In-class Exercise 1

- In this exercise, we will apply `ABC` and `abstractmethod` to create Shape3D class and its subclasses.

Tasks (1-4):

1. **Create Shape3D class with the following requirements:**
   - Abstract method:
     - `volume`: computes the volume of a Shape3D instance
     - `surface_area`: computes the surface area of a Shape3D instance
   - Concrete method:
     - `describe`: returns a description of an instance.

In-class Exercise 1

2. **Create its subclasses named `Cube`, `Sphere`, and `Cylinder`.**
   - `Cube` subclass:
     - `__init__` with argument: `side`
   - `Sphere` subclass:
     - `__init__` with argument: `radius`
   - `Cylinder` subclass:
     - `__init__` with arguments: `radius`, `height`

3. **Test polymorphism by creating the following instances:**
   - `Cube` with the side of 2
   - `Sphere` with the radius of 3
   - `Cylinder` with the radius of 2 and the height of 5.

4. **Finally, display the results.**

In-class Exercise 1

- Starter code available in LearnUs.

In-class Exercise 2

```
class A:
    def hello(self):
        print("Hello from A")

class B(A):
    def hello(self):
        print("Hello from B")
        super().hello()

class C(A):
    def hello(self):
        print("Hello from C")
        super().hello()
```

```
class D(B, C):
    def hello(self):
        print("Hello from D")
        super().hello()
```

Tasks (1-3):

1. Create a new class E that inherits from C.
2. Create class F(B, E). Print F.mro() and explain the order.
3. Try to design a class where Python raises an MRO conflict error.