Name: Abd Elrhaman Magdy **ID**:1261E

1.1 the difference between shallow and deep copying:-

1. Shallow Copy

Shallow copying creates a new copy of an object, but instead of copying nested objects (such as lists within another list), it only copies references or pointers to them.

How it works:

Imagine you have a parent folder (the original list) containing regular files and subfolders. Shallow copying creates a new parent folder, copies the regular files to it, but places only shortcuts to the original subfolders.

Result:

If you modify a regular object (such as a number or a string), the change will be reflected only in the new copy.

However, if you modify a modifiable object (such as a list or dictionary) within the new copy, this change will also be reflected in the original object, since both refer to the same nested object in memory.

Shallow copying methods:

Use the copy() function from the copy library.
Use a full slice, such as list[:].

2. Deep Copy

Deep copy is the process of creating a new copy of an object and recursively copying all nested objects.

How it works:

Using the same example, deep copy creates a new parent folder and completely copies all files and subfolders within it.

Result:

A completely new, independent object is created.
Any modifications to nested objects in the new copy will not affect the original object.

Deep copy methods:

Using the deepcopy() function from the copy library.

1.2 multiple inheritance:-

What is multiple inheritance?
Multiple inheritance is a feature that allows a class to inherit from more than one parent class.

In other words, a class can inherit attributes and methods from multiple parent classes. This allows you to create classes that combine different functions from multiple sources, rather than having to rewrite the code.

A real-life example:
Imagine you have a Car class and a Boat class. You can create a new class called AmphibiousVehicle that inherits from both. This way, the amphibious vehicle can have both the properties of a car (such as wheels and an engine) and the properties of a boat (such as buoyancy and oars).

How to Implement Multiple Inheritance in Python
In Python, multiple inheritance is implemented simply by placing the parent class names inside the parent parentheses of the new class definition, separated by commas.

```
class Parent1:
    pass

class Parent2:
    pass

class Child(Parent1, Parent2):
    pass
```

Method Resolution Order (MRO)

Here comes the important part of multiple inheritance. If both parent classes have the same method with the same name, how will the child class know which one to call?

This is called Method Resolution Order (MRO). Python follows an algorithm known as C3 Linearization to determine the order in which to search for a method.

Simplified, Python searches for a method first in the child class itself, then in the parent classes in the order in which they are defined, then in their ancestor classes, and so on.

You can check the MRO order of any class using the special attribute __mro__ or the help() function.

A practical example illustrates the concept and MRO.

```python
# Define the first parent class
class Engineer:
    def code(self):
        print("I am writing code.")

# Define the second parent class
class Musician:
    def play_instrument(self):
        print("I am playing the guitar.")

    def greet(self):
        print("Hello, I am a musician.")

# Define the third parent class (to show MRO)
class Athlete:
    def greet(self):
        print("Hello, I am an athlete.")

# Define the child class that inherits from Engineer, Musician, and Athlete
class SoftwareEngineerMusician(Engineer, Musician, Athlete):
    def work(self):
        print("I am a software engineer and a musician.")

# Create an instance of the child class
person = SoftwareEngineerMusician()

# The child class can use methods from both parent classes
person.code()
person.play_instrument()
person.work()

# Demonstrate Method Resolution Order (MRO)
# The "greet" method from Musician will be called because it comes first in the
inheritance list.
person.greet()

# You can check the MRO of the class
print("\nMethod Resolution Order (MRO):")
print(SoftwareEngineerMusician.__mro__)

# You can also use the built-in help() function
# help(SoftwareEngineerMusician)
```

As you can see in the example, the SoftwareEngineerMusician class inherits all methods from Engineer, Musician, and Athlete. When person.greet() is called, Python executes the method from the Musician class because it comes before Athlete in the inheritance sequence.

Advantages and Disadvantages

Advantages:

Flexibility: Allows complex classes to be built from simple components.

Code Reusability: Reduces the need for code duplication.

Disadvantages:

Complexity: Can lead to complex class structures and make code behavior difficult to track.

MRO Issues: Problems can occur if the method resolution order is not well understood, leading to the wrong method being called.