**1 Which principle does this illustrate?**

**class Dog:**

**def sound(self):**

**print("Bark")**

**class Cat:**

**def sound(self):**

**print("Meow")**

**def animal\_sound(animal):**

**animal.sound()**

A. Inheritance  
B. Encapsulation  
C. Polymorphism  
D. Overloading

Ans C

**2 Which of these is an example of polymorphism?**  
A. def \_\_init\_\_()  
B. def add(a, b) and def add(a, b, c)  
C. len("abc") and len([1,2,3])  
D. super().\_\_init\_\_()

Ans C

**3 Which Python feature allows method polymorphism naturally?**  
A. Duck typing  
B. Type casting  
C. Function overloading  
D. Dynamic memory

Ans A

4 **How do you make a variable private in Python?**  
A. Start with \_  
B. Start with \_\_  
C. Use private keyword  
D. Use @private decorator

Ans B

5 **What does name mangling do in Python?**  
A. Encrypts data  
B. Converts variable to binary  
C. Renames \_\_var to \_ClassName\_\_var  
D. Hides all class attributes

Ans C

6 **What happens if you define the same method twice in a class?**  
A. Both methods will run  
B. Error  
C. Last method overrides previous  
D. Random behavior

Ans C

7 **Which keyword is used to call the parent class method in overriding?**  
A. base()  
B. parent()  
C. super()  
D. self()

Ans C

8 **Which statement is true about method overriding?**  
A. Both methods must have different names  
B. Overriding is done in the same class  
C. Signature must match in parent and child  
D. Only constructors can be overridden

Ans C

9 **Which of the following is NOT an encapsulation practice in Python?**  
A. Using \_\_var for private members  
B. Wrapping data and methods in classes  
C. Using decorators for class security  
D. Making class variables global

Ans D

10 What is the following code print

class A:

def \_\_init\_\_(self):

self.\_\_value = 10

a = A()

print(a.\_\_value)

A. 10  
B. None  
C. Error  
D. 0

Ans C