Object-oriented Programming (OOP): Inheritance

**OOP: Data classes in Python**

1. What are Data Classes?

Data classes are a new feature available in Python built-ins since version 3.7. They provide a decorator and functions to create simpler, more convenient, and safer classes that are mainly used to process data, hence the name.

One of the main benefits of Data Classes is that they automatically generate several special methods for you, such as \_\_init\_\_, \_\_repr\_\_ and \_\_eq\_\_. This can save you a lot of time and redundant code when defining classes that are primarily used to process data.

Another main benefit of Data Classes is the use of strong typing, which ensures that the attributes of an instance are defined. This is achieved through the use of type annotations, which allow you to specify the type of each attribute when defining the class. This can prevent bugs due to type ambiguities and also makes the code easier to catch for other developers.

1. How to use Data Classes

To use Data Classes, you first need to import the dataclass decorator from the dataclasses module. This decorator is natively included in Python 3.7 and higher.

Using Data Classes is very simple. Just decorate your class definition with the @dataclass decorator to define a dataclass. Here is an example of a simple dataclass with default parameters:

from dataclasses import dataclass, field

@dataclass

class Point:

x: float

y: float

p = Point(1.0, 2.0)

print(p) # Output: Point(x=1.0, y=2.0)

We have defined a Point dataclass with two fields, x and y, both of which are floats. When you create an instance of the Point class, you can specify the values for x and y as arguments to the constructor.

By default, Data Classes will generate a \_\_init\_\_ method for you that takes the fields of the class as arguments. They will also generate a \_\_repr\_\_ method that returns a string representation of the object, which is what is printed when you call print(p) in the example above.

You can customize which methods are generated by the decorator by passing additional arguments, such as repr=False to deactivate the \_\_repr\_\_ method.