**Python Input And Output**

Python can be used to read and write data. Also it supports reading and writing data to Files.

**"print" statement:**

"print" statement is used to print the output on the screen.

print statement is used to take string as input and place that string to standard output.

Whatever you want to display on output place that expression inside the inverted commas. The expression whose value is to printed place it without inverted commas.

**Syntax:**

1. print "expression"  or  print expression.

eg:

1. a=10
2. print "Welcome to the world of Python"
3. print a

Output:

1. >>>
2. Welcome to the world of Python
3. 10
4. >>>

**Input from Keyboard:**

Python offers two in-built functions for taking input from user. They are:

**1) input()**

**2) raw\_input()**

**1) input() function**input() function is used to take input from the user. Whatever expression is given by the user, it is evaluated and result is returned back.

**Syntax:**

1. input("Expression")

**eg:**

1. n=input("Enter your expression ");
2. print "The evaluated expression is ", n

**Output:**

1. >>>
2. Enter your expression 10\*2
3. The evaluated expression is  20
4. >>>

**2) raw\_input()**raw\_input() function is used to take input from the user. It takes the input from the Standard input in the form of a string and reads the data from a line at once.

**Syntax:**

1. raw\_input(?statement?)

**eg:**

1. n=raw\_input("Enter your name ");
2. print "Welcome ", n

**Output:**

1. >>>
2. Enter your name Rajat
3. Welcome  Rajat
4. >>>

raw\_input() function returns a string. Hence in case an expression is to be evaluated, then it has to be type casted to its following data type. Some of the examples are given below:

**Program to calculate Simple Interest.**

1. prn=int(raw\_input("Enter Principal"))
2. r=int(raw\_input("Enter Rate"))
3. t=int(raw\_input("Enter Time"))
4. si=(prn\*r\*t)/100
5. print "Simple Interest is ",si

**Output:**

1. >>>
2. Enter Principal1000
3. Enter Rate10
4. Enter Time2
5. Simple Interest is  200
6. >>>

**Program to enter details of an user and print them.**

1. name=raw\_input("Enter your name ")
2. math=float(raw\_input("Enter your marks in Math"))
3. physics=float(raw\_input("Enter your marks in Physics"))
4. chemistry=float(raw\_input("Enter your marks in Chemistry"))
5. rollno=int(raw\_input("Enter your Roll no"))
6. print "Welcome ",name
7. print "Your Roll no is ",rollno
8. print "Marks in Maths is ",math
9. print "Marks in Physics is ",physics
10. print "Marks in Chemistry is ",chemistry
11. print "Average marks is ",(math+physics+chemistry)/3

**Output:**

1. >>>
2. Enter your name rajat
3. Enter your marks in Math76.8
4. Enter your marks in Physics71.4
5. Enter your marks in Chemistry88.4
6. Enter your Roll no0987645672
7. Welcome  rajat
8. Your Roll no is  987645672
9. Marks in Maths is  76.8
10. Marks in Physics is  71.4
11. Marks in Chemistry is  88.4
12. Average marks is  78.8666666667
13. >>>

**File Handling:**

Python provides the facility of working on Files. A File is an external storage on hard disk from where data can be stored and retrieved.

**Operations on Files:**

**1) Opening a File:** Before working with Files you have to open the File. To open a File, Python built in function open() is used. It returns an object of File which is used with other functions. Having opened the file now you can perform read, write, etc. operations on the File.

**Syntax:**

1. obj=open(filename , mode , buffer)

**here,**

**filename:**It is the name of the file which you want to access.

**mode:**It specifies the mode in which File is to be opened.There are many types of mode. Mode depends the operation to be performed on File. Default access mode is read.

**2) Closing a File:**Once you are finished with the operations on File at the end you need to close the file. It is done by the close() method. close() method is used to close a File.

**Syntax:**

1. fileobject.close()

**3) Writing to a File:**write() method is used to write a string into a file.

**Syntax:**

1. fileobject.write(string str)

**4) Reading from a File:**read() method is used to read data from the File.

**Syntax:**

1. fileobject.read(value)

here, value is the number of bytes to be read. In case, no value is given it reads till end of file is reached.

**Program to read and write data from a file.**

1. obj=open("abcd.txt","w")
2. obj.write("Welcome to the world of Python")
3. obj.close()
4. obj1=open("abcd.txt","r")
5. s=obj1.read()
6. print s
7. obj1.close()
8. obj2=open("abcd.txt","r")
9. s1=obj2.read(20)
10. print s1
11. obj2.close()

**Output:**

1. >>>
2. Welcome to the world of Python
3. Welcome to the world
4. >>>

**Attributes of File:**

There are following File attributes.

|  |  |
| --- | --- |
| **Attribute** | **Description** |
| Name | Returns the name of the file. |
| Mode | Returns the mode in which file is being opened. |
| Closed | Returns Boolean value. True, in case if file is closed else false. |

**Eg:**

1. obj = open("data.txt", "w")
2. print  obj.name
3. print  obj.mode
4. print  obj.closed

**Output:**

1. >>>
2. data.txt
3. w
4. False
5. >>>

**Modes of File:**

There are different modes of file in which it can be opened. They are mentioned in the following table.

A File can be opened in two modes:

1) Text Mode.

2) Binary Mode.

|  |  |
| --- | --- |
| **Mode** | **Description** |
| R | It opens in Reading mode. It is default mode of File. Pointer is at beginning of the file. |
| rb | It opens in Reading mode for binary format. It is the default mode. Pointer is at beginning of file. |
| r+ | Opens file for reading and writing. Pointer is at beginning of file. |
| rb+ | Opens file for reading and writing in binary format. Pointer is at beginning of file. |
| W | Opens file in Writing mode. If file already exists, then overwrite the file else create a new file. |
| wb | Opens file in Writing mode in binary format. If file already exists, then overwrite the file else create a new file. |
| w+ | Opens file for reading and writing. If file already exists, then overwrite the file else create a new file. |
| wb+ | Opens file for reading and writing in binary format. If file already exists, then overwrite the file else create a new file. |
| a | Opens file in Appending mode. If file already exists, then append the data at the end of existing file, else create a new file. |
| ab | Opens file in Appending mode in binary format. If file already exists, then append the data at the end of existing file, else create a new file. |
| a+ | Opens file in reading and appending mode. If file already exists, then append the data at the end of existing file, else create a new file. |
| ab+ | Opens file in reading and appending mode in binary format. If file already exists, then append the data at the end of existing file, else create a new file. |

**Methods:**

There are many methods related to File Handling. They are given in the following table:

There is a module "os" defined in Python that provides various functions which are used to perform various operations on Files. To use these functions 'os' needs to be imported.

|  |  |
| --- | --- |
| **Method** | **Description** |
| rename() | It is used to rename a file. It takes two arguments, existing\_file\_name and new\_file\_name. |
| remove() | It is used to delete a file. It takes one argument. Pass the name of the file which is to be deleted as the argument of method. |
| mkdir() | It is used to create a directory. A directory contains the files. It takes one argument which is the name of the directory. |
| chdir() | It is used to change the current working directory. It takes one argument which is the name of the directory. |
| getcwd() | It gives the current working directory. |
| rmdir() | It is used to delete a directory. It takes one argument which is the name of the directory. |
| tell() | It is used to get the exact position in the file. |

**1) rename():**

**Syntax:**

1. os.rename(existing\_file\_name, new\_file\_name)

**eg:**

1. import os
2. os.rename('mno.txt','pqr.txt')

**2) remove():**

**Syntax:**

1. os.remove(file\_name)

**eg:**

1. import os
2. os.remove('mno.txt')

**3) mkdir()**

**Syntax:**

os.mkdir("file\_name")

**eg:**

1. import os
2. os.mkdir("new")

**4) chdir()**

**Syntax:**

os.chdir("file\_name")

**eg:**

1. import os
2. os.chdir("new")

**5) getcwd()**

**Syntax:**

os.getcwd()

**eg:**

1. import os
2. print os.getcwd()

**6) rmdir()**

**Syntax:**

os.rmdir("directory\_name)

**eg:**

1. import os
2. os.rmdir("new")

NOTE: In order to delete a directory, it should be empty. In case directory is not empty first delete the files.