An example of a Python program which uses polymorphism is shown below.

```
class Cat:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def info(self):
        print(f"I am a cat. My name is {self.name}. I am {self.age} years old.")

    def make_sound(self):
        print("Meow")


class Dog:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def info(self):
        print(f"I am a dog. My name is {self.name}. I am {self.age} years old.")

    def make_sound(self):
        print("Bark")


cat1 = Cat("Kitty", 2.5)
dog1 = Dog("Fluffy", 4)

for animal in (cat1, dog1):
    animal.make_sound()
    animal.info()
    animal.make_sound()
```

Source: Programiz. (n.d.) **Polymorphism in Python**.

1. **Write a Python program with a polymorphism that is usable within the summative assessment for the driverless car.**

The following is a summative assessment of my Python software that uses a driverless car to demonstrate polymorphism:

```
class Vehicle:
    def __init__(self, make, model, year):
        self.make = make
        self.model = model
        self.year = year

    def make_sound(self):
        print("Beep beep!")

    def drive(self):
        print("I'm driving!")
```

```
class SelfDrivingCar(Vehicle):
    def __init__(self, make, model, year, max_speed):
        super().__init__(make, model, year)
        self.max_speed = max_speed

    def make_sound(self):
        print("Beep beep! I'm a self-driving car!")

    def drive(self):
        print("I'm driving autonomously at a speed of {} mph!".format(self.max_speed))

def main():
    vehicle = Vehicle("Honda", "Accord", 2023)
    car = SelfDrivingCar("Tesla", "Model 3", 2022, 30)

    vehicle.make_sound()
    vehicle.drive()

    car.make_sound()
    car.drive()

if __name__ == "__main__":
    main()
```

(Daulton, R., 2018).

Vehicle and SelfDrivingCar are the two classes that this programme defines. The SelfDrivingCar class derives from the primary class, the Vehicle class. This means that in addition to any new methods and characteristics defined in the SelfDrivingCar class, the SelfDrivingCar class also contains all of the methods and attributes of the Vehicle class.

This programme's Vehicle and SelfDrivingCar classes define the make_sound() and drive() functions. This exemplifies polymorphism. Because of polymorphism, a method's implementation can vary based on the class from which it is called.

Depending on whether the make_sound() method is invoked from a Vehicle or a SelfDrivingCar object in this programme, it will output a different message. It will print “Beep beep!” if it receives a call from a vehicle object. It prints “Beep beep! I'm a self-driving car!” if invoked from a SelfDrivingCar object.

The drive() method has a distinct implementation depending on the class it is called from. It will print “I'm driving!” if invoked from a vehicle object. “I'm driving autonomously at a speed of {} mph!” it will print if it is called from a SelfDrivingCar object.

References:

Daulton, R. (2018). *Python 3: vehicle inventory using class*. [online] Available at: https://stackoverflow.com/questions/50722188/python-3-vehicle-inventory-using-class.