1. **Select the correct option after executing the below code :-**

```
class Triangle:
    def __init__(self, side1, side2, side3):
        self.side1 = side1
        self.side2 = side2
        self.side3 = side3

    def calculate_perimeter(self):
        return self.side1 + self.side2 + self.side3

my_triangle = Triangle(3, 4, 5)

result = my_triangle.perimeter
```

   a. AttributeError: 'Triangle' object has no attribute 'perimeter'
   b. TypeError: 'Triangle' object has no attribute 'perimeter'
   c. No error, it will output the perimeter correctly
   d. SyntaxError: invalid syntax

2. **Select the correct option after executing the below code :-**

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

my_dog = Dog("Buddy", 3)

dog_age = my_dog['age']
```

   a. AttributeError: 'Dog' object has no attribute 'age'
   b. TypeError: 'Dog' object is not subscriptable
   c. No error, it will access the age correctly
   d. SyntaxError: invalid syntax

3. **Select the correct option after executing the below code :-**

```
class Employee:
    def __init__(self, name, salary):
        self.name = name
        self.salary = salary

# Create an instance of Employee
new_employee = Employee("Alice", 50000)
```

```
income = new_employee.salary()
```

a. AttributeError: 'Employee' object has no attribute 'salary'
b. TypeError: 'int' object is not callable
c. No error, it will return the salary correctly
d. SyntaxError: invalid syntax

4. **Select the correct option after executing the below code :-**

```
class Point:
  def __init__(self, x, y):
    self.x = x
    self.y = y


point = Point()
```

a. AttributeError: 'Point' object has no attribute 'x'
b. TypeError: init() missing 2 required positional arguments: 'x' and 'y'
c. No error, it will create a Point instance with default values for x and y
d. SyntaxError: invalid syntax

5. **Select the correct option after executing the below code :-**

```
class Product:
  def __init__(self, name, price):
    self.name = name
    self.price = price

my_product = Product("Laptop", 1200)

quantity = my_product.quantity
```

a. AttributeError: 'Product' object has no attribute 'quantity'
b. TypeError: 'Product' object has no attribute 'quantity'
c. No error, it will return None for 'quantity'
d. SyntaxError: invalid syntax

6. **Select the correct option after executing the below code :-**

```
class Fruit:
  def __init__(self, name, color):
    self.name = name
    self.color = color
```

```
my_fruit = Fruit("Banana", "Yellow")

my_fruit['color'] = "Green"
```

a. AttributeError: 'Fruit' object has no attribute 'color'
b. TypeError: 'Fruit' object is not subscriptable
c. No error, it will change the color correctly
d. SyntaxError: invalid syntax

7. **Select the correct option after executing the below code :-**

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

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

converter = TemperatureConverter(20)

result = converter.to_celsius()
```

a. AttributeError: 'TemperatureConverter' object has no attribute 'to_celsius'
b. TypeError: 'TemperatureConverter' object has no attribute 'to_celsius'
c. No error, it will output the result correctly
d. SyntaxError: invalid syntax

8. **Select the correct option after executing the below code :-**

```
class StudentRecord:
    def __init__(self, name, age, grade):
        self.name = name
        self.age = age
        self.grade = grade

record = StudentRecord("Bob", 22, "B")

marks = record.marks
```

a. AttributeError: 'StudentRecord' object has no attribute 'marks'

b. TypeError: 'StudentRecord' object has no attribute 'marks'
c. No error, it will return None for 'marks'
d. SyntaxError: invalid syntax

9. **Select the correct option after executing the below code :-**

```
class Movie:
    def __init__(self, title, genre, release_year):
        self.title = title
        self.genre = genre
        self.release_year = release_year

my_movie = Movie("Inception", "Sci-Fi", 2010)

my_movie.play()
```

a. AttributeError: 'Movie' object has no attribute 'play'
b. TypeError: 'Movie' object has no attribute 'play'
c. No error, it will play the movie correctly
d. SyntaxError: invalid syntax

10. **Select the correct option after executing the below code :-**

```
class CoffeeMachine:
    def __init__(self, brand, capacity):
        self.brand = brand
        self.capacity = capacity

    def make_coffee(self):
        return f"Brewing {self.capacity}ml of coffee"

my_coffee_machine = CoffeeMachine("XYZ", 500)

machine_brand = my_coffee_machine['brand']
```

a. AttributeError: 'CoffeeMachine' object has no attribute 'brand'
b. TypeError: 'CoffeeMachine' object is not subscriptable
c. No error, it will access the brand correctly
d. SyntaxError: invalid syntax

11. What is the output of the below code?

```
class Employee:
```

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

class Manager(Employee):
    def __init__(self, name, salary, bonus):
        super().__init__(name, salary)
        self.bonus = bonus

manager = Manager("Alice", 50000, 10000)
print(manager.name, manager._Employee__salary, manager.bonus)
```

a. "Alice None 10000"
b. "Alice 50000 10000"
c. Error
d. None

12. **What is the output of the below code?**

```
class Shape:
    def area(self):
        return 0

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

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

class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return 3.14 * self.radius ** 2

shapes = [Rectangle(4, 5), Circle(3)]
for shape in shapes:
    print(shape.area())
```

a. 20, 28.26
b. 0, 0
c. 20, 9.42
d. 0, 28.26

13. **What is the output of the below code?**

```
class Animal:
    def speak(self):
        return "Generic animal sound"

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

class Cat(Animal):
    def speak(self):
        return "Meow!"

def animal_speak(animal):
    return animal.speak()

animals = [Dog(), Cat()]
for animal in animals:
    print(animal_speak(animal))
```

a. Error
b. "Generic animal sound"

   "Generic animal sound"

c. "Meow!"

   "Woof!"

d. "Woof!"

   "Meow!"

14. **What is the output of the below code?**

```
class Publication:
    def __init__(self, title, author):
        self.title = title
        self._author = author

class Book(Publication):
    def __init__(self, title, author, genre):
        super().__init__(title, author)
        self.genre = genre

class EBook(Book):
    def __init__(self, title, author, genre, file_size):
        super().__init__(title, author, genre)
        self.file_size = file_size

ebook = EBook("Python Programming", "John Smith", "Programming", 5)
print(ebook.title, ebook._author, ebook.genre, ebook.file_size)
```

a. "Python Programming John Smith Programming 5"
b. "Python Programming None Programming 5"
c. Error
d. None

15. **What is the output of the below code?**

```
class Animal:
    def __init__(self):
        self.__legs = 4      #self.__legs is a private attribute


    def get_legs(self):
        return self.__legs

class Dog(Animal):
    def __init__(self):
        super().__init__()
        self.__legs = 3

dog = Dog()
print(dog.get_legs())
```

a. 4
b. 3
c. Error
d. None

16. **What is the output of the below code?**

```
class Car:
    def __init__(self, make, model):
        self._make = make          #protected attribute
        self.__model = model       #private attribute

class SportsCar(Car):
    def __init__(self, make, model, top_speed):
        super().__init__(make, model)
        self.top_speed = top_speed

sports_car = SportsCar("Ferrari", "F40", 200)
print(sports_car._make, sports_car.top_speed)
```

a. "Ferrari F40 200"
b. "Ferrari None 200"
c. "Ferrari F40 None"
d. “Ferrari 200”

17. **What is the output of the below code?**

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

    def display(self):
        return f"Person: {self.name}"

class Employee:
    def __init__(self, salary):
        self.salary = salary

    def display(self):
        return f"Employee Salary: {self.salary}"
```

```
class Manager(Person, Employee):
    def __init__(self, name, salary, bonus):
        Person.__init__(self, name)
        Employee.__init__(self, salary)
        self.bonus = bonus

manager = Manager("Alice", 50000, 10000)
print(manager.display())
```

a. "Person: Alice"
b. "Employee Salary: 50000"
c. "Manager: Alice, Salary: 50000, Bonus: 10000"
d. Error

18. **What is the output of the below code?**

```
class Base:
    def __init__(self):
        self.value = 10

class A(Base):
    def __init__(self):
        super().__init__()
        self.value += 1

class B(Base):
    def __init__(self):
        super().__init__()
        self.value += 2

class C(A, B):
    def __init__(self):
        super().__init__()

obj = C()
print(obj.value)
```

a. 10
b. 11
c. 12
d. 13

19. **What is the output of the below code?**

```
class X:
    def show(self):
        return "X"

class Y(X):
    def show(self):
        return "Y"

class Z(X):
    def show(self):
        return "Z"

class W(Y, Z):
    pass

obj = W()
print(obj.show())
```

a. "X"
b. "Y"
c. "Z"
d. Error

20. **What is the output of the below code?**

```
class CoffeeMachine:
    def __init__(self):
        self._water_level = 1000
        self._coffee_beans = 200

class EspressoMachine(CoffeeMachine):
    def make_espresso(self):
        self._water_level -= 30
        self._coffee_beans -= 15

espresso_machine = EspressoMachine()
espresso_machine.make_espresso()
print(espresso_machine._water_level, espresso_machine._coffee_beans)
```

a. 970 185
b. 1000 200
c. 970 None
d. Error