Classes

This lesson introduces object-oriented programming using classes, attributes, and methods.

Classes

In object-oriented programming (OOP), a class is a structure that allows you to group together a set of properties (called **attributes**) and functions (called **methods**) to manipulate those properties. Take the following class that defines a person with the `name` and `age` properties and the `greet()` method.

```
class Person:

      def __init__(self, name, age): # class constructor
          self.name = name # class variable
          self.age = age # class variable

      def greet(self): # class function to print a greeting
          print("Hello, my name is %s!" % self.name)
```

Most classes will need the constructor method (`__init__`) to **initialize** the class's attributes. In the above code snippet, the constructor of the class receives the person's name and age and stores that information in the **class's instance** (referenced by the `self` keyword). Finally, the `greet()` method prints the name of the person as stored in a **specific class instance (object)**.

Here's how we can instantiate two objects:

```
class Person:

     def __init__(self, name, age): # class constructor
         self.name = name # class variable
         self.age = age # class variable
     def greet(self): # class function to print a greeting
         print("Hello, my name is %s!" % self.name)



a = Person("Peter", 20) # instantiation
b = Person("Anna", 19) # instantiation

a.greet() # call a's greet method
b.greet() # call b's greet method

print(a.name)
print(a.age)  # We can also access the attributes of an object

print(b.name)
print(b.age)  # We can also access the attributes of an object
```

Let’s solve an exercise to practice the concepts you just learnt.