**Python programming language**

Table of Contents

[Variables 4](#_Toc132473278)

[Keywords 4](#_Toc132473279)

[Input Function 4](#_Toc132473280)

[Multiple Values in Single Line 5](#_Toc132473281)

[Multiple Line String Input 6](#_Toc132473282)

[Additional function 7](#_Toc132473283)

[Single and Multiline Comment 7](#_Toc132473284)

[Type casting 8](#_Toc132473285)

[String and String Functions 9](#_Toc132473286)

[String Manipulation 11](#_Toc132473287)

[swaping program 12](#_Toc132473288)

[Operators 12](#_Toc132473289)

[Arithmetic Operators 12](#_Toc132473290)

[Assignment Operators 13](#_Toc132473291)

[Comparison Operators or Relational Operators 14](#_Toc132473292)

[Logical Operators 16](#_Toc132473293)

[Bitwise Operators 16](#_Toc132473294)

[Conditional statement 17](#_Toc132473295)

[IF Statement 17](#_Toc132473296)

[IF - Else Statement 18](#_Toc132473297)

[Single if else statement 18](#_Toc132473298)

[Elif Statement 19](#_Toc132473299)

[Nested If Statement 19](#_Toc132473300)

[Loop 21](#_Toc132473301)

[While Loop 21](#_Toc132473302)

[Continue using While Loop 22](#_Toc132473303)

[Break using While Loop 23](#_Toc132473304)

[Range in Python 23](#_Toc132473305)

[For loop 24](#_Toc132473306)

[Nested For Loop 25](#_Toc132473307)

[While Else 27](#_Toc132473308)

[For Else 28](#_Toc132473309)

[List 29](#_Toc132473310)

[Tuple 32](#_Toc132473311)

[Set 35](#_Toc132473312)

[Dictionary 37](#_Toc132473313)

[Differenece between d1=d2 and d1 = d1.copy in dictionary 39](#_Toc132473314)

[Identity Operators 40](#_Toc132473315)

[Membership operators 41](#_Toc132473316)

[Function 41](#_Toc132473317)

[No Return Type Without Argument Function in Python 42](#_Toc132473318)

[No Return Type With Argument Function in Python 42](#_Toc132473319)

[Return Type Without Argument Function in Python 42](#_Toc132473320)

[Return Type With Argument Function in Python 43](#_Toc132473321)

[Arbitrary Arguments Function in Python (\*) 43](#_Toc132473322)

[Keyword Arguments Function in Python 43](#_Toc132473323)

[Arbitrary Keyword Arguments in Python(\*\*) 43](#_Toc132473324)

[Default Parameter Function in Python 43](#_Toc132473325)

[Passing a List as an Argument in Function Python 44](#_Toc132473326)

[Recursive function 44](#_Toc132473327)

[Lambda function 45](#_Toc132473328)

[Date Time Functions 46](#_Toc132473329)

[Math Functions 49](#_Toc132473330)

[Docstrings 50](#_Toc132473331)

[Try Block in Python 50](#_Toc132473332)

[try block in Python 51](#_Toc132473333)

[Try Else 51](#_Toc132473334)

[Try else finally 51](#_Toc132473335)

[Type of Exceptions in Python 51](#_Toc132473336)

[Class & object 54](#_Toc132473337)

[Class Attributes 55](#_Toc132473338)

[Instance Attributes 56](#_Toc132473339)

[Class Method 57](#_Toc132473340)

[Instance method 57](#_Toc132473341)

[Init Method 58](#_Toc132473342)

[Property Decorator 59](#_Toc132473343)

[Property Decorator Getter Setter 60](#_Toc132473344)

[Property Method 61](#_Toc132473345)

[Class Method Decorator 62](#_Toc132473346)

[Static Method 63](#_Toc132473347)

[Abstraction and Encapsulation 63](#_Toc132473348)

[Inheritance 66](#_Toc132473349)

[Single inheritance 67](#_Toc132473350)

[Multiple inheritance 68](#_Toc132473351)

[Multi level inheritance 69](#_Toc132473352)

[Function Overriding 70](#_Toc132473353)

[Handling Diamond Problem in Python 71](#_Toc132473354)

[Operator Overloading 72](#_Toc132473355)

[Abstract Base Class 74](#_Toc132473356)

[File HANDLING 75](#_Toc132473357)

[Open a File 75](#_Toc132473358)

[Delete a file 76](#_Toc132473359)

[Abstraction and Encapsulation 63](#_Toc132473348)

**Output**

|  |  |  |
| --- | --- | --- |
| **Compound Operator** | **Sample Expression** | **Expanded Form** |
|  |  |  |
| **+=** | **a+=2** | **a=a+2** |
|  |  |  |
| **-=** | **a-=6** | **a=a-6** |
|  |  |  |
| **\*=** | **a\*=7** | **a=a\*7** |
|  |  |  |
| **/=** | **a/=4** | **a=a/4** |
|  |  |  |
| **%=** | **a%=9** | **a=a%9** |
|  |  |  |
| **\*\*=** | **a\*\*=3** | **a=a\*\*3** |
|  |  |  |
| **//=** | **a//=2** | **a=a//2** |

|  |  |
| --- | --- |
| **Operator** | **uses** |
| **==** | **Equal operator** |
|  |  |
| **!=** | **Not Equal operator** |
|  |  |
| **<** | **Less than operator** |
|  |  |
| **>** | **Greater than operator** |
|  |  |
| **<=** | **Less than or equal to operator** |
|  |  |
| **>=** | **Greater than or equal to operator** |

|  |  |
| --- | --- |
| **Operator** | **Description** |
|  |  |
| **&** | **Bitwise AND** |
|  |  |
| **|** | **Bitwise OR** |
|  |  |
| **^** | **Bitwise XOR** |
|  |  |
| **~** | **Bitwise NOT** |
|  |  |
| **<<** | **Left shift** |
|  |  |
| **>>** | **Right shift** |
|  |  |

### Date Time Functions

**The strftime() function is used to convert date and time objects to their string representation.**

* **datetime.now(): Returns the current date and time.**
* **datetime.date(): Returns date object of today's date.**
* **datetime.time(): Returns the current time.**
* **datetime.datetime(): Returns the current date and time as a datetime object.**
* **datetime.timedelta(days=0, seconds=0, microseconds=0, milliseconds=0, minutes=0, hours=0, weeks=0): Represents the difference between two date or time values.**

**Syntax :  
           strftime ( format )**

**List of format codes**

|  |  |  |
| --- | --- | --- |
| **Directive** | **Description** | **Example** |
|  |  |  |
| **%A** | **Weekday full name** | **Monday** |
|  |  |  |
| **%a** | **Weekday short name** | **Mon** |
|  |  |  |
| **%d** | **Day of month (1-31)** | **26** |
|  |  |  |
| **%b** | **Month of short name** | **Dec** |
|  |  |  |
| **%B** | **Month of full name** | **December** |
|  |  |  |
| **%Y** | **Year of full version , without century** | **2022** |
|  |  |  |
| **%y** | **Year of short version** | **22** |
|  |  |  |
| **%w** | **Weekday as a number ( 0-Sun , 1-Mon , 2-Tue , 3-Wed , 4-Thu , 5-Fri , 6-Sat )** | **1(Monday)** |
|  |  |  |
| **%W** | **Week number of Year ( Monday as the first day of week ( 00-53 ) )** | **48** |
|  |  |  |
| **%m** | **Month as a Number ( 1(Jun) - 12(Dec) )** | **12(Dec)** |
|  |  |  |
| **%H** | **Hours ( 00-23 )** | **15** |
|  |  |  |
| **%M** | **Minute ( 00-59 )** | **50** |
|  |  |  |
| **%S** | **Second ( 00-59 )** | **23** |
|  |  |  |
| **%p** | **PM / AM** | **PM** |
|  |  |  |
| **%c** | **Local version of date and time** | **Mon Dec 26 15 : 50 : 23 2022** |
|  |  |  |
| **%X** | **Local version of time** | **15:50:23** |
|  |  |  |
| **%x** | **Local version of date** | **12/26/22** |
|  |  |  |

# Class & object

**A class is a blueprint or serves as a template from which individual objects are created**

**Object is an instance of a class which consists of methods and properties**

**Syntax of Class:  
      class Class\_Name :  
            # statements**

**Example of Class:  
      class student :  
            name = " Tutor joe's "  
            age = 30**

**Syntax of Object:  
      object\_name = class\_name ( arguments )**

**Example of Object:  
      s = student ( )**

**class car():**

**pass**

**a = 10**

**print(type(a))**

**print(type(car))**

**swift=car()**

**print(isinstance(swift,car))**

**print(isinstance(a,int))**

**print(type(swift))**

**Output**

**<class 'int'>**

**<class 'type'>**

**True**

**True**

**<class '\_\_main\_\_.car'>**

## Class Attributes

**Class attributes belong to the class itself they will be shared by all the instances. Such attributes are defined in the class body parts usually at the top, for legibility.**

**class Student():**

**name = "Ram Kumar"**

**age = 25**

**''' This is Class Attributes '''**

***# getattr method***

**print(getattr(Student, 'name'))**

**print(getattr(Student, 'age'))**

**print(getattr(Student, 'gender', 'No Such Attribute Found'))**

***# Dot Notation***

**print(Student.name)**

**print(Student.age)**

***# setattr***

**setattr(Student, 'name', 'Tutor Joes')**

**print(Student.name)**

**setattr(Student, 'gender', 'Male')**

**print(Student.gender)**

**Student.city = "Salem"**

**print(Student.city)**

**print(Student.\_\_dict\_\_)**

**delattr(Student,"city")**

**print(Student.\_\_dict\_\_)**

**del Student.gender**

**print(Student.\_\_dict\_\_)**

**Output**

**Ram Kumar**

**25**

**No Such Attribute Found**

**Ram Kumar**

**25**

**Tutor Joes**

**Male**

**Salem**

**{'\_\_module\_\_': '\_\_main\_\_', 'name': 'Tutor Joes', 'age': 25, '\_\_dict\_\_': <attribute '\_\_dict\_\_' of 'Student' objects>, '\_\_weakref\_\_': <attribute '\_\_weakref\_\_' of 'Student' objects>, '\_\_doc\_\_': None, 'gender': 'Male', 'city': 'Salem'}**

**{'\_\_module\_\_': '\_\_main\_\_', 'name': 'Tutor Joes', 'age': 25, '\_\_dict\_\_': <attribute '\_\_dict\_\_' of 'Student' objects>, '\_\_weakref\_\_': <attribute '\_\_weakref\_\_' of 'Student' objects>, '\_\_doc\_\_': None, 'gender': 'Male'}**

**{'\_\_module\_\_': '\_\_main\_\_', 'name': 'Tutor Joes', 'age': 25, '\_\_dict\_\_': <attribute '\_\_dict\_\_' of 'Student' objects>, '\_\_weakref\_\_': <attribute '\_\_weakref\_\_' of 'Student' objects>, '\_\_doc\_\_': None}**

## Instance Attributes

**class user:**

**course = 'Java'**

**o = user()**

**print(user.\_\_dict\_\_)**

**print(user.course) *# Print Class attribute***

**print(o.\_\_dict\_\_)**

**print(o.course) 🡪 instance attribute**

**o.course = "C++"**

**print(o.\_\_dict\_\_)**

**print(o.course)**

**o2 = user()**

**print(o2.course)**

**Output**

**{'\_\_module\_\_': '\_\_main\_\_', 'course': 'Java', '\_\_dict\_\_': <attribute '\_\_dict\_\_' of 'user' objects>, '\_\_weakref\_\_': <attribute '\_\_weakref\_\_' of 'user' objects>, '\_\_doc\_\_': None}**

**Java**

**{}**

**Java**

**{'course': 'C++'}**

**C++**

**Java**

## Class Method

***# Class Methods***

**class Student:**

**name = "Tutor Joes"**

**age = 25**

**def printall():**

**print("Name : ", Student.name)**

**print("Age : ", Student.age)**

**Student.printall()**

**print(Student.\_\_dict\_\_)**

**print(getattr(Student, "printall"))**

**getattr(Student, "printall")()**

**Student.\_\_dict\_\_['printall']()**

**Output**

**Name : Tutor Joes**

**Age : 25**

**{'\_\_module\_\_': '\_\_main\_\_', 'name': 'Tutor Joes', 'age': 25, 'printall': <function Student.printall at 0x000001F08DD5B5E0>, '\_\_dict\_\_': <attribute '\_\_dict\_\_' of 'Student' objects>>, '\_\_weakref\_\_': , '\_\_doc\_\_': None}**

**Name : Tutor Joes**

**Age : 25**

**Name : Tutor Joes**

**Age : 25**

## Instance method

***# instance Methods***

**class Student:**

**name = "Tutor Joes"**

**age = 25**

**def printall(self,gender): 🡪 used self keyword means, it is instance method and**

**directliy call by object**

**print("Name : ", Student.name)**

**print("Age : ", Student.age)**

**print("Gender : ", gender)**

**o=Student()**

**"""**

**o.printall()**

**Student.printall(o)**

**"""**

**o.printall("Male")**

**Student.printall(o,"Male")**

**Output**

**Name : Tutor Joes**

**Age : 25**

**Gender : Male**

**Name : Tutor Joes**

**Age : 25**

**Gender : Male**

## Init Method

***# init method in Python***

**class user:**

**def \_\_init\_\_(self, name):**

**print("Call When new Instance Created")**

**self.name = name 🡪instance attribute**

**def printall(self):**

**print("Name : ", self.name)**

**o1 = user("Tutor Joes")**

**o1.printall()**

**print(o1.\_\_dict\_\_)**

**o2 = user("Joes")**

**o2.printall()**

**print(o2.\_\_dict\_\_)**

**print(user.\_\_dict\_\_)**

**Output**

**Call When new Instance Created**

**Name : Tutor Joes**

**{'name': 'Tutor Joes'}**

**Call When new Instance Created**

**Name : Joes**

**{'name': 'Joes'}**

**{'\_\_module\_\_': '\_\_main\_\_', '\_\_init\_\_': <function user.\_\_init\_\_ at 0x000002485E95B5E0>, 'printall': <function user.printall at 0x000002485E95B670>, '\_\_dict\_\_': <attribute '\_\_dict\_\_' of 'user' objects>, '\_\_weakref\_\_': <attribute '\_\_weakref\_\_' of 'user' objects> '\_\_doc\_\_': None}**

## Property Decorator

***# Property Decorator***

**class user:**

**def \_\_init\_\_(self, name, age):**

**self.name = name**

**self.age = age**

***# self.msg = self.name + " is " + str(self.age) + " years old"***

**@property**

**def msg(self):**

**return self.name + " is " + str(self.age) + " years old"**

**o = user("Tutor Joes", 25)**

**print(o.name)**

**print(o.age)**

**print(o.msg)**

**o.age = 45**

**print(o.msg)**

**Output**

**Tutor Joes**

**25**

**Tutor Joes is 25 years old**

**Tutor Joes is 45 years old**

## Property Decorator Getter Setter

**In Python, property decorators are used to define getter, setter, and deleter methods for class properties. They allow for the encapsulation of data, by controlling access to the underlying data. Property decorators are applied to methods and define how a property value can be retrieved, set, or deleted.**

***# Property Decorators Getter Setter***

**class student:**

**def \_\_init\_\_(self, total):**

**self.\_total = total**

**def average(self):**

**return self.\_total / 5.0**

**@property**

**def total(self):**

**return self.\_total**

**@total.setter**

**def total(self, t):**

**if t < 0 or t > 500:**

**print("Invalid Total and can't Change")**

**else:**

**self.\_total = t**

**o = student(450)**

**print("Total : ", o.total)**

**print("Average : ", o.average())**

**o.total = 550**

**print("Total : ", o.total)**

**print("Average : ", o.average())**

**Output**

**Total : 450**

**Average : 90.0**

**Invalid Total and can't Change**

**Total : 450**

**Average : 90.0**

## Property Method

***# Property Method***

**class student:**

**def \_\_init\_\_(self, total):**

**self.\_total = total**

**def average(self):**

**return self.\_total / 5.0**

**def getter(self):**

**return self.\_total**

**def setter(self, t):**

**if t < 0 or t > 500:**

**print("Invalid Total and can't Change")**

**else:**

**self.\_total = t**

**total = property(getter, setter)**

**o = student(450)**

**print("Total : ", o.total)**

**print("Average : ", o.average())**

**o.total = 350**

**print("Total : ", o.total)**

**print("Average : ", o.average())**

**Output**

**Total : 450**

**Average : 90.0**

**Total : 350**

**Average : 70.0**

## Class Method Decorator

**class student:**

**count = 0**

**def \_\_init\_\_(self, name, age):**

**self.name = name**

**self.age = age**

**student.count += 1**

**def printDetail(self):**

**print("Name : ", self.name, " Age : ", self.age)**

**@classmethod**

**def total(cls):**

**return cls.count**

**o = student("Joes", 25)**

**o.printDetail()**

**a = student("Raja", 45)**

**a.printDetail()**

**print("Total Admission :", student.total())**

**print("Total Admission :", o.total())**

**Output**

**Name : Joes Age : 25**

**Name : Raja Age : 45**

**Total Admission : 2**

**Total Admission : 2**

## Static Method

***# Static Method in Python***

**class student:**

**def \_\_init\_\_(self, name, age):**

**self.name = name**

**self.age = age**

**def printDetail(self):**

**print("Name : ", self.name, " Age : ", self.age)**

**@staticmethod**

**def welcome():**

**print("Welcome to our Institution")**

**s1 = student("Joes", 25)**

**s1.printDetail()**

**s1.welcome()**

**s2 = student("Raja", 45)**

**s2.printDetail()**

**s2.welcome()**

**Output**

**Name : Joes Age : 25**

**Welcome to our Institution**

**Name : Raja Age : 45**

**Welcome to our Institution**

# Abstraction and Encapsulation

**Data abstraction**

* **Data abstraction refers to providing only essential information to the outside world hiding their background details**
* **To present the needed information in program without presenting the details**

**Data encapsulation**

* **Encapsulation is a process of wrapping code and data together into a single unit**

***# Abstraction and Encapsulation in Python***

**class Library:**

**def \_\_init\_\_(self, books):**

**self.books = books**

**def list\_books(self):**

**print("Available Books")**

**for book in self.books:**

**print(book)**

**def borrow\_book(self, borrow\_book):**

**if borrow\_book in self.books:**

**print("Get Your Book Now")**

**self.books.remove(borrow\_book)**

**else:**

**print("Book not Available")**

**def receive\_book(self, receive\_book):**

**print("You have returned the book")**

**self.books.append(receive\_book)**

**books = ['C', 'C++', 'Java']**

**o = Library(books)**

**msg = """**

**1.Display Book**

**2.Borrow Book**

**3.Return Book**

**"""**

**while True:**

**print(msg)**

**ch = int(input("Enter Your Choice : "))**

**if ch == 1:**

**o.list\_books()**

**elif ch == 2:**

**book = input("Enter Book Name To Borrow : ")**

**o.borrow\_book(book)**

**elif ch == 3:**

**book = input("Enter Book Name To Return : ")**

**o.receive\_book(book)**

**else:**

**print("Thank You come again")**

**quit()**

**Output**

**1.Display Book**

**2.Borrow Book**

**3.Return Book**

**Enter Your Choice : 1**

**Available Books**

**C**

**C++**

**Java**

**1.Display Book**

**2.Borrow Book**

**3.Return Book**

**Enter Your Choice : 2**

**Enter Book Name To Borrow : C**

**Get Your Book Now**

**1.Display Book**

**2.Borrow Book**

**3.Return Book**

**Enter Your Choice : 1**

**Available Books**

**C++**

**Java**

**1.Display Book**

**2.Borrow Book**

**3.Return Book**

**Enter Your Choice : 3**

**Enter Book Name To Return : Python**

**You have returned the book**

**1.Display Book**

**2.Borrow Book**

**3.Return Book**

**Enter Your Choice : 1**

**Available Books**

**C++**

**Java**

**Python**

**1.Display Book**

**2.Borrow Book**

**3.Return Book**

**Enter Your Choice : 4**

**Thank You come again**

# Inheritance

**Inheritance is a process in which one object acquires all the properties and behavior of its parent object automatically**

## Single inheritance

**It is defined as the inheritance in which a derived class is inherited from the only one base class**

Base

**Syntax :  
         class base1 :  
              body of base class  
         class derived( base1) :  
              body of derived class**

Derived

**class Nokia:**

**company = "Nokia India"**

**webiste = "www.nokia-india.com"**

**def contact\_details(self):**

**print("Address : Cherry Road,Near Bus Stand ,Salem")**

**class Nokia1100(Nokia):**

**def \_\_init\_\_(self):**

**self.name = "Nokia 1100"**

**self.year = 1998**

**def product\_details(self):**

**print("Name : ", self.name)**

**print("Year : ", self.year)**

**print("Company : ", self.company)**

**print("Website : ", self.webiste)**

**mobile = Nokia1100()**

**mobile.product\_details()**

**mobile.contact\_details()**

**Output**

**Name : Nokia 1100**

**Year : 1998**

**Company : Nokia India**

**Website : www.nokia-india.com**

**Address : Cherry Road,Near Bus Stand ,Salem**

## Multiple inheritance

**Multiple inheritance is a future of object oriented concept, where a class can inherit properties of more than one parent class**

**Syntax :  
         class Parent1 :  
                  # attributes and methods of Parent1  
         class Parent2 :  
                  # attributes and methods of Parent2  
         class Child( Parent1, Parent2 ) :  
                  # attributes and methods of Child**

C

B

A

**class Father:**

**def fishing(self):**

**print("Fishing in Rivers")**

**def chess(self):**

**print("Playing Chess From Father")**

**class Mother:**

**def cooking(self):**

**print("Cooking Food")**

**def chess(self):**

**print("Playing Chess From Mother")**

**class Son(Mother,Father):**

**def ride(self):**

**print("Riding Bicycle")**

**o = Son()**

**o.ride()**

**o.fishing()**

**o.cooking()**

**o.chess()**

**Output**

**Riding Bicycle**

**Fishing in Rivers**

**Cooking Food**

**Playing Chess From Mother**

## Multi level inheritance

A

**Syntax :  
         class base1 :  
                  body of base class  
         class derived1( base1 ) :  
                  body of derived class  
         class derived2( derived1 ) :  
                  body of derived class**

B

C

***# Multilevel Inheritance***

**class GrandFather:**

**def ownHouse(self):**

**print("Grandpa House")**

**class Father(GrandFather):**

**def ownBike(self):**

**print("Father's Bike")**

**class Son(Father):**

**def ownBook(self):**

**print("Son Have a Book")**

**o = Son()**

**o.ownHouse()**

**o.ownBike()**

**o.ownBook()**

**Output**

**Grandpa House**

**Father's Bike**

**Son Have a Book**

# Function Overriding

***# Function Overriding***

**class Employee:**

**def WorkingHrs(self):**

**self.hrs = 50**

**def printHrs(self):**

**print("Total Working Hrs : ", self.hrs)**

**class Trainee(Employee):**

**def WorkingHrs(self):**

**self.hrs = 60**

**def resetHrs(self):**

**super().WorkingHrs()**

**employee = Employee()**

**employee.WorkingHrs()**

**employee.printHrs()**

**trainee=Trainee()**

**trainee.WorkingHrs()**

**trainee.printHrs()**

***# Reset Trainee Hrs***

**trainee.resetHrs()**

**trainee.printHrs()**

**Output**

**Total Working Hrs : 50**

**Total Working Hrs : 60**

**Total Working Hrs : 50**

# Handling Diamond Problem in Python

A

C

B

D

**class A:**

**def display(self):**

**print("I am the display of Class A")**

**class B(A):**

**def display(self):**

**print("I am the display of Class B")**

**class C(A):**

**def display(self):**

**print("I am the display of Class C")**

**class D(B, C):**

**def display(self):**

**print("I am the display of Class D")**

**o = D()**

**o.display()**

**Output**

**I am the display of Class D**

# Operator Overloading

**"""**

**a = 10**

**b = 20**

**print(a + b)**

**a = "Tutor"**

**b = "Joes"**

**print(a + b)**

**"""**

**class Addition:**

**def \_\_init\_\_(self, a):**

**self.a = a**

**def \_\_add\_\_(o1, o2):**

**return o1.a + o2.a**

**def \_\_sub\_\_(o1, o2):**

**return o1.a - o2.a**

**o1 = Addition(10)**

**o2 = Addition(20)**

**print("Total : ", (o1 + o2))**

**print("Difference : ", (o1 - o2))**

**"""**

**Operator Magic Method**

**+ \_\_add\_\_(self, other)**

**- \_\_sub\_\_(self, other)**

**\* \_\_mul\_\_(self, other)**

**/ \_\_truediv\_\_(self, other)**

**// \_\_floordiv\_\_(self, other)**

**% \_\_mod\_\_(self, other)**

**\*\* \_\_pow\_\_(self, other)**

**>> \_\_rshift\_\_(self, other)**

**<< \_\_lshift\_\_(self, other)**

**& \_\_and\_\_(self, other)**

**| \_\_or\_\_(self, other)**

**^ \_\_xor\_\_(self, other)**

**Comparison Operators :**

**Operator Magic Method**

**< \_\_LT\_\_(SELF, OTHER)**

**> \_\_GT\_\_(SELF, OTHER)**

**<= \_\_LE\_\_(SELF, OTHER)**

**>= \_\_GE\_\_(SELF, OTHER)**

**== \_\_EQ\_\_(SELF, OTHER)**

**!= \_\_NE\_\_(SELF, OTHER)**

**Assignment Operators :**

**Operator Magic Method**

**-= \_\_ISUB\_\_(SELF, OTHER)**

**+= \_\_IADD\_\_(SELF, OTHER)**

**\*= \_\_IMUL\_\_(SELF, OTHER)**

**/= \_\_IDIV\_\_(SELF, OTHER)**

**//= \_\_IFLOORDIV\_\_(SELF, OTHER)**

**%= \_\_IMOD\_\_(SELF, OTHER)**

**\*\*= \_\_IPOW\_\_(SELF, OTHER)**

**>>= \_\_IRSHIFT\_\_(SELF, OTHER)**

**<<= \_\_ILSHIFT\_\_(SELF, OTHER)**

**&= \_\_IAND\_\_(SELF, OTHER)**

**|= \_\_IOR\_\_(SELF, OTHER)**

**^= \_\_IXOR\_\_(SELF, OTHER)**

**Unary Operators :**

**Operator Magic Method**

**- \_\_NEG\_\_(SELF, OTHER)**

**+ \_\_POS\_\_(SELF, OTHER)**

**~ \_\_INVERT\_\_(SELF, OTHER)**

**"""**

**Output**

**Total : 30**

**Difference : -10**

# Abstract Base Class

**from abc import ABC, abstractmethod**

**class Bank(ABC):**

**@abstractmethod**

**def loan(self): pass**

**@abstractmethod**

**def credit(self): pass**

**@abstractmethod**

**def debit(self): pass**

**class HDFC(Bank):**

**def loan(self):**

**print("We can Provide 7.5% Interest Loan")**

**def credit(self):**

**print("HDFC Provide Credit")**

**def debit(self):**

**print("HDFC Provide Debit")**

**def card(self):**

**print("HDFC Provide Credit Card")**

**o=HDFC()**

**o.loan()**

**o.credit()**

**o.debit()**

**o.card()**

**Output**

**We can Provide 7.5% Interest Loan**

**HDFC Provide Credit**

**HDFC Provide Debit**

**HDFC Provide Credit Card**

# File HANDLING

**File handling is an important part of any web application. Python has several functions for creating, reading, updating, and deleting files**

## Open a File

**try:**

**f=open("ram.txt",'w')**

***#f=open("data.txt",'a')***

***#f=open("data.txt",'r')***

***#print(f.read())***

***#print(f.readline())***

***#print(f.readlines())***

**"""**

**for line in f:**

**print(line)**

**"""**

**f.write("\nThis is New Line")**

**except FileNotFoundError:**

**print("File not Found")**

**else:**

**print("Thank You")**

**f.close()**

**Output**

**This is a New Line**

**Thank You**

## Delete a file

**import os**

**if os.path.exists("data.txt"):**

**os.remove("data.txt")**

**else:**

**print("File Not Found")**