Think Python

Write a Python program with polymorphism that is usable within the summative assessment for the humanoid robot.

```
# Base class
class Robot:
    def __init__(self, name):
        self.name = name

    def move(self):
        raise NotImplementedError("Subclasses should implement this method")

    def speak(self):
        raise NotImplementedError("Subclasses should implement this method")

# Humanoid Robot subclass
class HumanoidRobot(Robot):
    def __init__(self, name, height, weight):
        super().__init__(name)
        self.height = height
        self.weight = weight

    def move(self):
        print(f"{self.name} is walking on two legs.")

    def speak(self):
        print(f"{self.name} says: Hello, I am a humanoid robot.")

# Wheeled Robot subclass
class WheeledRobot(Robot):
    def __init__(self, name, wheel_count):
        super().__init__(name)
        self.wheel_count = wheel_count

    def move(self):
        print(f"{self.name} is rolling on {self.wheel_count} wheels.")

    def speak(self):
        print(f"{self.name} says: Beep boop, I am a wheeled robot.")

# Flying Robot subclass
class FlyingRobot(Robot):
    def __init__(self, name, wing_span):
        super().__init__(name)
        self.wing_span = wing_span

    def move(self):
        print(f"{self.name} is flying with a wing span of {self.wing_span} meters.")

    def speak(self):
        print(f"{self.name} says: Whoosh, I am a flying robot.")

# Function to test polymorphism
def robot_action(robot):
    robot.move()
    robot.speak()

# Creating instances of the robots
humanoid = HumanoidRobot("RoboMan", 1.8, 75)
wheeled = WheeledRobot("RoboCar", 4)
flying = FlyingRobot("RoboFly", 2.5)

# Using polymorphism: All robots can be interacted with via the base class reference
```

```
robots = [humanoid, wheeled, flying]

for robot in robots:
    robot_action(robot)
    print()  # Empty line between robot actions
```

Base Class: Robot, Defines common behaviour (methods move() and speak()) that all robot types should implement, but leaves the implementation to the subclasses.

Subclasses: HumanoidRobot, WheeledRobot, FlyingRobot. Each of these subclasses implements its own version of the move() and speak() methods.

Polymorphism: The function robot_action() accepts any object that is an instance of the Robot class (or its subclasses), demonstrating polymorphism by calling the overridden methods (move() and speak()) without needing to know the exact type of the robot.

```
RoboMan is walking on two legs.
RoboMan says: Hello, I am a humanoid robot.

RoboCar is rolling on 4 wheels.
RoboCar says: Beep boop, I am a wheeled robot.

RoboFly is flying with a wing span of 2.5 meters.
RoboFly says: Whoosh, I am a flying robot.
```