(/)

65%

Python - import & modules

Python

Novice

By: Guillaume, CTO at Holberton School

Weight: 1

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

Resources

Read or watch:

- Modules (/rltoken/BSjlo-phgAvB9FWYeTo-bQ)
- Command line arguments (/rltoken/wF0YjiM9DAGAkVLFsC7OdQ)
- Errors and Exceptions (/rltoken/dgTLhFa7VKGOfCy8sIVhfw)
- Learn to Program 11 Static & Exception Handling (/rltoken/KI0rZ0gVrDszVd6fzV6XtQ) (*starting at minute 7*)
- PEP 8 – Style Guide for Python Code (/rltoken/jO9By8tpVv_vKzNjc1TeHA)

man or help:

- `python3`

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone (/rltoken/MeWMQXhWHDbHC8-0a11Cbg), **without the help of Google**:

General

- Why Python programming is awesome
- How to import functions from another file
- How to use imported functions
- How to create a module
- How to use the built-in function `dir()`

- How to prevent code in your script from being executed when imported
- How to use command line arguments with your Python programs
- What's the difference between errors and exceptions
- What are exceptions and how to use them
- When do we need to use exceptions
- How to correctly handle an exception
- What's the purpose of catching exceptions
- How to raise a builtin exception
- When do we need to implement a clean-up action after an exception

Requirements

General

- Recommended editor: `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`

Quiz questions

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

Tasks

0. Import a simple function from a simple file

mandatory

Write a program that imports the function `def add(a, b):` from the file `add_0.py` and prints the result of the addition `1 + 2 = 3`

- You have to use `print` function with string format to display integers
- You have to assign:
  - the value `1` to a variable called `a`
  - the value `2` to a variable called `b`
  - and use those two variables as arguments when calling the functions `add` and `print`
- `a` and `b` must be defined in 2 different lines: `a = 1` and another `b = 2`
- Your program should print: `<a value> + <b value> = <add(a, b) value>` followed with a new line
- You can only use the word `add_0` once in your code
- You are not allowed to use `*` for importing or `__import__`
- Your code should not be executed when imported - by using `__import__` , like the example below

0)

```
guillaume@ubuntu:~/$ cat add_0.py
#!/usr/bin/python3
def add(a, b):
    """My addition function

    Args:
        a: first integer
        b: second integer

    Returns:
        The return value. a + b
    """
    return (a + b)

guillaume@ubuntu:~/$ ./0-add.py
1 + 2 = 3
guillaume@ubuntu:~/$ cat 0-import_add.py
__import__("0-add")
guillaume@ubuntu:~/$ python3 0-import_add.py
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alx_python`
- Directory: `python-import_modules`
- File: `0-add.py`

Help | Check your code | >_ Get a sandbox

9.1/14 pts

1. How to make a script dynamic!

mandatory

Write a program that prints the number of and the list of its arguments.

- The output should be:
  - Number of argument(s) followed by `argument` (if number is one) or `arguments` (otherwise), followed by
  - `:` (or `.` if no arguments were passed) followed by
  - a new line, followed by (if at least one argument),
  - one line per argument:
    - the position of the argument (starting at `1`) followed by `:`, followed by the argument value and a new line
- Your code should not be executed when imported
- The number of elements of `argv` can be retrieved by using: `len(argv)`
- You do not have to fully understand lists yet, but imagine that `argv` can be used just like a collection of arguments: you can use an index to walk through it. There are other ways (which will be preferred for future project tasks), if you know them you can use them.

```
guillaume@ubuntu:~/$ ./1-args.py
0 arguments.
guillaume@ubuntu:~/$ ./1-args.py Hello
1 argument:
1: Hello
guillaume@ubuntu:~/$ ./1-args.py Hello Holberton School 98 Battery street
6 arguments:
1: Hello
2: Holberton
3: School
4: 98
5: Battery
6: street
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alx_python`
- Directory: `python-import_modules`
- File: `1-args.py`

Help | Check your code | >_ Get a sandbox

9.1/14 pts

2. Everything can be imported

mandatory

Write a program that imports the variable `a` from the file `variable_load_2.py` and prints its value.

- You are not allowed to use `*` for importing or `__import__`
- Your code should not be executed when imported

```
guillaume@ubuntu:~/$ cat variable_load_2.py
#!/usr/bin/python3
a = 98
"""Simple variable
"""

guillaume@ubuntu:~/$ ./2-variable_load.py
98
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alx_python`
- Directory: `python-import_modules`
- File: `2-variable_load.py`

Help | Check your code | >_ Get a sandbox **7.15/11** pts

3. Integers division with debug

mandatory

Write a function that divides 2 integers and prints the result.

- Prototype: `def safe_print_division(a, b):`
- You can assume that `a` and `b` are integers
- The result of the division should print on the `finally:` section preceded by `Inside result:`
- Returns the value of the division, otherwise: `None`
- You have to use `try: / except: / finally:`
- You have to use `"{}".format()` to print the result
- You are not allowed to import any module

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

a = 12
b = 2
result = safe_print_division(a, b)
print("{:d} / {:d} = {}".format(a, b, result))

a = 12
b = 0
result = safe_print_division(a, b)
print("{:d} / {:d} = {}".format(a, b, result))

guillaume@ubuntu:~/$ ./3-main.py
Inside result: 6.0
12 / 2 = 6.0
Inside result: None
12 / 0 = None
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alx_python`
- Directory: `python-import_modules`
- File: `3-safe_print_division.py`

Help | Check your code | >_ Get a sandbox **5.85/9** pts

4. Raise exception

mandatory

Write a function that raises a type exception.

- Prototype: `def raise_exception():`
- You are not allowed to import any module

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

try:
    raise_exception()
except TypeError as te:
    print("Exception raised")

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

Repo:

- GitHub repository: `alx_python`
- Directory: `python-import_modules`
- File: `4-raise_exception.py`

Help | Check your code | >_ Get a sandbox

6.5/10 pts

5. Raise a message

mandatory

Write a function that raises a name exception with a message.

- Prototype: `def raise_exception_msg(message=""):`
- You are not allowed to import any module

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

try:
    raise_exception_msg("C is fun")
except NameError as ne:
    print(ne)

guillaume@ubuntu:~/$ ./5-main.py
C is fun
guillaume@ubuntu:~/$
```

Repo:

- GitHub repository: `alx_python`
- Directory: `python-import_modules`
- File: `5-raise_exception_msg.py`

Help | Check your code | >_ Get a sandbox

5.2/8 pts

Score

65%

Congratulations! You made it!

Next project: Python - Data Structures: Lists, Tuples

Open the next project (/projects/2057)

Previous project (/projects/100075)