3. Class Practice questions

Q1. Create a class `Student` with attributes `name` and `age`. Add a method to display them.

```python
class Student:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def display(self):
        print(f"Name: {self.name}, Age: {self.age}")

s = Student("Neeraj", 37)
s.display()

# Output
Name: Neeraj, Age: 37
```

Q2. Create a class `Calculator` with a method `add(a, b)` that returns the sum.

```python
class Calculator:
    def add(self, a, b):
        return a + b

c = Calculator()
print(c.add(5, 7))

# Output
12
```

Q3. Create a class `Dog` with a class variable `country` and an instance variable `name`.

```python
class Dog:
    country = "Indian"

    def __init__(self, name):
        self.name = name
```

```python
d = Dog("sultan")
print(d.name, d.country)

# Output
sultan Indian
```

Q4. Create a class `Book` with a default constructor (title = "Unknown"). Print the title.

```python
class Book:
    def __init__(self, title="Unknown"):
        self.title = title

b = Book()
print(b.title)

# Output
Unknown
```

Q5. Create a class `Car` with attributes `brand`, `model`, and a `__str__()` method to display them.

```python
class Car:
    def __init__(self, brand, model):
        self.brand = brand
        self.model = model

    def __str__(self):
        return f"{self.brand} {self.model}"

# Test
c = Car("Toyota", "Corolla")
print(c)
```

Q6. Create a class `Employee` with attributes `name`, `salary`, and a method `annual_salary()`.

```python
class Employee:
    def __init__(self, name, salary):
        self.name = name
        self.salary = salary
    def annual_salary(self):
        return self.salary * 12

# Test
e = Employee("Neeraj", 82250)
print(e.annual_salary())

# Output
987000
```

Q7. Create a class `BankAccount` with methods `deposit()`, `withdraw()`, and `display_balance()`.

```
class BankAccount:
    def __init__(self, owner, balance=0):
        self.owner = owner
        self.balance = balance

    def deposit(self, amount):
        self.balance += amount

    def withdraw(self, amount):
        if amount <= self.balance:
            self.balance -= amount
        else:
            print("Insufficient funds")

    def display_balance(self):
        print(f"Balance: {self.balance}")

acc = BankAccount("Neeraj", 1000)
acc.deposit(500)
acc.withdraw(200)
acc.display_balance()

# Output
Balance: 1300
```

Q8. Create a base class `Animal` with a method `speak()`. Create a subclass `Dog` that overrides `speak()`.

```python
class Animal:
    def __init__(self, name):
        self.name = name

    def speak(self):
        print("I am an animal")

class Dog(Animal):
    def speak(self):
        print("Woof!")

a = Animal("Generic")
d = Dog("sultan")
a.speak()
d.speak()

# Output
I am an animal
Woof!
```

Q9. Create a class `Rectangle` with methods `area()` and `perimeter()`.

```python
class Rectangle:
    def __init__(self, length, width):
        self.length = length
        self.width = width

    def area(self):
        return self.length * self.width

    def perimeter(self):
        return 2 * (self.length + self.width)

r = Rectangle(10, 5)
print("Area:", r.area())
print("Perimeter:", r.perimeter())

# Output
Area: 50
Perimeter: 30
```

Q10. Create a class `Circle` with methods `area()` and `circumference()`.

```python
import math
class Circle:
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return math.pi * self.radius ** 2

    def circumference(self):
        return 2 * math.pi * self.radius

c = Circle(7)
print("Area:", c.area())
print("Circumference:", c.circumference())

# Output
Area: 153.93804002589985
Circumference: 43.982297150257104
```

Q11. Create a class `Timer` with an attribute `seconds`. Add methods `add_seconds(s)` and `display_time()` in format HH:MM:SS.

```python
class Timer:
    def __init__(self, seconds=0):
        self.seconds = seconds

    def add_seconds(self, s):
        self.seconds += s

    def display_time(self):
        h = self.seconds // 3600
        m = (self.seconds % 3600) // 60
        s = self.seconds % 60
        print(f"{h:02d}:{m:02d}:{s:02d}")

t = Timer(3665)
t.display_time()
t.add_seconds(125)
t.display_time()

# Output
01:01:05
01:03:10
```

Q12. Create a class `Temperature` with attribute `celsius`. Add methods `to_fahrenheit()` and `to_kelvin()`.

```python
class Temperature:
    def __init__(self, celsius):
        self.celsius = celsius

    def to_fahrenheit(self):
        return (self.celsius * 9/5) + 32

    def to_kelvin(self):
        return self.celsius + 273.15

temp = Temperature(25)
print("Fahrenheit:", temp.to_fahrenheit())
print("Kelvin:", temp.to_kelvin())

# Output
Fahrenheit: 77.0
Kelvin: 298.15
```

Q13. Create a class `Counter` with a class variable `count` and methods `increment()` and `decrement()`.

```python
class Counter:
    count = 0  # class variable
    def increment(self):
        Counter.count += 1
    def decrement(self):
        Counter.count -= 1

c1 = Counter()
c2 = Counter()
c1.increment()
c2.increment()
c2.decrement()
print("Current Count:", Counter.count)

# Output
Current Count: 1
```

Q14. Create a class `Point` with attributes `x`, `y`. Add methods `distance(other_point)` and `midpoint(other_point)`.

```
import math
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y
    def distance(self, other):
        return math.sqrt((self.x - other.x)**2 + (self.y - other.y)**2)
    def midpoint(self, other):
        return Point((self.x + other.x)/2, (self.y + other.y)/2)

p1 = Point(2, 3)
p2 = Point(4, 7)
print("Distance:", p1.distance(p2))
mid = p1.midpoint(p2)
print("Midpoint:", (mid.x, mid.y))

# Output
Distance: 4.47213595499958
Midpoint: (3.0, 5.0)
```

Q15. Create a class `Car` with attributes `brand`, `model`, and `speed`. Add methods `accelerate(value)`, `brake(value)`, and `display_speed()`.

```
class Car:
    def __init__(self, brand, model, speed=0):
        self.brand = brand
        self.model = model
        self.speed = speed
    def accelerate(self, value):
        self.speed += value
    def brake(self, value):
        self.speed = max(0, self.speed - value)
    def display_speed(self):
        print(f"{self.brand} {self.model} speed: {self.speed} km/h")

c = Car("Toyota", "Corolla", 50)
c.accelerate(30)
c.display_speed()
c.brake(50)
c.display_speed()
```

```
# Output
Toyota Corolla speed: 80 km/h
Toyota Corolla speed: 30 km/h
```

Q16. Create a class `Person` with attributes `name`, `age`. Create subclasses `Teacher(subject)` and `Student(roll_number)`.

```python
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
class Teacher(Person):
    def __init__(self, name, age, subject):
        super().__init__(name, age)
        self.subject = subject
class Student(Person):
    def __init__(self, name, age, roll_number):
        super().__init__(name, age)
        self.roll_number = roll_number

t = Teacher("Umendra", 35, "Math")
s = Student("Neeraj", 18, 101)

print(f"Teacher: {t.name}, {t.age}, {t.subject}")
print(f"Student: {s.name}, {s.age}, {s.roll_number}")

# Output
Teacher: Umendra, 35, Math
Student: Neeraj, 18, 101
```

Q17. Create a class `Polygon` with sides. Add methods `perimeter()` and `number_of_sides()`. Create a subclass `Triangle` with method `area()` using Heron's formula.

```python
import math

class Polygon:
    def __init__(self, sides):
        self.sides = sides
    def perimeter(self):
        return sum(self.sides)
    def number_of_sides(self):
        return len(self.sides)
class Triangle(Polygon):
    def __init__(self, a, b, c):
        super().__init__([a, b, c])
    def area(self):
        s = self.perimeter() / 2
        return math.sqrt(s*(s-self.sides[0])*(s-self.sides[1])*(s-self.sides[2]))

tri = Triangle(3, 4, 5)
print("Perimeter:", tri.perimeter())
print("Area:", tri.area())

# Output
Perimeter: 12
Area: 6.0
```

Q18. Create a class `Employee` with attributes `name`, `salary`. Create subclass `Intern` with attribute `stipend` and override `annual_salary()`.

```
class Employee:
    def __init__(self, name, salary):
        self.name = name
        self.salary = salary
    def annual_salary(self):
        return self.salary * 12

class Intern(Employee):
    def __init__(self, name, stipend):
        self.name = name
        self.stipend = stipend
    def annual_salary(self):
        return self.stipend * 12

e = Employee("Neeraj", 500000)
i = Intern("Rahul", 10000)

print("Employee annual salary:", e.annual_salary())
print("Intern annual salary:", i.annual_salary())

# Output
Employee annual salary: 6000000
Intern annual salary: 120000
```

Q19. Create a class `Inventory` to manage items with methods `add_item()`, `remove_item()`, and `display_inventory()`.

```
class Inventory:
    def __init__(self):
        self.items = {}
    def add_item(self, item, quantity):
        self.items[item] = self.items.get(item, 0) + quantity
    def remove_item(self, item, quantity):
        if item in self.items:
            self.items[item] = max(0, self.items[item] - quantity)
    def display_inventory(self):
        print("Inventory:", self.items)
inv = Inventory()
inv.add_item("Pen", 10)
inv.add_item("Notebook", 5)
inv.remove_item("Pen", 3)
inv.display_inventory()

# Output
Inventory: {'Pen': 7, 'Notebook': 5}
```

Q20. Create a class `Store` with a dictionary of products. Add methods `add_product()`, `remove_product()`, `total_value()`. Create subclass `DiscountStore` with `apply_discount(percent)`.

```
class Store:
    def __init__(self):
        self.products = {}
    def add_product(self, name, price):
        self.products[name] = price
    def remove_product(self, name):
        self.products.pop(name, None)
    def total_value(self):
        return sum(self.products.values())
class DiscountStore(Store):
    def apply_discount(self, percent):
        for k in self.products:
            self.products[k] *= (1 - percent/100)

s = DiscountStore()
s.add_product("Laptop", 1000)
```

```python
s.add_product("Phone", 500)
print("Total before discount:", s.total_value())
s.apply_discount(10)
print("Total after 10% discount:", s.total_value())

# Output
Total before discount: 1500
Total after 10% discount: 1350.0
```