(/)

0x06. Python - Classes and Objects

Python OOP

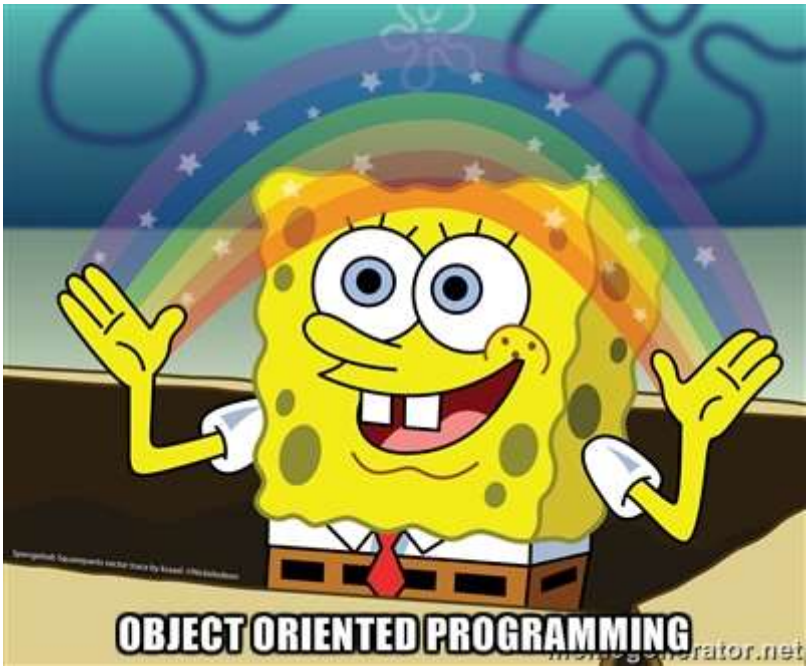
By Guillaume

Weight: 1

Ongoing project - started Jul 19, 2022, must end by Jul 20, 2022 - you're done with 0% of tasks.

Checker was released at Jul 19, 2022 12:00 PM

An auto review will be launched at the deadline



Background Context

OOP is a totally new concept for all of you (especially those who think they know about it :)). It's VERY important that you read at least all the material that is listed bellow (and skip what we recommend you to skip, you will see them later in the curriculum).

As usual, make sure you type (never copy and paste), test, understand all examples shown in the following links (including those in the video), test again etc. The biggest and most important takeaway of this project is: experiment by yourself OOP, play with it!

Read or watch the below resources in the order presented.

Resources

Read or watch:

- Object Oriented Programming (/rltoken/i49z6HxrBGRNnixo7ZWbEQ) (*Read everything until the paragraph "Inheritance" excluded. You do NOT have to learn about class attributes, `classmethod` and `staticmethod` yet*)
- Object-Oriented Programming (/rltoken/qz3KSn154ia4H2DPaabOzg) (*Please *be careful**: *in most of the following paragraphs, the author shows things the way you should not use or write a class in order to help you better understand some concepts and how everything works in Python 3. Make sure you read everything in the following paragraphs: General Introduction, First-class Everything, A Minimal Class in Python, Attributes (You DON'T have to learn about class attributes), Methods, The `__init__` Method, "Data Abstraction, Data Encapsulation, and Information Hiding," "Public, Protected, and Private Attributes"*)
- Properties vs. Getters and Setters (/rltoken/Wy2djWXK5b4rnnYIAq_wIA)
- Learn to Program 9 : Object Oriented Programming (/rltoken/MxIOanLf5vG5QeCWek2nqQ)
- Python Classes and Objects (/rltoken/AoLH4xp5StrQST-Cu0Fg8w)
- Object Oriented Programming (/rltoken/-vVnWzwR3a3X0H8Oia78Ug)

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone (/rltoken/WxAcwZ7gFDS8MKYqhB9Gzw), **without the help of Google**:

General

- Why Python programming is awesome
- What is OOP
- "first-class everything"
- What is a class
- What is an object and an instance
- What is the difference between a class and an object or instance
- What is an attribute
- What are and how to use public, protected and private attributes
- What is `self`
- What is a method
- What is the special `__init__` method and how to use it
- What is Data Abstraction, Data Encapsulation, and Information Hiding
- What is a property
- What is the difference between an attribute and a property in Python
- What is the Pythonic way to write getters and setters in Python
- How to dynamically create arbitrary new attributes for existing instances of a class
- How to bind attributes to object and classes
- What is the `__dict__` of a class and/or instance of a class and what does it contain
- How does Python find the attributes of an object or class
- How to use the `getattr` function

Copyright - Plagiarism

- You are tasked to come up with solutions for the tasks below yourself to meet with the above learning objectives.
- You will not be able to meet the objectives of this or any following project by copying and pasting someone else's work.
- You are not allowed to publish any content of this project.
- Any form of plagiarism is strictly forbidden and will result in removal from the program.

Requirements

General

- Allowed editors: `vi`, `vim`, `emacs`
- All your files will be interpreted/compiled on Ubuntu 20.04 LTS using python3 (version 3.8.5)
- All your files should end with a new line
- The first line of all your files should be exactly `#!/usr/bin/python3`
- A `README.md` file, at the root of the folder of the project, is mandatory
- Your code should use the pycodestyle (version `2.8.*`)
- All your files must be executable
- The length of your files will be tested using `wc`
- All your modules should have a documentation (`python3 -c 'print(__import__("my_module").__doc__)'`)
- All your classes should have a documentation (`python3 -c 'print(__import__("my_module").MyClass.__doc__)'`)
- All your functions (inside and outside a class) should have a documentation (`python3 -c 'print(__import__("my_module").my_function.__doc__)'` and `python3 -c 'print(__import__("my_module").MyClass.my_function.__doc__)'`)
- A documentation is not a simple word, it's a real sentence explaining what's the purpose of the module, class or method (the length of it will be verified)

More Info

Documentation is now mandatory! Each module, class, and method must contain docstring as comments. Example Google Style Python Docstrings (/rltoken/dOO785g5EQYkRU2E1wri0g)

Quiz questions

Great! You've completed the quiz successfully! Keep going! (Hide quiz)

Question #0

In this following code, what is `User` ?

```
class User:
    id = 89
    name = "no name"
    __password = None

    def __init__(self, new_name=None):
        self.is_new = True
        if new_name is not None:
            self.name = new_name
```

- [x] A class
- [ ] A string
- [ ] An attribute
- [ ] A method
- [ ] An instance

Question #1

In this following code, what is `id` ?

```
class User:
    id = 89
    name = "no name"
    __password = None

    def __init__(self, new_name=None):
        self.is_new = True
        if new_name is not None:
            self.name = new_name
```

- [ ] A public instance attribute
- [x] A public class attribute
- [ ] A public class method
- [ ] A public instance method
- [ ] A private class attribute
- [ ] A protected class attribute

Question #2

In this following code, what is `__password` ?

```
class User:
    id = 89
    name = "no name"
    __password = None

    def __init__(self, new_name=None):
        self.is_new = True
        if new_name is not None:
            self.name = new_name
```

- [ ] A public class attribute
- [ ] A public instance attribute
- [ ] A protected class attribute
- [ ] A protected instance attribute
- [x] A private class attribute
- [ ] A private instance attribute

Question #3

In this following code, what is `is_new` ?

```
class User:
    id = 89
    name = "no name"
    __password = None

    def __init__(self, new_name=None):
        self.is_new = True
        if new_name is not None:
            self.name = new_name
```

- [ ] A public class attribute
- [x] A public instance attribute
- [ ] A protected class attribute
- [ ] A protected instance attribute
- [ ] A private class attribute
- [ ] A private instance attribute

Question #4

What do these lines print?

```
>>> class User:
>>>     id = 89
>>>     name = "no name"
>>>     __password = None
>>>
>>>     def __init__(self, new_name=None):
>>>         self.is_new = True
>>>         if new_name is not None:
>>>             self.name = new_name
>>>
>>> u = User()
>>> u.is_new
```

- [ ] is_new
- [ ] Nothing
- [ ] False
- [x] True

Question #5

What do these lines print?

```
>>> class User:
>>>     id = 89
>>>     name = "no name"
>>>     __password = None
>>>
>>>     def __init__(self, new_name=None):
>>>         self.is_new = True
>>>         if new_name is not None:
>>>             self.name = new_name
>>>
>>> u = User()
>>> u.id
```

- [x] 89
- [ ] id
- [ ] User.id
- [ ] Nothing

Question #6

What do these lines print?

```
>>> class User:
>>>     id = 89
>>>     name = "no name"
>>>     __password = None
>>>
>>>     def __init__(self, new_name=None):
>>>         self.is_new = True
>>>         if new_name is not None:
>>>             self.name = new_name
>>>
>>> u = User("John")
>>> u.name
```

- [ ] name
- [ ] None
- [x] John
- [ ] no name

Question #7

What do these lines print?

```
>>> class User:
>>>     id = 89
>>>     name = "no name"
>>>     __password = None
>>>
>>>     def __init__(self, new_name=None):
>>>         self.is_new = True
>>>         if new_name is not None:
>>>             self.name = new_name
>>>
>>> u = User()
>>> u.name
```

- [ ] name
- [ ] None
- [ ] John
- [x] no name

Tasks

0. My first square

mandatory

(/)

Write an empty class `Square` that defines a square:

- You are not allowed to import any module

```
guillaume@ubuntu:~/0x06$ cat 0-main.py
#!/usr/bin/python3
Square = __import__('0-square').Square

my_square = Square()
print(type(my_square))
print(my_square.__dict__)

guillaume@ubuntu:~/0x06$ ./0-main.py
<class '0-square.Square'>
{}
guillaume@ubuntu:~/0x06$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x06-python-classes`
- File: `0-square.py`

Done? | Help | Check your code | >_ Get a sandbox

1. Square with size

mandatory

Write a class `Square` that defines a square by: (based on `0-square.py`)

- Private instance attribute: `size`
- Instantiation with `size` (no type/value verification)
- You are not allowed to import any module

Why?

Why `size` is private attribute?

The size of a square is crucial for a square, many things depend of it (area computation, etc.), so you, as class builder, must control the type and value of this attribute. One way to have the control is to keep it privately. You will see in next tasks how to get, update and validate the size value.

```
guillaume@ubuntu:~/0x06$ cat 1-main.py
#!/usr/bin/python3
Square = __import__('1-square').Square

my_square = Square(3)
print(type(my_square))
print(my_square.__dict__)

try:
    print(my_square.size)
except Exception as e:
    print(e)

try:
    print(my_square.__size)
except Exception as e:
    print(e)

guillaume@ubuntu:~/0x06$ ./1-main.py
<class '1-square.Square'>
{'_Square__size': 3}
'Square' object has no attribute 'size'
'Square' object has no attribute '__size'
guillaume@ubuntu:~/0x06$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x06-python-classes`
- File: `1-square.py`

☐ Done? | Help | Check your code | >_ Get a sandbox

2. Size validation

mandatory

Write a class `Square` that defines a square by: (based on `1-square.py`)

- Private instance attribute: `size`
- Instantiation with optional `size`: `def __init__(self, size=0):`
  - `size` must be an integer, otherwise raise a `TypeError` exception with the message `size must be an integer`
  - if `size` is less than `0`, raise a `ValueError` exception with the message `size must be >= 0`
- You are not allowed to import any module

7)

```
guillaume@ubuntu:~/0x06$ cat 2-main.py
#!/usr/bin/python3
Square = __import__('2-square').Square

my_square_1 = Square(3)
print(type(my_square_1))
print(my_square_1.__dict__)

my_square_2 = Square()
print(type(my_square_2))
print(my_square_2.__dict__)

try:
    print(my_square_1.size)
except Exception as e:
    print(e)

try:
    print(my_square_1.__size)
except Exception as e:
    print(e)

try:
    my_square_3 = Square("3")
    print(type(my_square_3))
    print(my_square_3.__dict__)
except Exception as e:
    print(e)

try:
    my_square_4 = Square(-89)
    print(type(my_square_4))
    print(my_square_4.__dict__)
except Exception as e:
    print(e)

guillaume@ubuntu:~/0x06$ ./2-main.py
<class '2-square.Square'>
{'_Square__size': 3}
<class '2-square.Square'>
{'_Square__size': 0}
'Square' object has no attribute 'size'
'Square' object has no attribute '__size'
size must be an integer
size must be >= 0
guillaume@ubuntu:~/0x06$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x06-python-classes`

- File: `2-square.py`

(/)

☐ Done? | Help | Check your code | >_ Get a sandbox

3. Area of a square

mandatory

Write a class `Square` that defines a square by: (based on `2-square.py`)

- Private instance attribute: `size`
- Instantiation with optional `size`: `def __init__(self, size=0):`
  - `size` must be an integer, otherwise raise a `TypeError` exception with the message `size must be an integer`
  - if `size` is less than `0`, raise a `ValueError` exception with the message `size must be >= 0`
- Public instance method: `def area(self):` that returns the current square area
- You are not allowed to import any module

```
guillaume@ubuntu:~/0x06$ cat 3-main.py
#!/usr/bin/python3
Square = __import__('3-square').Square

my_square_1 = Square(3)
print("Area: {}".format(my_square_1.area()))

try:
    print(my_square_1.size)
except Exception as e:
    print(e)

try:
    print(my_square_1.__size)
except Exception as e:
    print(e)

my_square_2 = Square(5)
print("Area: {}".format(my_square_2.area()))

guillaume@ubuntu:~/0x06$ ./3-main.py
Area: 9
'Square' object has no attribute 'size'
'Square' object has no attribute '__size'
Area: 25
guillaume@ubuntu:~/0x06$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x06-python-classes`

- File: `3-square.py`

(/)

☐ Done? | Help | Check your code | >_ Get a sandbox

4. Access and update private attribute

mandatory

Write a class `Square` that defines a square by: (based on `3-square.py`)

- Private instance attribute: `size` :
  - property `def size(self):` to retrieve it
  - property setter `def size(self, value):` to set it:
    - `size` must be an integer, otherwise raise a `TypeError` exception with the message `size must be an integer`
    - if `size` is less than `0` , raise a `ValueError` exception with the message `size must be >= 0`
- Instantiation with optional `size` : `def __init__(self, size=0):`
- Public instance method: `def area(self):` that returns the current square area
- You are not allowed to import any module

Why?

Why a getter and setter?

Reminder: `size` is a private attribute. We did that to make sure we control the type and value. Getter and setter methods are not 100% Python, but more OOP. With them, you will be able to validate the assignment of a private attribute and also define how getting the attribute value will be available from outside - by copy? by assignment? etc. Also, adding type/value validation in the setter will centralize the logic, since you will do it in only one place.

```
guillaume@ubuntu:~/0x06$ cat 4-main.py
#!/usr/bin/python3
Square = __import__('4-square').Square

my_square = Square(89)
print("Area: {} for size: {}".format(my_square.area(), my_square.size))

my_square.size = 3
print("Area: {} for size: {}".format(my_square.area(), my_square.size))

try:
    my_square.size = "5 feet"
    print("Area: {} for size: {}".format(my_square.area(), my_square.size))
except Exception as e:
    print(e)

guillaume@ubuntu:~/0x06$ ./4-main.py
Area: 7921 for size: 89
Area: 9 for size: 3
size must be an integer
guillaume@ubuntu:~/0x06$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x06-python-classes`
- File: `4-square.py`

☐ Done? | Help | Check your code | >_ Get a sandbox

5. Printing a square

mandatory

Write a class `Square` that defines a square by: (based on `4-square.py`)

- Private instance attribute: `size` :
  - property `def size(self):` to retrieve it
  - property setter `def size(self, value):` to set it:
    - `size` must be an integer, otherwise raise a `TypeError` exception with the message `size must be an integer`
    - if `size` is less than `0` , raise a `ValueError` exception with the message `size must be >= 0`
- Instantiation with optional `size` : `def __init__(self, size=0):`
- Public instance method: `def area(self):` that returns the current square area
- Public instance method: `def my_print(self):` that prints in stdout the square with the character `#`:
  - if `size` is equal to 0, print an empty line
- You are not allowed to import any module

7)

```
guillaume@ubuntu:~/0x06$ cat 5-main.py
#!/usr/bin/python3
Square = __import__('5-square').Square

my_square = Square(3)
my_square.my_print()

print("--")

my_square.size = 10
my_square.my_print()

print("--")

my_square.size = 0
my_square.my_print()

print("--")

guillaume@ubuntu:~/0x06$ ./5-main.py
###
###
###
--
##########
##########
##########
##########
##########
##########
##########
##########
##########
##########
--

--
guillaume@ubuntu:~/0x06$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x06-python-classes`
- File: `5-square.py`

☐ Done? | Help | Check your code | >_ Get a sandbox

6. Coordinates of a square

mandatory

(/)

Write a class `Square` that defines a square by: (based on `5-square.py`)

- Private instance attribute: `size`:
  - property `def size(self):` to retrieve it
  - property setter `def size(self, value):` to set it:
    - `size` must be an integer, otherwise raise a `TypeError` exception with the message `size must be an integer`
    - if `size` is less than `0`, raise a `ValueError` exception with the message `size must be >= 0`
- Private instance attribute: `position`:
  - property `def position(self):` to retrieve it
  - property setter `def position(self, value):` to set it:
    - `position` must be a tuple of 2 positive integers, otherwise raise a `TypeError` exception with the message `position must be a tuple of 2 positive integers`
- Instantiation with optional `size` and optional `position`: `def __init__(self, size=0, position=(0, 0)):`
- Public instance method: `def area(self):` that returns the current square area
- Public instance method: `def my_print(self):` that prints in stdout the square with the character `#`:
  - if `size` is equal to 0, print an empty line
  - `position` should be use by using space - **Don't fill lines by spaces** when `position[1] > 0`
- You are not allowed to import any module

7)

```
guillaume@ubuntu:~/0x06$ cat 6-main.py
#!/usr/bin/python3
Square = __import__('6-square').Square

my_square_1 = Square(3)
my_square_1.my_print()

print("--")

my_square_2 = Square(3, (1, 1))
my_square_2.my_print()

print("--")

my_square_3 = Square(3, (3, 0))
my_square_3.my_print()

print("--")

guillaume@ubuntu:~/0x06$ ./6-main.py | tr " " "_" | cat -e
###$
###$
###$
--$
$
_###$
_###$
_###$
--$
___###$
___###$
___###$
--$
guillaume@ubuntu:~/0x06$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x06-python-classes`
- File: `6-square.py`

Done? | Help | Check your code | >_ Get a sandbox

Done with the mandatory tasks? Unlock 4 advanced tasks now! (/projects/247/unlock_optionals)

(/)