



## OOP - Part 1

### Essentials of Object Oriented Programming

Michael Lu [mlu@student.42.fr](mailto:mlu@student.42.fr)

*Summary: This project will help you learn the essentials of objective oriented programming.*



*This work is licensed under a [Creative Commons Attribution-Noncommercial-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-nc-sa/4.0/).*

# Contents

<b>I</b>	<b>Foreword</b>	<b>2</b>
<b>II</b>	<b>Goals</b>	<b>3</b>
<b>III</b>	<b>General instructions</b>	<b>4</b>
<b>IV</b>	<b>Exercise 00</b>	<b>5</b>
<b>V</b>	<b>Exercise 01</b>	<b>6</b>
<b>VI</b>	<b>Exercise 02</b>	<b>7</b>
<b>VII</b>	<b>Exercise 03</b>	<b>8</b>
<b>VIII</b>	<b>Exercise 04</b>	<b>9</b>
<b>IX</b>	<b>Exercise 05</b>	<b>10</b>

# Chapter I

## Foreword

Did you know I love cooking and cooking is a great way to learn stuff?

Cooking teaches you a lot of skills that is beneficial to you. Regardless if you prefer your mom's home cooking or maybe your dad's barbecue, but have you ever tried following a recipe and learning it yourself?

If you ever start learning how to cook you will soon realize you need a couple of things first. You need measuring tools, some kind of hot plate, a way to cut or prep ingredients. Each of these objects are their own entity but they work together to provide you a delicious meal.

Object oriented programming is very similar to this concept. Hah! Bet you thought I wouldn't bring up programming eh? Just like cooking, you will be creating objects in objected oriented programming (I wonder why it's called that), and learning how to utilize them to help you create some cool stuff.



# Chapter II

## Goals

The goal of this project is to introduce you to the basics object oriented programming. By the end of this project you should know how to:

- Create classes
- Initiate an instance of a class
- Assign class methods and attributes
- Use variables specific to each instance
- Design classes that inherit from each other.

You will be exploring a fundamental topic of object oriented programming so take advantage of all the resources including articles, videos, your neighbor, StackOverflow and so forth. There are many tutorials on classes and inheritance.


# Chapter III

## General instructions

- This project will only be corrected by actual human beings. You are therefore free to organize and name your files as you wish, although you must respect some requirements listed below.
- You must follow the exercise details and instruction clearly
- You must turn in all the requested files
- Ask your peers, mentor, slack or anywhere else if you need any help, and make sure to have fun!

# Chapter IV

## Exercise 00

	ex00: Your first class
Topics to study : User-defined classes, importing from a file	
Files to turn in : main.py, first_class.py	
Notes : n/a	

Write a class called FirstClass in a file named first\_class.py. The constructor does not need to set any variables, but it should print "Hello World" to the terminal when an instance of the class is created. Test it by creating an instance of the class inside the file main.py.


```
?> python3 first_class.py
Hello World
?>
```



```
from first_class import FirstClass
```

# Chapter V

## Exercise 01

	ex01: Your second class and first inheritance
Topics to study : Inheritance	
Files to turn in : main.py, first_class.py, second_class.py	
Notes :	

Make your second class (a class named SecondClass in a file named second\_class) that will inherit from the first class. Its constructor should directly call the first class constructor that says "Hello World". Create an instance of the second class in your file main.py.


```
?> python3 main.py
Hello World
?>
```



Put a copy of the first\_class.py file in the same "ex01" folder as second\_class.py and main.py.

# Chapter VI

## Exercise 02

	ex02 : Your first parameter and passing parameter
Topics to study : <code>super</code>	
Files to turn in : <code>main.py</code> , <code>first_class.py</code> , <code>second_class.py</code>	
Notes :	


Edit a copy of your second class so that its init function takes in a parameter "name" (which will be your login name) and passes it into the first class which will display "Hello, <intraname>". You can hard-code your login name into the program without calling `input()` to fetch it. Create an instance of `second_class` in your name to see it work.

```
?> python3 main.py
Hello, mlu
?>
```



# Chapter VII

## Exercise 03


	ex03 : Your first method
Topics to study : Class methods	
Files to turn in : main.py, first_class.py, second_class.py	
Notes :	

Inside a copy of your first class, create a method called `say_hello`. `Say_hello` should take the "name" parameter from the constructor and print out "Hello, <intraname>". When the method is called print out a sentence stating that it has been called. Demonstrate how it works by creating an instance of `second_class` in your main.

```
?> python3 main.py
Method say_hello in FirstClass was called!
Hello, mlu
?>
```

# Chapter VIII

## Exercise 04

	ex04 : Your second method
Topics to study : <code>super</code>	
Files to turn in : <code>main.py</code> , <code>first_class.py</code> , <code>second_class.py</code>	
Notes :	

You now need to create a method inside your second class called `roll_dice` which will randomly generate a number from 1 to 6. `Roll_dice` will then call the `Hello` method in its parent class and pass it the random number. You should see an output of "Hello <username>, your number is <number>". To keep track of what's going on, every method you create should print something to announce it has been called. You must instantiate the second class only in your main.

```
?> python3 main.py
Method roll_dice in SecondClass called
Method hello in FirstClass is called
Hello mlu, your number is 2
?>
```

# Chapter IX

## Exercise 05



ex05 : Your first class variables, and more methods!

Topics to study : Return values, Getters and setters

Files to turn in : main.py, first\_class.py, second\_class.py

Notes :

Your second class will now take in a second parameter, a string called "hobby", when it is instantiated. Store the hobby in an instance variable. You will write a method called `get_hobby` in your second class that returns this variable. In your main, call the `get_hobby` function on your instance of the second class, and print it the return value in the statement "Your hobby is <hobby>."

```
?> python3 main.py
Method roll_dice in SecondClass called
Method hello in FirstClass is called
Hello mlu, your number is 5
Your hobby is being lazy.
?>
```



Read about the difference between class variables and instance variables. What is an instance of a class?