

(/)



Curriculum

SE Foundations ^
Average: 61.08% v

You have a captain's log due before 2024-05-12 (in 2 days)! Log it now!
(/captain_logs/5661831/edit)

You just released the advanced tasks of this project. Have fun!

0x08. Python - More Classes and Objects

Python **OOP**

⚙ Weight: 1

📅 Project over - took place from Jan 4, 2024 6:00 AM to Jan 5, 2024 6:00 AM

☑ An auto review will be launched at the deadline

In a nutshell...

- **Auto QA review:** 53.0/106 mandatory & 0.0/15 optional
- **Altogether: 50.0%**
 - Mandatory: 50.0%
 - Optional: 0.0%
 - Calculation: $50.0\% + (50.0\% * 0.0\%) == 50.0\%$

Resources

Read or watch:

- Object Oriented Programming (/rltoken/M-MFweENpRdEfRto_Gzlvq) (*Read everything until the paragraph "Inheritance" (excluded)*)
- Object-Oriented Programming (/rltoken/_Awd8Gn4SBdq2FRd_bY8KA) (*Please be careful: in most of the following paragraphs, the author shows 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 sure you read only the following paragraphs: "General Introduction," "First-class Everything,*



- (/ Minimal Class in Python," "Attributes," "Methods," "The `__init__` Method," "Data Abstraction, Data Encapsulation, and Information Hiding," "`__str__` - and `__repr__` -Methods," "Public-Protected- and Private Attributes," & "Destructor")
- Class and Instance Attributes (/rltoken/SGQlevRxW6ITgr4jGDzXbw)
 - classmethods and staticmethods (/rltoken/lj1EnTg02gtlknOkNv4xGA)
 - Properties vs. Getters and Setters (/rltoken/xjpk-jUNe0uGEzcNXbwIHQ) (Mainly the last part "Public instead of Private Attributes")
 - str vs repr (/rltoken/lu1ILT-t6FMuZvk7vRvfUQ)

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone (/rltoken/hOViVT2nJU8jeBxvw52bjw), **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
- What are the special `__str__` and `__repr__` methods and how to use them
- What is the difference between `__str__` and `__repr__`
- What is a class attribute
- What is the difference between a object attribute and a class attribute
- What is a class method
- What is a static method
- How to dynamically create arbitrary new attributes for existing instances of a class
- How to bind attributes to object and classes
- What is and what does contain `__dict__` of a class and of an instance of a class
- 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`

Quiz questions

Great! You've completed the quiz successfully! Keep going! ([Show quiz](#))

Tasks

0. Simple rectangle

mandatory

Score: 50.0% (*Checks completed: 100.0%*)

Write an empty class `Rectangle` that defines a rectangle:

- You are not allowed to import any module

```
guillaume@ubuntu:~/0x08$ cat 0-main.py
#!/usr/bin/python3
Rectangle = __import__('0-rectangle').Rectangle

my_rectangle = Rectangle()
print(type(my_rectangle))
print(my_rectangle.__dict__)

guillaume@ubuntu:~/0x08$ ./0-main.py
<class '0-rectangle.Rectangle'>
{}
guillaume@ubuntu:~/0x08$
```

No test cases needed

Repo:



- GitHub repository: alx-higher_level_programming
- (/). Directory: 0x08-python-more_classes
- File: 0-rectangle.py

☐ Done?

Check your code

> Get a sandbox

QA Review

1. Real definition of a rectangle

mandatory

Score: 50.0% (Checks completed: 100.0%)

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

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

```
guillaume@ubuntu:~/0x08$ cat 1-main.py
#!/usr/bin/python3
Rectangle = __import__('1-rectangle').Rectangle

my_rectangle = Rectangle(2, 4)
print(my_rectangle.__dict__)

my_rectangle.width = 10
my_rectangle.height = 3
print(my_rectangle.__dict__)


guillaume@ubuntu:~/0x08$ ./1-main.py
{'_Rectangle__height': 4, '_Rectangle__width': 2}
{'_Rectangle__height': 3, '_Rectangle__width': 10}
guillaume@ubuntu:~/0x08$
```

**No test cases needed****Repo:**

- GitHub repository: alx-higher_level_programming
- (/). Directory: 0x08-python-more_classes
- File: 1-rectangle.py

☐ Done?

Check your code

 Get a sandbox

QA Review

2. Area and Perimeter

mandatory

Score: 50.0% (Checks completed: 100.0%)

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

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



```
guillaume@ubuntu:~/0x08$ cat 2-main.py
#!/usr/bin/python3

Rectangle = __import__('2-rectangle').Rectangle

my_rectangle = Rectangle(2, 4)
print("Area: {} - Perimeter: {}".format(my_rectangle.area(), my_rectangle.perimeter()))

print("--")

my_rectangle.width = 10
my_rectangle.height = 3
print("Area: {} - Perimeter: {}".format(my_rectangle.area(), my_rectangle.perimeter()))

guillaume@ubuntu:~/0x08$ ./2-main.py
Area: 8 - Perimeter: 12
--
Area: 30 - Perimeter: 26
guillaume@ubuntu:~/0x08$
```

No test cases needed

Repo:

- GitHub repository: alx-higher_level_programming
- Directory: 0x08-python-more_classes
- File: 2-rectangle.py

☐ Done?

3. String representation

mandatory

Score: 50.0% (Checks completed: 100.0%)

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

- Private instance attribute: `width` :
 - property `def width(self)`: to retrieve it
 - property setter `def width(self, value)`: to set it:
 - `width` must be an integer, otherwise raise a `TypeError` exception with the message `width must be an integer`
 - if `width` is less than `0`, raise a `ValueError` exception with the message `width must be >= 0`
- Private instance attribute: `height` :
 - property `def height(self)`: to retrieve it
 - property setter `def height(self, value)`: to set it:
 - `height` must be an integer, otherwise raise a `TypeError` exception with the message `height must be an integer`



(//)

- if height is less than 0, raise a ValueError exception with the message height must be >= 0

- instantiation with optional width and height: def __init__(self, width=0, height=0):
- Public instance method: def area(self): that returns the rectangle area
- Public instance method: def perimeter(self): that returns the rectangle perimeter:
 - if width or height is equal to 0, perimeter has to be equal to 0
- print() and str() should print the rectangle with the character #: (see example below)
 - if width or height is equal to 0, return an empty string
- You are not allowed to import any module

```
guillaume@ubuntu:~/0x08$ cat 3-main.py
#!/usr/bin/python3
Rectangle = __import__('3-rectangle').Rectangle

my_rectangle = Rectangle(2, 4)
print("Area: {} - Perimeter: {}".format(my_rectangle.area(), my_rectangle.perimeter()))

print(str(my_rectangle))
print(repr(my_rectangle))

print("--")

my_rectangle.width = 10
my_rectangle.height = 3
print(my_rectangle)
print(repr(my_rectangle))

guillaume@ubuntu:~/0x08$ ./3-main.py
Area: 8 - Perimeter: 12
##
##
##
##
<3-rectangle.Rectangle object at 0x7f92a75a2eb8>
--
#####
#####
#####
<3-rectangle.Rectangle object at 0x7f92a75a2eb8>
guillaume@ubuntu:~/0x08$
```

Object address can be different

No test cases needed

Repo:

- GitHub repository: alx-higher_level_programming
- Directory: 0x08-python-more_classes
- File: 3-rectangle.py



(7) Done?

Check your code

>_ Get a sandbox

QA Review

4. Eval is magic

mandatory

Score: 50.0% (Checks completed: 100.0%)

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

- Private instance attribute: `width` :
 - property `def width(self)`: to retrieve it
 - property setter `def width(self, value)`: to set it:
 - `width` must be an integer, otherwise raise a `TypeError` exception with the message `width must be an integer`
 - if `width` is less than `0`, raise a `ValueError` exception with the message `width must be >= 0`
- Private instance attribute: `height` :
 - property `def height(self)`: to retrieve it
 - property setter `def height(self, value)`: to set it:
 - `height` must be an integer, otherwise raise a `TypeError` exception with the message `height must be an integer`
 - if `height` is less than `0`, raise a `ValueError` exception with the message `height must be >= 0`
- Instantiation with optional `width` and `height`: `def __init__(self, width=0, height=0)`:
- Public instance method: `def area(self)`: that returns the rectangle area
- Public instance method: `def perimeter(self)`: that returns the rectangle perimeter:
 - if `width` or `height` is equal to `0`, perimeter has to be equal to `0`
- `print()` and `str()` should print the rectangle with the character `#`: (see example below)
 - if `width` or `height` is equal to `0`, return an empty string
- `repr()` should return a string representation of the rectangle to be able to recreate a new instance by using `eval()` (see example below)
- You are not allowed to import any module




```
guillaume@ubuntu:~/0x08$ cat 4-main.py
#!/usr/bin/python3

Rectangle = __import__('4-rectangle').Rectangle

my_rectangle = Rectangle(2, 4)
print(str(my_rectangle))
print("--")
print(my_rectangle)
print("--")
print(repr(my_rectangle))
print("--")
print(hex(id(my_rectangle)))
print("--")

# create new instance based on representation
new_rectangle = eval(repr(my_rectangle))
print(str(new_rectangle))
print("--")
print(new_rectangle)
print("--")
print(repr(new_rectangle))
print("--")
print(hex(id(new_rectangle)))
print("--")

print(new_rectangle is my_rectangle)
print(type(new_rectangle) is type(my_rectangle))

guillaume@ubuntu:~/0x08$ ./4-main.py
##
##
##
##
--
##
##
##
##
--
Rectangle(2, 4)
--
0x7f09ebf7cc88
--
##
##
##
##
--
##
##
##
##
--
Rectangle(2, 4)
--
```



```
0x7f09ebf7ccc0
```

```
(/)
```

```
False
```

```
True
```

```
guillaume@ubuntu:~/0x08$
```

No test cases needed

Repo:

- GitHub repository: alx-higher_level_programming
- Directory: 0x08-python-more_classes
- File: 4-rectangle.py

☐ Done?

Check your code

> Get a sandbox

QA Review

5. Detect instance deletion

mandatory

Score: 50.0% (Checks completed: 100.0%)

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

- Private instance attribute: `width` :
 - property `def width(self)`: to retrieve it
 - property setter `def width(self, value)`: to set it:
 - `width` must be an integer, otherwise raise a `TypeError` exception with the message `width must be an integer`
 - if `width` is less than `0`, raise a `ValueError` exception with the message `width must be >= 0`
- Private instance attribute: `height` :
 - property `def height(self)`: to retrieve it
 - property setter `def height(self, value)`: to set it:
 - `height` must be an integer, otherwise raise a `TypeError` exception with the message `height must be an integer`
 - if `height` is less than `0`, raise a `ValueError` exception with the message `height must be >= 0`
- Instantiation with optional `width` and `height`: `def __init__(self, width=0, height=0):`
- Public instance method: `def area(self)`: that returns the rectangle area
- Public instance method: `def perimeter(self)`: that returns the rectangle perimeter:
 - if `width` or `height` is equal to `0`, perimeter has to be equal to `0`
- `print()` and `str()` should print the rectangle with the character `#`:
 - if `width` or `height` is equal to `0`, return an empty string
- `repr()` should return a string representation of the rectangle to be able to recreate a new instance by using `eval()`
- Print the message `Bye rectangle...` (`...` being 3 dots not ellipsis) when an instance of `Rectangle` is deleted
- You are not allowed to import any module



```
guillaume@ubuntu:~/0x08$ cat 5-main.py
#!/usr/bin/python3

Rectangle = __import__('5-rectangle').Rectangle

my_rectangle = Rectangle(2, 4)
print("Area: {} - Perimeter: {}".format(my_rectangle.area(), my_rectangle.perimeter()))

del my_rectangle

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

guillaume@ubuntu:~/0x08$ ./5-main.py
Area: 8 - Perimeter: 12
Bye rectangle...
[NameError] name 'my_rectangle' is not defined
guillaume@ubuntu:~/0x08$
```

No test cases needed

Repo:

- GitHub repository: alx-higher_level_programming
- Directory: 0x08-python-more_classes
- File: 5-rectangle.py

☐ Done?

6. How many instances

mandatory

Score: 50.0% (Checks completed: 100.0%)

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

- Private instance attribute: `width` :
 - property `def width(self)`: to retrieve it
 - property setter `def width(self, value)`: to set it:
 - `width` must be an integer, otherwise raise a `TypeError` exception with the message `width must be an integer`
 - if `width` is less than 0, raise a `ValueError` exception with the message `width must be >= 0`
- Private instance attribute: `height` :
 - property `def height(self)`: to retrieve it
 - property setter `def height(self, value)`: to set it:
 - `height` must be an integer, otherwise raise a `TypeError` exception with the message `height must be an integer`



(/)

- if height is less than 0, raise a ValueError exception with the message height must be >= 0

- Public class attribute number_of_instances :
 - Initialized to 0
 - Incremented during each new instance instantiation
 - Decrementd during each instance deletion
- Instantiation with optional width and height: def __init__(self, width=0, height=0):
- Public instance method: def area(self): that returns the rectangle area
- Public instance method: def perimeter(self): that returns the rectangle perimeter:
 - if width or height is equal to 0, perimeter has to be equal to 0
- print() and str() should print the rectangle with the character # :
 - if width or height is equal to 0, return an empty string
- repr() should return a string representation of the rectangle to be able to recreate a new instance by using eval()
- Print the message Bye rectangle... (... being 3 dots not ellipsis) when an instance of Rectangle is deleted
- You are not allowed to import any module

```
guillaume@ubuntu:~/0x08$ cat 6-main.py
#!/usr/bin/python3
Rectangle = __import__('6-rectangle').Rectangle

my_rectangle_1 = Rectangle(2, 4)
my_rectangle_2 = Rectangle(2, 4)
print("{:d} instances of Rectangle".format(Rectangle.number_of_instances))
del my_rectangle_1
print("{:d} instances of Rectangle".format(Rectangle.number_of_instances))
del my_rectangle_2
print("{:d} instances of Rectangle".format(Rectangle.number_of_instances))

guillaume@ubuntu:~/0x08$ ./6-main.py
2 instances of Rectangle
Bye rectangle...
1 instances of Rectangle
Bye rectangle...
0 instances of Rectangle
guillaume@ubuntu:~/0x08$
```

No test cases needed

Repo:

- GitHub repository: alx-higher_level_programming
- Directory: 0x08-python-more_classes
- File: 6-rectangle.py

☐ Done?



7. Change representation

mandatory

Score: 50.0% (Checks completed: 100.0%)

(/)

Write a class `Rectangle` that defines a rectangle by: (based on `6-rectangle.py`)

- Private instance attribute: `width` :
 - property `def width(self)`: to retrieve it
 - property setter `def width(self, value)`: to set it:
 - `width` must be an integer, otherwise raise a `TypeError` exception with the message `width must be an integer`
 - if `width` is less than 0, raise a `ValueError` exception with the message `width must be >= 0`
- Private instance attribute: `height` :
 - property `def height(self)`: to retrieve it
 - property setter `def height(self, value)`: to set it:
 - `height` must be an integer, otherwise raise a `TypeError` exception with the message `height must be an integer`
 - if `height` is less than 0, raise a `ValueError` exception with the message `height must be >= 0`
- Public class attribute `number_of_instances` :
 - Initialized to 0
 - Incremented during each new instance instantiation
 - Decrementd during each instance deletion
- Public class attribute `print_symbol` :
 - Initialized to `#`
 - Used as symbol for string representation
 - Can be any type
- Instantiation with optional `width` and `height`: `def __init__(self, width=0, height=0):`
- Public instance method: `def area(self)`: that returns the rectangle area
- Public instance method: `def perimeter(self)`: that returns the rectangle perimeter:
 - if `width` or `height` is equal to 0, perimeter has to be equal to 0
- `print()` and `str()` should print the rectangle with the character(s) stored in `print_symbol` :
 - if `width` or `height` is equal to 0, return an empty string
- `repr()` should return a string representation of the rectangle to be able to recreate a new instance by using `eval()`
- Print the message `Bye rectangle...` (... being 3 dots not ellipsis) when an instance of `Rectangle` is deleted
- You are not allowed to import any module



```
guillaume@ubuntu:~/0x08$ cat 7-main.py
#!/usr/bin/python3

Rectangle = __import__('7-rectangle').Rectangle

my_rectangle_1 = Rectangle(8, 4)
print(my_rectangle_1)
print("--")
my_rectangle_1.print_symbol = "&"
print(my_rectangle_1)
print("--")

my_rectangle_2 = Rectangle(2, 1)
print(my_rectangle_2)
print("--")
Rectangle.print_symbol = "C"
print(my_rectangle_2)
print("--")

my_rectangle_3 = Rectangle(7, 3)
print(my_rectangle_3)

print("--")

my_rectangle_3.print_symbol = ["C", "is", "fun!"]
print(my_rectangle_3)

print("--")

guillaume@ubuntu:~/0x08$ ./7-main.py
#####
#####
#####
#####
--
&&&&&&&&
&&&&&&&&
&&&&&&&&
&&&&&&&&
--
##
--
CC
--
CCCCCCC
CCCCCCC
CCCCCCC
--
['C', 'is', 'fun!']['C', 'is', 'fun!']['C', 'is', 'fun!']['C', 'is', 'fun!']['C',
'is', 'fun!']['C', 'is', 'fun!']['C', 'is', 'fun!']
['C', 'is', 'fun!']['C', 'is', 'fun!']['C', 'is', 'fun!']['C', 'is', 'fun!']['C',
'is', 'fun!']['C', 'is', 'fun!']['C', 'is', 'fun!']
['C', 'is', 'fun!']['C', 'is', 'fun!']['C', 'is', 'fun!']['C', 'is', 'fun!']['C',
'is', 'fun!']['C', 'is', 'fun!']['C', 'is', 'fun!']
--
Bye rectangle...
```

```
Bye rectangle...  
Bye rectangle...  
quillaume@ubuntu:~/0x08$
```


No test cases needed

Repo:

- GitHub repository: alx-higher_level_programming
- Directory: 0x08-python-more_classes
- File: 7-rectangle.py

☐ Done?

Check your code

 Get a sandbox

QA Review

8. Compare rectangles

mandatory

Score: 50.0% (Checks completed: 100.0%)

Write a class `Rectangle` that defines a rectangle by: (based on `7-rectangle.py`)

- Private instance attribute: `width` :
 - property `def width(self)`: to retrieve it
 - property setter `def width(self, value)`: to set it:
 - `width` must be an integer, otherwise raise a `TypeError` exception with the message `width must be an integer`
 - if `width` is less than `0`, raise a `ValueError` exception with the message `width must be >= 0`
- Private instance attribute: `height` :
 - property `def height(self)`: to retrieve it
 - property setter `def height(self, value)`: to set it:
 - `height` must be an integer, otherwise raise a `TypeError` exception with the message `height must be an integer`
 - if `height` is less than `0`, raise a `ValueError` exception with the message `height must be >= 0`
- Public class attribute `number_of_instances` :
 - Initialized to `0`
 - Incremented during each new instance instantiation
 - Decrementd during each instance deletion
- Public class attribute `print_symbol` :
 - Initialized to `#`
 - Used as symbol for string representation
 - Can be any type
- Instantiation with optional `width` and `height`: `def __init__(self, width=0, height=0):`
- Public instance method: `def area(self)`: that returns the rectangle area
- Public instance method: `def perimeter(self)`: that returns the rectangle perimeter:
 - if `width` or `height` is equal to `0`, perimeter has to be equal to `0`
- `print()` and `str()` should print the rectangle with the character `#` :

- if width or height is equal to 0, return an empty string
- (/). `repr()` should return a string representation of the rectangle to be able to recreate a new instance by using `eval()`
- Print the message `Bye rectangle...` (... being 3 dots not ellipsis) when an instance of `Rectangle` is deleted
- Static method `def bigger_or_equal(rect_1, rect_2):` that returns the biggest rectangle based on the area
 - `rect_1` must be an instance of `Rectangle`, otherwise raise a `TypeError` exception with the message `rect_1 must be an instance of Rectangle`
 - `rect_2` must be an instance of `Rectangle`, otherwise raise a `TypeError` exception with the message `rect_2 must be an instance of Rectangle`
 - Returns `rect_1` if both have the same area value
- You are not allowed to import any module

```
guillaume@ubuntu:~/0x08$ cat 8-main.py
#!/usr/bin/python3
Rectangle = __import__('8-rectangle').Rectangle

my_rectangle_1 = Rectangle(8, 4)
my_rectangle_2 = Rectangle(2, 3)

if my_rectangle_1 is Rectangle.bigger_or_equal(my_rectangle_1, my_rectangle_2):
    print("my_rectangle_1 is bigger or equal to my_rectangle_2")
else:
    print("my_rectangle_2 is bigger than my_rectangle_1")


my_rectangle_2.width = 10
my_rectangle_2.height = 5
if my_rectangle_1 is Rectangle.bigger_or_equal(my_rectangle_1, my_rectangle_2):
    print("my_rectangle_1 is bigger or equal to my_rectangle_2")
else:
    print("my_rectangle_2 is bigger than my_rectangle_1")


guillaume@ubuntu:~/0x08$ ./8-main.py
my_rectangle_1 is bigger or equal to my_rectangle_2
my_rectangle_2 is bigger than my_rectangle_1
Bye rectangle...
Bye rectangle...
guillaume@ubuntu:~/0x08$
```

No test cases needed

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x08-python-more_classes`
- File: `8-rectangle.py`



☐ Done?

Check your code

Get a sandbox

QA Review

9. A square is a rectangle

mandatory

Score: 50.0% (Checks completed: 100.0%)

Write a class `Rectangle` that defines a rectangle by: (based on `8-rectangle.py`)

- Private instance attribute: `width` :
 - property `def width(self)`: to retrieve it
 - property setter `def width(self, value)`: to set it:
 - `width` must be an integer, otherwise raise a `TypeError` exception with the message `width must be an integer`
 - if `width` is less than `0` ,raise a `ValueError` exception with the message `width must be >= 0`
- Private instance attribute: `height` :
 - property `def height(self)`: to retrieve it
 - property setter `def height(self, value)`: to set it:
 - `height` must be an integer, otherwise raise a `TypeError` exception with the message `height must be an integer`
 - if `height` is less than `0` ,raise a `ValueError` exception with the message `height must be >= 0`
- Public class attribute `number_of_instances` :
 - Initialized to `0`
 - Incremented during each new instance instantiation
 - Decrementd during each instance deletion
- Public class attribute `print_symbol` :
 - Initialized to `#`
 - Used as symbol for string representation
 - Can be any type
- Instantiation with optional `width` and `height`: `def __init__(self, width=0, height=0):`
- Public instance method: `def area(self)`: that returns the rectangle area
- Public instance method: `def perimeter(self)`: that returns the rectangle perimeter:
 - if `width` or `height` is equal to `0` ,perimeter has to be equal to `0`
- `print()` and `str()` should print the rectangle with the character `#` :
 - if `width` or `height` is equal to `0` ,return an empty string
- `repr()` should return a string representation of the rectangle to be able to recreate a new instance by using `eval()`
- Print the message `Bye rectangle...` (... being 3 dots not ellipsis) when an instance of `Rectangle` is deleted
- Static method `def bigger_or_equal(rect_1, rect_2)`: that returns the biggest rectangle based on the area
 - `rect_1` must be an instance of `Rectangle` ,otherwise raise a `TypeError` exception with the message `rect_1 must be an instance of Rectangle`
 - `rect_2` must be an instance of `Rectangle` ,otherwise raise a `TypeError` exception with the message `rect_2 must be an instance of Rectangle`
 - Returns `rect_1` if both have the same area value
- Class method `def square(cls, size=0)`: that returns a new `Rectangle` instance with `width == height == size`
- You are not allowed to import any module

```
guillaume@ubuntu:~/0x08$ cat 9-main.py
#!/usr/bin/python3

Rectangle = __import__('9-rectangle').Rectangle

my_square = Rectangle.square(5)
print("Area: {} - Perimeter: {}".format(my_square.area(), my_square.perimeter()))
print(my_square)

guillaume@ubuntu:~/0x08$ ./9-main.py
Area: 25 - Perimeter: 20
#####
#####
#####
#####
#####
Bye rectangle...
guillaume@ubuntu:~/0x08$
```

No test cases needed

Repo:

- GitHub repository: alx-higher_level_programming
- Directory: 0x08-python-more_classes
- File: 9-rectangle.py

☐ Done?

Check your code

 Get a sandbox

QA Review

10. N queens

#advanced

Score: 50.00% (Checks completed: 100.00%)





Chess grandmaster Judit Polgár (/rltoken/bsRwbT64OvYjWaClriv0jg), the strongest female chess player of all time

The N queens puzzle is the challenge of placing N non-attacking queens on an $N \times N$ chessboard. Write a program that solves the N queens problem.

- Usage: nqueens N
 - If the user called the program with the wrong number of arguments, print Usage: nqueens N , followed by a new line, and exit with the status 1
- where N must be an integer greater or equal to 4
 - If N is not an integer, print N must be a number , followed by a new line, and exit with the status 1
 - If N is smaller than 4 , print N must be at least 4 , followed by a new line, and exit with the status 1
- The program should print every possible solution to the problem
 - One solution per line
 - Format: see example
 - You don't have to print the solutions in a specific order
- You are only allowed to import the sys module

Read: Queen (/rltoken/dAQmi8RxMnLH-iHBzkz-lw), Backtracking (/rltoken/TGXZXdY2Awg8m4mSjlrjjA)

```
julien@ubuntu:~/0x08. N Queens$ ./101-nqueens.py 4
[[0, 1], [1, 3], [2, 0], [3, 2]]
[[0, 2], [1, 0], [2, 3], [3, 1]]
julien@ubuntu:~/0x08. N Queens$ ./101-nqueens.py 6
[[0, 1], [1, 3], [2, 5], [3, 0], [4, 2], [5, 4]]
[[0, 2], [1, 5], [2, 1], [3, 4], [4, 0], [5, 3]]
[[0, 3], [1, 0], [2, 4], [3, 1], [4, 5], [5, 2]]
[[0, 4], [1, 2], [2, 0], [3, 5], [4, 3], [5, 1]]
julien@ubuntu:~/0x08. N Queens$
```



Repo:

- GitHub repository: alx-higher_level_programming
- (/).
- Directory: 0x08-python-more_classes
- File: 101-nqueens.py

☐ Done?

Check your code

Ask for a new correction

 Get a sandbox

QA Review

Copyright © 2024 ALX, All rights reserved.

