Python Programming

Lesson 3 – Object Oriented Programming (OOP)

Lesson 3 - Outline

- Class and Object
- Instantiate Object
- Add attributes to Class
- Define methods in a class
- Pass arguments to methods
- Four basic concepts of OOP

What is Object Oriented Programming (OOP)

What is Object Oriented Programming

- A programming concept/model
- Use the concept of classes and objects
- Make the structure of a software program become simple and reusable pieces of code
- Create individual instances of objects
- Examples of programming language
  - Python, Java, C#, etc.

Class and Object

- Relationship between class and object
  - Example of cookie cutters and cookies
  - The cookie cutter is the class which defines the characteristics of each cookie, for example size and shape
  - The class is used to create objects
  - The objects are the cookies

Class and Object

- Definition of Class
  - A class describes the **class variables (~ attributes)** and **methods, etc.** of an object
  - Blueprint of objects
- Definition of Object
  - Each object in Python is defined by a class
  - Objects are instances of classes, you can create as many objects you need once you have defined a class
  - Basically, you need to create an object before you can access its members

How to create a class

OOP Examples

- Let’s go through an example step by step!

OOP Example

- Create a class
- Instantiate an object
- Add attributes to class
- Define methods in a class
- Pass arguments to methods

Create a class

Create a class

- **Define a class**
- **<u>Syntax</u>**

Use the class keyword, followed by the class name and a colon

```
class Classname:
```

- **<u>Example 1a</u>**

```
class Dog:
```

Create a class

- **Use of** `__init__` within the class
- For initialization of object
  (similar to a constructor in Java)
- **Syntax**

1) Inside the class, an `__init__` method has to be defined with `def`
2) With argument `self`, refers to the object itself
3) Inside the method, the `pass` keyword is temporary used because Python expects something there

```
def __init__(self):
    pass
```

Create a class

- **Use of** `__init__` within the class
- For initialization of object
  (similar to a constructor in Java)
- **Example 1b**

```
def __init__(self):
    pass
```

- **Remark**
- Be careful of the indentation!

Instantiate an object

Instantiate an object

- **Instantiate an object**
- **Syntax**

1) Type the class name, followed by two brackets
2) Normally, assign this to a variable to keep track of the object

```
varname = Classname()
```

- **Example 1c**

```
golden = Dog()
```

Instantiate an object

- **Instantiate an object**
- **<u>Example 1d</u>**

```
golden = Dog()

print(golden) #print to view the info of object
```

Add attributes to class

Add attributes to class

- **Add attributes to class**
- **<u>Syntax</u>**

  Give attribute to class with its name and assignment

```
def __init__(self, attribute_name):
        self.attribute_name = attribute_name
```

Add attributes to class

- **Add attributes to class**
- **Example 1e**

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

- **Explanation**
  - `__init__` takes two arguments after `self`: `name` and `age`
  - Assign the the arguments to `self.name` and `self.age`

Add attributes to class

- **Create a new object with initialized attributes**
- **<u>Example 1f</u>**

```
golden = Dog("Golden", 4)
```

- **<u>Explanation</u>**
  - Create a new object, with its initialized attributes
    (i.e. create a new dog object with the dog's name and age)

Add attributes to class

- **Use of the attributes under an object**
- **<u>Example 1g</u>**

```
golden = Dog("Golden", 4)

print(golden.name)

print(golden.age)
```

- **<u>Explanation</u>**
  - Use the dot notation “.”
  - First typing the name of the object, followed by a dot and the attribute's name

Add attributes to class

- **Use of the attributes under an object**
- **Example 1h**

```
golden = Dog("Golden", 4)

print(golden.name + " is " + str(golden.age) + " year(s) old.")
```

- **Explanation**
  - Another example on using `object.attribute`

Define methods in a class

Define methods in a class

- **Define methods in a class**
- **Syntax**

1) Inside the class, give a method name and is defined with `def`
2) With argument
3) Inside the method, build some logics / return a value

```
def methodname(argument):
        # Some logics here
```

Define methods in a class

- **Define methods in a class**
- **Example 1i**

```
def bark(self):
    print("bark bark!")
```

- **Explanation**
  - A Dog class at before
  - Instance method named `bark` is created
  - Argument is the object itself (i.e. `self`)
  - Logic is to print a statement

Define methods in a class

- **Use of methods in an instantiated object**
- **Example 1j**

```
golden = Dog("Golden", 4)

golden.bark()
```

- **Explanation**
  - Use the dot notation "."
  - First typing the name of the object, followed by a dot and the method's name

Define methods in a class

- **Define another method in the class**
- **Example 1k**

```
def doginfo(self):
    print(self.name + " is " + str(self.age) + " year(s) old.")
```

- **Explanation**
  - Create a method that print the dog’s name and age

Define methods in a class

- **Use of methods in an instantiated object**
- **<u>Example 1l</u>**

```
golden = Dog("Golden", 4)
silver = Dog("Silver", 6)
smallq = Dog("Small Q", 8)

golden.doginfo()
silver.doginfo()
smallq.doginfo()
```

- **<u>Explanation</u>**
  - Use the dot notation "."
  - First typing the name of the object, followed by a dot and the method's name

Change the attribute value of an object

- **Change the attribute value of an object**
- **<u>Example 1m</u>**

```
golden.age = 5
```

- **<u>Explanation</u>**
  - Change the `age` of `golden` to 5

Change the attribute value of an object

- **Change the attribute value of an object**
- **<u>Example 1n</u>**

```
print(golden.age)
```

- **<u>Explanation</u>**
  - Print and verify the `age` of `golden` has been changed to 5

Change the attribute value of an object

- **Use a method to change the attribute value of an object**
- **Example 1o**

```
def birthday(self):
    self.age +=1
```

- **Explanation**
  - Change the `age` of the dog by passing the dog object to the `birthday` method

Change the attribute value of an object

- **Use a method to change the attribute value of an object**
- **Example 1p**

```
print(golden.age)    #Before Birthday - Age

golden.birthday()

print(golden.age)    #After Birthday - Age
```

- **Explanation**
  - Print and verify the `age` of the dog before and after passing the dog object to the `birthday` method

Passing arguments to methods

- **Passing arguments to methods**
- **Syntax**

1) Inside the class, give a method name and is defined with `def`
2) With argument (that can be more than one)
3) Inside the method, build some logics / return a value

```
def methodname(argument1, argument2, …):
        # Some logics here
```

Passing arguments to methods

- **Passing arguments to methods**
- **<u>Example 1q</u>**

```
def setBuddy(self, buddy):
    self.buddy = buddy
    buddy.buddy = self
```

- **<u>Explanation</u>**
  - Define a method that can assign the buddy to each other dogs

Passing arguments to methods

- **Passing arguments to methods**
- **Example 1r**

```
golden = Dog("Golden", 4)
silver = Dog("Silver", 6)

golden.setBuddy(silver)
```

- **Explanation**
  - Set golden and silver as buddy each other

Passing arguments to methods

- **Passing arguments to methods**
- **Example 1s**

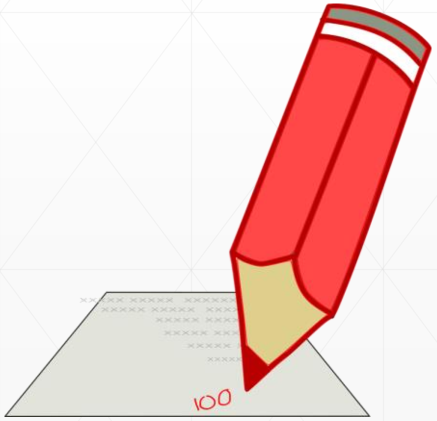
```
print(golden.buddy.name)
print(golden.buddy.age)

print(silver.buddy.name)
print(silver.buddy.age)

print(golden.buddy.doginfo())
```

- **Explanation**
  - Able to print the value of golden’s buddy (i.e. silver) name and age and also able to print the value of silver’s buddy (i.e. golden) name and age. *(And call the method doginfo() too.)*

More examples in OOP

More Examples in OOP

- Let’s try some more examples in OOP!



Four basic concepts of OOP

Four basic concepts of OOP

- Encapsulation
- Abstraction
- Inheritance
- Polymorphism

Four basic concepts of OOP

- To be continue…

Reference

- OOP Example https://www.datacamp.com/community/tutorials/python-oop-tutorial

Thank you