poly-and-encap

May 18, 2024

0.0.1 Q1a. Create a base class Animal with a method make_sound(). Define two subclasses, Dog and Cat, that inherit from Animal and override the make_sound() method.

[1]:
```
class Animal:
    def make_sound(self):
        pass

class Dog(Animal):
    def make_sound(self):
        print("Bark")

class Cat(Animal):
    def make_sound(self):
        print("Meow")

# Test the classes
dog = Dog()
cat = Cat()
dog.make_sound()  # Output: Bark
cat.make_sound()  # Output: Meow
```

```
Bark
Meow
```

0.0.2 b. Create a function that can calculate the sum of two numbers or concatenate two strings.

[2]:
```
def add(a, b):
    return a + b

# Test the function
print(add(3, 5))          # Output: 8
print(add("Hello, ", "world!"))  # Output: Hello, world!
```

```
8
Hello, world!
```

0.0.3 **Q2a. Create a BankAccount class with a protected attribute _balance. Implement a method get_balance() to retrieve the balance.**

[3]:
```
class BankAccount:
    def __init__(self, initial_balance=0):
        self._balance = initial_balance

    def get_balance(self):
        return self._balance

# Test the class
account = BankAccount(100)
print(account.get_balance())  # Output: 100
```

```
100
```

0.0.4 **b. Create a Python module calculator with a private function ___add that adds two numbers. Implement a public function add_numbers that calls the private function.**

[4]:
```
class Calculator:
    def __init__(self):
        pass

    def __add(self, a, b):
        return a + b

    def add_numbers(self, a, b):
        return self.__add(a, b)

# Test the module
calc = Calculator()
print(calc.add_numbers(3, 4))  # Output: 7
```

```
7
```

0.0.5 Q3)Create an abstract class Shape with an abstract method calculate_area(). Define two subclasses, Rectangle and Circle, that inherit from Shape and implement the calculate_area() method.

[5]:
```
from abc import ABC, abstractmethod
import math

class Shape(ABC):
    @abstractmethod
    def calculate_area(self):
        pass
```

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

    def calculate_area(self):
        return self.width * self.height

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

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

# Test the classes
rectangle = Rectangle(4, 5)
circle = Circle(3)

print(rectangle.calculate_area())  # Output: 20
print(circle.calculate_area())     # Output: 28.274333882308138
```

```
20
28.274333882308138
```