Simplon Normes:

Python

We use `pylint` to verify that your file is formatted as you wish. You can install and run `pylint` like this

```
$ sudo apt install pylint
$ pylint --rcfile=simplon.rc my_python_package my_file.py
```

`Pylint` will ignore some folder and file, like the ones starting with `test`.

```
$ ls package_name
  __init__.py  main.py  math.py  test_math.py  tests_mains.py

# Does not read test_math.py and tests_mains.py
$ pylint --rcfile=pylint_simplon package_name
```

Also ignore folders named `docs dockerfiles _build logs .env tests test`.

DocStrings

Docstring are excepted Everywhere but on modules and should be at least one line.

```
def main():
  """Required Documentation"""

class MyClass():
  """Required Documentation"""
```

```
$ head myfile.py
#/usr/bin/env python3
"""Not Required Module documentation"""

def open_a_file():
  """Required Documentation"""
```

Limits

- a file can not be longer than 160 lines.
- a file can not contain more than 5 function/class.
- a function or methods can not have more than 25 lines.
- a line can not be longer than 80 character.
- a Class can not have less than 2 methods (unless explicit in a subject).
- a function or method can not have more than 4 argument.
- a Class can not have more than 5 attributes.

- No more than 3 boolean value in if statement.
- No more 4 return in a function body.
- No more than 4 Branches.
- No more than 5 local variables.
- You will not inheritate more than 3 time.
- class can not have more than 6 public methods
- Wildcard import are NOT allowed

```python
# bad
def this_line_is_longer_than_heighty_characters_and_it_is_way_too_long_for_vti_tty_format_wh
  pass

def less_than_heigty_characters():
  pass

# bad: too many boolean in if statement
if i % 5 and i % 3 and i % 10 and i % 6:
  pass

# good
if i % 5 and i % 3:
  if i % 10 and i % 6:
    pass

# Bad: too many branches
if i == 1:
  pass
elif i == 2:
  pass
elif i == 3:
  pass
elif i == 4:
  pass
elif i == 5:
  pass
elif i == 6:
  pass
```

Indentation

Two space you will use as an indentation (You can probably set your IDE/VSCode to put two spaces with the `tab` key).

```python
def right_indent(indent):
  if indent:
    return True
  return False
```

```python
def bad_indent(ind):
    if ind:
          return True
    return False
```

Nested Block

You can't have more than three nested block:

```python
def good_nested():
  for i in range(10):
    if i % 5:
      if i % 2:
        print("Hello World")

# too many nested level
def bad_nested():
  for i in range(10):
    if i % 5:
      if i % 2:
    if True:
          print("Hello World")
```

Quote Consistency

In a file a quote (`'|"`) you will choose. In this file no other quote will be used.

```python
def wrong_quote_consistency():
  string = "Hello"
  word = 'World'
```

```python
def quote_consistency():
  string = "Hello"
  word = "World"
```

Argument, Function, Class, Methods and Attributes naming

All your constant value will use `UPPER_CASE` style.

The `snake_case` style you will use for:

- argument
- attribute
- function
- module

- variables

The `PascalCase` style you will use for:

- Class

The `camelCase` style you will use for:

- methods

```
# Good
import somemodule

CONST_VALUE = "World"

def some_func():
  variable = "toto"
  return variable

class MyClass():
  def myMethod(argument):
    self.attribute = "tata"

# Bad
import SomeModule

constvalue = "World"

def SomeFunc():
  Variable = "toto"
  return Variable

class myclass():
  def my_method(WrongArgument):
    self.AttributeBad = "tata"
    if wrongArgument == 0:
      return 0
    if wrongArgument == 1:
      return 1
    if wrongArgument == 2:
      return 1
    if wrongArgument == 3:
      return 3
    return 5
```

Return Values

If your function return a value in branches, every branches should return.

```
def not_enought_return(integer):
  if integer == 5:
    return 0
  elif integer == 3:
    return 1

def return_in_all_braches(integer)
  if integer == 5:
    return 0
  elif integer == 3:
    return 1
  return -1
```

Variable Naming and Usage

All variable should have a named between 2 and 30 characters, excepted for `i, j, k, x, y, _` which can be used as index variables. all declared variable should be used or indicated otherwise.

```
# Bad
def some_func(attr, notused):
  again_not_used, value = attr
  print(value)

# good
def some_func(attr, _):
  unused_vtuple, value = attr
  print(value)
```

End Of Line format

Your end of line should always be `LF` ones (linux mode).

```
$ cat -e good.py
def main():$
  my_var = "toto"$

$ cat -e bad.py
def main():^M$
  my_var = "toto"^M$
```

Score

When running `pylint` it gives you a score. For easy maintenance I would suggest to try to keep it always higher than 80%.

Remember that now, we ask you to 100% when finishing a projets. If you have any question. Please ask to `@lumy` first.