Python Inheritance

Inheritance allows us to define a class that inherits all the methods and properties from another class.

`Parent` class is the class being `inherited from` , also called `base` class.

`Child` class is the class that `inherits from another class` , also called `derived` class.

Create a Parent Class

Any class can be a parent class, so the syntax is the same as creating any other class:

In [18]:
```
# Example
# Create a class named Person,
# with firstname and lastname properties, and a printname method:

class Person:
    def __init__(self, fname, lname):
        self.firstname = fname
        self.lastname = lname
        print("Hello, that's parent class")
    def printname(self):
        print(self.firstname, self.lastname)

#Use the Person class to create an object,
# and then execute the printname method:

x = Person("John", "Doe")
x.printname()
```

```
Hello, that's parent class
John Doe
```

Create a Child Class

To create a class that inherits the functionality from another class, send the parent class as a parameter when creating the child class:

In [2]:
```
# Example
# Create a class named Student, which will inherit
# the properties and methods from the Person class:

class Student(Person):
    pass
```

Note: Use the pass keyword when you do not want to add any other properties or methods to the class.

Now the Student class has the same properties and methods as the Person class.

In [19]:
```
# Example
# Use the Student class to create an object,
# and then execute the printname method:

x = Student("Mike", "Olsen")
x.printname()
```

```
His name is Mike and his father name is Olsen
Mike Olsen
```

Add the init() Function

So far we have created a child class that inherits the properties and methods from its parent.

We want to add the **init**() function to the child class (instead of the pass keyword).

Note: The init() function is called automatically every time the class is being used to create a new object.

In [23]:
```
# Example
# Add the __init__() function to the Student class:

class Student(Person):
    def __init__(self, fname, lname):
        self.firstname = fname
        self.lastname = lname
        print(f"His name is {self.firstname} and his father name is {self.lastn
z = Student("Mahmoud","Samir")
z.printname()
```

```
His name is Mahmoud and his father name is Samir
Mahmoud Samir
```

When you add the **init**() function, the child class will no longer inherit the parent's **init**() function.

Note: The child's init() function overrides the inheritance of the parent's init() function.

To keep the inheritance of the parent's **init**() function, add a call to the parent's **init**() function:

In [25]:
```
# Example
class Student(Person):
    def __init__(self, fname, lname):
        Person.__init__(self, fname, lname)
```

Now we have successfully added the init() function, and kept the inheritance of the

parent class, and we are ready to add functionality in the init() function

Use the super() Function

Python also has a super() function that will make the child class inherit all the methods and properties from its parent:

In [26]:
```
# Example
class Student(Person):
    def __init__(self, fname, lname):
        super().__init__(fname, lname)
```

By using the super() function, you do not have to use the name of the parent element, it will automatically inherit the methods and properties from its parent.

Add Properties

In [27]:
```
# Example
# Add a property called graduationyear to the Student class:

class Student(Person):
    def __init__(self, fname, lname):
        super().__init__(fname, lname)
        self.graduationyear = 2019
```

In the example below, the year 2019 should be a variable, and passed into the Student class when creating student objects. To do so, add another parameter in the init() function:

In [28]:
```
# Example
# Add a year parameter, and pass the correct year when creating objects:

class Student(Person):
    def __init__(self, fname, lname, year):
        super().__init__(fname, lname)
        self.graduationyear = year

x = Student("Mike", "Olsen", 2019)
```

```
Hello, that's parent class
```

Add Methods

In [29]:
```
# Example
# Add a method called welcome to the Student class:

class Student(Person):
    def __init__(self, fname, lname, year):
        super().__init__(fname, lname)
        self.graduationyear = year

    def welcome(self):
        print("Welcome", self.firstname, self.lastname,
              "to the class of", self.graduationyear)
```

If you add a method in the child class with the same name as a function in the parent class, the inheritance of the parent method will be overridden.

In [31]:
```
x = Student("Mike", "Olsen", 2019)
x.welcome()
```

```
Hello, that's parent class
Welcome Mike Olsen to the class of 2019
```

In [ ]: