Module 3. Object oriented programming

Classes

Classes

A class is a new type that you can define in Python. Classes are so-called templates or blueprints for objects.

How to create a class

You can create a class like you define functions, but using the **class** keyword instead.

```
class Person():
  #define it
  pass
```

To give it more content you need to use the initializer method.

Initializer method

```
class Person():
  def __init__(self, name, age):
      self.name = name
      self.age= age
```

Notice that when we defined __init__ we specified we expected to receive three arguments: self, name and age. But when we created a new instance of the class we only gave it two arguments. This is because self is a special argument that is always passed to methods.

Instance methods

You can create more methods inside a class by writing functions within the class.

```
class Person():
  def __init__(self, name, age):
     self.name = name
     self.age= age
  def greet(self):
     print(f"Hi {self.name}! You are {self.age} years old.")
```

Objects/Instance

A construction created from a class with an instance

```
bob = Person('Bob', 42)
```

Name the variable and use the class and the desired arguments inside the parenthesis.

Call instance methods

Once you've created a class with method inside, you can call them by adding a dot and the name of the method to the name of the variable.

```
print(bob.age)
#Will print 42
bob.greet()
#Will print the greeting defined within the class
```

Value error

In Python, a ValueError is a type of exception that is raised when a function or operation receives an argument with the correct data type but an inappropriate value. In other words, the data type is correct, but the value of the argument is not allowed or valid for the given operation.

```
def divide(a, b):
    if b == 0:
        raise ValueError("The second argument (b) cannot be zero.")
    return a / b
```

Use the raise keyword indented after the if statement and then a string with the error that you want to show.

Getattr()

getattr(object, name)
Return the value of the named attribute of *object*. *name* must be a string. If the string is the name of one of the object's attributes, the result is the value of that attribute.