Object Oriented Programming (OOPS)

---

Class

---

1) What is a class ? ---> A blueprint (or) description (or) template (or) sample of an object

2) What does a class contain ? ---> A group of methods

3) What is the keyword to define a class ? ---> class

What is the keyword to define a function (or) method ? ---> def

What is the keyword to define a lambda function ? ---> lambda

4) class c1:

```
    def method1(self):
        statements
    def method2(self):
        statements
```

What is the above procedure called ? ---> Class definition

5) What is c1 called ? ---> classname

6) What does class definition do ? ---> Informs pvm about the methods present in the class

7) Can object be created without defining the class ? ---> No

8) In other words, define the class first and then create object

9) Can there be multiple classes with same name ? ---> Yes

10) What happens when there are multiple classes with same name ? --->
Last class is recognized and rest are discarded

11) What are pre-defined class names ? ---> int , float , complex , bool , str , list , tuple , set , dict and so on

What is c1 called ? ---> A user defined class

12) Can there be a class without methods ? ---> Yes and it is called empty class

13) How to define an empty class ? --->

```
class c1:
    pass
```

Note:

1) What does a java class contain ? ---> Instance variables and methods

2) What about python class ? ---> Methods only

Object

---

1) What is an object ? ---> An instance of a class

2) What is 25 called ? ---> An instance of int class

What is 10.8 called ? ---> An instance of float class

What is 'Hyd' called ? ---> An instance of str class

What is [10, 20 , 15 , 18] called ? ---> An instance of list class

3) How is an object created ? ---> ref = classname()

4) What does a = c1() do ? ---> Creates c1 class object and assigns id of the object to reference 'a'

Where does refernce 'a' point to ? ---> c1 class object

What is the name of object ? ---> 'a' itself

5) In other words, reference name is nothing but object name

6) What does object 'a' initially contain ? ---> Nothing and it is an empty object

*7) In other words, every python object (i.e. user defined object) is initially empty

8) How to determine id of the object ? ---> id(object)

9) How to determine type of the object ? ---> type(object)

What is the type of user defined object ? ---> <class '__main__ . c1'>

10) How to delete an object ? ---> del reference

11) Is object alive after program terminates in case it is not deleted ? --->

No and every object is lost soon after program terminates

Method

---

1) What does a method do ? ---> Performs operation on object such as

a) Reads inputs into an object

b) Prints an object

c) Adds two objects

d) Compares two objects and so on

2) Where is a method defined ? ---> Inside the class

3) Method is a member of what ? ---> Class becoz it is defined inside the class

Class is a member of what ? ---> Module

4) class c1:

def m1(self):

statements

Is m1() a method ? ---> Yes becoz it is defined inside the class

5) What is the mandatory argument to a method ? ---> self

6) What are the two ways to call a method ? --->object . method() and

classname . method(object)

7) Can a method be called without an object ? ---> No and object is mandatory to call a method

self object

---

1) What is self ? ---> Invoking object (or) owner object

2) a . m1()

What is self wrt the above method call ? ---> Object 'a' becoz method m1() is called wrt object 'a'

3) c . add(a , b)

What is self wrt the above method call ? ---> Object 'c' becoz method add() is called wrt object 'c'

4) c1 . m1(b)

What is self wrt the above method call ? ---> Object 'b' becoz it is passed to m1() method

5) a . m1()

Where does self point to ? ---> The same object where 'a' points

If reference 'a' is 1000, self is also 1000

6) Who is initializing self and when ? ---> PVM initializes self as soon as method is called

7) Is Self valid ? ---> Yes becoz it is a user defined word but not keyword

8) Who can use self ? ---> Only method

Who can not use self ? ---> Function and outside the class

9) Why function can not use self ? ---> Since it is not called wrt object

Instance Variable

---

1) What is an instance variable ? ---> A variable of the object

What is a local variable ? ---> A variable of the function (or) method

What is a global variable ? ---> A variable of the module (or) program

2) Where does instance variable reside ? ---> In the object

3) Instance variable is a member of what ? ---> Object

4) What are instance variables of date object ? ---> dd , mm and yy

What are instance variables of rational object ? ---> num and den

What are instance variables of time object ? ---> hours , minutes and seconds

What are instance variables of complex object ? ---> real and imag

What are instance variables of employee object ? --->

empnumber , empname , salary and so on

5) How to add an instance variable to an object ? ---> object . new-variable = value

6) Where can instance variable be added to the object ? ---> Any where in the program

7) a = emp() ---> Empty object

What does a . empno = 25 do ? ---> Adds variable empno to object 'a' with value 25

What does a . empno = 35 do ? ---> Modifies value of empno in object 'a' to 35

8) object . new-variable = 25

What does the above statement do ? ---> Adds a new variable to the object with value 25

9) object . existing-variable = 35

What does the above statement do ? ---> Modifies value of the variable to 35

10) How to delete instance variable of the object ? ---> del object . variable

11) What does del a . empno do ? ---> Removes variable empno from object 'a'

12) del object . valid-variable

What does the above statement do ? ---> Removes variable from the object

13) del object . invalid-variable

What does the above statement do ? --->Throws AttributeError

14) Python objects are growable and shrinkable

i.e. New variables can be added to the object and existing variables can be removed from the object any where in the program

15) In other words, python objects are dynamic

Note:

What about java and c++ objects ? ---> They are neither growable nor shrinkable i.e. static objects

# Identify error (Home work)

```
class c1:
        def m1(self):  # Empty method
                pass
class c2: # Empty class
    pass
class c3: # Error : No methods in the class nor pass
```

Empty class

---

1) What is an empty class ? ---> A class without methods

2) How is an empty class denoted ? ---> class c1:
Pass

# Find outputs (Home work)

```
class c1: # Empty class
        pass
# End of the class

a = c1()  # Creates empty c1 class object
print(id(a)) # Address of object 'a'
print(type(a)) # Type of object 'a' i.e. <class '__main__ . c1'>
print(a . __dict__) # Converts empty object 'a' to {}
print(a) # __str__() method of object class returns type and address of object 'a'
del a # Deletes object 'a'
#print(a) # Error : Object 'a' does not exist
```

1) a = c1()

print(a)

Which method is executed ? ---> __str__() method of object class becoz there is no __str__() method in class c1

2) What is object class ? ---> Parent class to all python classes

3) Where is object class defined ? ---> In builtins module

4) What does __str__() method of object class do ? ---> Returns type and address of the object

Eg: class object:

def __str__(self):

returns type and address of object

'''

Method Vs Function

---

1) class c1:

def m1(self):

statements

# End of the class

def f1():

statements

2) Where is a method defined ? ---> Inside the class

Where is a function defined ? ---> Outside the class

3) Method is a member of what ? ---> Class

Function is a member what ? ---> Module

4) How is a method called ? ---> object . method() (or) classname . method(object)

How is a function called ? ---> function()

5) What is the mandatory argument for a method ? ---> self which is a user defined word

Does function have self argument ? ---> No becoz it is not called wrt object

6) a = c1()

a . m1()

Where is m1() searched ? ---> In class c1 becoz it is called wrt c1 class object 'a'

f1()

Where is f1() searched ? ---> In same module i.e. current module

7) What is the issue with m1() ? --->

Executes function m1() and throws error when there is no function m1()

a = c1()

What is the issue with a . f1() ? --->

Executes method f1() of class c1 and throws error when there is no method f1() in class c1

8) What is the pre-requisite to call a method ? ---> There should be at least one object

What about a function ? ---> It can be called directly even though there are no objects

9) What does a method do ? ---> Performs operation on object such as read object , print object and so on

What does a function do ? ---> It is a general purpose function and does not perform any operation on object

10) Is sort() a method (or) function ? ---> A method of list class

What about sorted() ? ---> A function of builtins module

# Find outputs (Home work)

```
def m1():
    print('Function')
class c1:
        def m1(self): # Discarded : Another method is defined with same name
            print('1st method')
        def m1(self): # Discarded : Another method is defined with same name
            print('2nd method')
        def m1(self): # Recognized : The last method
            print('3rd method')
# End of class c1
a = c1() # Creates c1 class object
a . m1() # Executes the last m1() method of class c1 i.e. 3rd method
m1() # Executes m1() function
```

3rd method

Function

Which method is executed when methods have same name ? ---> The last method

```
# Find outputs (Home work)
class c1:
        def m1(self): # Discarded : Another method is defined with same name
                print('No argument method')
        def m1(self , x): # Discarded : Another method is defined with same name
                print('Single argument method : ' , x)
        def m1(self , x , y): # Recognized : The last method
                print('Two argument method : ' , x , y)
# End of class c1
a = c1() # Creates c1 class object
a . m1(10 , 20) # Two argument method : <space> 10 <space> 20
#a . m1(30) # Error : Arg is not passed for 'y'
#a . m1() # Error : Args are not passed for 'x' and 'y'
```

1) Does python support method overloading ? ---> No

2) Does python support function overloading ? ---> No

3) Does python support operator overloading ? ---> Yes

```
# Find outputs (Home work)
class c1:
        def m1(self):
                print('No argument method')
        def m1(self , x):
                print('Single argument method : ' , x)
        def m1(self , x = 1 , y = 2):
                print('Two argument method : ' , x , y)
# End of class c1
a = c1() # Creates c1 class object
a . m1(10 , 20) # Two argument method : <space> 10 <space> 20
a . m1(30) # Two argument method : <space> 30 <space> 2
a . m1() # Two argument method : <space> 1 <space> 2
```

```
# Find outputs (Home work)
class c1: # Discarded : Another class is defined with same name
    def m1(self):
        print('Method of first c1 class')
class c1: # Discarded : Another class is defined with same name
    def m1(self):
        print('Method of second c1 class')
class c1: # Recognized : The last class
    def m1(self):
        print('Method of third c1 class')
a = c1() # 3rd c1 class object
a . m1() # Method of third c1 class
```

1) Which class is recognized when multiple classes have same name ? ---> The last class

2) Can two classes have same name in c++ and java ? ---> No

```
# Find outputs (Home work)
class c1: # Discarded : Another class is defined with same name
    def m1(self):
        print('Method of first c1 class')
class c1: # Discarded : Another class is defined with same name
    def m1(self):
        print('Method of second c1 class')
class c1: # Recognized : The last class
    pass
a = c1() # Creates 3rd c1 class object
#a . m1() # Error : No m1() method in 3rd c1 class

# Find outputs (Home work)
class c1:
    pass
```

```
# End of class
a = c1() # Creates an empty c1 class object
print(a . __dict__) # { }
a . x = 10 # Adds variable 'x' to object 'a' with value 10
print(a . __dict__) # {'x' : 10}
a . y = 20 # Adds variable 'y' to object 'a' with value 20
print(a . __dict__) # {'x' : 10, 'y' : 20}
a . x = 30 # Modifies a . x to 30
print(a . __dict__) # {'x' : 30, 'y' : 20}
a . y = 40 # Modifies a . y to 40
print(a . __dict__) # {'x' : 30, 'y' : 40}
del a . x # Removes variable 'x' from object 'a'
print(a . __dict__) # {'y' : 40}
del a . y # Removes variable 'y' from object 'a'
print(a . __dict__) # {}
del a # Deletes object 'a'
#print(a . __dict__) # Error : Object 'a' does not exist
```

__dict__ variable

---

1) What is __dict__ variable ? ---> A dictionary representation of the object

2) What does object . __dict__ do ? ---> Converts object to dictionary

3) What are the keys of dictionary ? ---> All the instance variables of the object

   What are the values of dictionary ? ---> Values of instance variables

4) If object 'a' has x = 10 and y = 20

   What is the result of a . __dict__ ? ---> {'x' : 10 , 'y' : 20}

''' (Home work)

Write a program to determine area and perimeter of triangle and represent triangle by an object

1) What is the area of triangle ? ---> sqrt(s (s - a) (s - b) * (s - c))

2) What is the formula for 's' ? ---> (a + b + c) / 2

3) What is the perimeter of triangle ? ---> a + b + c

```
import math
class triangle:
  def get(self):
    self . a = eval(input('Enter first side of triangle : ')) # Adds variable 'a' to object self with user input
    self . b = eval(input('Enter second side of triangle : ')) # Adds variable 'b' to object self with user input
    self . c = eval(input('Enter third side of triangle : ')) # Adds variable 'c' to object self with user input
  def test(self):
    if self . a + self . b > self . c and self . b + self . c > self . a and self . c + self . a > self . b:
      pass
    else:
      print('Not a triangle')
      exit()
      def area(self):
                  s = (self . a + self . b + self . c) / 2
                  return math . sqrt(s * (s - self . a) * (s - self . b) * (s - self . c))
      def peri(self):
                  return self . a + self . b + self . c
# End of the class
if __name__ == '__main__': # True when prog5a is executed and False when prog5a is imported
      t = triangle() # Create an empty triangle class object
      t . get() # Reads inputs to object 't'
      t . test() # Are values of object 't' valid ?
      print('Area : ', t . area()) # Area of values in object 't'
      print('Perimeter : ', t . peri()) # Perimeter of values in object 't'
```

Repeat prog5a such that methods are called in another way

1) What are the two ways to call a method ? ---> object . method() and classname . method(object)

2) Import triangle class defined in prog5a but do not define triangle class again

```
from prog5a import triangle # Imports triangle class and the if statement present outside triangle class

t = triangle() # Creates empty triangle class object

triangle . get(t) # Executes get() method of triangle class and passes object 't' to the method

triangle . test(t) # Executes test() method of triangle class and passes object 't' to the method

print('Area : ', triangle . area(t) ) # # Executes area() method of triangle class and passes object 't' to the method

print('Perimeter: ', triangle . peri(t)) # Executes peri() method of triangle class and passes object 't' to the method
```

1) from prog5a import triangle

What are imported ? ---> triangle class and the if statement outside triangle class

2) py prog5a.py

What is the value of __name__ variable ? ---> '__main__' becoz prog5a is executed

3) from prog5a import triangle

What is the value of __name__ variable ? ---> Module name (i.e. 'prog5a') becoz triangle class is imported from prog5a

Is if statement executed ? ---> No becoz if condition is false

*4) In general, statements outside the class should be under if condition so that they are not executed when the program (or) module is imported

5) classes , functions and variables defined in a program can be reused by a different program.

This is possible with import (or) from statement

6) import prog5a

What are imported ? ---> prog5a and if statement

Is triangle class imported ? ---> No becoz members are not imported

How to use triangle class ? ---> prog5a . triangle

```
# Find outputs (Home work)
class c1:
        def m1(self): # self is object 'a'
                x = 10 # Lv of m1() method
                self . x = 20 # Adds variable 'x' to object 'a' with value 20
                print(x) # Lv i.e. 10
                print(self . x) # a . x i.e. 20
                x += 5 # Modifies Lv to 15
                self . x += 7 # Modifies a . x to 27
  # Lv 'x' is lost
        def m2(self):
                #print(x) # Errror : No Lv 'x' in method m2()
                print(self . x) # a . x i.e. 27
                self . x += 6 # Modifies a . x to 33
        # Nothing is lost
# End of the class
a = c1() # Creates an empty c1 class object
a . m1() # Executes m1() method of class c1
a . m2() # Executes m2() method of class c1
print(a . x) # 33
#print(self . x) # Error : No self outside the class
#print(x) # Error : No Gv 'x'
```

```
10
20
27
33
```

Local variable ---> x = 10 ---> 15

Object a ---> x = 20 ---> 27 ---> 33

1) Who can use self ? ---> Just method

2) Who can not use self ? ---> Function and outside the class

Local Variable Vs Instance Variable

---

1) class c1:

    def m1(self):

        self . x = 10

        y = 20

What is 'x' called ? ---> Instance variable due to self . x

What is 'y' called ? ---> Local variable

2) What is an instance variable ? ---> A variable of the object

What is a local variable ? ---> A variable of function (or) method

3) How is an instance variable initialized ? ---> object . variable = value

How is a local variable initalized ? ---> variable = value

4) Where can instance variable be initialized ? ---> Any where in the program

Where can local variable be initialized ? ---> Inside function (or) method

5) What does object contain ? ---> Instance variables but not local and global variables

6) Where is instance variable represented ? ---> Inside the object

Where is local variable represented ? ---> Outside the object

7) How is an instance variable accessed ? ---> object . variable

How is local variable accessed ? ---> Just Variable

8) Who can use instance variable ? ---> Whole program i.e. Whereever object is visible

Who can use local variable ? ---> That method (or) function where where the variable is initialized

9) What does object . __dict__ return ? ---> All the instance variables of object but not local variables

10) Can instance variable and local variable have same name ? ---> Yes

11) How are they distinguished when they have same name ? --->

Thru object i.e. obj . variable for instance variable

and

just variable for local variable

12) Instance variable is a member of what ? ---> Object

Local variable is a member of what ? ---> Function (or) method

# object 't' ---> a = 3 , b = 4 , c = 8

1) What are a , b , c called ? ---> Instance variables of object 't'

2) What is 's' called ? ---> Local variable of area() method

3) What is the issue with t . s = 10 ? ---> Adds variable 's' to object 't'

but triangle class object should have three variables but not four

4) How many variables are in object 't' after execution of t = triangle() ? --->

Nothing becoz every object is initially empty

5) How many variables are in object 't' after execution of t . get() ? --->

Three Variables a , b and c becoz get() method adds three variables to object 't'

(Home work)

Write a program to add two objects where each object contains three values and store results in third object

1st object ---> x = 10 , y = 20 , z = 30

2nd object ---> x = 40 , y = 50 , z = 60

3rd object ---> x = 10 + 40 = 50 , y = 20 + 50 = 70 , z = 30 + 60 = 90

```
class Test:
    def get(self):
        self . x = eval(input('Enter 1st number:')) # Adds variable 'x' to object self with user input
        self . y = eval(input('Enter 2nd number:')) # Adds variable 'y' to object self with user input
        self . z = eval(input('Enter 3rd number : ')) # Adds variable 'z' to object self with user input
    def add(self , m , n): #  self is object 'c' , 'm' is object 'a' , 'n' is object 'b'
                self . x = m . x + n . x # Adds variable 'x' to object self with the result
                self . y = m . y + n . y # Adds variable 'y' to object self with the result
                self . z = m . z + n . z # Adds variable 'z' to object self with the result
    def disp(self): #  self is object 'c'
                print('x : ' , self . x) # prints c . x
                print('y : ' , self . y) # prints c . y
                print('z : ' , self . z) # prints c . z
# End of the class
a = Test() # Create three empty Test class objects
b = Test()
```

c = Test()

print('First Object')

a . get() # Reads inputs to object 'a'

print('Second Object')

b . get() # Reads inputs to object 'b'

c . add(a , b) # Adds objects a and b and stores results in object 'c'

print('Addition results')

c . disp() # Prints values of object 'c'

object 'a' ---> x = 10 , y = 20 , z = 30

object 'b' ---> x == 40 , y = 50 , z = 60

object 'c' ---> x = 50 , y = 70 , z = 90

1) Is c = a + b valid ? ---> No becoz there is no __add__() method in Test class

2) What does print(c) do ? ---> Executes __str__() method of object class which returns type and address of object 'c'

Types of objects

---

1) Owner object (or) Invoking object

2) Parameter object

Method call : a . m1(b)

---

1) What is 'a' called ? ---> Owner object becoz method is called wrt object 'a'

2) What is 'b' called ? ---> Parameter object becoz it is being passed to m1() method

3) How to access variables of object 'b' in m1() method ? ---> b . variable

4) How to access variables of object 'a' in m1() method ? ---> self . variable where self is object 'a'

# Find outputs (Home work)

class Date:

```
        def   disp(self):
                print('dd : ' , self . dd)
                print('mm : ' , self . mm)
                print('yy : ' , self . yy)
        def  __str__(self):
                return  F'{self . dd}-{self . mm}-{self . yy}'
#end of the class
a =  Date() # Creates  a  Date  class  object
a . dd = 15 # Adds  variable  dd  to  object  'a'  with  value  15
a . mm = 8  # Adds  variable  mm  to  object  'a'  with  value  8
a . yy = 1947  # Adds  variable  yy  to  object  'a'  with  value  1947
a . disp()  #  Executes  method  of  Date class
print(a) #  __str__()  method  of  Date  class  returns  '15-8-1947'
print(a . __str__())  #  __str__()  method  of  Date  class  returns  '15-8-1947'
print(a . __dict__) # Converts  object 'a'  to  dictionary

dd : 15
mm : 8
yy : 1947
15 - 8 - 1947
15 - 8 - 1947
{'dd' : 15 , 'mm' : 8 , 'yy' : 1947}

# object  'a'  --->  dd = 15 , mm = 8 , yy = 1947
1) How  to  print  object  in  dictionary  form ?  --->  print(object . __dict__)
2)        How  to  print  object  in  string  form ?  ---> print(object)
3) How  to  print  object  in  object  form ?  ---> object . disp()

#  Find  outputs (Home  work)
class  Date:
        pass
```

```
# End of the class

a = Date() # Creates an empty Date class object

a . dd = 15 # Adds variable dd to object 'a' with value 15

a . mm = 8 # Adds variable mm to object 'a' with value 8

a . yy = 1947 # Adds variable yy to object 'a' with value 1947

print(a) # Executes __str__() method of object class which returns type and address of object 'a'
```

1) Why is __str__() method of object class executed ? ---> Since there is no __str__() method in Date class

2) What does __str__() method of object class do ? ---> Returns type and address of the object

```
# Find outputs (Home work)

class c1:
        def __str__(self): # self is object 'a'
                return '25'

class c2:
        def __str__(self): # self is object 'b'
                return 35

class c3:
        def __str__(self): # self is object 'c'
                print('Hyd')

class c4:
        def __str__(self , x): # self is object 'd' and 'x' is 50
                return F'{x}'

#end of the class

a = c1() # Creates an empty c1 class object

b = c2() # Creates an empty c2 class object

c = c3() # Creates an empty c3 class object

d = c4() # Creates an empty c4 class object

print(a) # Executes __str__() method of c1 class which returns '25'
```

#print(b) # Error : __str__() method of class c2 returns non-string i.e. 35

#print(c) # Error : __str__() method of class c3 returns non-string i.e. None

#print(d) # Error : __str__() method of class c4 has an arg 'x'

print(b . __str__()) # Executes __str__() method of class c2 which returns 35

print(c . __str__()) # Executes __str__() method of class c3 which prints 'Hyd' and returns None

print(d . __str__(50)) # Executes __str__() method of class c4 which returns '50'

25

35

Hyd

None

50

1) Can __str__() method return non-string ? ---> Yes when __str__() method is called explicitly and no when __str__() method is automatically executed

2) Can __str__() method have an argument ? ---> Yes when __str__() method is called explicitly and no when __str__() method is automatically executed

Write a program to determine total , average and grade of a student

Inputs are Roll Number , Stud Name , Marks of 3 subjects and Gender

```
class student:
    def get(self): # self is object 's'
        self . rno = int(input('Enter roll number: ')) # Adds variable rno to object 's' with user input
        self . sname = input('Enter student name : ') # Adds variable sname to object 's' with user input
        self . gender = input(('Enter gender (m/f) :')) # Adds variable gender to object 's' with user input
        self . m = [] # Adds empty list 'm' to object 's'
        for i in range(3): # Marks of 3 subjects
            marks = int(input(F'Enter marks of subject {i + 1} : ')) # Reads input to local variable
            self . m . append(marks) # Appends value of Lv to list s . m
    def compute(self): # self is object 's'
```

```
        self . tot = sum(self . m) # Adds variable tot to object 's' with sum of marks
        self . avg = self . tot / 3 # Adds variable avg to object 's' with average marks
        if min(self . m) < 40:  # At least one subject is below 40 marks
                self . grade = 'Fail' # Adds variable grade to object 's' with 'Fail'
        elif self . avg >= 70:
          self . grade = 'Distinction' # Adds variable grade to object 's' with 'Distinction'
        elif self . avg >= 60:
          self . grade = 'First class' # Adds variable grade to object 's' with 'First class'
        elif self . avg >= 50:
          self . grade = 'Second class' # Adds variable grade to object 's' with 'Second class'
        else:
          self . grade = 'Third class'  # Adds variable grade to object 's' with 'Third class'
    def disp(self): # self is object 's'
        print('Roll Number : ' , self . rno)
        print('Student Name : ' , self . sname)
        print('Gender : ' , self . gender)
        print('Total Marks : ' , self . tot)
        print(F'Average : {self . avg:.2f}')
        print('Grade : ' , self . grade)
    def __str__(self):
        return F'{self . rno} \t {self . sname} \t {self . gender} \t {self . tot} \t {self . avg:.2f} \t {self . grade}' # Concatenates all the value of object self to form a string
#End of the class
if __name__ == '__main__': # True when prog9a is executed and False when prog9a is imported
    s = student() # Creates an empty student class object
    s . get() # Reads inputs to object 's'
    s . compute() # Stores results in object 's'
    s . disp() # Prints values of object 's'
    print(s) # Executes __str__() method of student class which returns all the values of object 's' in the form of string
```

# Object 's' ---> rno = 25 , sname = 'Rama Rao' , gender = 'm' , m = [52,48,55] , total = 155 , average = 51.66 , grade = '2nd class'

Can Fail be handled at the end ? ---> No and it should be handled only at the beginning

'''

Write a program to add , subtract , multiply and divide two rational numbers

1) 1st rational number ---> 2 / 3

2nd rational number ---> 5 / 9

What is the sum ? ---> 2 / 3 + 5 / 9 = (18 + 15) / 27 = 33 / 27 = 11 / 9

What is the difference ? ---> 2 / 3 - 5 / 9 = (18 - 15) / 27 = 3 / 27 = 1 / 9

What is the product ? ---> 2 / 3 * 5 / 9 = 10 / 27 = 10 / 27

What is the division ? ---> 2 / 3 / 5 / 9 = 2 / 3 * 9 / 5 = 18 / 15 = 6 / 5 ---> Succesful division and return True

2) 1st rational number ---> 2 / 3

2nd rational number ---> 0 / 9

What is the sum ? ---> 2 / 3 + 0 / 9 = (18 + 0) / 27 = 18 / 27 = 2 / 3

What is the difference ? ---> 2 / 3 - 0 / 9 = (18 - 0) / 27 = 18 / 27 = 2 / 3

What is the product ? ---> 2 / 3 * 0 / 9 = 0 / 27 = 0 / 27 ---> Simplification is not required becoz numerator is 0

What is the division ? ---> 2 / 3 / 0 / 9 = 2 / 3 * 9 / 0 = 18 / 0 ---> Division is not permitted and return False

3) When is simplification required ? ---> When numerator is non-zero

4) What does div() method return ? ---> True when division is succesful and False otherwise

```
import math

class rat:

        def get(self):

                self . nr = int(input('Enter numerator : ')) # Adds variable nr to object self with user input

                self . dr = int(input('Enter denominator : ')) # Adds variable dr to object self with user input

                self . test() # Is denom of object self zero

        def test(self):
```

```
            while self . dr == 0: # Repeat until dr is non-zero
                self . dr = int(input('Denom can not be zero , reenter : ')) # Reads non-zero denominator to object self
    def __str__(self):
                return F'{self . nr} / {self . dr}' # Concatenates values of self to form a string with '/'
    def add(self , a , b):
            self . nr = a . nr * b . dr + a . dr * b . nr # Adds variable nr to object self with the result
            self . dr = a . dr * b . dr # Adds variable dr to object self with the result
            self . simplify() # Simplifies values of object self

    c . add(a , b)
    object a ---> 2 / 3
    object b ---> 5 / 9
    object c ---> 2 / 3 + 5 / 9 = (2 * 9 + 5 * 3) / (5 * 9) = 33 / 27 = 11 / 9
    def sub(self , a , b):
            self . nr = a . nr * b . dr - a . dr * b . nr # Adds variable nr to object self with the result
            self . dr = a . dr * b . dr # Adds variable dr to object self with the result
            self . simplify() # Simplifies values of object self
    d . sub(a , b)
    object a ---> 2 / 3
    object b ---> 5 / 9
    object d ---> 2 / 3 - 5 / 9 = (2 * 9 - 5 * 3) / (5 * 9) = 3 / 27 = 1 / 9
    def mul(self , a , b):
            self . nr = a . nr * b . nr # Adds variable nr to object self with the result
            self . dr = a . dr * b . dr # Adds variable dr to object self with the result
            self . simplify() # Simplifies values of object self
    e . mul(a , b)
    object a ---> 2 / 3
    object b ---> 5 / 9
```

```
        object  e  --->  2 / 3 * 5 / 9 = (2 * 5) / (3 * 9) = 10 / 27
        def    div(self , a , b):
                self . nr = a . nr * b . dr  # Adds  variable  nr  to  object  self  with  the  result
                self . dr = a . dr * b . nr  # Adds  variable  dr  to  object  self  with  the  result
                self . simplify()  #  Simplifies  values  of  object  self
        f . div(a , b)
        object  a  --->  2 / 3
        object  b  --->  5 / 9
        object  f  --->  2 / 3 / 5 / 9 = 2 / 3 * 9 / 5 = (2 * 9) / (3 * 5) = 18 / 15 = 6 / 5

        def   simplify(self):
                if  self . nr != 0:
                        ans = math . gcd(self . nr , self . dr)  #  gcd  of  values  of  self
                        self . nr = self . nr // ans  #  Simplifies  nr  of  object  self
                        self . dr = self . dr // ans   #  Simplifies  dr  of  object  self
        c . simplify()
        1)  12 / 15  --->  4 / 5
        2) 10 / 27   --->  10 / 27
        3) 0 / 27  --->   0 / 27
        '''
# End  of the class
if __name__ == '__main__': # True when prog10a is executed and False when prog10a is imported
        a = rat() #  Creates  6  empty  rat  class  objects
        b = rat()
        c = rat()
        d = rat()
        e = rat()
        f = rat()
        a . get()  #  Reads  inputs  to  object  'a'
        b . get()  #  Reads  inputs  to  object  'b'
```

c . add(a , b) # Adds objects 'a' and 'b' and stores results in object 'c'

d . sub(a , b) # Subtracts objects 'a' and 'b' and stores results in object 'd'

e . mul(a , b) # Multipliees objects 'a' and 'b' and stores results in object 'd'

f . div(a , b) # Divides objects 'a' and 'b' and stores results in object 'f'

print('Sum : ' , c) # __str__() method of rat class returns values of object 'c' in the form of string

print('Difference : ' , d) # __str__() method of rat class returns values of object 'd' in the form of string

print('Product : ' , e) # __str__() method of rat class returns values of object 'e' in the form of string

if b . nr != 0:

print('Division : ' , f) # __str__() method of rat class returns values of object 'f' in the form of string

else:

print('Division is not permitted')

1) Can a method call another method of same class ? ---> Yes with self . method()

2) get() calls which method ? ---> Method test() of the same class

3) If get() method is called wrt obj 'a' ,

test() method is called wrt which object ? ---> Same object 'a' due to self . test()

4) add() calls which method ? ---> Method simplify() of the same class

5) If add() method is called wrt object 'c',

simplify() method is called wrt which object ? ---> Same object 'c' due to self . simplify()

6) In which order can methods of the class be defined ? ---> Any order

7) Can a method be called before it is defined ? ---> Yes

Can a function be called before it is defined ? ---> No

dir() function

---

1) What does dir(classname) do ? ---> Returns a list of all the methods of the class

2) What does dir(int) do ? ---> Returns a list of all the methods of int class

3) What does dir(obj) do ? ---> Returns a list of all the instance variables of the object and methods of the clas

4) a = 3 + 4j

What does dir(a) do ? ---> Returns a list of all the instance variables of the object 'a' (i.e. real and imag) and methods of complex class

5) What does dir(module) do ? ---> Returns a list of all the members of the module and environment variables

6) What does dir(No-args) do ? ---> Returns a list of all the members of current module and environment variables

7) Where is dir() function defined ? ---> In builtins module

8) What is the argument of dir() function ? ---> classname , object (or) module

```
# setattr() , getattr() and hasattr() functions demo program

class Emp:

    pass

# End of the class

e = Emp() # Creates an empty Emp class object

print(e . __dict__) # { }

setattr(e , 'empno' , 25) # Adds variable empno to object 'e' with value 25 (Same as e . empno = 25)

setattr(e , 'ename' , 'Rama Rao') # Adds variable ename to object 'e' with value 'Rama Rao' (Same as e . ename = 'Rama Rao')

setattr(e , 'sal' , 10000.0) # Adds variable sal to object 'e' with value 10000.0 (same as e . sal = 10000.0)

setattr(e , 'sal' , 20000.0) # Modifies variable sal of object 'e' to 20000.0 (same as e . sal = 20000.0)

print(e . __dict__) # {'empno' : 25 , 'ename' : 'Rama Rao' , 'sal' : 20000.0}

print('Emp number : ' , getattr(e , 'empno')) # Returns value of empno in object 'e' i.e. 25

print('Emp name : ' , getattr(e , 'ename')) # Returns value of ename in object 'e' i.e. Rama Rao

print('Salary : ' , getattr(e , 'sal')) # Returns value of sal in object 'e' i.e. 20000.0

#print('Gender : ' , getattr(e , 'gender')) # Error : No variable gender in object 'e'

print(hasattr(e , 'empno')) # True : Variable empno exists in object 'e'

print(hasattr(e , 'ename')) # True : Variable ename exists in object 'e'

print(hasattr(e , 'sal')) # True : Variable sal exists in object 'e'

print(hasattr(e , 'gender')) # False : No variable gender in object 'e'
```

setattr() function

---

1) What does setattr() function do ? ---> Adds variable(i.e. instance vaiable) to the object at runtime

2) What are the three arguments of setattr() function ? ---> Object , variable and value of the variable

3) What does setattr(object , new-variable , value) do ? ---> Adds new variable to the object with the value

4) What does setattr(object , valid-variable , value) do ? ---> Modifies value of the variable in the object

5) obj . variable = value

setattr(object , variable , value)

What is the difference between the two statements ? --->

obj . variable = value ---> Adds variable to the object at design time setattr() function adds variable to the object at runtime

6) object . variable = value ---> Programmer adds variable to the object at design time

setattr(object , input() , input()) ---> Client adds variable to the object at runtime

7) Can setattr() function add method to the class ? ---> No

8) Where is setattr() function defined ? ---> In builtins module

getattr() function

---

1) What does getattr() function do ? ---> Returns value of variable in the object

2) What are the two arguments of getattr() function ? ---> Object and Variable

3) What does getattr(object , valid-variable) do ? ---> Returns value of the variable which is in the object

What does getattr(object , invalid-variable) do ? ---> Throws AttributeError

4) print(object . variable)

print(getattr(object , variable))

What is the difference between the above two statements ? --->

object . variable returns value of the variable at design time but getattr() returns value of the variable at runtime

5) object . variable ---> Programmer decides which variable is to be accessed at design time

getattr(object , input()) ---> Client decides which variable is to be accessed at runtime

6) Where is getattr() function defined ? ---> In builtins module

hasattr() function

---

1) What does hasattr(object , variable) do ? ---> Returns True if the variable is present in the object and False otherwise

2) What does hasattr(classname , method) do ? --->

Returns True if the method is present in the class and False otherwise

3) What does hasattr(object , method) do ? ---> Returns True if the method is present in the class and False otherwise

4) What does hasattr(classname , variable) do ? ---> Returns False becoz class does not contain variable

5) What is the second argument of hasattr() function if 1st argument is object ? ---> Variable (or) method

What is the second argument of hasattr() function if 1st argument is classname ? ---> Just method but not variable

6) Where is hasattr() function defined ? ---> In builtins module