Programming for data science - Question 01 - Part A - COMScDS221P-015

Inheritance is one of the key concepts in object-oriented programming (OOP) in Python. It uses where a child class inherits the features, attributes, parameters, and behaviors from the parent class. The main advantage of inheritance is the ability to reuse the components of parent class can be used in the child class defined earlier and reduces the size of the program.

In the real-world cars have common features that we call as a car. As a basic car has its own car color, model name, manufacturer name, engine, car seats, and exterior components. For instance, if the parent class as a cars could have common features of above mentioned properties and each class which could have bearing inherited those properties can be called as child classes. The concept of inheritance is explained in the below Python code.

class Cars:

```
    def _init_(self, make, model):
        self.make = make
        self.model = model

    def get_details(self):
        return "Make: {self.make}, Model: {self.model}"
```

class Sedan(Cars):

```
    def _init_(self, make, model, year):
    # Initialize attributes of parent class
        super()._init_(make, model)
        self.year = year

    def get_details(self):
        # Override method of parent class
        return "Make: {self.make}, Model: {self.model}, Year: {self.year}"
```

class SUV(Cars):

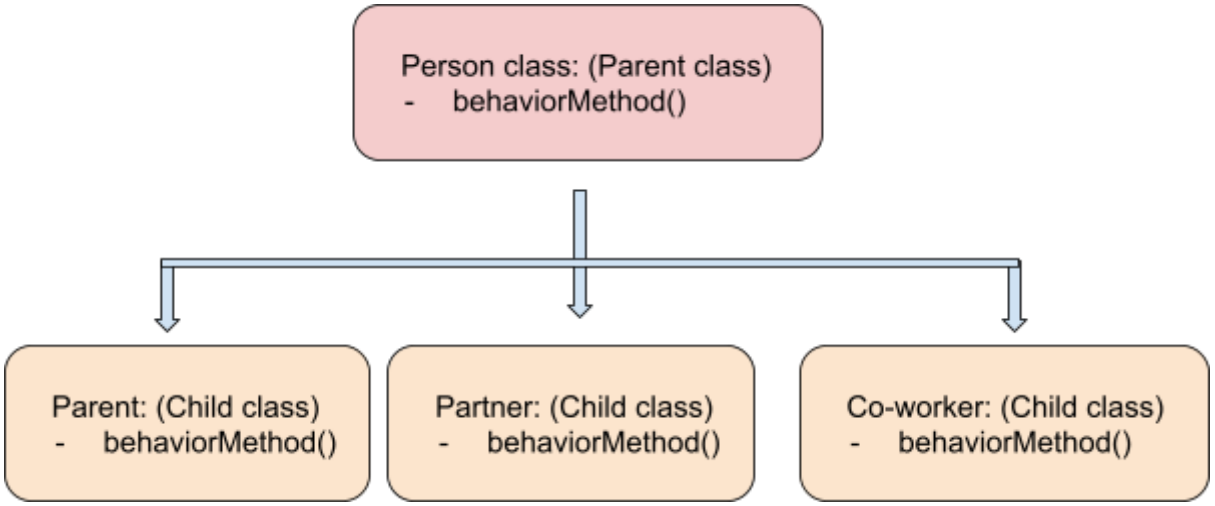
```
    def _init_(self, make, model, seating_capacity):
        # Initialize attributes of parent class
        super()._init_(make, model)
        self.seating_capacity = seating_capacity
```

Question 01 - Part G

Polymorphism

Polymorphism means to have many forms. As a real-world example, a person can play many roles in his life such as a partner, a parent, as a co-worker, as a manager, and so on. But all together is playing all the roles in one person.

Polymorphism is the objects, parameters, and functions which are owned by certain parent class or child classes and also having code objects of same identical names in the same inherited hierarchical tree but may have different actions and behaviors according to the situation or use cases.



Polymorphism has two main concepts which are overloading and overriding. These two concepts use in different use-case scenarios.

Overloading is when a child class inherits the parameters and properties of a parent class and also adds more parameters in the same name to the existing or new child classes to function more features in the program. As an example, a person has many common qualities and behaviors. Such as first name, last name, salutation, age, and education level. In order to use features as parameters in a sub-child class while adding new parameters which are not available in the parent class or other child classes.

Overriding is when the existing function name, parameters, and functions are the same but the function body is different, creating a change in an existing code method to a new code method in an existing class or new class.

The concept of Polymorphism is explained in the following Python code.

```
class Person:
  def _init_(self, fname, lname):
    self.firstname = fname
    self.lastname = lname

  #overloading
  def printname(self):
    print(self.firstname, self.lastname)

  def printname(self,salutation):
    print(salutation,self.firstname, self.lastname)

  def printname(self,salutation,education):
    print(salutation,education,self.firstname, self.lastname)


class Student(Person):
  #overriding
  def printname(self):
      print("Student's name is ", self.firstname, self.lastname)
```