(/)

Curriculum
SE Foundations
Average: **62.22%**

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!

0x07. Python - Test-driven development

Python **UnitTests** **TDD**

Weight: 1

Project over - took place from Jan 2, 2024 6:00 AM to Jan 8, 2024 6:00 AM

An auto review will be launched at the deadline

In a nutshell…

- **Auto QA review:** 71.0/167 mandatory & 0.0/104 optional
- **Altogether: 42.51%**
  - Mandatory: 42.51%
  - Optional: 0.0%
  - Calculation: 42.51% + (42.51% * 0.0%) == **42.51%**

Concepts

For this project, we expect you to look at this concept:

- Never forget a test (/concepts/47)

(/)

Background Context

Important notice on intranet checks for Python projects

Starting from today:

- Based on the requirements of each task, **you should always write the documentation (module(s) + function(s)) and tests first**, before you actually code anything
- The intranet checks for Python projects won't be released before their first deadline, in order for you to focus more on TDD and think about all possible cases
- We strongly encourage you to work together on test cases, so that you don't miss any edge case. **But not in the implementation of them!**
- **Don't trust the user**, always think about all possible edge cases

Resources

Read or watch:

- doctest — Test interactive Python examples (/rltoken/BwZJVq2MQ1_Vg_3gphoitQ) (*until "26.2.3.7. Warnings" included*)
- doctest – Testing through documentation (/rltoken/96kLRRIOHzsn3VDDXT21HA)
- Unit Tests in Python (/rltoken/wfuUl81Q3Nku1qCzdDHAfA)
- Unittest module (/rltoken/1v-d9Ol13JabJq8Ul6MlPg)
- Interactive and Non-interactive tests (/rltoken/lB65hNMXBziXy4A0YLlOog)

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone (/rltoken/tYtzLvssHW_9zR6SZQlNrQ), **without the help of Google**:

General

- Why Python programming is awesome
- What's an interactive test
- Why tests are important
- How to write Docstrings to create tests
- How to write documentation for each module and function

- What are the basic option flags to create tests
- How to find edge cases

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

Python Scripts

- 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`

Python Test Cases

- Allowed editors: `vi`, `vim`, `emacs`
- All your files should end with a new line
- All your test files should be inside a folder `tests`
- All your test files should be text files (extension: `.txt`)
- All your tests should be executed by using this command: `python3 -m doctest ./tests/*`
- All your modules should have a documentation (`python3 -c 'print(__import__("my_module").__doc__)'`)
- All your functions should have a documentation (`python3 -c 'print(__import__("my_module").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)
- We strongly encourage you to work together on test cases, so that you don't miss any edge case – The Checker is checking for tests!

Quiz questions

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

Tasks

0. Integers addition

mandatory

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

Write a function that adds 2 integers.

- Prototype: `def add_integer(a, b=98):`
- `a` and `b` must be integers or floats, otherwise raise a `TypeError` exception with the message `a must be an integer` or `b must be an integer`
- `a` and `b` must be first casted to integers if they are float
- Returns an integer: the addition of `a` and `b`
- You are not allowed to import any module

```
guillaume@ubuntu:~/0x07$ cat 0-main.py
#!/usr/bin/python3
add_integer = __import__('0-add_integer').add_integer

print(add_integer(1, 2))
print(add_integer(100, -2))
print(add_integer(2))
print(add_integer(100.3, -2))
try:
    print(add_integer(4, "School"))
except Exception as e:
    print(e)
try:
    print(add_integer(None))
except Exception as e:
    print(e)

guillaume@ubuntu:~/0x07$ ./0-main.py
3
98
100
98
b must be an integer
a must be an integer
guillaume@ubuntu:~/0x07$ python3 -m doctest -v ./tests/0-add_integer.txt | tail -
2
9 passed and 0 failed.
Test passed.
guillaume@ubuntu:~/0x07$ python3 -c 'print(__import__("0-add_integer").__doc__)'
| wc -l
5
guillaume@ubuntu:~/0x07$ python3 -c 'print(__import__("0-add_integer").add_intege
r.__doc__)' | wc -l
3
guillaume@ubuntu:~/0x07$
```

Repo:

(/)

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x07-python-test_driven_development`
- File: `0-add_integer.py, tests/0-add_integer.txt`

Done? | Check your code | Get a sandbox | QA Review

1. Divide a matrix

mandatory

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

Write a function that divides all elements of a matrix.

- Prototype: `def matrix_divided(matrix, div):`
- `matrix` must be a list of lists of integers or floats, otherwise raise a `TypeError` exception with the message `matrix must be a matrix (list of lists) of integers/floats`
- Each row of the `matrix` must be of the same size, otherwise raise a `TypeError` exception with the message `Each row of the matrix must have the same size`
- `div` must be a number (integer or float), otherwise raise a `TypeError` exception with the message `div must be a number`
- `div` can't be equal to `0`, otherwise raise a `ZeroDivisionError` exception with the message `division by zero`
- All elements of the matrix should be divided by `div`, rounded to 2 decimal places
- Returns a new matrix
- You are not allowed to import any module

```
guillaume@ubuntu:~/0x07$ cat 2-main.py
#!/usr/bin/python3
matrix_divided = __import__('2-matrix_divided').matrix_divided

matrix = [
    [1, 2, 3],
    [4, 5, 6]
]
print(matrix_divided(matrix, 3))
print(matrix)

guillaume@ubuntu:~/0x07$ ./2-main.py
[[0.33, 0.67, 1.0], [1.33, 1.67, 2.0]]
[[1, 2, 3], [4, 5, 6]]
guillaume@ubuntu:~/0x07$ python3 -m doctest -v ./tests/2-matrix_divided.txt | tai
l -2
5 passed and 0 failed.
Test passed.
guillaume@ubuntu:~/0x07$
```

Note: you might have a different number of tests than in the above example. As usual, your tests should cover all possible cases.

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x07-python-test_driven_development`
- File: `2-matrix_divided.py, tests/2-matrix_divided.txt`

(/)

Done? | Check your code | Get a sandbox | QA Review

2. Say my name

mandatory

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

Write a function that prints `My name is <first name> <last name>`

- Prototype: `def say_my_name(first_name, last_name=""):`
- `first_name` and `last_name` must be strings otherwise, raise a `TypeError` exception with the message `first_name must be a string` or `last_name must be a string`
- You are not allowed to import any module

```
guillaume@ubuntu:~/0x07$ cat 3-main.py
#!/usr/bin/python3
say_my_name = __import__('3-say_my_name').say_my_name

say_my_name("John", "Smith")
say_my_name("Walter", "White")
say_my_name("Bob")
try:
    say_my_name(12, "White")
except Exception as e:
    print(e)

guillaume@ubuntu:~/0x07$ ./3-main.py | cat -e
My name is John Smith$
My name is Walter White$
My name is Bob $
first_name must be a string$
guillaume@ubuntu:~/0x07$ python3 -m doctest -v ./tests/3-say_my_name.txt | tail -
2
5 passed and 0 failed.
Test passed.
guillaume@ubuntu:~/0x07$
```

Note: you might have a different number of tests than in the above example. As usual, your tests should cover all possible cases.

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x07-python-test_driven_development`
- File: `3-say_my_name.py, tests/3-say_my_name.txt`

(/) Done? | Check your code | >_ Get a sandbox | QA Review

3. Print square

mandatory

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

Write a function that prints a square with the character `#`.

- Prototype: `def print_square(size):`
- `size` is the size length of the square
- `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`
- if `size` is a float and is less than 0, raise a `TypeError` exception with the message `size must be an integer`
- You are not allowed to import any module

(7)

```
guillaume@ubuntu:~/0x07$ cat 4-main.py
#!/usr/bin/python3
print_square = __import__('4-print_square').print_square

print_square(4)
print("")
print_square(10)
print("")
print_square(0)
print("")
print_square(1)
print("")
try:
    print_square(-1)
except Exception as e:
    print(e)
print("")

guillaume@ubuntu:~/0x07$ ./4-main.py
####
####
####
####

##########
##########
##########
##########
##########
##########
##########
##########
##########
##########


#

size must be >= 0

guillaume@ubuntu:~/0x07$ python3 -m doctest -v ./tests/4-print_square.txt
guillaume@ubuntu:~/0x07$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x07-python-test_driven_development`
- File: `4-print_square.py, tests/4-print_square.txt`

Done? | Check your code | Get a sandbox | QA Review

4. Text indentation

mandatory

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

Write a function that prints a text with 2 new lines after each of these characters: `.`, `?` and `:`

- Prototype: `def text_indentation(text):`
- `text` must be a string, otherwise raise a `TypeError` exception with the message `text must be a string`
- There should be no space at the beginning or at the end of each printed line
- You are not allowed to import any module

(7)

```
guillaume@ubuntu:~/0x07$ cat 5-main.py
#!/usr/bin/python3
text_indentation = __import__('5-text_indentation').text_indentation

text_indentation("""Lorem ipsum dolor sit amet, consectetur adipiscing elit. \
Quonam modo? Utrum igitur tibi litteram videor an totas paginas commovere? \
Non autem hoc: igitur ne illud quidem. Fortasse id optimum, sed ubi illud: \
Plus semper voluptatis? Teneo, inquit, finem illi videri nihil dolere. \
Transfer idem ad modestiam vel temperantiam, quae est moderatio cupiditatum \
rationi oboediens. Si id dicis, vicimus. Inde sermone vario sex illa a Dipylo \
stadia confecimus. Sin aliud quid voles, postea. Quae animi affectio suum \
cuique tribuens atque hanc, quam dico. Utinam quidem dicerent alium alio \
beatiorem! Iam ruinas videres""")

guillaume@ubuntu:~/0x07$ ./5-main.py | cat -e
Lorem ipsum dolor sit amet, consectetur adipiscing elit.$
$
Quonam modo?$
$
Utrum igitur tibi litteram videor an totas paginas commovere?$
$
Non autem hoc:$
$
igitur ne illud quidem.$
$
Fortasse id optimum, sed ubi illud:$
$
Plus semper voluptatis?$
$
Teneo, inquit, finem illi videri nihil dolere.$
$
Transfer idem ad modestiam vel temperantiam, quae est moderatio cupiditatum ratio
ni oboediens.$
$
Si id dicis, vicimus.$
$
Inde sermone vario sex illa a Dipylo stadia confecimus.$
$
Sin aliud quid voles, postea.$
$
Quae animi affectio suum cuique tribuens atque hanc, quam dico.$
$
Utinam quidem dicerent alium alio beatiorem! Iam ruinas videresguillaume@ubuntu:
~/0x07$
guillaume@ubuntu:~/0x07$ python3 -m doctest -v ./tests/5-text_indentation.txt
guillaume@ubuntu:~/0x07$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x07-python-test_driven_development`
- File: `5-text_indentation.py, tests/5-text_indentation.txt`

(/) Done? | Check your code | >_ Get a sandbox | QA Review

5. Max integer - Unittest

mandatory

Score: 0.0% (*Checks completed: 0.0%*)

Since the beginning you have been creating “Interactive tests”. For this exercise, you will add Unittests.

In this task, you will write unittests for the function `def max_integer(list=[]):.`

- Your test file should be inside a folder `tests`
- You have to use the unittest module (/rltoken/hX5a13o-1mXGTQASWBitFQ)
- Your test file should be python files (extension: `.py`)
- Your test file should be executed by using this command: `python3 -m unittest tests.6-max_integer_test`
- All tests you make must be passable by the function below
- We strongly encourage you to work together on test cases, so that you don’t miss any edge case

(7)

```
guillaume@ubuntu:~/0x07$ cat 6-max_integer.py
#!/usr/bin/python3
"""Module to find the max integer in a list
"""


def max_integer(list=[]):
    """Function to find and return the max integer in a list of integers
        If the list is empty, the function returns None
    """
    if len(list) == 0:
        return None
    result = list[0]
    i = 1
    while i < len(list):
        if list[i] > result:
            result = list[i]
        i += 1
    return result

guillaume@ubuntu:~/0x07$
guillaume@ubuntu:~/0x07$ cat 6-main.py
#!/usr/bin/python3
max_integer = __import__('6-max_integer').max_integer

print(max_integer([1, 2, 3, 4]))
print(max_integer([1, 3, 4, 2]))
guillaume@ubuntu:~/0x07$
guillaume@ubuntu:~/0x07$ ./6-main.py
4
4
guillaume@ubuntu:~/0x07$
guillaume@ubuntu:~/0x07$ python3 -m unittest tests.6-max_integer_test 2>&1 | tail
-1
OK
guillaume@ubuntu:~/0x07$
guillaume@ubuntu:~/0x07$ head -7 tests/6-max_integer_test.py
#!/usr/bin/python3
"""Unittest for max_integer([..])
"""
import unittest
max_integer = __import__('6-max_integer').max_integer

class TestMaxInteger(unittest.TestCase):
guillaume@ubuntu:~/0x07$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x07-python-test_driven_development`
- File: `tests/6-max_integer_test.py`

(/)Done? | Check your code | Ask for a new correction | >_ Get a sandbox | QA Review

6. Matrix multiplication

#advanced

Score: 47.87% (*Checks completed: 95.74%*)

Write a function that multiplies 2 matrices:

- Read: Matrix multiplication - only Matrix product (two matrices) (/rltoken/Qw_rYR3IYYL5DHDH-iCWCA)
- Prototype: `def matrix_mul(m_a, m_b):`
- `m_a` and `m_b` must be validated with these requirements in this order
- `m_a` and `m_b` must be an list of lists of integers or floats:
  - if `m_a` or `m_b` is not a list: raise a `TypeError` exception with the message `m_a must be a list` or `m_b must be a list`
  - if `m_a` or `m_b` is not a list of lists: raise a `TypeError` exception with the message `m_a must be a list of lists` or `m_b must be a list of lists`
  - if `m_a` or `m_b` is empty (it means: `= []` or `= [[]]`): raise a `ValueError` exception with the message `m_a can't be empty` or `m_b can't be empty`
  - if one element of those list of lists is not an integer or a float: raise a `TypeError` exception with the message `m_a should contain only integers or floats` or `m_b should contain only integers or floats`
  - if `m_a` or `m_b` is not a rectangle (all 'rows' should be of the same size): raise a `TypeError` exception with the message `each row of m_a must be of the same size` or `each row of m_b must be of the same size`
- If `m_a` and `m_b` can't be multiplied: raise a `ValueError` exception with the message `m_a and m_b can't be multiplied`
- You are not allowed to import any module

```
guillaume@ubuntu:~/0x07$ cat 100-main.py
#!/usr/bin/python3
matrix_mul = __import__('100-matrix_mul').matrix_mul

print(matrix_mul([[1, 2], [3, 4]], [[1, 2], [3, 4]]))
print(matrix_mul([[1, 2]], [[3, 4], [5, 6]]))

guillaume@ubuntu:~/0x07$ ./100-main.py
[[7, 10], [15, 22]]
[[13, 16]]
guillaume@ubuntu:~/0x07$ python3 -m doctest -v ./tests/100-matrix_mul.txt | tail
-2
6 passed and 0 failed.
Test passed.
guillaume@ubuntu:~/0x07$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x07-python-test_driven_development`
- File: `100-matrix_mul.py, tests/100-matrix_mul.txt`

Done? | Check your code | Ask for a new correction | Get a sandbox | QA Review

7. Lazy matrix multiplication

#advanced

Score: 50.00% (*Checks completed: 100.00%*)

Write a function that multiplies 2 matrices by using the module NumPy (/rltoken/sXnBuOVSyhKEGt-biOyOWg)

To install it: `pip3 install numpy==1.15.0`

- Prototype: `def lazy_matrix_mul(m_a, m_b):`
- Test cases should be the same as `100-matrix_mul` but with new exception type/message

```
guillaume@ubuntu:~/0x07$ cat 101-main.py
#!/usr/bin/python3
lazy_matrix_mul = __import__('101-lazy_matrix_mul').lazy_matrix_mul

print(lazy_matrix_mul([[1, 2], [3, 4]], [[1, 2], [3, 4]]))
print(lazy_matrix_mul([[1, 2]], [[3, 4], [5, 6]]))

guillaume@ubuntu:~/0x07$ ./101-main.py
[[ 7 10]
 [15 22]]
[[13 16]]
guillaume@ubuntu:~/0x07$ python3 -m doctest -v ./tests/101-lazy_matrix_mul.txt
guillaume@ubuntu:~/0x07$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x07-python-test_driven_development`
- File: `101-lazy_matrix_mul.py, tests/101-lazy_matrix_mul.txt`

Done? | Check your code | Ask for a new correction | Get a sandbox | QA Review

8. CPython #3: Python Strings

#advanced

Score: 50.00% (*Checks completed: 100.00%*)

Create a function that prints Python strings.

- Prototype: `void print_python_string(PyObject *p);`
- Format: see example
- If `p` is not a valid string, print an error message (see example)
- Read: Unicode HOWTO (/rltoken/UkkHHaILiYf9d_a3nc4Bxw)

About:

- Python version: 3.4
- You are allowed to use the C standard library
- Your shared library will be compiled with this command line: `gcc -shared -Wl,-soname,libPython.so -o libPython.so -fPIC -I/usr/include/python3.4 102-python.c`

```
julien@ubuntu:~/0x07. Pyhton Strings$ cat 102-tests.py
import ctypes

lib = ctypes.CDLL('./libPython.so')
lib.print_python_string.argtypes = [ctypes.py_object]
s = "The spoon does not exist"
lib.print_python_string(s)
s = "ложка не существует"
lib.print_python_string(s)
s = "La cuillère n'existe pas"
lib.print_python_string(s)
s = "勺子不存在"
lib.print_python_string(s)
s = "숟가락은 존재하지 않는다."
lib.print_python_string(s)
s = "スプーンは存在しない"
lib.print_python_string(s)
s = b"The spoon does not exist"
lib.print_python_string(s)
julien@ubuntu:~/0x07. Pyhton Strings$ gcc -shared -Wl,-soname,libPython.so -o lib
Python.so -fPIC -I/usr/include/python3.4 102-python.c
julien@ubuntu:~/0x07. Pyhton Strings$ python3 ./102-tests.py
[.] string object info
  type: compact ascii
  length: 24
  value: The spoon does not exist
[.] string object info
  type: compact unicode object
  length: 19
  value: ложка не существует
[.] string object info
  type: compact unicode object
  length: 24
  value: La cuillère n'existe pas
[.] string object info
  type: compact unicode object
  length: 5
  value: 勺子不存在
[.] string object info
  type: compact unicode object
  length: 14
  value: 숟가락은 존재하지 않는다.
[.] string object info
  type: compact unicode object
  length: 10
  value: スプーンは存在しない
[.] string object info
  [ERROR] Invalid String Object
julien@ubuntu:~/0x07. Pyhton Strings$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x07-python-test_driven_development`

- File: `102-python.c`

(/)

Done? | Check your code | Ask for a new correction | Get a sandbox | QA Review