**Activity 4: Identify the Concept**

**Task 1**

**Instructions:**

1. Look at the following two Python code snippets:

**Snippet 1:**

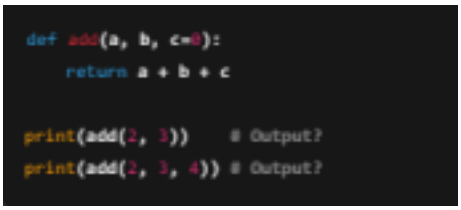
```
class Dog:
    def speak(self):
        return "Woof!"

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

animals = [Dog(), Cat()]

for animal in animals:
    print(animal.speak())
```

**Snippet 2:**



```
def add(a, b, c=0):
    return a + b + c

print(add(2, 3))    # Output?
print(add(2, 3, 4)) # Output?
```

2. What do you observe in both snippets?

ANSWER:

Snippet 1 demonstrates method overriding (polymorphism).

Snippet 2 illustrates function argument handling and the absence of function overloading in

Python.

3. How does Python handle different implementations of the speak() method in Snippet 1?

ANSWER:

Python allows different classes to have methods with the same name but different implementations.

4. How does Python handle the add() function when given different numbers of arguments in Snippet 2?

ANSWER:

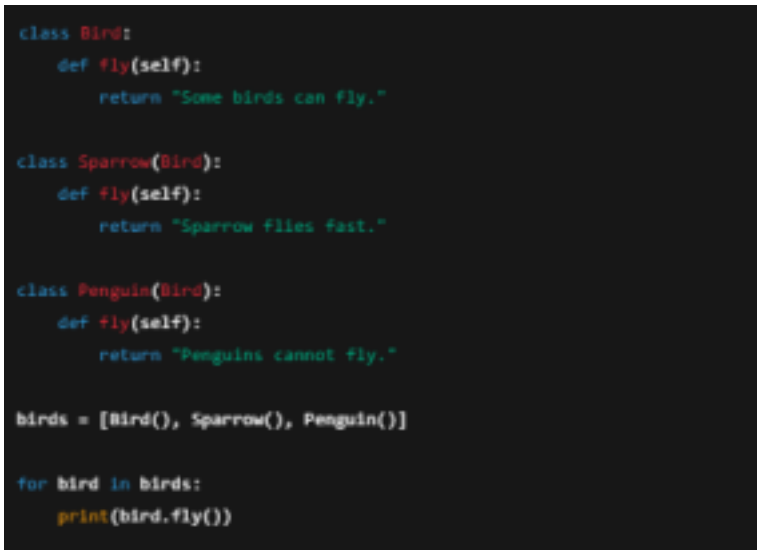
If add() is called with missing arguments, Python throws a TypeError.

To fix this, we can provide a default value for c, e.g., def add(a, b, c=0):.

**Task 2**

**Instructions:**

Analyze the following code and predict the output before running it.



```
class Bird:
    def fly(self):
        return "Some birds can fly."

class Sparrow(Bird):
    def fly(self):
        return "Sparrow flies fast."

class Penguin(Bird):
    def fly(self):
        return "Penguins cannot fly."

birds = [Bird(), Sparrow(), Penguin()]

for bird in birds:
    print(bird.fly())
```

1. What do you think the output will be?

ANSWER:

Some bird can fly.

Sparrow flies fast.

Penguins cannot fly.

2. Why do different objects return different results for the same method (fly())?

ANSWER:

The reason different objects return different results for the same method, like fly(), is due to in object-oriented programming.