Experiment 9 - Classes, Constructors, and Destructors in Python

Name: Mohammad Sayeed

Roll no : C56 Div: C Class: TY CSE

1. Student Class

Section: Classes

Problem Statement:

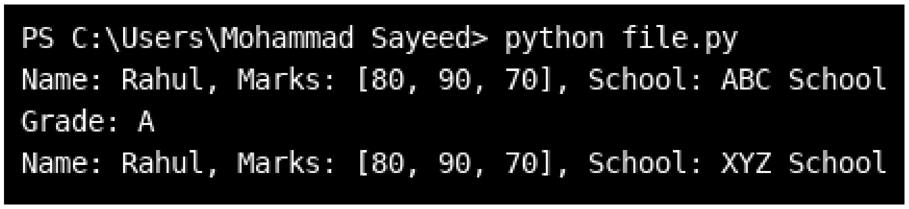
Create a class Student with name and marks. Include methods to display details, change school name (class method), calculate grade (static method), and a destructor message.

Code:

```
class Student:
    school = "ABC School"
    def __init__(self, name, marks):
        self.name = name
        self.marks = marks
    def display(self):
        print(f"Name: {self.name}, Marks: {self.marks}, School: 
{Student.school}")
    @classmethod
    def change_school(cls, new):
        cls.school = new
    @staticmethod
    def grade(marks):
        avg = sum(marks)/len(marks)
        if avg>=75: return 'A'
        elif avg>=60: return 'B'
        elif avg>=50: return 'C'
        else: return 'D'
    def __del__(self):
        print(f"Student object for {self.name} destroyed")

s=Student('Rahul',[80,90,70])
s.display()
print('Grade:', Student.grade(s.marks))
Student.change_school('XYZ School')
s.display()
```

Output:



```
PS C:\Users\Mohammad Sayeed> python file.py
Name: Rahul, Marks: [80, 90, 70], School: ABC School
Grade: A
Name: Rahul, Marks: [80, 90, 70], School: XYZ School
```

2. Employee Salary

Section: Classes

Problem Statement:

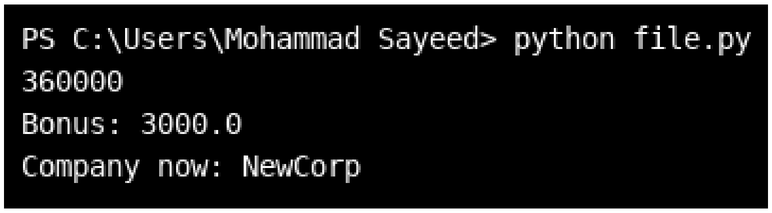
Create a class Employee with emp_id, name, and salary. Include methods to calculate yearly salary, update company name (class method), calculate bonus (static method), and a destructor message.

Code:

```
class Employee:
    company = 'ABC Corp'
    def __init__(self, emp_id, name, salary):
        self.emp_id = emp_id; self.name = name; self.salary = salary
    def yearly_salary(self):
        print(self.salary * 12)
    @classmethod
    def update_company(cls, name):
        cls.company = name
    @staticmethod
    def bonus(salary, percent):
        return salary * percent/100
    def __del__(self):
        print(f'Employee {self.name} destroyed')

e=Employee(101,'Asha',30000)
e.yearly_salary()
print('Bonus:', Employee.bonus(e.salary,10))
Employee.update_company('NewCorp')
print('Company now:', Employee.company)
```

Output:



```
PS C:\Users\Mohammad Sayeed> python file.py
360000
Bonus: 3000.0
Company now: NewCorp
```

3. Bank Account

Section: Classes

Problem Statement:

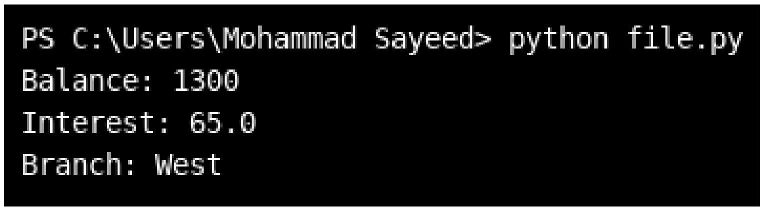
Create a class BankAccount with acc_no and balance. Include methods for deposit/withdrawal, update bank branch (class method), calculate interest (static method), and a destructor message.

Code:

```
class BankAccount:
    branch = 'Main'
    def __init__(self, acc_no, balance=0):
        self.acc_no = acc_no; self.balance = balance
    def deposit(self, amt):
        self.balance += amt
    def withdraw(self, amt):
        if amt<=self.balance: self.balance -= amt
        else: print('Insufficient')
    @classmethod
    def update_branch(cls, b): cls.branch = b
    @staticmethod
    def interest(balance, rate, years):
        return balance * ((1+rate/100)**years -1)
    def __del__(self):
        print(f'Account {self.acc_no} closed')

a=BankAccount('A123',1000)
a.deposit(500); a.withdraw(200)
print('Balance:', a.balance)
print('Interest:', round(BankAccount.interest(a.balance,5,1),2))
BankAccount.update_branch('West'); print('Branch:', BankAccount.branch)
```

Output:



```
PS C:\Users\Mohammad Sayeed> python file.py
Balance: 1300
Interest: 65.0
Branch: West
```

4. Rectangle Operations

Section: Classes

Problem Statement:

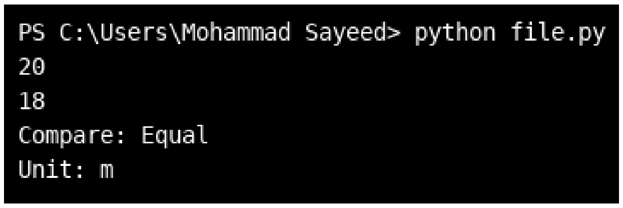
Create a class Rectangle with length and width. Include methods to calculate area/perimeter, update unit (class method), compare two rectangles (static method), and destructor message.

Code:

```
class Rectangle:
    unit='cm'
    def __init__(self,l,w): self.l=l; self.w=w
    def area(self): print(self.l*self.w)
    def perimeter(self): print(2*(self.l+self.w))
    @classmethod
    def set_unit(cls,u): cls.unit=u
    @staticmethod
    def compare(r1,r2):
        return 'Equal' if r1.l*r1.w==r2.l*r2.w else ('Larger' if
r1.l*r1.w>r2.l*r2.w else 'Smaller')
    def __del__(self): print('Rectangle destroyed')

r1=Rectangle(4,5); r2=Rectangle(2,10)
r1.area(); r1.perimeter()
print('Compare:', Rectangle.compare(r1,r2))
Rectangle.set_unit('m'); print('Unit:', Rectangle.unit)
```

Output:



```
PS C:\Users\Mohammad Sayeed> python file.py
20
18
Compare: Equal
Unit: m
```

5. Car Information

Section: Classes

Problem Statement:

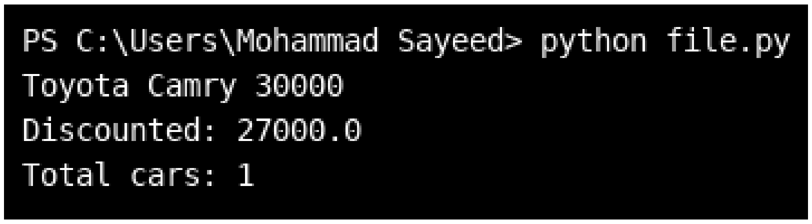
Create a class Car with brand, model, and price. Include methods to display details, update total cars (class method), calculate discounted price (static method), and destructor message.

Code:

```
class Car:
    total=0
    def __init__(self,brand,model,price):
        self.brand=brand; self.model=model; self.price=price;
Car.total+=1
    def display(self): print(self.brand,self.model,self.price)
    @classmethod
    def reset_total(cls): cls.total=0
    @staticmethod
    def discounted(price,percent): return price*(1-percent/100)
    def __del__(self): print('Car object destroyed')

c=Car('Toyota','Camry',30000); c.display()
print('Discounted:', Car.discounted(c.price,10))
print('Total cars:', Car.total)
```

Output:



```
PS C:\Users\Mohammad Sayeed> python file.py
Toyota Camry 30000
Discounted: 27000.0
Total cars: 1
```

6. Circle Calculations

Section: Classes

Problem Statement:

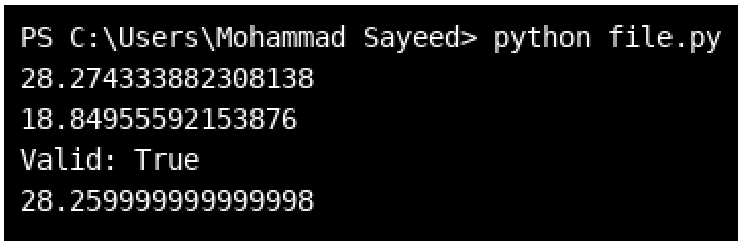
Create a class Circle with radius. Include methods to calculate area/circumference, update π value (class method), validate radius (static method), and destructor message.

Code:

```
import math
class Circle:
    pi=math.pi
    def __init__(self,r): self.r=r
    def area(self): print(Circle.pi*self.r*self.r)
    def circumference(self): print(2*Circle.pi*self.r)
    @classmethod
    def set_pi(cls,val): cls.pi=val
    @staticmethod
    def validate(r): return r>0
    def __del__(self): print('Circle destroyed')

c=Circle(3); c.area(); c.circumference()
print('Valid:', Circle.validate(3))
Circle.set_pi(3.14); c.area()
```

Output:



```
PS C:\Users\Mohammad Sayeed> python file.py
28.274333882308138
18.84955592153876
Valid: True
28.259999999999998
```

7. Book Details

Section: Classes

Problem Statement:

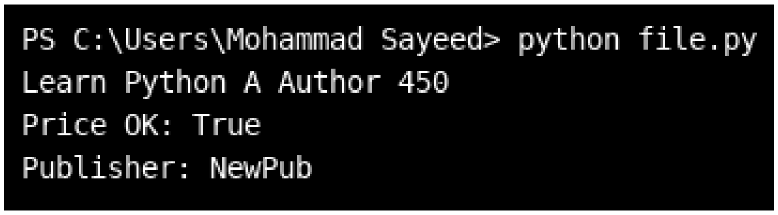
Create a class Book with title, author, and price. Include methods to display details, update publisher (class method), check price limit (static method), and destructor message.

Code:

```
class Book:
    publisher='PubX'
    def __init__(self,title,author,price): self.title=title;
self.author=author; self.price=price
    def display(self): print(self.title,self.author,self.price)
    @classmethod
    def set_publisher(cls,p): cls.publisher=p
    @staticmethod
    def price_ok(price): return price<=500
    def __del__(self): print('Book destroyed')

b=Book('Learn Python','A Author',450); b.display()
print('Price OK:', Book.price_ok(b.price))
Book.set_publisher('NewPub'); print('Publisher:', Book.publisher)
```

Output:



```
PS C:\Users\Mohammad Sayeed> python file.py
Learn Python A Author 450
Price OK: True
Publisher: NewPub
```

8. Laptop Specifications

Section: Classes

Problem Statement:

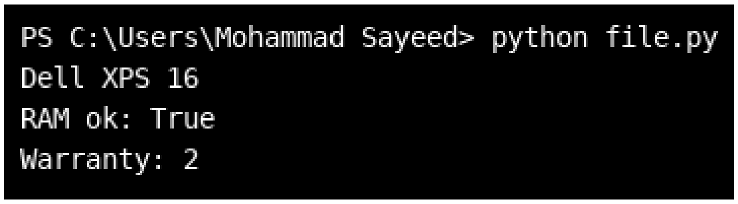
Create a class Laptop with brand, model, and ram. Include methods to display specs, update warranty (class method), check RAM sufficiency (static method), and destructor message.

Code:

```
class Laptop:
    warranty=1
    def __init__(self,brand,model,ram): self.brand=brand;
self.model=model; self.ram=ram
    def specs(self): print(self.brand,self.model,self.ram)
    @classmethod
    def set_warranty(cls,y): cls.warranty=y
    @staticmethod
    def ram_ok(ram): return ram>=8
    def __del__(self): print('Laptop destroyed')

l=Laptop('Dell','XPS',16); l.specs()
print('RAM ok:', Laptop.ram_ok(l.ram))
Laptop.set_warranty(2); print('Warranty:', Laptop.warranty)
```

Output:



```
PS C:\Users\Mohammad Sayeed> python file.py
Dell XPS 16
RAM ok: True
Warranty: 2
```

9. Payroll System

Section: Classes

Problem Statement:

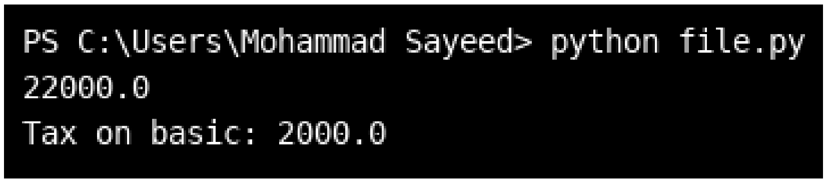
Create a class Payroll with name and basic_salary. Include methods to calculate total salary, update HRA percentage (class method), calculate tax (static method), and destructor message.

Code:

```
class Payroll:
    hra_pct=10
    def __init__(self,name,basic): self.name=name; self.basic=basic
    def total_salary(self):
        hra=self.basic*Payroll.hra_pct/100
        print(self.basic+hra)
    @classmethod
    def set_hra(cls,p): cls.hra_pct=p
    @staticmethod
    def tax(amount): return amount*0.1
    def __del__(self): print('Payroll object destroyed')

p=Payroll('Sara',20000); p.total_salary()
print('Tax on basic:', Payroll.tax(p.basic))
```

Output:



```
PS C:\Users\Mohammad Sayeed> python file.py
22000.0
Tax on basic: 2000.0
```

10. Temperature Converter

Section: Classes

Problem Statement:

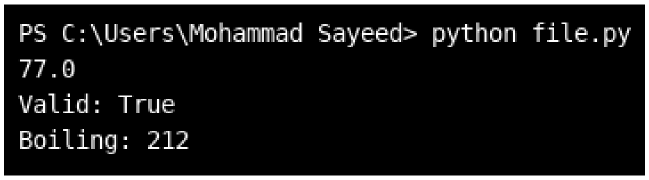
Create a class Temperature with a Celsius value. Include methods to convert to Fahrenheit, set boiling point (class method), validate temperature (static method), and destructor message.

Code:

```
class Temperature:
    boiling=100
    def __init__(self,c): self.c=c
    def to_f(self): print((self.c*9/5)+32)
    @classmethod
    def set_boiling(cls,b): cls.boiling=b
    @staticmethod
    def valid(c): return isinstance(c,(int,float))
    def __del__(self): print('Temperature object destroyed')

t=Temperature(25); t.to_f()
print('Valid:', Temperature.valid(25))
Temperature.set_boiling(212); print('Boiling:', Temperature.boiling)
```

Output:



```
PS C:\Users\Mohammad Sayeed> python file.py
77.0
Valid: True
Boiling: 212
```

11. Product Management

Section: Classes

Problem Statement:

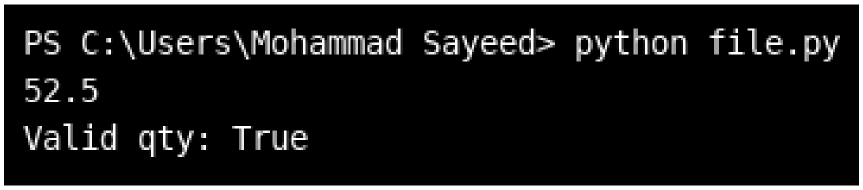
Create a class Product with name, price, and quantity. Include methods to calculate total cost, update tax rate (class method), validate quantity (static method), and destructor message.

Code:

```
class Product:
    tax=0.05
    def __init__(self,name,price,qty): self.name=name;
self.price=price; self.qty=qty
    def total_cost(self): print(self.price*self.qty*(1+Product.tax))
    @classmethod
    def set_tax(cls,t): cls.tax=t
    @staticmethod
    def valid_qty(q): return q>=0
    def __del__(self): print('Product destroyed')

prd=Product('Pen',10,5); prd.total_cost()
print('Valid qty:', Product.valid_qty(prd.qty))
```

Output:



```
PS C:\Users\Mohammad Sayeed> python file.py
52.5
Valid qty: True
```

12. Employee Attendance

Section: Classes

Problem Statement:

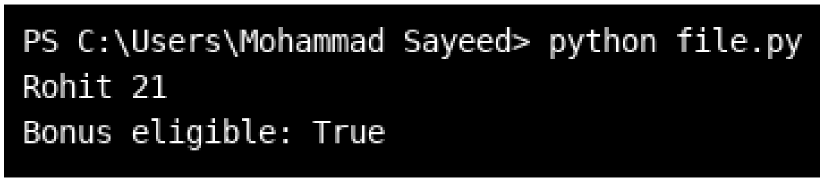
Create a class Attendance with emp_name and days_present. Include methods to display attendance, update total working days (class method), check bonus eligibility (static method), and destructor message.

Code:

```
class Attendance:
    total_days=22
    def __init__(self,emp,days): self.emp=emp; self.days=days
    def display(self): print(self.emp, self.days)
    @classmethod
    def set_total(cls,d): cls.total_days=d
    @staticmethod
    def bonus_elig(days): return days>20
    def __del__(self): print('Attendance object destroyed')

a=Attendance('Rohit',21); a.display()
print('Bonus eligible:', Attendance.bonus_elig(a.days))
```

Output:



```
PS C:\Users\Mohammad Sayeed> python file.py
Rohit 21
Bonus eligible: True
```