

Python Practical’s

# TASK 7

Smit Joshi | 21-09-2023

View On [**github.com/smit-joshi814**](https://github.com/smit-joshi814/Learning-python/tree/main/collage/Task7)

# Practical 1

Write a python program for creating Student Management System.

# Student Management System

class Student():

    def \_\_init\_\_(self,rollNo,name,marks1,marks2):

        self.rollNo=rollNo

        self.name=name

        self.marks1=marks1

        self.marks2=marks2

    def accept(self):

        # Will Add the current Student to list

        ls.append(self)

        print("\n! Student Details Added SuccessFully !\n")

    # display the given Student details

    def display(self,obj):

        print()

        print(f"rollNo : {obj.rollNo}")

        print(f"Name : {obj.name}")

        print(f"Marks 1 : {obj.marks1}")

        print(f"Marks 2: {obj.marks2}")

        print()

    # searches for Student by rollNo

    def search(self,rollNo):

        for i in ls:

            if i.rollNo==rollNo:

                return i

    # Deletes the Student by rollNo

    def delete(self,rollNo):

        obj=self.search(rollNo)

        if obj != None:

            ls.remove(obj)

            print("\nStudent removed Successfully\n")

        else:

            print(f"\nInvalid oldRollNo no Student Found With {rollNo}\n")

    # Updates the Student rollNo with newRollNo

    def update(self,oldNo,newRollNo):

        obj=self.search(oldNo)

        if obj != None:

            obj.rollNo=newRollNo

            print("\nStudent Updated Successfully\n")

        else: print(f"\nInvalid oldRollNo no Student Found With {oldNo}\n")

# Global list

ls=[]

# To get the Student Details From User

def getStudentDetails():

    rollNo=int(input("Enter Roll No: "))

    name=input("Enter name: ")

    marks1=int(input("Enter Marks 1: "))

    marks2=int(input("Enter Marks 2: "))

    return rollNo,name,marks1,marks2

# Get Student RollNo

def getRollNo(bool=True):

    if bool:  return int(input("Enter rollNo: "))

    else: return int(input("Enter new rollNo: "))

# Main method

def main():

    obj=Student(0,'',0,0)

    while True:

        print("\n-----------------------\n1. Enter Student Details: ")

        print("2. See Available Students: ")

        print("3. Search Student by rollNo: ")

        print("4. Update Student rollno: ")

        print("5. Delete Student: ")

        print("6. exit\n")

        choice=int(input("Enter choice: "))

        match choice:

            case 1:

                rollNo,name,marks1,marks2=getStudentDetails()

                obj=Student(rollNo,name,marks1,marks2)

                obj.accept()

            case 2:

                for i in ls:

                    obj.display(i)

            case 3:

                obj.display(obj.search(getRollNo()))

            case 4:

                obj.update(getRollNo(),getRollNo(False))

            case 5:

                obj.delete(getRollNo())

            case 6:

                break

            case default:

                print("Invalid choice! Try Again")

# Main Calling

main()

### Output:

PS D:\LEARNING\COLLAGE\SAM7\Python\collage\Task7> py practical1.py

-----------------------

1. Enter Student Details:

2. See Available Students:

3. Search Student by rollNo:

4. Update Student rollno:

5. Delete Student:

6. exit

Enter choice: 1

Enter Roll No: 101

Enter name: Smit Joshi

Enter Marks 1: 90

Enter Marks 2: 90

! Student Details Added SuccessFully !

-----------------------

1. Enter Student Details:

2. See Available Students:

3. Search Student by rollNo:

4. Update Student rollno:

5. Delete Student:

6. exit

Enter choice: 1

Enter Roll No: 201

Enter name: Switi Patel

Enter Marks 1: 89

Enter Marks 2: 92

! Student Details Added SuccessFully !

-----------------------

1. Enter Student Details:

2. See Available Students:

3. Search Student by rollNo:

4. Update Student rollno:

5. Delete Student:

6. exit

Enter choice: 2

rollNo : 101

Name : Smit Joshi

Marks 1 : 90

Marks 2: 90

rollNo : 201

Name : Switi Patel

Marks 1 : 89

Marks 2: 92

-----------------------

1. Enter Student Details:

2. See Available Students:

3. Search Student by rollNo:

4. Update Student rollno:

5. Delete Student:

6. exit

Enter choice: 3

Enter rollNo: 101

rollNo : 101

Name : Smit Joshi

Marks 1 : 90

Marks 2: 90

-----------------------

1. Enter Student Details:

2. See Available Students:

3. Search Student by rollNo:

4. Update Student rollno:

5. Delete Student:

6. exit

Enter choice: 4

Enter rollNo: 201

Enter new rollNo: 102

Student Updated Successfully

-----------------------

1. Enter Student Details:

2. See Available Students:

3. Search Student by rollNo:

4. Update Student rollno:

5. Delete Student:

6. exit

Enter choice: 2

rollNo : 101

Name : Smit Joshi

Marks 1 : 90

Marks 2: 90

rollNo : 102

Name : Switi Patel

Marks 1 : 89

Marks 2: 92

-----------------------

1. Enter Student Details:

2. See Available Students:

3. Search Student by rollNo:

4. Update Student rollno:

5. Delete Student:

6. exit

Enter choice: 5

Enter rollNo: 101

Student removed Successfully

-----------------------

1. Enter Student Details:

2. See Available Students:

3. Search Student by rollNo:

4. Update Student rollno:

5. Delete Student:

6. exit

Enter choice: 2

rollNo : 102

Name : Switi Patel

Marks 1 : 89

Marks 2: 92

-----------------------

1. Enter Student Details:

2. See Available Students:

3. Search Student by rollNo:

4. Update Student rollno:

5. Delete Student:

6. exit

Enter choice: 6

PS D:\LEARNING\COLLAGE\SAM7\Python\collage\Task7>

# Practical 2

Write a python program to create an Employee management System.

# Employee management System

class Employee():

    def \_\_init\_\_(self,empId,empName,experiance,salary,designation):

        self.empId=empId

        self.empName=empName

        self.experience=experiance

        self.salary=salary

        self.designation=designation

    # adds Employee To List

    def addEmployee(self):

        employees.append(self)

        print("Employee Added Successfully")

    # removes Employee

    def removeEmployee(self,empId):

        obj=self.search(empId)

        if obj != None:

            employees.remove(obj)

            print("\nEmployee Removed Successfully\n")

        else:  print("\nInvalid Employee Id ca't Delete Employee\n")

    # Updates Employee

    def updateEmployee(self,empId,newEmpId):

        obj=self.search(empId)

        if obj != None:

            obj.empId=newEmpId

            print("\nEmployee Id Updated Successfully\n")

        else:

            print("\nInvalid Employee Id\n")

    #  Search Employee By empId

    def search(self,empId):

        for i in employees:

            if i.empId==empId:

                return i

    # Disployee Employee Details

    def display(self,employee):

        print()

        print(f"Eployee Id: {employee.empId}")

        print(f"Employee Name: {employee.empName}")

        print(f"Employee Experience: {employee.experience}+ Years")

        print(f"Employee salary: {employee.salary}")

        print(f"Employee Designation: {employee.designation}")

        print()

employees=[]

# To GetEmployeeDetails

def getEmployeeDetails():

    empId=int(input("Enter Employee Id: "))

    empName=input("Enter Employee name: ")

    experience=int(input("Enter Experience in Years: "))

    salary=int(input("Enter Employee Salary: "))

    designation=input("Enter Employee Designation: ")

    return empId,empName,experience,salary,designation

def getEmployeeId(bool=True):

    if bool: return int(input("Enter Employee Id: "))

    else: return int(input("Enter New Id: "))

def main():

    obj=Employee(0,'',0,0,'')

    while True:

        print("\n-----------------------\n1. Enter Employee Details: ")

        print("2. See Available Employees: ")

        print("3. Search Employee by Id: ")

        print("4. Update Employee Id: ")

        print("5. Delete Employee: ")

        print("6. exit\n")

        choice=int(input("Enter choice: "))

        match choice:

            case 1:

                empId,empname,experience,salary,designation=getEmployeeDetails()

                obj=Employee(empId,empname,experience,salary,designation)

                obj.addEmployee()

            case 2:

                for i in employees:

                    obj.display(i)

            case 3:

                obj.display(obj.search(getEmployeeId()))

            case 4:

                obj.updateEmployee(getEmployeeId(),getEmployeeId(False))

            case 5:

                obj.removeEmployee(getEmployeeId())

            case 6:

                break

            case default:

                print("Invalid choice! Try Again")

# Main Calling

main()

### Output:

PS D:\LEARNING\COLLAGE\SAM7\Python\collage\Task7> py practical2.py

-----------------------

1. Enter Employee Details:

2. See Available Employees:

3. Search Employee by Id:

4. Update Employee Id:

5. Delete Employee:

6. exit

Enter choice: 1

Enter Employee Id: 101

Enter Employee name: Smit Joshi

Enter Experience in Years: 9

Enter Employee Salary: 1200000

Enter Employee Designation: Product Manager

Employee Added Successfully

-----------------------

1. Enter Employee Details:

2. See Available Employees:

3. Search Employee by Id:

4. Update Employee Id:

5. Delete Employee:

6. exit

Enter choice: 1

Enter Employee Id: 201

Enter Employee name: Switi Patel

Enter Experience in Years: 2

Enter Employee Salary: 60000

Enter Employee Designation: Software Engineer

Employee Added Successfully

-----------------------

1. Enter Employee Details:

2. See Available Employees:

3. Search Employee by Id:

4. Update Employee Id:

5. Delete Employee:

6. exit

Enter choice: 2

Eployee Id: 101

Employee Name: Smit Joshi

Employee Experience: 9+ Years

Employee salary: 1200000

Employee Designation: Product Manager

Eployee Id: 201

Employee Name: Switi Patel

Employee Experience: 2+ Years

Employee salary: 60000

Employee Designation: Software Engineer

-----------------------

1. Enter Employee Details:

2. See Available Employees:

3. Search Employee by Id:

4. Update Employee Id:

5. Delete Employee:

6. exit

Enter choice: 3

Enter Employee Id: 101

Eployee Id: 101

Employee Name: Smit Joshi

Employee Experience: 9+ Years

Employee salary: 1200000

Employee Designation: Product Manager

-----------------------

1. Enter Employee Details:

2. See Available Employees:

3. Search Employee by Id:

4. Update Employee Id:

5. Delete Employee:

6. exit

Enter choice: 4

Enter Employee Id: 201

Enter New Id: 102

Employee Id Updated Successfully

-----------------------

1. Enter Employee Details:

2. See Available Employees:

3. Search Employee by Id:

4. Update Employee Id:

5. Delete Employee:

6. exit

Enter choice: 2

Eployee Id: 101

Employee Name: Smit Joshi

Employee Experience: 9+ Years

Employee salary: 1200000

Employee Designation: Product Manager

Eployee Id: 102

Employee Name: Switi Patel

Employee Experience: 2+ Years

Employee salary: 60000

Employee Designation: Software Engineer

-----------------------

1. Enter Employee Details:

2. See Available Employees:

3. Search Employee by Id:

4. Update Employee Id:

5. Delete Employee:

6. exit

Enter choice: 5

Enter Employee Id: 102

Employee Removed Successfully

-----------------------

1. Enter Employee Details:

2. See Available Employees:

3. Search Employee by Id:

4. Update Employee Id:

5. Delete Employee:

6. exit

Enter choice: 2

Eployee Id: 101

Employee Name: Smit Joshi

Employee Experience: 9+ Years

Employee salary: 1200000

Employee Designation: Product Manager

-----------------------

1. Enter Employee Details:

2. See Available Employees:

3. Search Employee by Id:

4. Update Employee Id:

5. Delete Employee:

6. exit

Enter choice: 6

PS D:\LEARNING\COLLAGE\SAM7\Python\collage\Task7>

# Practical 3

Write a python program to demonstrate the concept of lambda function for finding the

multiple of 7 from the list and adding 10 to all the members of the list.

numbers=[5,2,7, 10,14,67, 21]

# multiples of 7

multiples=list(filter(lambda x:x%7==0,numbers))

print("Multiples Of 7: ",multiples)

numbers=list(map(lambda i:i+10,numbers))

print("Added 10: ",numbers)

### Output:

PS D:\LEARNING\COLLAGE\SAM7\Python\collage\Task7> py practical3.py

Multiples Of 7: [7, 14, 21]

Added 10: [15, 12, 17, 20, 24, 77, 31]

PS D:\LEARNING\COLLAGE\SAM7\Python\collage\Task7>

# Practical 4

Write a python program to demonstrate the concept of method overloading. Write three

functions named surface area() for calculating the surface area of cube, sphere and cylinder.

from multipledispatch import dispatch

import math

class Area:

    @dispatch(int)

    def area(self,length):

       print("Area of cube is: ",6\*length\*\*2)

    @dispatch(float)

    def area(self,radious):

        print("Area of Shpare is: ",4\*math.pi\*radious\*\*2)

    @dispatch(float,int)

    def area(self,radious,height):

        print("Area of cylinder is: ",2\*math.pi\*radious\*(radious+height))

obj=Area()

obj.area(10)

obj.area(2.0)

obj.area(2.0,2)

### Output:

PS D:\LEARNING\COLLAGE\SAM7\Python\collage\task7> py practical4.py

Area of cube is: 600

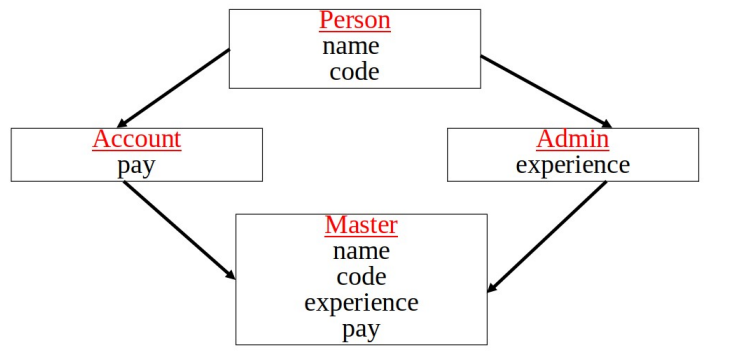
Area of Shpare is: 50.26548245743669

Area of cylinder is: 50.26548245743669

PS D:\LEARNING\COLLAGE\SAM7\Python\collage\task7>

# Practical 5

Write a python program to implement the following hierarchy.



class Person():

    def \_\_init\_\_(self,name,code):

        self.name=name

        self.code=code

class Account(Person):

    def \_\_init\_\_(self, name, code,pay):

        # super().\_\_init\_\_(self,name,code)  -- Error

        Person.\_\_init\_\_(self,name, code)

        self.pay=pay

class Admin(Person):

    def \_\_init\_\_(self,name,code,experience):

        # super().\_\_init(name,code)     -- Error

        Person.\_\_init\_\_(self,name,code)

        self.experience=experience

class Master(Account,Admin):

    def \_\_init\_\_(self, name, code, pay,experience):

        # super().\_\_init\_\_(name,code,pay,experience)    --Error

        Account.\_\_init\_\_(self,name,code,pay)

        Admin.\_\_init\_\_(self,name,code,experience)

    def display\_details(self):

        print(f"Name: {self.name}")

        print(f"Code: {self.code}")

        print(f"Pay: {self.pay}")

        print(f"Experience: {self.experience}")

master = Master("Joshi Smit", 123, 50000, 5)

master.display\_details()

### Output:

PS D:\LEARNING\COLLAGE\SAM7\Python\collage\Task7> py practical5.py

Name: Joshi Smit

Code: 123

Pay: 50000

Experience: 5

PS D:\LEARNING\COLLAGE\SAM7\Python\collage\Task7>

# Practical 6

Write a python program for creating class Vehicle with members Model no, type and price.

Derive class car and bike from class vehicle. Class car has members engine number, color

and fueltype. Class Bike has members as machine CC and mileage. Write proper

constructors and display functions to display all the details.

class Vehicle():

    def \_\_init\_\_(self,modelNo,type,price):

        self.modelNo=modelNo

        self.type=type

        self.price=price

    def display(self):

        print()

        print("Model No:",self.modelNo)

        print("Type:",self.type)

        print(f"Price: ₹{self.price}")

class Car(Vehicle):

    def \_\_init\_\_(self,modelNo,type,price,engineNumber,color,fuelType):

        super().\_\_init\_\_(modelNo,type,price)

        self.engineNumber=engineNumber

        self.color=color

        self.fuelType=fuelType

  def display(self):

        super().display()

        print("Engine Number:",self.engineNumber)

        print("Color:",self.color)

        print("Fuel Type:",self.fuelType)

class Bike(Vehicle):

    def \_\_init\_\_(self, modelNo, type, price,machineCC,mileage):

        super().\_\_init\_\_(modelNo, type, price)

        self.machineCC=machineCC

        self.mileage=mileage

    def display(self):

        super().display()

        print("Machine CC:",self.machineCC)

        print("Mileage:",self.mileage)

# Bike Objects

Hero=Bike("Hero Splendor Plus","Commuter",65000,97.2,65)

Bajaj=Bike("Bajaj Pulsar 180","StreetFighter",120000,178.6,45)

# Car Objects

Sedun=Car(1, 'Sedan', 20000, '1234567890', 'Red', 'Gasoline')

SUV=Car(2, 'SUV', 30000, '9876543210', 'Blue', 'Diesel')

Truck=Car(3, 'Truck', 40000, '0987654321', 'Green', 'Electric')

# printing All The Objects Data

print('\n------HERO------')

Hero.display()

print('\n------BAJAJ------')

Bajaj.display()

print('\n------SEDUN------')

Sedun.display()

print('\n------SUV------')

SUV.display()

print('\n------TRUCK------')

Truck.display()

### Output:

PS D:\LEARNING\COLLAGE\SAM7\Python\collage\Task7> py practical6.py

------HERO------

Model No: Hero Splendor Plus

Type: Commuter

Price: ₹65000

Machine CC: 97.2

Mileage: 65

------BAJAJ------

Model No: Bajaj Pulsar 180

Type: StreetFighter

Price: ₹120000

Machine CC: 178.6

Mileage: 45

------SEDUN------

Model No: 1

Type: Sedan

Price: ₹20000

Engine Number: 1234567890

Color: Red

Fuel Type: Gasoline

------SUV------

Model No: 2

Type: SUV

Price: ₹30000

Engine Number: 9876543210

Color: Blue

Fuel Type: Diesel

------TRUCK------

Model No: 3

Type: Truck

Price: ₹40000

Engine Number: 0987654321

Color: Green

Fuel Type: Electric

PS D:\LEARNING\COLLAGE\SAM7\Python\collage\Task7>

# Practical 7

Write a Python program to filter a list of integers using Lambda into positive, negative and

zero numbers. (Create three different lists and display)

numbers=[10,0,20,-47,-134]

print("Original: ",numbers)

positive=list(filter(lambda x: x > 0, numbers))

negative=list(filter(lambda x: x < 0, numbers))

zero=list(filter(lambda x: x==0 , numbers))

print("Poritive:",positive,"\nNegative:",negative,"\nZero:",zero)

### Output:

PS D:\LEARNING\COLLAGE\SAM7\Python\collage\Task7> py practical7.py

Original: [10, 0, 20, -47, -134]

Poritive: [10, 20]

Negative: [-47, -134]

Zero: [0]

PS D:\LEARNING\COLLAGE\SAM7\Python\collage\Task7>