfile.dat","rb")
    try:
        print("File m.dat stores data: ")
        while True:
            m=pickle.load(f)
```

```
            print(m)
    except EOFError:
        f.close()
write()
readfile()
```

Output:

```
Enter the String: jai
Enter the String: kamal
Enter the String: sumanth
File m.dat stores data:
['jai']
['jai', 'kamal']
['jai', 'kamal', 'sumanth']
```

Example

```
import pickle
emp1 = {"Empno" : 1207, "Name" : "Saurav", "Age" : 36,}
emp2 = {"Empno" : 1208, "Name" : "nidhi", "Age" : 37,}
emp3 = {"Empno" : 1209, "Name" : "Kishan", "Age" : 43,}
fobj=open("myfile.dat","ab")
pickle.dump(emp1,fobj)
pickle.dump(emp2,fobj)
pickle.dump(emp3,fobj)
print("succesfully written in dictionary..")

stu = {}
fobj=open("myfile.dat","rb")
try:
    print("fobj.stu Record")
#x=pickle.load(fobj)
    while True:
        str=pickle.load(fobj)
        print(str)
except EOFError:
    fobj.close()
```

Output:

```
succesfully written in dictionary..
fobj.stu Record
{'Empno': 1204, 'Name': 'Saurav', 'Age': 28}
{'Empno': 1205, 'Name': 'Kalyan', 'Age': 36}
{'Empno': 1206, 'Name': 'Kishan', 'Age': 39}
{'Empno': 1207, 'Name': 'Saurav', 'Age': 36}
{'Empno': 1208, 'Name': 'nidhi', 'Age': 37}
{'Empno': 1209, 'Name': 'Kishan', 'Age': 43}
```

Example

Input in dictionary in Binary File

```
import pickle
fobj=open("mygrv.dat","wb")
st = {}
ans = 'y'
while ans=='y':
    r=int(input("Enter the rollno :"))
    n=input("Enter the name : ")
    m=int(input("Enter the marks : "))
    st['rno'] = r
    st['nm']  = n
    st['mrk'] = m
    pickle.dump(st,fobj)
    ans=in
    put("Want to Add : ? (y/n)..")
fobj.close()
```

Output:

```
Enter the rollno :108
Enter the name : jaikamal
Enter the marks : 70
Want to Add : ? (y/n)..y
Enter the rollno :111
Enter the name : hemant
Enter the marks : 72
Want to Add : ? (y/n)..n
```

What is a CSV ?

CSV (Comma Separated Values) is a simple file format used to store tabular data, such as a spreadsheet or database. A CSV file stores tabular data (numbers and text) in plain text. Each line of the file is a data record. Each record consists of one or more fields, separated by commas. The use of the comma as a field separator is the source of the name for this file format.

Python CSV module of Python provides to read and write objects make sure to import it in your program by giving a statement as:

import csv

Opening and Closing CSV Files

Dfile = open("stu.csv","w") --> Open CSV file in Write mode

File1 = open("stu.csv","r") --> Open CSV file in Read mode

Dfile.close() --> Close the CSV file.

As CSV file text are text file such as translation of end of line (EOL)

EOL Characters used in different types of Operating System

Symbol/Char	Meaning	Operating System
CR[\r]	Carriage Return	Macintosh
LF[\n]	Line Feed	UNIX
CR/LF[\r \n]	Carriage Return/ Line Feed	Windows, MSDOS
NULL[\0]	Null Charaters	Other OS

if you specify the newline argument while writing onto a csv file,
it'll create csv file with no EOL translation.

```
Dfile = open("stu.csv","w", newline='')
                              ^
                              |_______CSV file opened in write mode with file
handle
                                      as Dfile (no EOL translation)

File1 = open("stu.csv","r", newline='')
                              ^
                              |_______CSV file opened in read mode with file
handle
                                      as File1 (no EOL translation)
```

Role of newline argument is to specify how would python handle
newline character while working with csv files.

For Example
[30,"Amit\rRawat",24,3.14] --> here \r is EOL character of the file in
Macintosh system due to different OS for EOL character, if you want to
avoid this problem use newline='' argument.

csv.writer --> returns a writer object which writes data into csv file.

writerow() and writerrow() function can be used for writting onto
the writer object.

writerows() method writes all given rows to the CSV file.

```
sturec = [[11,naman,78],[12,manoj,80.0],[13,alok,46.0]]

<writerobject>.writerows(Sturec)
```

write a program to create a CSV file named student,csv and
write 3 records of student in it
having details like rno, name, area, percentage.

Example

```
import csv
f = open("student.csv","w")
wr = csv.writer(f)
head = ['rno','name','address','perc']
wr.writerow(head)
for i in range(3)
        r = int(input("Enter Roll No. "))
        nm = input("Enter Name ")
        addr = input("Enter Area of Address. ")
        p = float(input("Enter Percentage. "))
        l = [r,nm,addr,p]  :-->  all input are into list
        wr.writerow(l)      :--> written the list into csv file
f.close()
```

CSV file save.

Example

```
import csv
f = open("student.csv","w")
wr = csv.writer(f)
head = ['rno','name','marks']
wr.writerow(head)
for i in range(3):
    print("student Record ",(i+1))
    r = int(input("Enter Roll No. "))
    nm = input("Enter Name ")
    mrk = float(input("Enter Marks "))
    strec = [r,nm,mrk]
    wr.writerow(strec)
f.close()
```

Example

Reading data from csv file and display the output.

```
import csv
f = open("D:\\aspnet\\stdlist.csv","r")
csv_reader = csv.reader(f)    # csv_reader is csv reader object
print("Content of Student File are :")
for row in csv_reader:
    print(row)
f.close()
```

Output:

```
['Name', 'Class', 'Marks']
['Amit', 'XI', '56']
['Ajay', 'XI', '58']
['Aman', 'XI', '64']
['Alok', 'XI', '51']
['Bhanu', 'XI', '50']
['Jai', 'XI', '69']
```

Program to create csv file

```
import csv
f = open("result.csv","w")
cwriter = csv.writer(f)
compdata = [
    ['name','Points','Rank'],
    ['Amit',1500,23],
    ['Alok',3600,10],
    ['Ajay',1800,11]]
cwriter.writerows(compdata)
f.close()
```

Output

result.csv file will be created default path of python

Example

CSV file reading without specifying the newline argument:

with open(<csv file>.<read mode>) as <file handle>

```
import csv
with open("result.csv","r",newline ='\r\n') as csvf:
    record = csv.reader(csvf)
    print("Content of Result File :")
    row = []
    for rec in record:
        row.append(rec)
   print(row)
```

Output:

```
Content of Result File :
[['name', 'Points', 'Rank'], ['Amit', '1500', '26'], ['Alok', '3600', '14'], ['Ajay',
'1800', '19']]
```

Example

```
import csv
with open("result.csv","r",newline ='\r\n') as csvf:
    record = csv.reader(csvf)
    print("Content of Result File :")
    #row = []
    for rec in record:
        print(rec)
```

Output:

```
Content of Result File :
['name', 'Points', 'Rank']
['Amit', '1500', '23']
['Alok', '3600', '10']
['Ajay', '1800', '11']
```

Example

```
import csv
fh = open("Empl.csv","w",newline ='')
ewriter = csv.writer(fh)
empdata = [
    ['EmpNo','Name','Desig','Sal'],
```

```
  ['110','Manoj','Admin','18000'],
    ['120','Manish','Admin','10000'],
    ['100','Ajay','Lecturer','20000'],
    ['111','Aman','Clerk','10000'],
    ['114','Naveen','Analyst','22000']
    ]
ewriter.writerows(empdata)
print("File Successfully created..")
fh.close()
```

Output:

Empl.csv file will be created default path of python and records saved into Empl.csv file

Example:

Display all line/record Number:

```
import csv
with open("Empl.csv","r",newline ='\r\n') as csvf:
    record = csv.reader(csvf)
    print("Content of Result File :")

    for rec in record:
        print(rec)
```

Output:

```
Content of Result File :
['EmpNo', 'Name', 'Desig', 'Sal']
['110', 'Manoj', 'Admin', '18000']
['120', 'Manish', 'Admin', '10000']
['100', 'Ajay', 'Lecturer', '20000']
['111', 'Aman', 'Clerk', '10000']
['114', 'Naveen', 'Analyst', '22000']
```