



Department of Computing Technologies

SRM IST, Kattankulathur – 603 203

Course Code: 18CSC207J

Course Name: Advanced Programming Practice

Experiment No	3
Title of Experiment	To complete all the 8 problems in Jupyter environment
Name of the Student	Abhijay Rajvansh (RA2011003010398)
Date of Experiment	07 - 03 - 2022

Staff Signature with date

1.

Aim: Write a Python class named SRMIST with six attributes school, dept1, dept2, dept2 and dept3. Add a new attribute specialization and display the entire attribute and their values of the said class. Now remove the dept1 and dept2 attribute and display the entire attribute with values.

CODE:

```
class SRMIST:
    school = 'SRMIST'
    dept1 = 'Computer Science'
    dept2 = 'Artificial Intelligence'
    dept3 = 'Mechanical Engineering'
    dept4 = 'Biotech'
print("Original attributes and their values of the Student class:")
for attr, value in SRMIST.__dict__.items():
    if not attr.startswith('_'):
        print(f'{attr} -> {value}')
print("\nAfter adding the specialization, attributes and their values with
the said class:")
SRMIST.specialization = 'Blockchain'
for attr, value in SRMIST.__dict__.items():
    if not attr.startswith('_'):
        print(f'{attr} -> {value}')
print("\nAfter removing the dept1,dept2 attributes and their values from
the said class:")
del SRMIST.dept1
del SRMIST.dept2
#delattr(Student, 'student_name')
for attr, value in SRMIST.__dict__.items():
    if not attr.startswith('_'):
        print(f'{attr} -> {value}')
```

```
In [1]: class SRMIST:
        school = 'SRMIST'
        dept1 = 'Computer Science'
        dept2 = 'Artificial Intelligence'
        dept3 = 'Mechanical Engineering'
        dept4 = 'Biotech'
        print("Original attributes and their values of the Student class:")
        for attr, value in SRMIST.__dict__.items():
            if not attr.startswith('_'):
                print(f'{attr} -> {value}')
        print("\nAfter adding the specialization, attributes and their values with the said class:")
        SRMIST.specialization = 'Blockchain'
        for attr, value in SRMIST.__dict__.items():
            if not attr.startswith('_'):
                print(f'{attr} -> {value}')
        print("\nAfter removing the dept1,dept2 attributes and their values from the said class:")
        del SRMIST.dept1
        del SRMIST.dept2
        #delattr(SRMIST, 'student_name')
        for attr, value in SRMIST.__dict__.items():
            if not attr.startswith('_'):
                print(f'{attr} -> {value}')

        Original attributes and their values of the Student class:
        school -> SRMIST
        dept1 -> Computer Science
        dept2 -> Artificial Intelligence
        dept3 -> Mechanical Engineering
        dept4 -> Biotech

        After adding the specialization, attributes and their values with the said class:
        school -> SRMIST
        dept1 -> Computer Science
        dept2 -> Artificial Intelligence
        dept3 -> Mechanical Engineering
        dept4 -> Biotech
        specialization -> Blockchain

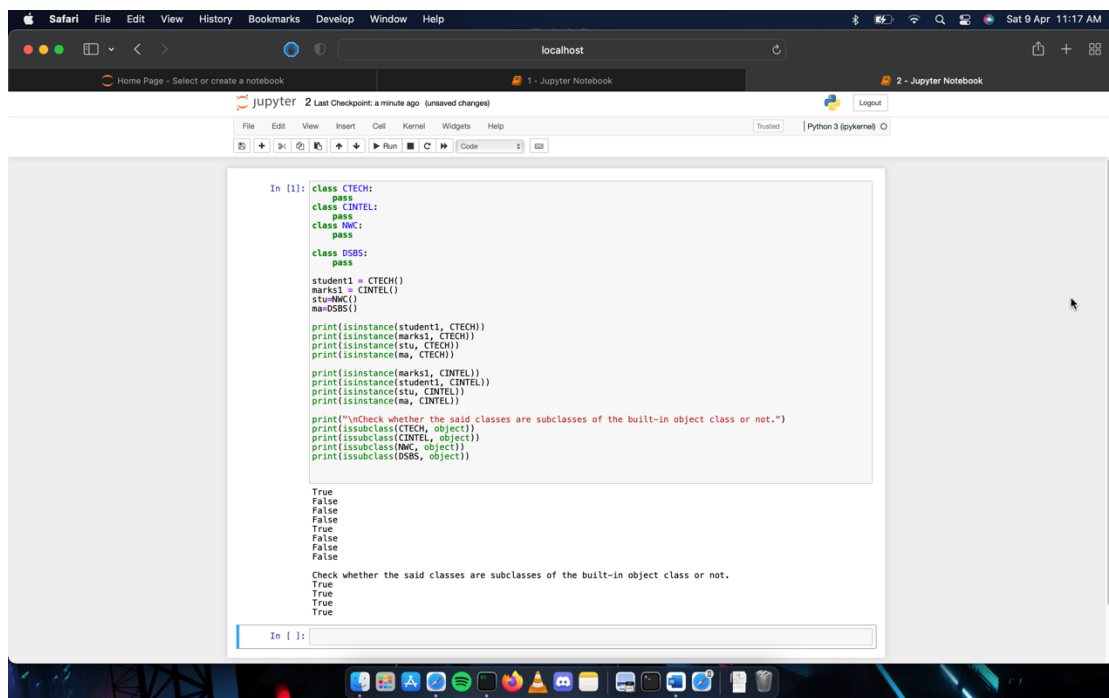
        After removing the dept1,dept2 attributes and their values from the said class:
        school -> SRMIST
        dept3 -> Mechanical Engineering
        dept4 -> Biotech
        specialization -> Blockchain
```

Result: Python class named SRMIST with six attributes school, dept1, dept2, dept2 and dept3 was completed.

2.

Aim: Write a Python program to create four empty classes, CTECH, CINTEL, NWC and DSBS. Now create some instances and check whether they are instances of the said classes or not. Also, check whether the said classes are subclasses of the built-in object class or not.

CODE:

A screenshot of a Jupyter Notebook interface in a Safari browser. The notebook contains a single code cell with the following Python code:

```
In [1]: class CTECH:
        pass
class CINTEL:
    pass
class NWC:
    pass
class DSBS:
    pass

student1 = CTECH()
marks1 = CINTEL()
stu=NWC()
na=DSBS()

print(isinstance(student1, CTECH))
print(isinstance(marks1, CTECH))
print(isinstance(stu, CTECH))
print(isinstance(na, CTECH))

print(isinstance(marks1, CINTEL))
print(isinstance(student1, CINTEL))
print(isinstance(stu, CINTEL))
print(isinstance(na, CINTEL))

print("\nCheck whether the said classes are subclasses of the built-in object class or not.")
print(issubclass(CTECH, object))
print(issubclass(CINTEL, object))
print(issubclass(NWC, object))
print(issubclass(DSBS, object))

True
False
False
False
True
False
False
False
False

Check whether the said classes are subclasses of the built-in object class or not.
True
True
True
True
```

The output of the code is displayed below the code cell, showing the results of the isinstance and issubclass checks.

Result: Python program to create four empty classes, CTECH, CINTEL, NWC and DSBS. Now create some instances and check whether they are instances of the said classes or not was completed.

3.

Aim: Write a program to print the names of the departments students by creating a Dept class. If no name is passed while creating an object of the Dept class, then the name should be "SCO", otherwise the name should be equal to the String value passed while creating the object of the Dept class.

CODE:

```
class Dept:
    def __init__(self, *args):
        if len(args) == 1:
            self.dept=args[0]

        elif len(args) == 0:
            self.dept="SCO"

    def deptname(self):
        print(self.dept)

d1=Dept()
d1.deptname()

d2=Dept("CSE")
d2.deptname()
```

```
In [1]: class Dept:
def __init__(self, *args):
    if len(args) == 1:
        self.dept=args[0]
    elif len(args) == 0:
        self.dept="SC0"
def deptname(self):
    print(self.dept)

d1=Dept()
d1.deptname()

d2=Dept("CSE")
d2.deptname()

SC0
CSE

In [ ]: |
```

Result: program to print the names of the departments students by creating a Dept class. If no name is passed while creating an object of the Dept class, then the name should be "SC0", was completed.

4.

Aim: Create a class named 'Rectangle' with two data members- length and breadth and a function to calculate the area which is 'length*breadth'.

The class has three constructors which are :

1 - having no parameter - values of both length and breadth are assigned zero.

2 - having two numbers as parameters - the two numbers are assigned as length and breadth respectively.

3 - having one number as parameter - both length and breadth are assigned that number.

Now, create objects of the 'Rectangle' class having none, one and two parameters and print their areas.

CODE:

```
def areaRectangle(a, b):  
    return (a * b)
```

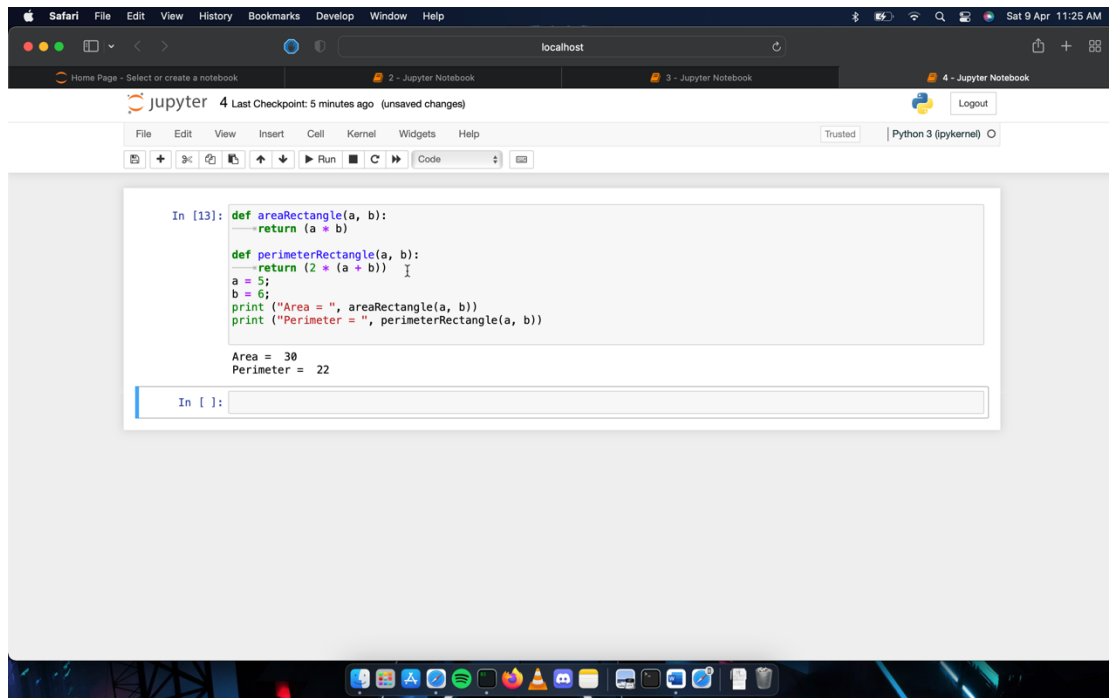
```
def perimeterRectangle(a, b):  
    return (2 * (a + b))
```

```
a = 5;
```

```
b = 6;
```

```
print ("Area = ", areaRectangle(a, b))
```

```
print ("Perimeter = ", perimeterRectangle(a, b))
```



The screenshot shows a Jupyter Notebook running in a Safari browser. The notebook has a title bar with 'Jupyter' and '4 Last Checkpoint: 5 minutes ago (unsaved changes)'. The code cell contains the following Python code:

```
In [13]: def areaRectangle(a, b):  
         return (a * b)  
  
         def perimeterRectangle(a, b):  
             return (2 * (a + b))  
         a = 5;  
         b = 6;  
         print ("Area = ", areaRectangle(a, b))  
         print ("Perimeter = ", perimeterRectangle(a, b))
```

The output of the code cell is:

```
Area = 30  
Perimeter = 22
```

Below the output, there is an input field for the next code cell, labeled 'In []:'.

Result: Python program to create a rectangle class was completed.

5.

Aim: Create a class named 'PrintDT' to print various numbers of different datatypes by creating different functions with the same name 'python_data' having a parameter for each datatype. (example : tuple, list, string)

CODE:

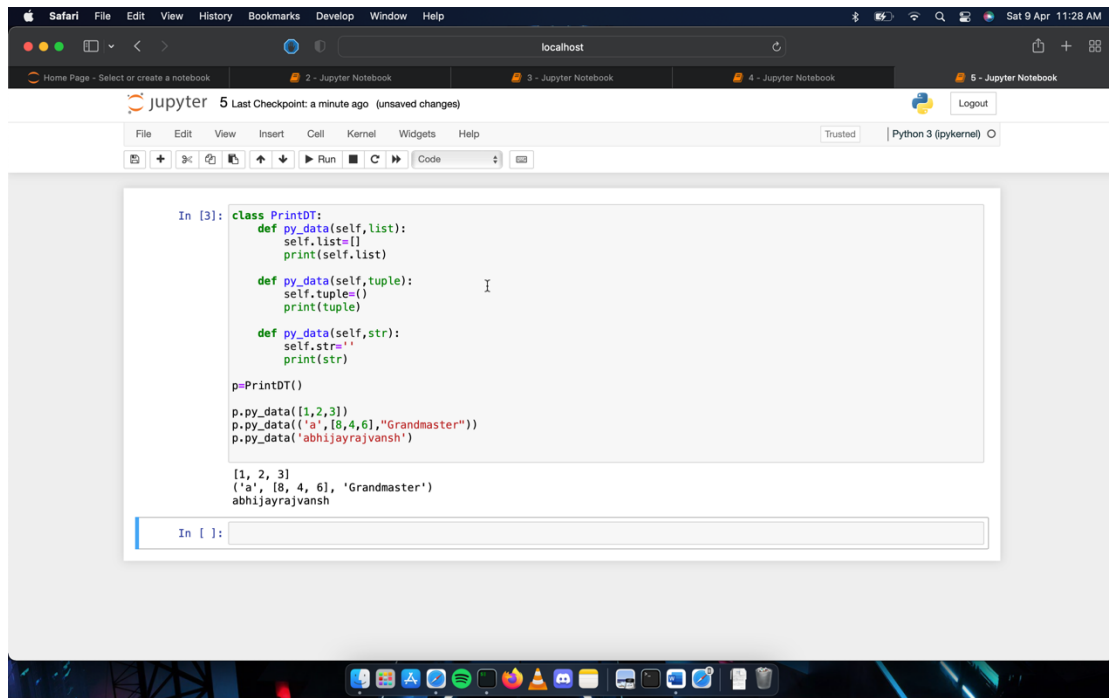
```
class PrintDT:
    def py_data(self,list):
        self.list=[]
        print(self.list)

    def py_data(self,tuple):
        self.tuple=()
        print(tuple)

    def py_data(self,str):
        self.str=""
        print(str)

p=PrintDT()

p.py_data([1,2,3])
p.py_data(('a',[8,4,6],"Grandmaster"))
p.py_data('abhijayrajvansh')
```



```
In [3]: class PrintDT:
        def py_data(self, list):
            self.list=list
            print(self.list)

        def py_data(self, tuple):
            self.tuple=()
            print(tuple)

        def py_data(self, str):
            self.str=str
            print(str)

p=PrintDT()

p.py_data([1,2,3])
p.py_data(('a', [8,4,6], "Grandmaster"))
p.py_data('abhiyrajvansh')

[1, 2, 3]
('a', [8, 4, 6], 'Grandmaster')
abhiyrajvansh

In [ ]:
```

Result: python program to create class named 'PrintDT' to print various numbers of different datatypes by creating different functions with the same name 'python_data' having a parameter for each datatype was completed.

6.

AIM: A student from SRMIST has his/her money deposited Rs.15000, Rs.30000 and Rs. 40,000 in banks-CUB, HDFC and Indian Bank respectively. We have to print the money deposited by him/her in a particular bank.

Create a class named 'Banks_SRMIST' with a function 'getBalance' which returns 0. Make its three subclasses named 'CUB', 'HDFC' and 'Indian_Bank' with a function with the same name 'getBalance' which returns the amount deposited in that particular bank. Call the function 'getBalance' by the object of each of the three banks.

CODE:

```
class Banks_SRMIST:
    def getBalance():
        return 0
class CUB(Banks_SRMIST):

    def getBalance(balance):
        return balance
class HDFC(Banks_SRMIST):

    def getBalance(balance):
        return balance
class Indian_Bank(Banks_SRMIST):

    def getBalance(balance):
        return balance
Banks_SRMIST()
print(CUB.getBalance(15000))
print(HDFC.getBalance(30000))
print(Indian_Bank.getBalance(40000))
```

```
In [1]: class Banks_SRMIST:
        def getBalance():
            return 0
        class CUB(Banks_SRMIST):
            def getBalance(balance):
                return balance
        class HDFC(Banks_SRMIST):
            def getBalance(balance):
                return balance
        class Indian_Bank(Banks_SRMIST):
            def getBalance(balance):
                return balance
        Banks_SRMIST()
        print(CUB.getBalance(15000))
        print(HDFC.getBalance(30000))
        print(Indian_Bank.getBalance(40000))

15000
30000
40000

In [ ]:
```

The screenshot shows a Jupyter Notebook running in a Safari browser. The notebook has a title bar with 'Safari', 'File', 'Edit', 'View', 'History', 'Bookmarks', 'Develop', 'Window', and 'Help'. The address bar shows 'localhost'. The notebook interface includes a toolbar with icons for file operations, a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help', and a status bar showing 'Python 3 (ipykernel)'. The code cell contains a Python class hierarchy for banks and the output of a program. The output shows the balances for CUB, HDFC, and Indian Bank.

Result: Python program of a student from SRMIST has his/her money deposited Rs.15000, Rs.30000 and Rs. 40,000 in banks-CUB, HDFC and Indian Bank respectively. We have to print the money deposited by him/her in a particular bank was completed.

7.

AIM: Create a Time class and initialize it with hours and minutes.

1. Make a method addTime which should take two time object and add them. E.g.- (2 hour and 50 min)+(1 hr and 20 min) is (4 hr and 10 min)
2. Make a method displayTime which should print the time.
3. Make a method DisplayMinute which should display the total minutes in the Time. E.g.- (1 hr 2 min) should display 62 minute.

CODE:

```
class Time():
```

```
    def __init__(self, hours, mins):  
        self.hours = hours  
        self.mins = mins
```

```
    def addTime(t1, t2):  
        t3 = Time(0,0)  
        if t1.mins+t2.mins > 60:  
            t3.hours = (t1.mins+t2.mins)//60  
            t3.hours = t3.hours+t1.hours+t2.hours  
            t3.mins = (t1.mins + t2.mins) % 60  
        return t3
```

```
    def displayTime(self):  
        print ("Time is",self.hours,"hours and",self.mins,"minutes.")
```

```
    def displayMinute(self):  
        print ((self.hours*60)+self.mins)
```

```
a = Time(2,40)  
b = Time(1,30)  
c = Time.addTime(a,b)  
c.displayTime()  
c.displayMinute()
```

```
In [1]: class Time():
        def __init__(self, hours, mins):
            self.hours = hours
            self.mins = mins

        def addTime(t1, t2):
            t3 = Time(0,0)
            if t1.mins+t2.mins > 60:
                t3.hours = (t1.mins+t2.mins)//60
                t3.hours = t3.hours+t1.hours+t2.hours
                t3.mins = (t1.mins + t2.mins) % 60
            return t3

        def displayTime(self):
            print ("Time is",self.hours,"hours and",self.mins,"minutes.")

        def displayMinute(self):
            print ((self.hours*60)+self.mins)

a = Time(2,40)
b = Time(1,30)
c = Time.addTime(a,b)
c.displayTime()
c.displayMinute()

Time is 4 hours and 10 minutes.
250

In [ ]:
```

Result: To create a python program of Time class and initialize it with hours and minutes was completed.

8.

AIM: Write a program to print the area and perimeter of a triangle having sides of 3, 4 and 5 units by creating a class named 'Triangle' with a function to print the area and perimeter.

CODE:

```
class Triangle:
    def findPerimeter(self, s1, s2, s3):
        return (s1 + s2 + s3)

    def findArea(self, s1, s2, s3):
        p = (s1 + s2 + s3)
        s = p/2
        return (s * (s-s1) * (s-s2)*(s-s3))**0.5

s1 = float(input("Enter the first side of the triangle : "))
s2 = float(input("Enter the second side of the triangle : "))
s3 = float(input("Enter the third side of the triangle : "))

u = Triangle()

print("The perimeter of the triangle is : {0:.2f}".format(
    u.findPerimeter(s1, s2, s3)))
print("The area of the triangle is : {0:.2f}".format(u.findArea(s1, s2, s3)))
```

```
In [1]: class Triangle:
        def findPerimeter(self, s1, s2, s3):
            return (s1 + s2 + s3)

        def findArea(self, s1, s2, s3):
            p = (s1 + s2 + s3)
            s = p/2
            return (s * (s-s1) * (s-s2)*(s-s3))**0.5

s1 = float(input("Enter the first side of the triangle : "))
s2 = float(input("Enter the second side of the triangle : "))
s3 = float(input("Enter the third side of the triangle : "))

u = Triangle()

print("The perimeter of the triangle is : {:.2f}".format(
    u.findPerimeter(s1, s2, s3)))
print("The area of the triangle is : {:.2f}".format(u.findArea(s1, s2, s3)))

Enter the first side of the triangle : 3
Enter the second side of the triangle : 4
Enter the third side of the triangle : 5
The perimeter of the triangle is : 12.00
The area of the triangle is : 6.00

In [ ]:
```

Result: Python program to print the area and perimeter of a triangle having sides of 3, 4 and 5 units by creating a class named 'Triangle' with a function to print the area and perimeter was completed.