200 EXERCISES FOR OOP PYTHON

OBJECT ORIENTED PYTHON

FRANK ANEMAET

PythonOOPBook

Frank Anemaet

This book is for sale at http://leanpub.com/pythonoopbook

This version was published on 2020-10-30

Contents

Introduction **1**
List of instructions 1
Functions 1
OOP 1
Example exercises 1

Chapter One **2**
Class and objects 2
Objects 2
Class attributes 3
Object types 4
Class methods 4
Exercises 5

Chapter Two **9**
Classes In-Depth 9
Constructor 9
Destructor 9
Private variables 9
Exercises 9

Chapter Three **10**
Inheritance 10
Inherit methods 10
Multiple Inheritance 10
Override methods 10
Exercises 10

Chapter four **11**
Method overloading 11
Interface 11
Factory method 11
Exercises 11

Chapter five **12**

Object Serialization 12
Object serialization with JSON 12
Object serialization with YAML 12
Object serialization with Pickle 12
Exercises 12

Chapter six **13**
Counter OOP 13
Static method 13
Class method 13
Exercises 13

Answers **14**
Chapter 1 14
Chapter 2 14
Chapter 3 14
Chapter 4 14
Chapter 5 14
Chapter 6 14

Introduction

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

List of instructions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

OOP

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Example exercises

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Chapter One

Class and objects

In the real world you are surrounded by objects. Why not in programming?

Objects *can be represented digitally*. You can describe objects by their object variables. Objects have a "parent" which is a class. A class is used to create an object.

Objects

In your garage you may have several vehicles. So you might have a car and you might have a motorbike and perhaps a just a simple bike.

Your car, motorbike and bike have things in common: each has wheels, an engine, lights and other things in common. These commonalities are stores in a **class**.

```
# Define properties
class Vehicle:
    wheels = 2
    speed = 100

# Create objects
car = Vehicle()
motorbike = Vehicle()
bike = Vehicle()
```

You can change the values of each object. You can set the maximum speed of one or more objects to 100.

Each object created here, is of type "Vehicle", meaning they are created from the class Vehicle.

An object is always created from one *and only one* class. But it doesn't have to be the same class.

For instance, a hovercraft doesn't have wheels, so it doesn't make sense to create it from this class Vehicle.

In this example, all objects are of type vehicle. All of them have the variables defined in class *Vehicle*: the variables wheels and maximum speeds.

You can output the variables of an object. The *values* are unique to the object:

```
print(car.wheels)
```

You see a car with two wheels. This applies for every object (car, motorbike, bike). You *can set the variable for every object*. Every object has two wheels. And for a car less probably more wheels so you can change that.

```
car.wheels = 4
```

So now we have four wheels for the car but two wheels for the other vehicles.We just print the car wheels only.

```
car.wheels = 4
print(car.wheels)
```

You can set the other variables. So if you see a car at maximum speed it will tell you that this object's maximum speed is one hundred.

So in this case we made three objects and we set some variables for those objects that we can then use.

Class attributes

You can represent objects as virtual objects. To do so, every object needs variables that describe it. Let's say you have some pets. Each pet has a name, an age and other properties. They can be of different types (classes).

```
class Dog:
    name = ""
    age = 0
    commands = []

class Cat:
    name = ""
    age = 0

pet1 = Cat()
pet1.name = "Bob"
pet1.age = 3

pet2 = Dog()
pet2.name = "Dave"
pet2.age = 5
```

In this example there are two objects: pet1 and pet2. Class Dog is used to create pet2, class Cat is used to create pet1.

Each object has the variables of their class. For pet2 that's three variables, for pet1 that's two variables.

So classes can be used to represent an abstract model of the objects. Each time an object is created, a class is used. In the example aboe we create two objects (pet1,pet2) using different classes (class Cat, class Dog).

Object types

If you have different types of classes, you can create different types of objects. (Virtual) objects may have unique properties. Car have a certain number of wheels, but if you have pets pets: no wheels.

You can describe objects in an abstract way: a **class**. These classes are used to create virtual objects.

Each virtual object can have properties (variables) and methods.

For instance, you can have these clases.

```
class Car:
    wheels = 4

class Pet:
    legs = 4
```

Then you can create objects, which will have unique variables.

```
obj1 = Car()
obj2 = Pet()
obj3 = Pet()
obj4 = Car()
obj5 = Car()
```

So there can be different types of objects. The object type is defined upon creation, using a class.

Class methods

In this world you are surrounded by objects. For example, different types of birds: parrots, pigeon, pinguin and eagle.

These have in common that they have certain features. Some can fly, different colors etc. You can have different objects.

```
# Define abstract object
class Bird:
    def fly():
        print('flying')

# Create objects
parrot = Bird()
pigeon = Bird()
eagle = Bird()

# Make objects do something
parrot.fly()
pigeon.fly()
```

Each object can have methods. In this code, the objects have a method fly. You can call objects methods as many times as you want.
You can have methods that change the objects variable(s).

```
class Bird:
    name = ""

    def setName(self, name):
        self.name = name

    def fly():
        print('flying')

eagle = Bird()
eagle.setName("Bob")
```

To do that, you need to specify self for methods inside a class. That is to say, the object changes their own variables.

Exercises

1. How is object orientated programming (OOP) different from procedural programming?
2. What is the difference between a Python object and a real world object?
3. Are all Python programs object oriented?
4. Write the code: Create an object Duck from class Animal
5. True or False: A class is created using an object
6. True or False: Object variables are the same for all objects
7. True or False: The program below is incorrect

```
obj1.score = 2
obj2.score = 5
```

8. True or False: Python supports only Object Orientated programming
9. True or False: In Python you can use procedural and object orientated programming at the same time

10. What is the right way to create an object?

```
A: person = Student
B: person = Student()
C: person = new Student
D: person = new Student()
```

11. What is the purpose of a class?
12. Can an object be created without a class?
13. How many objects can be created?
14. How many classes can be created?
15. What is the right way to set an objects variable?

```
A: person->name = "Alice"
B: person..name = "Alice"
C: person.name = "Alice"
D: person.name() = "Alice"
```

16. What is the use of a constructor?
17. What is the user of a destructor?
18. What is the correct way to define a constructor?

```
A: def const():
B: def constructor(self):
C: def init(self):
D: def __init__(self):
```

19. What is the use of the **self** keyword?
20. Finish the code, create three objects using class Student

```python
class Student:
    def __init__(self, id, name):
        self.id = id
        self.name = name

# Create 3 students
```

21. Finish the code below

```python
# define class
# your code here

# create objects
cofee = Drink()
tea = Drink()
soda = Drink()
```

22. In soccer, which classes should exist? And which objects should exist?
23. Define the classes and objects required for the Tennis game
24. Define all the chess pieces, as if they were Python objects
25. Which classes are required for the game of chess?
26. Create a class PhoneBook with the functions insert(), delete() and lookup()
27. Create a class AddressBook with the functions add() and lookup()
28. Now implement the code for exercise 24
29. Create a class Magazine with title and pages. Then create two objects (magazines) with a different title and page content
30. Finish the code below

```python
class PasswordManager:
    def __init__(self):
        pass

pm = PasswordManager()
pm.add("twitter.com","frank","password")
````

31. Finish the code

```python
class Window:
    def __init__(self, title):
```

```
        self.title = title

# Create objects
chrome = Window("Chrome Browser")
notepad = ...
word = ...
```

32. Finish the code, write the class

```
# define class
class Car:
    pass


obj = Car()
obj.topspeed = 130
obj.wheels = 4
obj.accelerate()
obj.brake()
obj.turnLeft()
obj.turnRight()
```

33. What's wrong with the code below?

```
class Notepad:
   def new():
        self.document = ""


window = Notepad()
window.new()
```

Chapter Two

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Classes In-Depth

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Constructor

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Destructor

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Private variables

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Chapter Three

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Inheritance

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Inherit methods

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Multiple Inheritance

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Override methods

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Chapter four

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Method overloading

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Interface

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Factory method

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Chapter five

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Object Serialization

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Object serialization with JSON

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Object serialization with YAML

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Object serialization with Pickle

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Chapter six

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Counter OOP

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Static method

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Class method

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Exercises

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Answers

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Chapter 1

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Chapter 2

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Chapter 3

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Chapter 4

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Chapter 5

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.

Chapter 6

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/pythonoopbook.