A23126510207

Hasitha Kalla

Lab exercise

Week-5

OOPS- Basic programs

Set-1

1. Create a Class with instance attributes

```
class student:
  def __init__ (self,name,dep,score):
    self.name = name
    self.dep = dep
    self.score = score
  def details(self):
    print("Name:",self.name)
    print("Department:",self.dep)
    print("Score:",self.score)
stud=student("Hasitha","CSE",9.15)
stud.details()
```

```
Name: Hasitha
Department: CSE
Score: 9.15
```

2. Create a Vehicle class without any variables and methods

```
class vehicle:
  def drive(self):
    pass
class car(vehicle):
  print("You are in a Car")
class bus(vehicle):
  print("You are in a Bus")
car=car()
bus=bus()
car.drive()
bus.drive()
```

```
You are in a Car
You are in a Bus
```

3. Create a child class Bus that will inherit all of the variables and methods of the Vehicle class

```
class vehicle:
  def drive():
    pass
class bus(vehicle):
  def drive(self):
    print("Bus is being driven")
bus=bus()
bus.drive()
```

```
Bus is being driven
```

4. Class Inheritance

```
class student:
  def __init__(self,name,rollno):
    self.name = name
    self.rollno = rollno
  def display(self):
    print("Name:",self.name)
    print("Roll no: ",self.rollno)
class student_year(student):
  def __init__(self,name,rollno,year):
    super().__init__(name,rollno)
    self.year = year
  def display(self):
    super().display()
    print("Year:",self.year)
student = student_year("Hasitha",207,2023)
print(student.display())
```

```
Name: Hasitha
Roll no:  207
Year: 2023
None
```

5: Define a property that must have the same value for every class instance (object)

```
class student:
    school_name = "Jujutsu High School"
    def __init__(self, name, rollno):
        self.name = name
        self.rollno = rollno
    def display(self):
        print("Name:", self.name)
        print("Roll no:", self.rollno)
        print("School name:", student.school_name)
class student_year(student):
    def __init__(self, name, rollno, year):
        super().__init__(name, rollno)
        self.year = year
    def display(self):
        super().display()
        print("Year:", self.year)
student1 = student_year("Hasitha", 207, 2023)
student2 = student_year("Kaushika", 208, 2024)
student1.display()
print("")
student2.display()
```

```
Name: Hasitha
Roll no: 207
School name: Jujutsu High School
Year: 2023

Name: Kaushika
Roll no: 208
School name: Jujutsu High School
Year: 2024
```

6. Check type of an object

```
class jjk:
  def __init__(self,name,domain):
    self.name = name
    self.domain = domain
sorcerer = jjk("Gojo Satoru",28)
print(type(jjk))
print(type(sorcerer.name))
print(type(sorcerer.domain))
```

```
<class 'type'>
<class 'str'>
<class 'int'>
```

7. Determine if School_bus is also an instance of the Vehicle class

```
class vehicle:
  def __init__(self,name):
    self.name = name
class bus(vehicle):
  pass
school_bus = bus("TATA")
print(isinstance(school_bus, vehicle))
print(isinstance(school_bus, bus))
```

```
True
True
```

Set-2

1. Write a Python program to create a person class. Include attributes like name, country and date of birth. Implement a method to determine the person’s age.

```
class person:
  def __init__(self,name,country,dob):
    self.name = name
    self.country = country
    self.dob = dob
  def age(self):
    year = int(input("Enter year: "))
    age = year-self.dob
    print("Age:",age)
p1=person("Hasitha","India",2006)
p1.age()
```

```
Enter year: 2024
Age: 18
```

2. Write a Python program to create a class representing a bank. Include methods for managing customer accounts and transactions.

```
class Bank:
    def add_customer(self, customer_id, initial_balance=0):
        self.customers[customer_id] = initial_balance
    def __init__(self):
        self.customers = {}
        self.transactions = []
    def deposit(self, customer_id, amount):
        self.customers[customer_id] += amount
        self.transactions.append((customer_id, 'deposit', amount))
    def withdraw(self, customer_id, amount):
         self.customers[customer_id] -= amount
         self.transactions.append((customer_id, 'withdraw', amount))
    def get_balance(self, customer_id):
        return self.customers[customer_id]
bank = Bank()
bank.add_customer('123', 1000)
bank.deposit('123', 500)
bank.withdraw('123', 200)
print(bank.get_balance('123'))
```

```
1300
```

3. Write a Python program to create a class representing a shopping cart. Include methods for adding and removing items, and calculating the total price.

```
class shopping:
  def __init__(self):
    self.items = {}
  def add_item(self,name,price):
    self.items[name] = price
  def remove_item(self, name):
    del self.items[name]
  def bill(self):
    total = sum(self.items.values())
    print(f"Total billing amount: {total}")
shopping = shopping()
shopping.add_item("Book", 70)
shopping.add_item("Pen", 10)
shopping.add_item("Board", 100)
shopping.remove_item("Board")
shopping.bill()
```

```
Total billing amount: 80
```

4. Write a Python program to create a calculator class. Include methods for basic arithmetic operations.

```
class calculator:
 def add(self, a, b):
  return a + b
 def sub(self, a, b):
  return a - b
 def multi(self, a, b):
  return a * b
 def div(self, a, b):
  return a / b
n=input()
c = calculator()
if(n[1]=="+"):
  print(c.add(a,b))
elif(n[1]=="-"):
  print(c.sub(n[0],n[2]))
elif(n[1]=="*"):
  print(c.multi(n[0],n[2]))
elif(n[1]=="/"):
  print(c.div(n[0],n[2]))
else:
  print("invalid choice")
```

```
2+6
8.0
```

Set-3

1. Write a Python program to create a class representing a Circle. Include methods to calculate its area and perimeter.

```
class circle:
  def __init__(self,radius):
    self.radius = radius
  def area(self):
    return (3.14*self.radius**2)
  def perimeter(self):
    return (2*3.14*self.radius)
r = int(input("Enter radius: "))
c = circle(r)
print()
print("Area: ",c.area())
print("Perimeter: ",c.perimeter())
```

```
Enter radius: 3

Area:  28.26
Perimeter:  18.84
```

2. Write a Python program to create a class that represents a shape. Include methods to calculate its area and perimeter. Implement subclasses for different shapes like circle, triangle, and square.

```
class circle:
  def __init__(self,radius):
    self.radius = radius
  def area(self):
    return (3.14*self.radius**2)
  def perimeter(self):
    return (2*3.14*self.radius)
class triangle:
  def __init__(self,base,height):
    self.base = base
    self.height = height
  def area(self):
    return (self.height*self.base*0.5)
  def perimeter(self):
    hypo = 0.5**((2**self.base)+(2**self.height))
    return self.base+self.height+hypo
class square:
  def __init__(self,side):
    self.side = side
  def area(self):
    return (self.radius**2)
  def perimeter(self):
    return 4*self.radius
print("1. Circle")
print("2. Triangle")
print("3. Square")
n = int(input("Enter your choice: "))
if (n == 1):
  r = int(input("Enter radius: "))
  c = circle(r)
  print("Area: ",c.area())
  print("Perimeter: ",c.perimeter())
elif (n == 2):
  b = int(input("Enter base: "))
  h = int(input("Enter height: "))
  c = triangle(b,h)
  print("Area: ",c.area())
  print("Perimeter: ",c.perimeter())
elif (n == 3):
  b = int(input("Enter side length: "))
  c = square(b)
  print("Area: ",c.area())
  print("Perimeter: ",c.perimeter())
else:
  print("Invalid choice")
```

```
1. Circle
2. Triangle
3. Square
Enter your choice: 2
Enter base: 3
Enter height: 4
Area:  6.0
Perimeter:  7.000000059604645
```

3. Write a Python program to create a class representing a stack data structure. Include methods for pushing and popping elements.

```
class stack:
 def __init__(self):
  self.items = []
 def push(self, item):
  self.items.append(item)
 def pop(self):
  self.items.pop()
 def display(self):
  return self.items
s = stack()
n = int(input("How many numbers do you want to push: "))
for i in range(n):
  a = int(input("Enter: "))
  s.push(a)
print(s.display())
b = input("do you want to remove any pop (y/n):")
m = int(input("How many numbers do you want to pop: "))
for i in range(m):
  if (b == "y"):
   s.pop()
print(s.display())
```

```
How many numbers do you want to push: 4
Enter: 2
```

```
Enter: 5
Enter: 7
Enter: 6
[2, 5, 7, 6]
do you want to remove any pop (y/n):y
How many numbers do you want to pop: 2
[2, 5]
```

4. 7. Write a Python program to create a class representing a queue data structure. Include methods for enqueueing and dequeueing elements.

```
class data:
 def __init__(self):
  self.items = []
 def push(self, item):
  self.items.append(item)
 def pop(self):
  self.items.pop()
 def display(self):
  return self.items
s = stack()
n = int(input("How many names do you want to add to the queue: "))
for i in range(n):
  a = input("Enter: ")
  s.push(a)
print(s.display())
b = input("do you want to remove any element (y/n):")
m = int(input("How many names do you want to remove: "))
for i in range(m):
  if (b == "y"):
```