Super task

1. If I don't call super(), which constructor will be executed?

Python will not automatically call the constructors of its parent classes. This means that the parent classes' constructors will not be executed unless you manually invoke them.

Example:

```
class A:
    def __init__(self):
        print("Initializing A")

class B:
    def __init__(self):
        print("Initializing B")

class C(A, B):
    def __init__(self):
        print("Initializing C")
```

Therefore, only the constructor of class C is called, and the constructors of classes A and B are not invoked.

2. If I use super() in a multiple inheritance scenario, which constructor will be called, and how can I ensure each one is called?

When using super() in a multiple inheritance scenario, Python follows the Method Resolution Order (MRO) to determine the sequence in which classes are initialized. The MRO is a list that defines the order in which base classes are considered when executing a method.

Example:

```
class A:
    def __init__(self):
        print("Initializing A")

class B:
    def __init__(self):
        print("Initializing B")

class C(A, B):
    def __init__(self):
        super().__init__()
```

It's lead to: super() calls the constructor of class A because it is the first class in the MRO of class C.

To ensure that each constructor is called, you can explicitly call each parent class's constructor:

```
class C(A, B):
    def __init__(self):
```

A.__init__(self)

B.__init__(self)

print("Initializing C")

In this case, all constructors are called explicitly.