



OOP Introduction

Basic Concepts of Object-Oriented-Programming

Wesley To wto@student.42.fr
42 Staff pedago@42.fr

Summary: These exercises will introduce you to the basic concepts of object-oriented-programming.

Contents

I	Foreword	2
II	Introduction	3
III	Goals	4
IV	General instructions	5
V	Exercise 00 : Your First Class	6
VI	Exercise 01 : Methods	7
VII	Exercise 02 : Inheritance & Polymorphism	8
VIII	Exercise 03 : Parameters	9
IX	Exercise 04 : Instance Variables	10
X	Exercise 05 : Class Variables	11
XI	Exercise 06 : Putting It All Together	12
XII	Exercise 07 : One More...	13
XIII	Bonus: Go Your Own Way...	14
XIV	Turn-in and peer-evaluation	15

Chapter I

Foreword

Here's 12 of the 29 "Dad Jokes That Are So Bad, They're Actually Good," according to BuzzFeed:

- What time did the man go to the dentist? Tooth hurt-y.
- Did you hear about the guy who invented Lifesavers? They say he made a mint.
- A ham sandwich walks into a bar and orders a beer. Bartender says, "Sorry, we don't serve food here."
- Whenever the cashier at the grocery store asks my dad if he would like the milk in a bag, he replies, "No, just leave it in the carton."
- Why do chicken coops only have two doors? Because if they had four, they would be chicken sedans!
- Me: "Dad, make me a sandwich!" Dad: "Poof! You're a sandwich!"
- Why did the clydesdale give the pony a glass of water? Because he was a little horse!
- I used to have a job at a calendar factory, but I got the sack because I took a couple of days off.
- How do you make a Kleenex dance? Put a little boogie in it.
- Two guys walk into a bar. The third one ducks.
- I had a dream that I was a muffler last night. I woke up exhausted.
- 5/4 people admit that they're bad with fractions.

See the other 17: <https://goo.gl/Da2sT4>

Chapter II

Introduction

The goal of this project is to complete a sequence of exercises which will teach you the basics of object oriented programming.

This course uses Python 3 as the approved language. Make sure you are using the correct version to prevent execution and grading errors.

So, what are you waiting for? Get going!



You can verify your Python version by running the commands below. At least one of them should work. If it turns out you don't have Python 3, you can run the `setup.sh` script provided to you. If you are having trouble, ask your neighbor or mentor. The script may take several minutes to finish.

```
?> python --version  
Python 3.6.5
```

```
?> which python3  
/nfs/2018/y/yourintra/.pyenv/shims/python3
```

Chapter III

Goals

The goal of this OOP introduction is to introduce you to the primary concepts of object oriented programming. By the end of this project, you should know how to:

- Create classes
- Create methods
- Create inheritances
- Create variables
- Be awesome

You will be exploring the fundamental topics of object-oriented-programming, so take advantage of all available resources, including your neighbor, and Google. There are also many tutorials on classes and inheritance.



The more specific you are with Google, the better. In addition to the topic you are looking up, include as many relevant technical terms as you can. For example, "python 2D list comprehension with default value" is better than "python list"

Chapter IV

General instructions

- This project will only be corrected by actual human beings.
- You are free to organize and name your files as you wish, although to use the provided filechecker, you must strictly follow the provided instructions for naming and organization.
- You must follow the exercise details and instruction clearly
- You must turn in all the requested files
- Your project must be written in a language approved by the Hack High School program (Python 3)
- Ask your peers, mentor, Slack/Discord, or anywhere else if you need help, and make sure to have fun
- If an exercise asks for a `main.py` file, you must follow this template:


```
#put your imports here

def main():
    #put your code here

if __name__ == "__main__":
    main()
```

Chapter V

Exercise 00 : Your First Class


	Exercise 00
Your First Class	
Turn-in directory : <i>ex00/</i>	
Files to turn in : first_class.py	
Forbidden functions : None	
Notes : n/a	

In the `first_class` file, make your first class (called `FirstClass`) with a constructor that says "First!". Your class will be instantiated in the provided `main` file.

```
?> python main.py
First!
```

Chapter VI

Exercise 01 : Methods

	Exercise 01
Methods	
Turn-in directory : <i>ex01/</i>	
Files to turn in : <code>cellphone.py</code> , <code>main.py</code>	
Forbidden functions : <code>None</code>	
Notes : <code>n/a</code>	

OOP got you down? Wish that you could call out for help? Now you can (kinda)! We'll be making a cellphone now!

Create a class `Cellphone`, and give it three methods: `call()`, `text()`, and `drop()`. Add print statements in each method to produce the output seen below. Then write a `main` that creates a cellphone and calls each method, producing output like this:


```
?> python main.py
Making A Call...
Sending A Text...
CRASH! CRACK! ...
```



Examine the provided `main.py` file from the previous exercise to see how yours should be structured

Chapter VII

Exercise 02 : Inheritance & Polymorphism

	Exercise 02
Inheritance & Polymorphism	
Turn-in directory : <i>ex02/</i>	
Files to turn in : <i>animal.py, main.py, dog.py, cat.py</i>	
Forbidden functions : None	
Notes : n/a	

Let's make some animals!

Create a class `Animal`. Give the class two methods: `speak()` and `sleep()`, and have `sleep()` output "Zzzz...". Have `speak()` do nothing.

Now create 2 subclasses, `Dog` and `Cat`, and override the `speak()` method in each one to match the example output. In the constructor for each class, output a message letting us know what is being created.

Finally, write a `main` that reproduces the output below. It should create all animal types (including `Animal`), and call both `speak()` and `sleep()` for each one.


```
?> python main.py
Making An Animal
Making A Dog
Making A Cat
Woof
Meow
Zzzz...
Zzzz...
Zzzz...
```



"Method overriding" is a core component of polymorphism. Google it!

Chapter VIII

Exercise 03 : Parameters

	Exercise 03
Parameters	
Turn-in directory : <i>ex03/</i>	
Files to turn in : <code>greeter.py</code> , <code>insult_comic.py</code> , <code>main.py</code>	
Forbidden functions : None	
Notes : n/a	


Up until now, all we've done is make our objects give us output. Let's give them some input now.

Create a class **Greeter** and give it a method **speak()** that takes in a name, and prints out "Hello \$name". Now do the same thing, but for an **InsultComic**. Match the example output for **speak()**. As always, make a **main**, create your objects, and call **speak()** for each one, passing in "Wesley" as the argument. It should reproduce the output below.

```
?> python main.py
Hello Wesley
I'd say it's good to see you Wesley, but I'd be lying...
```

Chapter IX

Exercise 04 : Instance Variables

	Exercise 04
Instance Variables	
Turn-in directory : <i>ex04/</i>	
Files to turn in : <i>building.py, campus.py, main.py</i>	
Forbidden functions : <i>None</i>	
Notes : <i>n/a</i>	

We've covered inheritance, we've covered input and output, and we've covered behavior. Now let's add the final ingredient to an object: data.

Create a class **Building** and have its constructor take in two parameters **name** and **capacity**. Store the two parameters as instance variables. Create a method **get_info()** that outputs the name of the building and its capacity (see example output below).


Next, create a class **Campus** with three instance variables, **buildings**, **num_buildings**, and **capacity**, that store a list of the buildings, the number of buildings, and the total capacity, respectively. Give this class two instance methods **get_info()** and **add_building()**. **add_building()** takes in a building object, adds the building to the list of buildings, increments the number of buildings, and adds the capacity of the building to the capacity of the campus. **get_info()** outputs the status of the campus.

In your **main**, create two buildings ("Math Building" and "Science Building") that hold 25 and 17 people, respectively. Then create a campus, call **get_info()** for both buildings, add both buildings to the campus, and call **get_info()** for the campus. Your main should reproduce the output below.

```
?> python main.py
Building "Math Building" can hold 25 people
Building "Science Building" can hold 17 people
The campus has 2 building(s) with a combined capacity of 42 people
```

Chapter X

Exercise 05 : Class Variables

	Exercise 05
Class Variables	
Turn-in directory : <i>ex05/</i>	
Files to turn in : <i>player.py</i> , <i>*.py</i> , <i>main.py</i>	
Forbidden functions : <i>None</i>	
Notes : <i>n/a</i>	

Pick your favorite team sport (or the one you know the most about). Create a class `Player` and give it a class variable `playbook`. Initialize the `playbook` with at least 2 strings representing the names of plays into this `playbook`.


Create classes for the different player positions that each inherit from `Player`, and give them a `run_play()` method which takes in the name of a play. Have the `run_play()` method print out something appropriate for each play (including if they don't recognize the play).

As always, write a `main` to demonstrate your classes and their methods. Your `main` should produce output similar to the one below.

```
?> python main.py
Point Guard: I'll dribble penetrate
Shooting Guard: I'll cut to the corner
Center: I'll set the first pick
Small Forward: I'll run to the wing
Power Forward: I'll set the second pick
Point Guard: I'll dribble past the screens to the post
Shooting Guard: I'll cut to the top of the key
Center: I'll set the first pick and roll to the basket
Small Forward: I'll maneuver to the low block and set a screen
Power Forward: I'll set the second pick and screen the shooting guard
Point Guard: I don't know that play...
Shooting Guard: I don't know that play...
Center: I don't know that play...
Small Forward: I don't know that play...
Power Forward: I don't know that play...
```

Chapter XI

Exercise 06 : Putting It All Together

	Exercise 06
Putting It All Together	
Turn-in directory : <i>ex06/</i>	
Files to turn in : <code>classroom.py</code> , <code>person.py</code> , <code>student.py</code> , <code>teacher.py</code> , <code>main.py</code>	
Forbidden functions : None	
Notes : n/a	

Let's get meta! We're going to create a classroom!


Create two classes `Classroom` and `Person`. Create two subclasses of `Person` called `Teacher` and `Student`. Give `Classroom` a class variable called `capacity` and initialize it to 20.

Using your knowledge from the previous exercises, as well as the provided `main` file as a guide, give the classes the appropriate methods and variables to produce the same output as below.

```
?> python main.py
This classroom has a standard capacity of 20 and is run by Wesley. It currently has 3 students
This classroom has a standard capacity of 20 and is run by Anand. It currently has 2 students
Wesley: Welcome to class, Alice
Alice: Good morning, Wesley
Wesley: Welcome to class, Carol
Carol: Good morning, Wesley
Wesley: Welcome to class, Eve
Eve: Good morning, Wesley
Anand: Welcome to class, Bob
Bob: Good morning, Anand
Anand: Welcome to class, Dave
Dave: Good morning, Anand
```

Chapter XII

Exercise 07 : One More...

	Exercise 07
One More...	
Turn-in directory : <i>ex07/</i>	
Files to turn in : <i>dad.py, joke.py, main.py</i>	
Forbidden functions : None	
Notes : n/a	

Create two classes Dad and Joke. Give Dad a `speak()` instance method, and put this in the method body:

```
arr = [87, 104, 97, 116, 32, 105, 115, 32, 116, 104, 101, 32, 111,
       98, 106, 101, 99, 116, 45, 111, 114, 105, 101, 110, 116, 101,
       100, 32, 119, 97, 121, 32, 111, 102, 32, 98, 101, 99, 111, 109,
       105, 110, 103, 32, 119, 101, 97, 108, 116, 104, 121, 63]
print(''.join([chr(n) for n in arr]))
```

Give Joke a `reply()` instance method, and put this in the method body:

```
arr = [73, 110, 104, 101, 114, 105, 116, 97, 110, 99, 101]
print(''.join([chr(n) for n in arr]))
```

Write a main where the dad speaks and the joke replies. Run the main, and enjoy!

Chapter XIII

Bonus: Go Your Own Way...

Congrats! You've completed all the exercises. You are now free to explore OOP and work on whatever you want. Create anything involving the principles of OOP (either the ones we covered or ones you research on your own). Here are a few topics we did not cover:

- Abstract classes
- Abstract methods
- Class methods
- "Gang of Four" design patterns
- SOLID
- Multiple inheritance

Here are a few possible ideas of things to create:

- A text-based board game
- A messaging application
- A chat bot
- A sudoku game

Chapter XIV

Turn-in and peer-evaluation

Turn your work in using your GiT repository. Only work present on your repository will be graded in defense. Good luck and remember to have fun!