(/)

Python - Almost a circle

Python

0%

Master

By: Guillaume, CTO at Holberton School

Weight: 5

Your score will be updated once you launch the project review.

Background Context

The AirBnB project is a big part of the Higher level curriculum. This project will help you be ready for it.

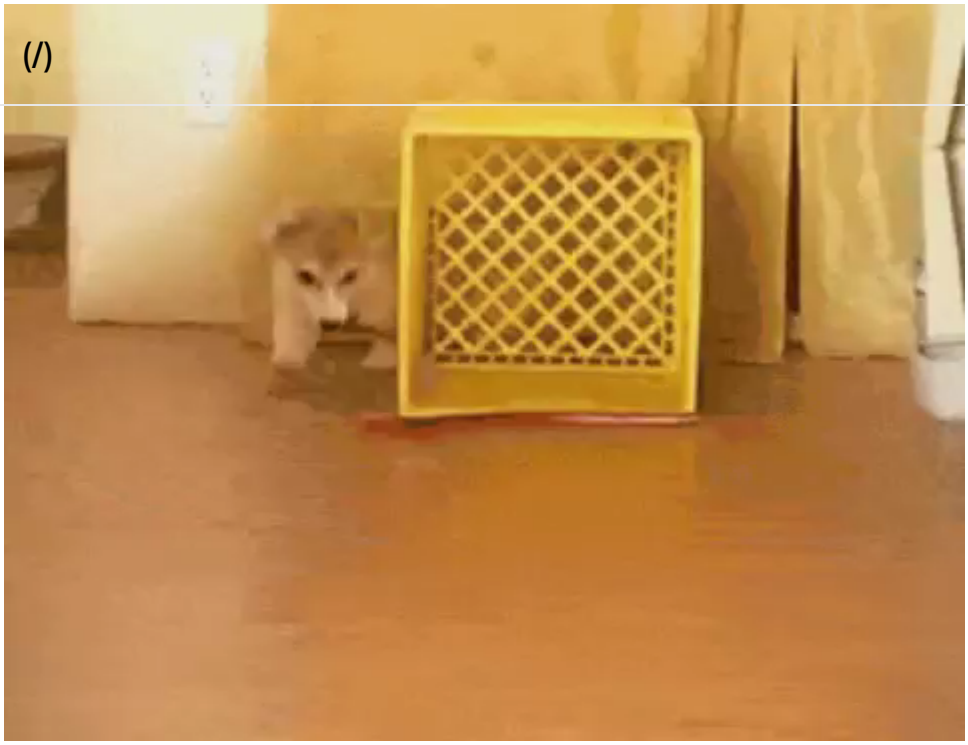
In this project, you will review everything about Python:

- Import
- Exceptions
- Class
- Private attribute
- Getter/Setter
- Class method
- Static method
- Inheritance
- Unittest
- Read/Write file

You will also learn about:

- `args` and `kwargs`
- Serialization/Deserialization
- JSON

(/)



Resources

Read or watch:

- args/kwargs (/rltoken/7zBCbNkZbTlhjBk3EIF51Q)
- JSON encoder and decoder (/rltoken/Z2J1HT8EQtKC5ppFRGN5JQ)
- unittest module (/rltoken/p5yLGmkQPUoKIXRMVYmNWA)
- Python test cheatsheet (/rltoken/wqELatpgT9UQIl0z7DPPAg)

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone (/rltoken/LnSf21WNEy0h1xxtCU1IuA), **without the help of Google**:

General

- What is Unit testing and how to implement it in a large project
- How to serialize and deserialize a Class
- How to write and read a JSON file
- What is `*args` and how to use it
- What is `**kwargs` and how to use it
- How to handle named arguments in a function

Requirements

Python Scripts

- Allowed editors: `Visual studio code`
- All your files will be interpreted/compiled on Ubuntu 20.04 LTS using `python3` (version 3.4.3)

- All your files should end with a new line
- A `README.md` file, at the root of the folder of the project, is mandatory
- Your code should use the `PEP 8` style (version 1.7.*)
- The length of your files will be tested using `wc`
- All your modules should be documented: `python3 -c 'print(__import__("my_module").__doc__)'`
- All your classes should be documented: `python3 -c 'print(__import__("my_module").MyClass.__doc__)'`
- All your functions (inside and outside a class) should be documented: `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)

Tasks

0. Base class

mandatory

Write the first class `Base`:

Create a folder named `models` with an empty file `__init__.py` inside - with this file, the folder will become a Python package

Create a file named `models/base.py`:

- Class `Base`:
  - private class attribute `__nb_objects = 0`
  - class constructor: `def __init__(self, id=None):`:
    - if `id` is not `None`, assign the public instance attribute `id` with this argument value - you can assume `id` is an integer and you don't need to test the type of it
    - otherwise, increment `__nb_objects` and assign the new value to the public instance attribute `id`

This class will be the "base" of all other classes in this project. The goal of it is to manage `id` attribute in all your future classes and to avoid duplicating the same code (by extension, same bugs)

```
guillaume@ubuntu:~/$ cat 0-main.py
#!/usr/bin/python3
""" 0-main """
from models.base import Base

if __name__ == "__main__":

    b1 = Base()
    print(b1.id)

    b2 = Base()
    print(b2.id)

    b3 = Base()
    print(b3.id)

    b4 = Base(12)
    print(b4.id)

    b5 = Base()
    print(b5.id)

guillaume@ubuntu:~/$ ./0-main.py
1
2
3
12
4
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alx_python`
- Directory: `python-almost_a_circle`
- File: `models/base.py, models/__init__.py`

Help | Check your code

0/20 pts

1. First Rectangle

mandatory

Write the class `Rectangle` that inherits from `Base` :

- In the file `models/rectangle.py`
- Class `Rectangle` inherits from `Base`
- Private instance attributes, each with its own public getter and setter:
  - `__width -> width`
  - `__height -> height`
  - `__x -> x`

- `__y -> y`
- Class constructor: `def __init__(self, width, height, x=0, y=0, id=None)`:
  - Call the super class with `id` - this super call with use the logic of the `__init__` of the `Base` class
  - Assign each argument `width`, `height`, `x` and `y` to the right attribute

Why private attributes with getter/setter? Why not directly public attribute?

Because we want to protect attributes of our class. With a setter, you are able to validate what a developer is trying to assign to a variable. So after, in your class you can "trust" these attributes.

```
guillaume@ubuntu:~/$ cat 1-main.py
#!/usr/bin/python3
""" 1-main """
from models.rectangle import Rectangle

if __name__ == "__main__":

    r1 = Rectangle(10, 2)
    print(r1.id)

    r2 = Rectangle(2, 10)
    print(r2.id)

    r3 = Rectangle(10, 2, 0, 0, 12)
    print(r3.id)

guillaume@ubuntu:~/$ ./1-main.py
1
2
12
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alx_python`
- Directory: `python-almost_a_circle`
- File: `models/rectangle.py`

Help | Check your code

0/36 pts

2. Validate attributes

mandatory

Update the class `Rectangle` by adding validation of all setter methods and instantiation (`id` excluded):

- If the input is not an integer, raise the `TypeError` exception with the message: `<name of the attribute> must be an integer`. Example: `width must be an integer`
- If `width` or `height` is under or equals 0, raise the `ValueError` exception with the message: `<name of the attribute> must be > 0`. Example: `width must be > 0`

- If `x` or `y` is under 0, raise the `ValueError` exception with the message: `<name of the attribute> must be >= 0`. Example: `x must be >= 0`

(/)

```
guillaume@ubuntu:~/$ cat 2-main.py
#!/usr/bin/python3
""" 2-main """
from models.rectangle import Rectangle

if __name__ == "__main__":

    try:
        Rectangle(10, "2")
    except Exception as e:
        print("[{}] {}".format(e.__class__.__name__, e))

    try:
        r = Rectangle(10, 2)
        r.width = -10
    except Exception as e:
        print("[{}] {}".format(e.__class__.__name__, e))

    try:
        r = Rectangle(10, 2)
        r.x = {}
    except Exception as e:
        print("[{}] {}".format(e.__class__.__name__, e))

    try:
        Rectangle(10, 2, 3, -1)
    except Exception as e:
        print("[{}] {}".format(e.__class__.__name__, e))

guillaume@ubuntu:~/$ ./2-main.py
[TypeError] height must be an integer
[ValueError] width must be > 0
[TypeError] x must be an integer
[ValueError] y must be >= 0
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alx_python`
- Directory: `python-almost_a_circle`
- File: `models/rectangle.py`

Help | Check your code

0/64 pts

3. Area first

mandatory

Update the class `Rectangle` by adding the public method `def area(self):` that returns the area value of the `Rectangle` instance.

```
guillaume@ubuntu:~/$ cat 3-main.py
#!/usr/bin/python3
""" 3-main """
from models.rectangle import Rectangle

if __name__ == "__main__":

    r1 = Rectangle(3, 2)
    print(r1.area())

    r2 = Rectangle(2, 10)
    print(r2.area())

    r3 = Rectangle(8, 7, 0, 0, 12)
    print(r3.area())

guillaume@ubuntu:~/$ ./3-main.py
6
20
56
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alx_python`
- Directory: `python-almost_a_circle`
- File: `models/rectangle.py`

Help | Check your code

0/16 pts

4. Display #0

mandatory

Update the class `Rectangle` by adding the public method `def display(self):` that prints in stdout the `Rectangle` instance with the character `#` - you don't need to handle `x` and `y` here.

```
guillaume@ubuntu:~/$ cat 4-main.py
#!/usr/bin/python3
""" 4-main """
from models.rectangle import Rectangle

if __name__ == "__main__":

    r1 = Rectangle(4, 6)
    r1.display()

    print("---")

    r1 = Rectangle(2, 2)
    r1.display()

guillaume@ubuntu:~/$ ./4-main.py
####
####
####
####
####
####
---
##
##
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alx_python`
- Directory: `python-almost_a_circle`
- File: `models/rectangle.py`

Help | Check your code

0/24 pts

5. __str__

mandatory

Update the class `Rectangle` by overriding the `__str__` method so that it returns `[Rectangle] (<id>) <x>/<y> - <width>/<height>`

```
guillaume@ubuntu:~/$ cat 5-main.py
#!/usr/bin/python3
""" 5-main """
from models.rectangle import Rectangle

if __name__ == "__main__":

    r1 = Rectangle(4, 6, 2, 1, 12)
    print(r1)

    r2 = Rectangle(5, 5, 1)
    print(r2)

guillaume@ubuntu:~/$ ./5-main.py
[Rectangle] (12) 2/1 - 4/6
[Rectangle] (1) 1/0 - 5/5
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alx_python`
- Directory: `python-almost_a_circle`
- File: `models/rectangle.py`

Help | Check your code

0/20 pts

6. Display #1

mandatory

Update the class `Rectangle` by improving the public method `def display(self):` to print in stdout the `Rectangle` instance with the character `#` by taking care of `x` and `y`

```
guillaume@ubuntu:~/$ cat 6-main.py
#!/usr/bin/python3
""" 6-main """
from models.rectangle import Rectangle

if __name__ == "__main__":

    r1 = Rectangle(2, 3, 2, 2)
    r1.display()

    print("---")

    r2 = Rectangle(3, 2, 1, 0)
    r2.display()

guillaume@ubuntu:~/$ ./6-main.py | cat -e
$
$
  ##$
  ##$
  ##$
---$
 ###$
 ###$
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alx_python`
- Directory: `python-almost_a_circle`
- File: `models/rectangle.py`

Help | Check your code

0/16 pts

7. Update #0

mandatory

Update the class `Rectangle` by adding the public method `def update(self, *args):` that assigns an argument to each attribute:

- 1st argument should be the `id` attribute
- 2nd argument should be the `width` attribute
- 3rd argument should be the `height` attribute
- 4th argument should be the `x` attribute
- 5th argument should be the `y` attribute

This type of argument is called a “no-keyword argument” - Argument order is super important.

7)

```
guillaume@ubuntu:~/$ cat 7-main.py
#!/usr/bin/python3
""" Doc """
from models.rectangle import Rectangle

if __name__ == "__main__":

    r1 = Rectangle(10, 10, 10, 10)
    print(r1)

    r1.update(89)
    print(r1)

    r1.update(89, 2)
    print(r1)

    r1.update(89, 2, 3)
    print(r1)

    r1.update(89, 2, 3, 4)
    print(r1)

    r1.update(89, 2, 3, 4, 5)
    print(r1)

guillaume@ubuntu:~/$ ./7-main.py
[Rectangle] (1) 10/10 - 10/10
[Rectangle] (89) 10/10 - 10/10
[Rectangle] (89) 10/10 - 2/10
[Rectangle] (89) 10/10 - 2/3
[Rectangle] (89) 4/10 - 2/3
[Rectangle] (89) 4/5 - 2/3
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alx_python`
- Directory: `python-almost_a_circle`
- File: `models/rectangle.py`

Help | Check your code

0/23 pts

8. Update #1

mandatory

Update the class `Rectangle` by updating the public method `def update(self, *args):` by changing the prototype to `update(self, *args, **kwargs)` that assigns a key/value argument to attributes:

- `**kwargs` can be thought of as a double pointer to a dictionary: key/value

(/)

- As Python doesn't have pointers, `**kwargs` is not literally a double pointer – describing it as such is just a way of explaining its behavior in terms you're already familiar with
- `**kwargs` must be skipped if `*args` exists and is not empty
- Each key in this dictionary represents an attribute to the instance

This type of argument is called a "key-worded argument". Argument order is not important.

```
guillaume@ubuntu:~/$ cat 8-main.py
#!/usr/bin/python3
""" 8-main """
from models.rectangle import Rectangle

if __name__ == "__main__":

    r1 = Rectangle(10, 10, 10, 10)
    print(r1)

    r1.update(height=1)
    print(r1)

    r1.update(width=1, x=2)
    print(r1)

    r1.update(y=1, width=2, x=3, id=89)
    print(r1)

    r1.update(x=1, height=2, y=3, width=4)
    print(r1)

guillaume@ubuntu:~/$ ./8-main.py
[Rectangle] (1) 10/10 - 10/10
[Rectangle] (1) 10/10 - 10/1
[Rectangle] (1) 2/10 - 1/1
[Rectangle] (89) 3/1 - 2/1
[Rectangle] (89) 1/3 - 4/2
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alx_python`
- Directory: `python-almost_a_circle`
- File: `models/rectangle.py`

Help | Check your code

0/40 pts

9. And now, the Square!

mandatory

Write the class `Square` that inherits from `Rectangle`:

- In the file `models/square.py`
- Class `Square` inherits from `Rectangle`
- Class constructor: `def __init__(self, size, x=0, y=0, id=None):`:
  - Call the super class with `id`, `x`, `y`, `width` and `height` - this super call will use the logic of the `__init__` of the `Rectangle` class. The `width` and `height` must be assigned to the value of `size`
  - You must not create new attributes for this class, use all attributes of `Rectangle` - As reminder: a Square is a Rectangle with the same width and height
  - All `width`, `height`, `x` and `y` validation must inherit from `Rectangle` - same behavior in case of wrong data
- The overloading `__str__` method should return `[Square] (<id>) <x>/<y> - <size>` - in our case, `width` or `height`

As you know, a Square is a special Rectangle, so it makes sense this class Square inherits from Rectangle. Now you have a Square class who has the same attributes and same methods.

9)

```
guillaume@ubuntu:~/$ cat 9-main.py
#!/usr/bin/python3
""" 9-main """
from models.square import Square

if __name__ == "__main__":

    s1 = Square(5)
    print(s1)
    print(s1.area())
    s1.display()

    print("---")

    s2 = Square(2, 2)
    print(s2)
    print(s2.area())
    s2.display()

    print("---")

    s3 = Square(3, 1, 3)
    print(s3)
    print(s3.area())
    s3.display()

guillaume@ubuntu:~/$ ./9-main.py
[Square] (1) 0/0 - 5
25
#####
#####
#####
#####
#####
---
[Square] (2) 2/0 - 2
4
  ##
  ##
---
[Square] (3) 1/3 - 3
9



 ###
 ###
 ###
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alx_python`
- Directory: `python-almost_a_circle`
- File: `models/square.py`

Help | Check your code

0/32 pts

10. Square size

mandatory

Update the class `Square` by adding the public getter and setter `size`

- The setter should assign (in this order) the `width` and the `height` - with the same value
- The setter should have the same value validation as the `Rectangle` for `width` and `height` - No need to change the exception error message (It should be the one from `width`)

```
guillaume@ubuntu:~/$ cat 10-main.py
#!/usr/bin/python3
""" 10-main """
from models.square import Square

if __name__ == "__main__":

    s1 = Square(5)
    print(s1)
    print(s1.size)
    s1.size = 10
    print(s1)

    try:
        s1.size = "9"
    except Exception as e:
        print("[{}] {}".format(e.__class__.__name__, e))

guillaume@ubuntu:~/$ ./10-main.py
[Square] (1) 0/0 - 5
5
[Square] (1) 0/0 - 10
[TypeError] width must be an integer
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alx_python`
- Directory: `python-almost_a_circle`
- File: `models/square.py`

Help Check your code **0/16** pts

Score

0%

Your score will be updated once you launch the project review.

Please review **all the tasks** before you start the peer review.

Review all the tasks

Skip this project

Previous project (/projects/2066)