(/)

Curriculum
SE Foundations
Average: **59.43%**

You have a captain's log due before 2024-05-12 (in 3 days)! Log it now! (/captain_logs/5661831/edit)

0x05. Python - Exceptions

Python

Weight: 1

Project over - took place from Dec 18, 2023 6:00 AM to Dec 19, 2023 6:00 AM

An auto review will be launched at the deadline

In a nutshell…

- **Auto QA review:** 7.15/84 mandatory & 0.0/51 optional
- **Altogether: 8.51%**
  - Mandatory: 8.51%
  - Optional: 0.0%
  - Calculation: 8.51% + (8.51% * 0.0%) == **8.51%**

Resources

Read or watch:

- Errors and Exceptions (/rltoken/Yj7sDOzmKwICSHr7WEAW3A)
- Learn to Program 11 Static & Exception Handling (/rltoken/xASzXarhF1sBhzYkJ14LvQ) (*starting at minute 7*)

Learning Objectives

At the end of this project, you are expected to be able to explain to anyone (/rltoken/ER6JlfkhcpsfFWZNN_BBvg), **without the help of Google**:

General

- Why Python programming is awesome
- 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

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`

Tasks

0. Safe list printing

mandatory

Score: 65.0% (*Checks completed: 100.0%*)

Write a function that prints `x` elements of a list.

- Prototype: `def safe_print_list(my_list=[], x=0):`
- `my_list` can contain any type (integer, string, etc.)
- All elements must be printed on the same line followed by a new line.
- `x` represents the number of elements to print
- `x` can be bigger than the length of `my_list`
- Returns the real number of elements printed

- You have to use `try: / except:`
- You are not allowed to import any module
- You are not allowed to use `len()`

```
guillaume@ubuntu:~/0x05$ cat 0-main.py
#!/usr/bin/python3
safe_print_list = __import__('0-safe_print_list').safe_print_list

my_list = [1, 2, 3, 4, 5]

nb_print = safe_print_list(my_list, 2)
print("nb_print: {:d}".format(nb_print))
nb_print = safe_print_list(my_list, len(my_list))
print("nb_print: {:d}".format(nb_print))
nb_print = safe_print_list(my_list, len(my_list) + 2)
print("nb_print: {:d}".format(nb_print))

guillaume@ubuntu:~/0x05$ ./0-main.py
12
nb_print: 2
12345
nb_print: 5
12345
nb_print: 5
guillaume@ubuntu:~/0x05$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x05-python-exceptions`
- File: `0-safe_print_list.py`

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

1. Safe printing of an integers list

mandatory

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

Write a function that prints an integer with `"{:d}".format()`.

- Prototype: `def safe_print_integer(value):`
- `value` can be any type (integer, string, etc.)
- The integer should be printed followed by a new line
- Returns `True` if `value` has been correctly printed (it means the `value` is an integer)
- Otherwise, returns `False`
- You have to use `try: / except:`
- You have to use `"{:d}".format()` to print as integer
- You are not allowed to import any module
- You are not allowed to use `type()`

```
guillaume@ubuntu:~/0x05$ cat 1-main.py
#!/usr/bin/python3
safe_print_integer = __import__('1-safe_print_integer').safe_print_integer

value = 89
has_been_print = safe_print_integer(value)
if not has_been_print:
    print("{} is not an integer".format(value))

value = -89
has_been_print = safe_print_integer(value)
if not has_been_print:
    print("{} is not an integer".format(value))

value = "School"
has_been_print = safe_print_integer(value)
if not has_been_print:
    print("{} is not an integer".format(value))

guillaume@ubuntu:~/0x05$ ./1-main.py
89
-89
School is not an integer
guillaume@ubuntu:~/0x05$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x05-python-exceptions`
- File: `1-safe_print_integer.py`

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

2. Print and count integers

mandatory

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

Write a function that prints the first `x` elements of a list and only integers.

- Prototype: `def safe_print_list_integers(my_list=[], x=0):`
- `my_list` can contain any type (integer, string, etc.)
- All integers have to be printed on the same line followed by a new line - other type of value in the list must be skipped (in silence).
- `x` represents the number of elements to access in `my_list`
- `x` can be bigger than the length of `my_list` - if it's the case, an exception is expected to occur
- Returns the real number of integers printed
- You have to use `try: / except:`
- You have to use `"{:d}".format()` to print an integer
- You are not allowed to import any module
- You are not allowed to use `len()`

(/)

```
guillaume@ubuntu:~/0x05$ cat 2-main.py
#!/usr/bin/python3
safe_print_list_integers = \
    __import__('2-safe_print_list_integers').safe_print_list_integers

my_list = [1, 2, 3, 4, 5]

nb_print = safe_print_list_integers(my_list, 2)
print("nb_print: {:d}".format(nb_print))

my_list = [1, 2, 3, "School", 4, 5, [1, 2, 3]]
nb_print = safe_print_list_integers(my_list, len(my_list))
print("nb_print: {:d}".format(nb_print))

nb_print = safe_print_list_integers(my_list, len(my_list) + 2)
print("nb_print: {:d}".format(nb_print))

guillaume@ubuntu:~/0x05$ ./2-main.py
12
nb_print: 2
12345
nb_print: 5
12345Traceback (most recent call last):
  File "./2-main.py", line 14, in <module>
    nb_print = safe_print_list_integers(my_list, len(my_list) + 2)
  File "/0x05/2-safe_print_list_integers.py", line 7, in safe_print_list_integers
    print("{:d}".format(my_list[i]), end="")
IndexError: list index out of range
guillaume@ubuntu:~/0x05$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x05-python-exceptions`
- File: `2-safe_print_list_integers.py`

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

3. Integers division with debug

mandatory

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

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:~/0x05$ 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:~/0x05$ ./3-main.py
Inside result: 6.0
12 / 2 = 6.0
Inside result: None
12 / 0 = None
guillaume@ubuntu:~/0x05$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x05-python-exceptions`
- File: `3-safe_print_division.py`

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

4. Divide a list

mandatory

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

Write a function that divides element by element 2 lists.

- Prototype: `def list_division(my_list_1, my_list_2, list_length):`
- `my_list_1` and `my_list_2` can contain any type (integer, string, etc.)
- `list_length` can be bigger than the length of both lists
- Returns a new list (length = `list_length`) with all divisions
- If 2 elements can't be divided, the division result should be equal to `0`
- If an element is not an integer or float:
  - print: `wrong type`
- If the division can't be done (`/0`):
  - print: `division by 0`
- If `my_list_1` or `my_list_2` is too short
  - print: `out of range`
- You have to use `try: / except: / finally:`
- You are not allowed to import any module

```
guillaume@ubuntu:~/0x05$ cat 4-main.py
#!/usr/bin/python3
list_division = __import__('4-list_division').list_division

my_l_1 = [10, 8, 4]
my_l_2 = [2, 4, 4]
result = list_division(my_l_1, my_l_2, max(len(my_l_1), len(my_l_2)))
print(result)

print("--")

my_l_1 = [10, 8, 4, 4]
my_l_2 = [2, 0, "H", 2, 7]
result = list_division(my_l_1, my_l_2, max(len(my_l_1), len(my_l_2)))
print(result)

guillaume@ubuntu:~/0x05$ ./4-main.py
[5.0, 2.0, 1.0]
--
division by 0
wrong type
out of range
[5.0, 0, 0, 2.0, 0]
guillaume@ubuntu:~/0x05$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x05-python-exceptions`
- File: `4-list_division.py`

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

5. Raise exception

mandatory

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

Write a function that raises a type exception.

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

```
guillaume@ubuntu:~/0x05$ cat 5-main.py
#!/usr/bin/python3
raise_exception = __import__('5-raise_exception').raise_exception

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

guillaume@ubuntu:~/0x05$ ./5-main.py
Exception raised
guillaume@ubuntu:~/0x05$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x05-python-exceptions`
- File: `5-raise_exception.py`

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

6. Raise a message

mandatory

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

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:~/0x05$ cat 6-main.py
#!/usr/bin/python3
raise_exception_msg = __import__('6-raise_exception_msg').raise_exception_msg

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

guillaume@ubuntu:~/0x05$ ./6-main.py
C is fun
guillaume@ubuntu:~/0x05$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x05-python-exceptions`
- File: `6-raise_exception_msg.py`

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

7. Safe integer print with error message

#advanced

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

Write a function that prints an integer.

- Prototype: `def safe_print_integer_err(value):`
- `value` can be any type (integer, string, etc.)
- The integer should be printed followed by a new line
- Returns `True` if `value` has been correctly printed (it means the `value` is an integer)
- Otherwise, returns `False` and prints in `stderr` the error precede by `Exception:`
- You have to use `try: / except:`
- You have to use `"{:d}".format()` to print as integer
- You are not allowed to use `type()`

```
guillaume@ubuntu:~/0x05$ cat 100-main.py
#!/usr/bin/python3
safe_print_integer_err = \
    __import__('100-safe_print_integer_err').safe_print_integer_err

value = 89
has_been_print = safe_print_integer_err(value)
if not has_been_print:
    print("{} is not an integer".format(value))

value = -89
has_been_print = safe_print_integer_err(value)
if not has_been_print:
    print("{} is not an integer".format(value))

value = "School"
has_been_print = safe_print_integer_err(value)
if not has_been_print:
    print("{} is not an integer".format(value))

guillaume@ubuntu:~/0x05$ ./100-main.py
89
-89
Exception: Unknown format code 'd' for object of type 'str'
School is not an integer
guillaume@ubuntu:~/0x05$ ./100-main.py 2> /dev/null
89
-89
School is not an integer
guillaume@ubuntu:~/0x05$
```

Repo:

- GitHub repository: `alx-higher_level_programming`

- Directory: `0x05-python-exceptions`
- File: `100-safe_print_integer_err.py`

(/)

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

8. Safe function

#advanced

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

Write a function that executes a function safely.

- Prototype: `def safe_function(fct, *args):`
- You can assume `fct` will be always a pointer to a function
- Returns the result of the function,
- Otherwise, returns `None` if something happens during the function and prints in `stderr` the error precede by `Exception:`
- You have to use `try: / except:`

(7)

```
guillaume@ubuntu:~/0x05$ cat 101-main.py
#!/usr/bin/python3
safe_function = __import__('101-safe_function').safe_function


def my_div(a, b):
    return a / b

result = safe_function(my_div, 10, 2)
print("result of my_div: {}".format(result))

result = safe_function(my_div, 10, 0)
print("result of my_div: {}".format(result))


def print_list(my_list, len):
    i = 0
    while i < len:
        print(my_list[i])
        i += 1
    return len

result = safe_function(print_list, [1, 2, 3, 4], 10)
print("result of print_list: {}".format(result))

guillaume@ubuntu:~/0x05$ ./101-main.py
result of my_div: 5.0
Exception: division by zero
result of my_div: None
1
2
3
4
Exception: list index out of range
result of print_list: None
guillaume@ubuntu:~/0x05$ ./101-main.py 2> /dev/null
result of my_div: 5.0
result of my_div: None
1
2
3
4
result of print_list: None
guillaume@ubuntu:~/0x05$
```

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x05-python-exceptions`
- File: `101-safe_function.py`

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

9. ByteCode -> Python #4 (/)

#advanced

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

Write the Python function `def magic_calculation(a, b):` that does exactly the same as the following Python bytecode:

(/)

```
3           0 LOAD_CONST               1 (0)
            3 STORE_FAST               2 (result)

4           6 SETUP_LOOP              94 (to 103)
            9 LOAD_GLOBAL              0 (range)
           12 LOAD_CONST               2 (1)
           15 LOAD_CONST               3 (3)
           18 CALL_FUNCTION            2 (2 positional, 0 keyword pair)
           21 GET_ITER
      >>   22 FOR_ITER                77 (to 102)
           25 STORE_FAST               3 (i)

5          28 SETUP_EXCEPT            49 (to 80)

6          31 LOAD_FAST                3 (i)
           34 LOAD_FAST                0 (a)
           37 COMPARE_OP               4 (>)
           40 POP_JUMP_IF_FALSE       58

7          43 LOAD_GLOBAL              1 (Exception)
           46 LOAD_CONST               4 ('Too far')
           49 CALL_FUNCTION            1 (1 positional, 0 keyword pair)
           52 RAISE_VARARGS            1
           55 JUMP_FORWARD            18 (to 76)

9     >>   58 LOAD_FAST                2 (result)
           61 LOAD_FAST                0 (a)
           64 LOAD_FAST                1 (b)
           67 BINARY_POWER
           68 LOAD_FAST                3 (i)
           71 BINARY_TRUE_DIVIDE
           72 INPLACE_ADD
           73 STORE_FAST               2 (result)
      >>   76 POP_BLOCK
           77 JUMP_ABSOLUTE           22

10    >>   80 POP_TOP
           81 POP_TOP
           82 POP_TOP

11         83 LOAD_FAST                1 (b)
           86 LOAD_FAST                0 (a)
           89 BINARY_ADD
           90 STORE_FAST               2 (result)

12         93 BREAK_LOOP
           94 POP_EXCEPT
           95 JUMP_ABSOLUTE           22
           98 END_FINALLY
           99 JUMP_ABSOLUTE           22
      >>  102 POP_BLOCK

13    >>  103 LOAD_FAST                2 (result)
          106 RETURN_VALUE
```

- Tip: Python bytecode (/rltoken/-eivu0w172OUPm-iCeKgtw)

(/)

Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x05-python-exceptions`
- File: `102-magic_calculation.py`

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

10. CPython #2: PyFloatObject

#advanced

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

Create three C functions that print some basic info about Python lists, Python bytes an Python float objects.

Python lists:

- Prototype: `void print_python_list(PyObject *p);`
- Format: see example
- If `p` is not a valid `PyListObject`, print an error message (see example)

Python bytes:

- Prototype: `void print_python_bytes(PyObject *p);`
- Format: see example
- Line "first X bytes": print a maximum of 10 bytes
- If `p` is not a valid `PyBytesObject`, print an error message (see example)

Python float:

- Prototype: `void print_python_float(PyObject *p);`
- Format: see example
- If `p` is not a valid `PyFloatObject`, print an error message (see example)
- Read `/usr/include/python3.4/floatobject.h`

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 -Wall -Werror -Wextra -pedantic -std=c99 -shared -Wl,-soname,libPython.so -o libPython.so -fPIC -I/usr/include/python3.4 103-python.c`
- You are not allowed to use the following macros/functions:
  - `Py_SIZE`
  - `Py_TYPE`
  - `PyList_Size`
  - `PyList_GetItem`
  - `PyBytes_AS_STRING`
  - `PyBytes_GET_SIZE`
  - `PyBytes_AsString`
  - `PyBytes_AsStringAndSize`
  - `PyFloat_AS_DOUBLE`
  - `PySequence_GetItem`
  - `PySequence_Fast_GET_SIZE`
  - `PySequence_Fast_GET_ITEM`
  - `PySequence_ITEM`
  - `PySequence_Fast_ITEMS`

NOTE:

- The python script will be launched using the `-u` option (Force `stdout` to be unbuffered).
- It is **strongly** advised to either use `setbuf(stdout, NULL);` or `fflush(stdout)` in your C functions IF you choose to use `printf`. The reason to that is that Python `s` print `and` `libC` s `printf` don't share the same buffer, and the output can appear disordered.

(/)

```
julien@ubuntu:~/CPython$ python3 --version
Python 3.4.3
julien@ubuntu:~/CPython$ gcc -Wall -Werror -Wextra -pedantic -std=c99 -shared -W
l,-soname,libPython.so -o libPython.so -fPIC -I/usr/include/python3.4 103-python.
c
julien@ubuntu:~/CPython$ cat 103-tests.py
#!/usr/bin/python3 -u

import ctypes

lib = ctypes.CDLL('./libPython.so')
lib.print_python_list.argtypes = [ctypes.py_object]
lib.print_python_bytes.argtypes = [ctypes.py_object]
lib.print_python_float.argtypes = [ctypes.py_object]
s = b"Hello"
lib.print_python_bytes(s);
b = b'\xff\xf8\x00\x00\x00\x00\x00\x00';
lib.print_python_bytes(b);
b = b'What does the \'b\' character do in front of a string literal?';
lib.print_python_bytes(b);
l = [b'Hello', b'World']
lib.print_python_list(l)
del l[1]
lib.print_python_list(l)
l = l + [4, 5, 6.0, (9, 8), [9, 8, 1024], b"School", "Betty"]
lib.print_python_list(l)
l = []
lib.print_python_list(l)
l.append(0)
lib.print_python_list(l)
l.append(1)
l.append(2)
l.append(3)
l.append(4)
lib.print_python_list(l)
l.pop()
lib.print_python_list(l)
l = ["School"]
lib.print_python_list(l)
lib.print_python_bytes(l);
f = 3.14
lib.print_python_float(f);
l = [-1.0, -0.1, 0.0, 1.0, 3.14, 3.14159, 3.14159265, 3.14159265358979323846264338
327950288419716939937510582097494459230781640628 6]
print(l)
lib.print_python_list(l);
lib.print_python_float(l);
lib.print_python_list(f);
julien@ubuntu:~/CPython$ ./103-tests.py
[.] bytes object info
  size: 5
  trying string: Hello
  first 6 bytes: 48 65 6c 6c 6f 00
[.] bytes object info
  size: 8
```

```
  trying string: ??
first 9 bytes: ff f8 00 00 00 00 00 00 00
[.] bytes object info
  size: 60
  trying string: What does the 'b' character do in front of a string literal?
  first 10 bytes: 57 68 61 74 20 64 6f 65 73 20
[*] Python list info
[*] Size of the Python List = 2
[*] Allocated = 2
Element 0: bytes
[.] bytes object info
  size: 5
  trying string: Hello
  first 6 bytes: 48 65 6c 6c 6f 00
Element 1: bytes
[.] bytes object info
  size: 5
  trying string: World
  first 6 bytes: 57 6f 72 6c 64 00
[*] Python list info
[*] Size of the Python List = 1
[*] Allocated = 2
Element 0: bytes
[.] bytes object info
  size: 5
  trying string: Hello
  first 6 bytes: 48 65 6c 6c 6f 00
[*] Python list info
[*] Size of the Python List = 8
[*] Allocated = 8
Element 0: bytes
[.] bytes object info
  size: 5
  trying string: Hello
  first 6 bytes: 48 65 6c 6c 6f 00
Element 1: int
Element 2: int
Element 3: float
[.] float object info
  value: 6.0
Element 4: tuple
Element 5: list
Element 6: bytes
[.] bytes object info
  size: 9
  trying string: School
  first 10 bytes: 48 6f 6c 62 65 72 74 6f 6e 00
Element 7: str
[*] Python list info
[*] Size of the Python List = 0
[*] Allocated = 0
[*] Python list info
[*] Size of the Python List = 1
[*] Allocated = 4
Element 0: int
[*] Python list info
```

```
[*] Size of the Python List = 5
[*] Allocated = 8
Element 0: int
Element 1: int
Element 2: int
Element 3: int
Element 4: int
[*] Python list info
[*] Size of the Python List = 4
[*] Allocated = 8
Element 0: int
Element 1: int
Element 2: int
Element 3: int
[*] Python list info
[*] Size of the Python List = 1
[*] Allocated = 1
Element 0: str
[.] bytes object info
  [ERROR] Invalid Bytes Object
[.] float object info
  value: 3.14
[-1.0, -0.1, 0.0, 1.0, 3.14, 3.14159, 3.14159265, 3.141592653589793]
[*] Python list info
[*] Size of the Python List = 8
[*] Allocated = 8
Element 0: float
[.] float object info
  value: -1.0
Element 1: float
[.] float object info
  value: -0.1
Element 2: float
[.] float object info
  value: 0.0
Element 3: float
[.] float object info
  value: 1.0
Element 4: float
[.] float object info
  value: 3.14
Element 5: float
[.] float object info
  value: 3.14159
Element 6: float
[.] float object info
  value: 3.14159265
Element 7: float
[.] float object info
  value: 3.141592653589793
[.] float object info
  [ERROR] Invalid Float Object
[*] Python list info
  [ERROR] Invalid List Object
julien@ubuntu:~/CPython$
```

(/)
Repo:

- GitHub repository: `alx-higher_level_programming`
- Directory: `0x05-python-exceptions`
- File: `103-python.c`

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