Codio activity: Inheritance concepts

The object-oriented programming (OOP) technique has been embraced by many programmers working in various programming languages, such as C++, Java, Python, etc. Inheritance, one of the four primary pillars of Object-Oriented Programming, is one of the most essential parts of the OOPs idea. The other three pillars are polymorphism, encapsulation, and data abstraction (Singh, 2020).

Furthermore, Singh (2020) added that in object-oriented programming (OOP), inheritance is essential for a class to inherit traits from another class. This core idea creates meaningful class interactions and is targeted towards code reuse.

A derived class (also known as a subclass or child class) inherits the properties of a base class (also known as a superclass or parent class) according to the inheritance hierarchy. In essence, the derived class extends the capabilities of the base class by incorporating new elements or replacing outdated ones.

The fundamental tenet of inheritance is that a derived class adds its unique attributes and behaviours while simultaneously inheriting all the base class's characteristics (data members) and behaviours (methods).

The ability to create class hierarchies with shared attributes and methods in Python is made possible by the language's robust inheritance feature (Srivastava1, S. 2018). We can create a parent or base class, a child or a derived class that inherits from it by deriving a class from another class. Because of inheritance, parent class code can be reused in the child class without being completely rebuilt. It also allows updating or adding new features to the child class without affecting the parent class. Code is easier to understand and maintain when objects are related in a real-world manner.

The syntax to use inheritance in Python is as follows based on W3schools (2019) examples:

```
class ParentClass:
   #body of parent class

class ChildClass(ParentClass):
   #body of child class
```

The child class can replace existing attributes and methods or define new ones while inheriting all of the parent class's attributes and methods. We can use the super() function or the parent class's name to access the properties or methods of the parent class from the child class.

Depending on how many parent classes of the classes are involved, there are many types of inheritance in Python. These are what they are according to Srivastava1, S. (2018):

Single inheritance is the process through which a child class derives from just one parent class. This was demonstrated in the preceding message.

Multiple inheritances occur when a child class derives from several parent classes. For instance:

```
class A:
   #body of class A

class B:
   #body of class B

class C(A, B):
   #body of class C
```

Class C in this instance, is descended from classes A and B. When combining the functionality of distinct classes, multiple inheritances can be helpful, but it can also be confusing and complex if the parent classes have contradictory methods or attributes.

Multilevel inheritance is the process by which a child class derives from another class, which in turn derives from a parent class. For instance:

```
class A:
    #body of class A

class B(A):
    #body of class B

class C(A):
    #body of class C
```

Here, class A is the ancestor of classes B and C. When we wish to develop various subclasses from a single-parent class with shared characteristics, hierarchical inheritance might be helpful.

References:

Singh, H. (2020). *Inheritance in Object Oriented Programming | Inheritance in OOPs*. [online] Analytics Vidhya. Available at: https://www.analyticsvidhya.com/blog/2020/10/inheritance-object-oriented-programming/.

Srivastava1, S. (2018). *Inheritance in Python*. [online] Available at: https://www.geeksforgeeks.org/inheritance-in-python/.

W3schools.com. (2019). *Python Inheritance*. [online] Available at: https://www.w3schools.com/python/python_inheritance.asp.