Inheritance and Polymorphism in Python

Inheritance

Inheritance in Python works similarly to Java, allowing one class to derive properties and methods from another.

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

class Student(User):
    def __init__(self, name, email, batch_name, psp):
        super().__init__(name, email)
        self.batch_name = batch_name
        self.psp = psp
```

Types of Inheritance

Python supports single, multi-level, and multiple inheritance. The diamond problem is also a consideration in Python when using multiple inheritance.

The Diamond Problem

Python has a method resolution order (MRO) that defines the order in which base classes are searched when executing a method. This solves the diamond problem.

Polymorphism

Polymorphism in Python is less formal than in Java. Python uses duck typing, which means that an object's suitability is determined by the presence of certain methods and properties, rather than the actual type of the object.

Duck Typing

If it looks like a duck and quacks like a duck, it's a duck. This idiom is applied in Python to determine if an object can be used for a particular purpose.

```
def quack_and_fly(thing):
    # Notice we don't check the type of `thing`
    thing.quack()
    thing.fly()
    print("This thing can quack and fly!")

class Duck:
    def quack(self):
        print("Quack, quack!")

    def fly(self):
        print("Flap, flap!")

class Person:
    def quack(self):
        print("I'm Quacking Like a Duck!")

    def fly(self):
        print("I'm Flapping my Arms!")

d = Duck()
p = Person()

quack_and_fly(d)
quack_and_fly(p)
```

Method Overriding

Method overriding is used in Python to alter the implementation of a method defined in a base class.

```
class User:
    def print_user(self):
        print(f"Name: {self.name}, Email: {self.email}")

class Student(User):
    def print_user(self):
        print(f"Name: {self.name}, Email: {self.email}, Batch: {self.batch_name}
```

Advantages of Polymorphism

- Code reusability
- Improved code readability
- Easier to debug

Problems with Polymorphism

- Increased complexity
- Downcasting issues
- Performance overhead in dynamic typing