



Python & ML - Módulo 02

Básicos 3

Resumen: Sigamos practicando con ejercicios de programación en Python más avanzados. Destino: Decoradores, lambda, gestor de contexto y paquete de construcción.

Capítulo I

Instrucciones generales


- La versión de Python que se recomienda utilizar es la 3.7, puedes comprobar la versión de Python con el siguiente comando: `python -V`
- La norma: durante esta piscina, se recomienda seguir los [estándares PEP 8](#), aunque no es obligatorio. Puedes instalar [pycodestyle](#) que es una herramienta para comprobar tu código Python.
- La función `eval` nunca está permitida.
- Los ejercicios están ordenados del más fácil al más difícil.
- Tus ejercicios van a ser evaluados por otras personas, así que asegúrate de que los nombres de tus variables y funciones sean apropiados y corteses.
- Tu manual es internet.
- Te animamos a crear programas de prueba para tu proyecto, aunque este trabajo **no tendrá que ser presentado y no será calificado**. Te dará la oportunidad de poner a prueba fácilmente tu trabajo y el de tus compañeros/as. Estos tests te serán especialmente útiles durante tu evaluación. De hecho, durante la evaluación, eres libre de utilizar tus pruebas y/o las pruebas del compañero/a al que estás evaluando.

Índice general

I.	Instrucciones generales	1
II.	Ejercicio 00	3
III.	Ejercicio 01	5
IV.	Ejercicio 02	7
V.	Ejercicio 03	10
VI.	Ejercicio 04	12
VII.	Ejercicio 05	14

Capítulo II

Ejercicio 00

	Ejercicio : 00
Map, filter, reduce	
Directorio de entrega : <code>ex00/</code>	
Archivos a entregar : <code>ft_map.py</code> , <code>ft_filter.py</code> , <code>ft_reduce.py</code>	
Funciones prohibidas : Ninguna	

Objetivo

El objetivo del ejercicio es trabajar con las funciones built-in `map`, `filter` and `reduce`.

Instrucciones

Implementa las funciones `ft_map`, `ft_filter` y `ft_reduce`. Tómate tu tiempo para comprender los casos de uso de las funciones built-in (`map` y `filter`) y la función `reduce` del módulo `functools`. No se espera que escribas clases específicas para crear objetos `ft_map`, `ft_filter` o `ft_reduce`, echa un vistazo a los ejemplos para saber qué hacer.

Aquí las características de las funciones:

```
def ft_map(function_to_apply, iterable):
    """Map the function to all elements of the iterable.
    Args:
        function_to_apply: a function taking an iterable.
        iterable: an iterable object (list, tuple, iterator).
    Return:
        An iterable.
        None if the iterable can not be used by the function.
    """
    # ... Your code here ...

def ft_filter(function_to_apply, iterable):
    """Filter the result of function apply to all elements of the iterable.
    Args:
        function_to_apply: a function taking an iterable.
        iterable: an iterable object (list, tuple, iterator).
    Return:
        An iterable.
        None if the iterable can not be used by the function.
    """
    # ... Your code here ...

def ft_reduce(function_to_apply, iterable):
    """Apply function of two arguments cumulatively.
    Args:
        function_to_apply: a function taking an iterable.
        iterable: an iterable object (list, tuple, iterator).
    Return:
        A value, of same type of elements in the iterable parameter.
        None if the iterable can not be used by the function.
    """
    # ... Your code here ...
```

Ejemplos

```
# Example 1:
x = [1, 2, 3, 4, 5]
ft_map(lambda dum: dum + 1, x)
# Output:
<generator object ft_map at 0x7f708faab7b0> # The adress will be different

list(ft_map(lambda t: t + 1, x))
# Output:
[2, 3, 4, 5, 6]

# Example 2:
ft_filter(lambda dum: not (dum % 2), x)
# Output:
<generator object ft_filter at 0x7f709c777d00> # The adress will be different


list(ft_filter(lambda dum: not (dum % 2), x))
# Output:
[2, 4]

# Example 3:
lst = ['H', 'e', 'l', 'l', 'o', ' ', 'w', 'o', 'r', 'l', 'd']
ft_reduce(lambda u, v: u + v, lst)
# Output:
"Hello world"
```

Se espera que generes una excepción para las funciones similar a las excepciones de map, filter y reduce cuando se dan parámetros incorrectos (pero no es necesario reproducir exactamente los mismos mensajes de excepción).

Capítulo III

Ejercicio 01

	Ejercicio : 01
¿args y kwargs?	
Directorio de entrega : <i>ex01/</i>	
Archivos a entregar : main.py	
Funciones prohibidas : Ninguna	

Objetivo

El objetivo del ejercicio es descubrir y manipular argumentos ***args** y ****kwargs**.

Instrucciones

En este ejercicio tienes que implementar una función llamada **what_are_the_vars** que devuelva una instancia de la clase **ObjectC**.

Los atributos **ObjectC** se establecen a través de los parámetros recibidos durante la instanciación. Tendrás que modificar la ‘instancia’ **ObjectC**, **NO** la clase.

Deberías echar un vistazo a las funciones integradas **getattr**, **setattr**.

```
def what_are_the_vars(...):
    """
    ...
    """
    # ... Your code here ...

class ObjectC(object):
    def __init__(self):
        # ... Your code here ...

def doom_printer(obj):
    if obj is None:
        print("ERROR")
        print("end")
        return
    for attr in dir(obj):
        if attr[0] != '_':
            value = getattr(obj, attr)
            print("{}: {}".format(attr, value))
    print("end")


if __name__ == "__main__":
    obj = what_are_the_vars(7)
    doom_printer(obj)
    obj = what_are_the_vars(None, [])
    doom_printer(obj)
    obj = what_are_the_vars("ft_lol", "Hi")
    doom_printer(obj)
    obj = what_are_the_vars()
    doom_printer(obj)
    obj = what_are_the_vars(12, "Yes", [0, 0, 0], a=10, hello="world")
    doom_printer(obj)
    obj = what_are_the_vars(42, a=10, var_0="world")
    doom_printer(obj)
    obj = what_are_the_vars(42, "Yes", a=10, var_2="world")
    doom_printer(obj)
```

Ejemplos

```
$> python main.py
var_0: 7
end
var_0: None
var_1: []
end
var_0: ft_lol
var_1: Hi
end
end
a: 10
hello: world
var_0: 12
var_1: Yes
var_2: [0, 0, 0]
end
ERROR
end
a: 10
var_0: 12
var_1: Yes
var_2: world
end
```

Capítulo IV

Ejercicio 02

	Ejercicio : 02
El registrador	
Directorio de entrega : <i>ex02/</i>	
Archivos a entregar : logger.py	
Funciones prohibidas : Ninguna	

Objetivo

En este ejercicio, aprenderás sobre decoradores, y no estamos hablando de la decoración de tu habitación. El `@log` escribirá información sobre la función decorada en un archivo `machine.log`.

Instrucciones

Tienes que crear el decorador de registro en el mismo archivo. Presta atención a las diferentes acciones registradas en la llamada de cada método. Puedes observar que el nombre de usuario de la variable de entorno se escribe en el archivo de registro.


```

import time
from random import randint
import os
#... your definition of log decorator...
class CoffeeMachine():
    water_level = 100
    @log
    def start_machine(self):
        if self.water_level > 20:
            return True
        else:
            print("Please add water!")
            return False

    @log
    def boil_water(self):
        return "boiling..."

    @log
    def make_coffee(self):
        if self.start_machine():
            for _ in range(20):
                time.sleep(0.1)
                self.water_level -= 1
                print(self.boil_water())
                print("Coffee is ready!")

    @log
    def add_water(self, water_level):
        time.sleep(randint(1, 5))
        self.water_level += water_level
        print("Blub blub blub...")
if __name__ == "__main__":
    machine = CoffeeMachine()
    for i in range(0, 5):
        machine.make_coffee()
    machine.make_coffee()
    machine.add_water(70)

```

Ejemplos

```

$> python logger.py
boiling...
Coffee is ready!
boiling...
Coffee is ready!
boiling...
Coffee is ready!
boiling...
Coffee is ready!
Please add water!
Please add water!
Blub blub blub...
$>

```

```

$> cat machine.log
(cmaxime)Running: Start Machine [ exec-time = 0.001 ms ]
(cmaxime)Running: Boil Water [ exec-time = 0.005 ms ]
(cmaxime)Running: Make Coffee [ exec-time = 2.499 s ]
(cmaxime)Running: Start Machine [ exec-time = 0.002 ms ]
(cmaxime)Running: Boil Water [ exec-time = 0.005 ms ]
(cmaxime)Running: Make Coffee [ exec-time = 2.618 s ]
(cmaxime)Running: Start Machine [ exec-time = 0.003 ms ]
(cmaxime)Running: Boil Water [ exec-time = 0.004 ms ]
(cmaxime)Running: Make Coffee [ exec-time = 2.676 s ]
(cmaxime)Running: Start Machine [ exec-time = 0.003 ms ]
(cmaxime)Running: Boil Water [ exec-time = 0.004 ms ]


```

```
(cmaxime)Running: Make Coffee [ exec-time = 2.648 s ]  
(cmaxime)Running: Start Machine [ exec-time = 0.011 ms ]  
(cmaxime)Running: Make Coffee [ exec-time = 0.029 ms ]  
(cmaxime)Running: Start Machine [ exec-time = 0.009 ms ]  
(cmaxime)Running: Make Coffee [ exec-time = 0.024 ms ]  
(cmaxime)Running: Add Water [ exec-time = 5.026 s ]  
$>
```

Presta atención, la longitud entre “:” y “[“ es 20]. Escribe las conclusiones correspondientes en esta parte de una entrada de registro.

Capítulo V

Ejercicio 03

	Ejercicio : 03
Cuestiones Json	
Directorio de entrega : <i>ex03/</i>	
Archivos a entregar : <code>csvreader.py</code>	
Funciones prohibidas : Ninguna	

Objetivo

El objetivo de este ejercicio es implementar un gestor de contexto como una clase. Por ello, te recomendamos encarecidamente que investigues sobre el gestor de contexto.

Instrucciones

Implementa una clase `CsvReader` que abra, lea y analice un archivo CSV. Esta clase será entonces un gestor de contexto como clase. Para crearla, tu clase necesita algunos métodos integrados:

- `__init__`,
- `__enter__`,
- `__exit__`.

Es obligatorio cerrar el fichero una vez finalizado el proceso. Se espera que gestionen correctamente un archivo CSV mal formateado (es decir, que gestionen la excepción):

- falta de correspondencia entre el número de campos y el número de registros,
- registros de diferente longitud.

```

class CsvReader():
    def __init__(self, filename=None, sep=',', header=False, skip_top=0, skip_bottom=0):
        # ... Your code here ...
    def __enter__(...):
        # ... Your code here ...

    def __exit__(...):
        # ... Your code here ...

    def getdata(self):
        """ Retrieves the data/records from skip_top to skip bottom.
        Return:
            nested list (list(list, list, ...)) representing the data.
        """
        # ... Your code here ...
    def getheader(self):
        """ Retrieves the header from csv file.
        Returns:
            list: representing the data (when self.header is True).
            None: (when self.header is False).
        """
        # ... Your code here ...

```

Un archivo CSV (Comma-Separated Values) es un archivo de texto delimitado que utiliza una coma para separar los valores. Por lo tanto, el separador de campos (o delimitador) suele ser una coma (,) pero con tu gestor de contexto debes ofrecer la posibilidad de cambiar este parámetro. Los parámetros `skip_top` y `skip_bottom` permiten decidir si la clase omite las líneas de la parte superior e inferior del archivo. También se debería poder mantener la primera línea como cabecera si `encabezamiento` (`header`) es `True`. El archivo no debe estar dañado (ya sea una línea con demasiados valores o una línea con muy pocos valores), de lo contrario devuelve `None`.

Tienes que manejar el caso de archivo no encontrado (`file not found`).

Se espera que implementes dos métodos:

- `getdata()`,
- `getheader()`.

```


from csvreader import CsvReader
if __name__ == "__main__":
    with CsvReader('good.csv') as file:
        data = file.getdata()
        header = file.getheader()

from csvreader import CsvReader
if __name__ == "__main__":
    with CsvReader('bad.csv') as file:
        if file == None:
            print("File is corrupted")

```

Capítulo VI

Ejercicio 04

	Ejercicio : 04
MiniPack	
Directorio de entrega : <i>ex04/</i>	
Archivos a entregar : <code>build.sh</code> , <code>*.py</code> , <code>*.md</code> , <code>*.cfg</code> , <code>*.txt</code>	
Funciones prohibidas : Ninguna	

Objetivo

El objetivo del ejercicio es aprender a construir un paquete y comprender la magnificencia de [PyPi](#).

Instrucciones

Tienes que crear un paquete llamado `my_minipack`.



RTFM

Tendrá 2 **módulos**:

- la barra de progreso (module00 ex10), que debe importarse mediante `import my_minipack.progress`
- el registrador (module02 ex02), que debe importarse mediante `import my_minipack.logger`.

El paquete se instalará mediante pip utilizando uno de los siguientes comandos (ambos deberían funcionar):

```
$> pip install ./dist/my_minipack-1.0.0.tar.gz
$> pip install ./dist/my_minipack-1.0.0-py3-none-any.whl
```

A partir de los siguientes comandos de terminal y las salidas correspondientes, extrae la conclusión necesaria.

```
$> python -m venv tmp_env && source tmp_env/bin/activate
(tmp_env) > pip list
# Output
Package      Version
-----
pip          19.0.3
setuptools 40.8.0

(tmp_env) $> cd ex04/ && bash build.sh
# Output ... No specific verbose expected, do as you wish ...
...
(tmp_env) $> ls dist
# Output
my_minipack-1.0.0-py3-none-any.whl my_minipack-1.0.0.tar.gz

(tmp_env) $> pip list
# Output
Package      Version
-----
my-minipack 1.0.0
pip          21.0.1 # the last version at the time
setuptools 54.2.0 # the last version at the time
wheel       0.36.2 # the last version at the time
(tmp_env) $> pip show -v my_minipack
# Output (minimum metadata asked)
Name: my-minipack
Version: 1.0.0
Summary: Howto create a package in python.
Home-page: None
Author: mdavid
Author-email: mdavid@student.42.fr
License: GPLv3
Location: [PATH TO BOOTCAMP PYTHON]/module02/tmp_env/lib/python3.7/site-packages
Requires:
Required-by:
Metadata-Version: 2.1
Installer: pip
Classifiers:
Development Status :: 3 - Alpha
Intended Audience :: Developers
Intended Audience :: Students
Topic :: Education
Topic :: HowTo
Topic :: Package
License :: OSI Approved :: GNU General Public License v3 (GPLv3)
Programming Language :: Python :: 3
Programming Language :: Python :: 3 :: Only
(tmp_env) $>
```


Añade también un archivo LICENSE.md (puedes elegir una licencia real o una falsa, no importa) y un archivo README que contenga una breve documentación sobre tu librería. El script 'build.sh' actualiza 'pip' y construye los paquetes de distribución en 'wheel' y el formato '.egg'.



Puedes asegurarte de que el paquete se ha instalado correctamente ejecutando el comando `pip list` que muestra la lista de paquetes instalados y comprobar los metadatos del paquete con `pip show -v my_minipack`. Por supuesto, no reproduzcas exactamente los mismos metadatos, cambia la información del autor, modifica el resumen de los elementos Tema y Audiencia si así lo deseas.

Capítulo VII

Ejercicio 05

	Ejercicio : 05
TinyStatistician	
Directorio de entrega : <code>ex05/</code>	
Archivos a entregar : <code>TinyStatistician.py</code>	
Funciones prohibidas : Cualquier función que le calcule la media (<code>mean</code>), mediana (<code>median</code>), cuartiles (<code>quartiles</code>), varianza (<code>var</code>) o desviación estándar (<code>std</code>).	

Objetivo

Iniciación a nociones muy básicas de estadística.

Instrucciones

Crea una clase llamada `TinyStatistician` que implemente los siguientes métodos:

- `mean(x)`: calcula la media de una lista o matriz `x` no vacía dada, utilizando un bucle `for`. El método devuelve la media como un `float`, en caso contrario `None` si `x` es una lista o matriz vacía. Dado un vector `x` de dimensión $m \times 1$, la fórmula matemática de su media es:

$$\mu = \frac{\sum_{i=1}^m x_i}{m}$$

- `median(x)`: calcula la mediana de una lista o matriz `x` no vacía dada. El método devuelve la mediana como un `float`, en caso contrario `None` si `x` es una lista o matriz vacía.
- `quartiles(x)`: calcula los cuartiles 1^{st} y 3^{rd} de una matriz `x` no vacía dada. El método devuelve el cuartil como un `float`, en caso contrario `None` si `x` es una lista o matriz vacía.

- **var(x)**: calcula la varianza de una lista o matriz **x** no vacía dada, utilizando un bucle for. El método devuelve la varianza como un float, en caso contrario **None** si **x** es una lista o matriz vacía. Dado un vector **x** de dimensión $m \times 1$, la fórmula matemática de su varianza es:

$$\sigma^2 = \frac{\sum_{i=1}^m (x_i - \mu)^2}{m} = \frac{\sum_{i=1}^m [x_i - (\frac{1}{m} \sum_{j=1}^m x_j)]^2}{m}$$

- **std(x)** : calcula la desviación estándar de una lista o matriz **x** no vacía dada, utilizando un bucle for. El método devuelve la desviación estándar como un float, en caso contrario **None** si **x** es una lista o matriz vacía. Dado un vector **x** de dimensión $m \times 1$, la fórmula matemática de su desviación estándar es:

$$\sigma = \sqrt{\frac{\sum_{i=1}^m (x_i - \mu)^2}{m}} = \sqrt{\frac{\sum_{i=1}^m [x_i - (\frac{1}{m} \sum_{j=1}^m x_j)]^2}{m}}$$

Todos los métodos reciben una **lista** o un **numpy.ndarray** como parámetro.

Asumimos que todas las entradas tienen un formato correcto, es decir, una lista o matriz de tipo numérico o una lista o matriz vacía. No tienes que proteger tus funciones contra errores de entrada.

Ejemplos

```
from TinyStatistician import TinyStatistician
tstat = TinyStatistician()
a = [1, 42, 300, 10, 59]
tstat.mean(a)
# Expected result: 82.4
tstat.median(a)
# Expected result: 42.0
tstat.quartile(a)
# Expected result: [10.0, 59.0]
tstat.var(a)
# Expected result: 12279.439999999999
tstat.std(a)
# Expected result: 110.8126346586862
```


Reconocimientos

Los módulos Python & ML son el resultado de un trabajo colectivo, al que queremos dar las gracias:

- Maxime Chouluka (cmaxime),
- Pierre Peigné (ppeigne, pierre@42ai.fr),
- Matthieu David (mdavid, matthieu@42ai.fr),
- Quentin Feuillade-Montixi (qfeuilla, quentin@42ai.fr)

que supervisó la creación, la mejora y esta transcripción.

- Louis Develle (ldevelle, louis@42ai.fr)
- Augustin Lopez (aulopez)
- Luc Lenotre (llenotre)
- Owen Roberts (oroberts)
- Thomas Flahault (thflahau)
- Amric Trudel (amric@42ai.fr)
- Baptiste Lefeuvre (blefeuvr@student.42.fr)
- Mathilde Boivin (mboivin@student.42.fr)
- Tristan Duquesne (tduquesn@student.42.fr)

por su inversión en la creación y desarrollo de estos módulos.

- Barthélémy Leveque (bleveque@student.42.fr)
- Remy Oster (roster@student.42.fr)
- Quentin Bragard (qbragard@student.42.fr)
- Marie Dufourq (madufour@student.42.fr)
- Adrien Vardon (advardon@student.42.fr)

que realizaron las pruebas beta de la primera versión de los módulos de aprendizaje automático.