CIBERTEC

PYTHON FUNDAMENTALS FOR DATA SCIENCE

Capítulo 2: Lenguaje PYTHON

OBJETIVOS

- Entender la sintaxis del lenguaje de programación y su aplicación en la Ciencia de Datos.

AGENDA

1. Python en Data Science.
2. OOP.
3. Manejo de errores.
4. List Comprehensions.
5. Funciones Lambda.
6. Map/Reduce.
7. Generadores.
8. Decoradores.

1. PYTHON EN DATA SCIENCE

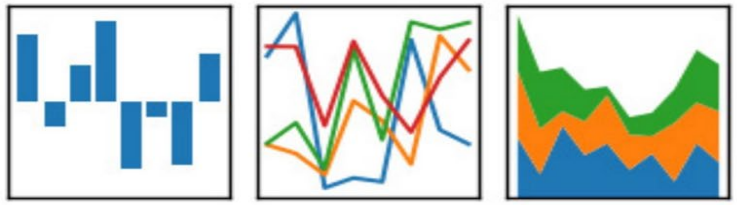
- Exploración de datos
- Visualización
- Machine learning
- NLP
- Computer Vision
- Deep learning
- Big data

1. PYTHON EN DATA SCIENCE

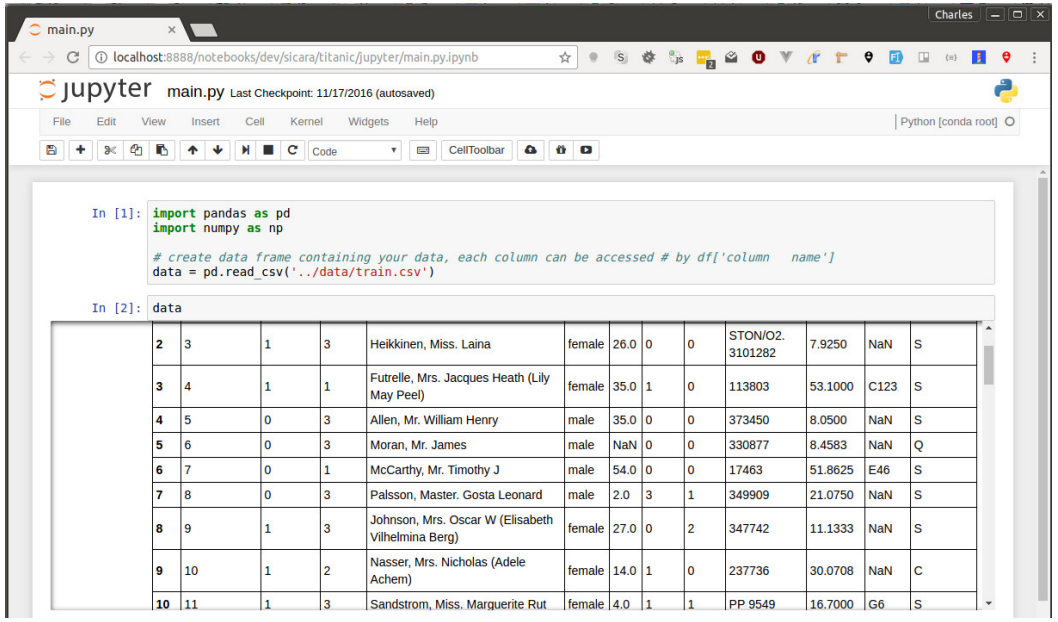
- Exploración de datos -

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



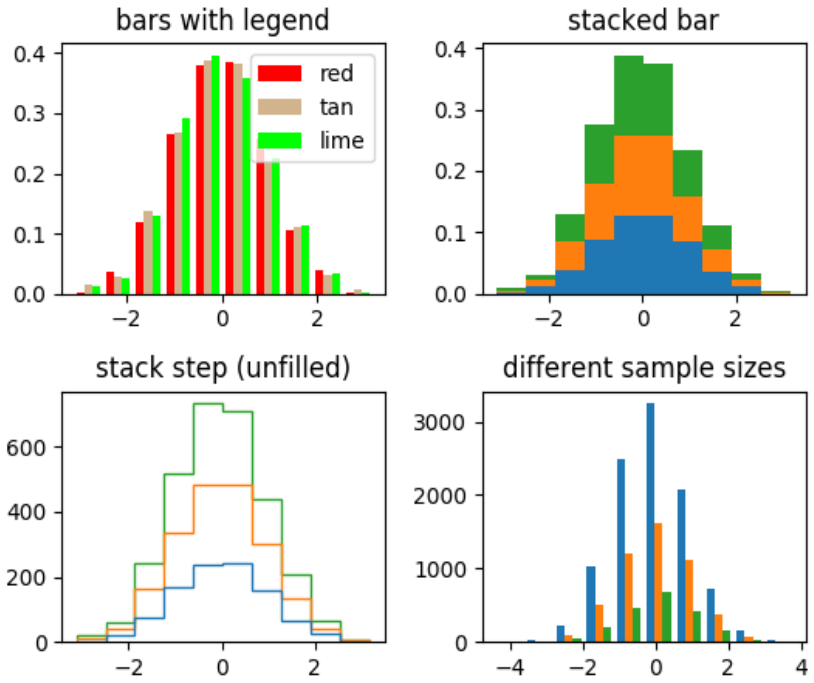
jupyter

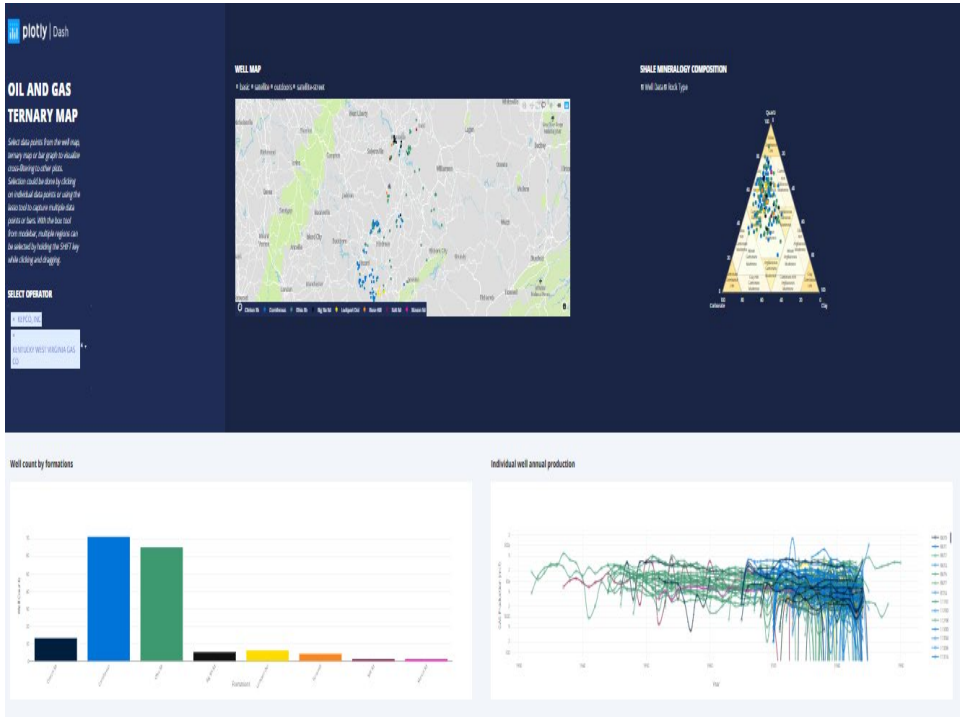


2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C
10	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.7000	G6	S

1. PYTHON EN DATA SCIENCE

- Visualización -



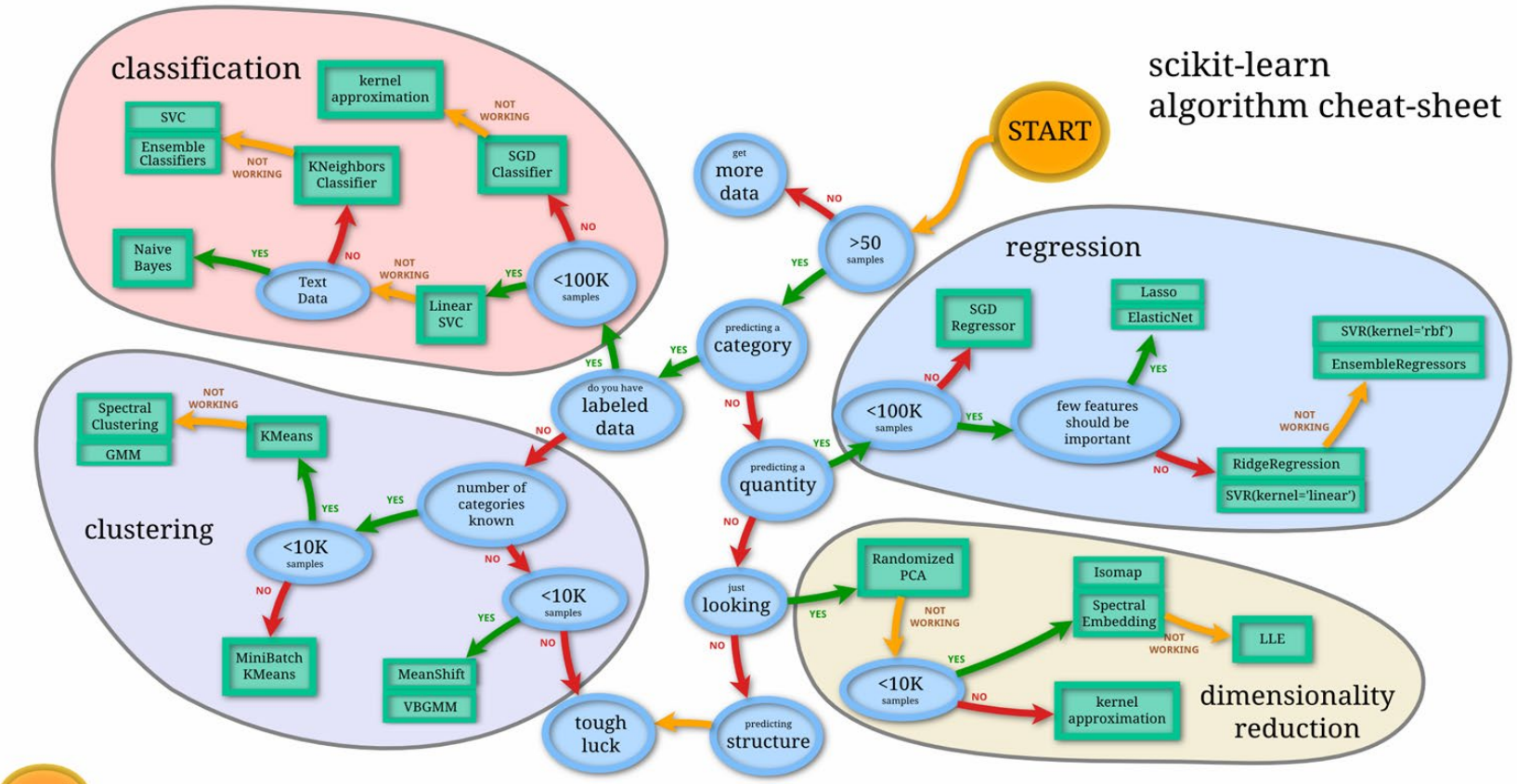


matplotlib

plotly | Dash

1. PYTHON EN DATA SCIENCE

- Machine Learning -

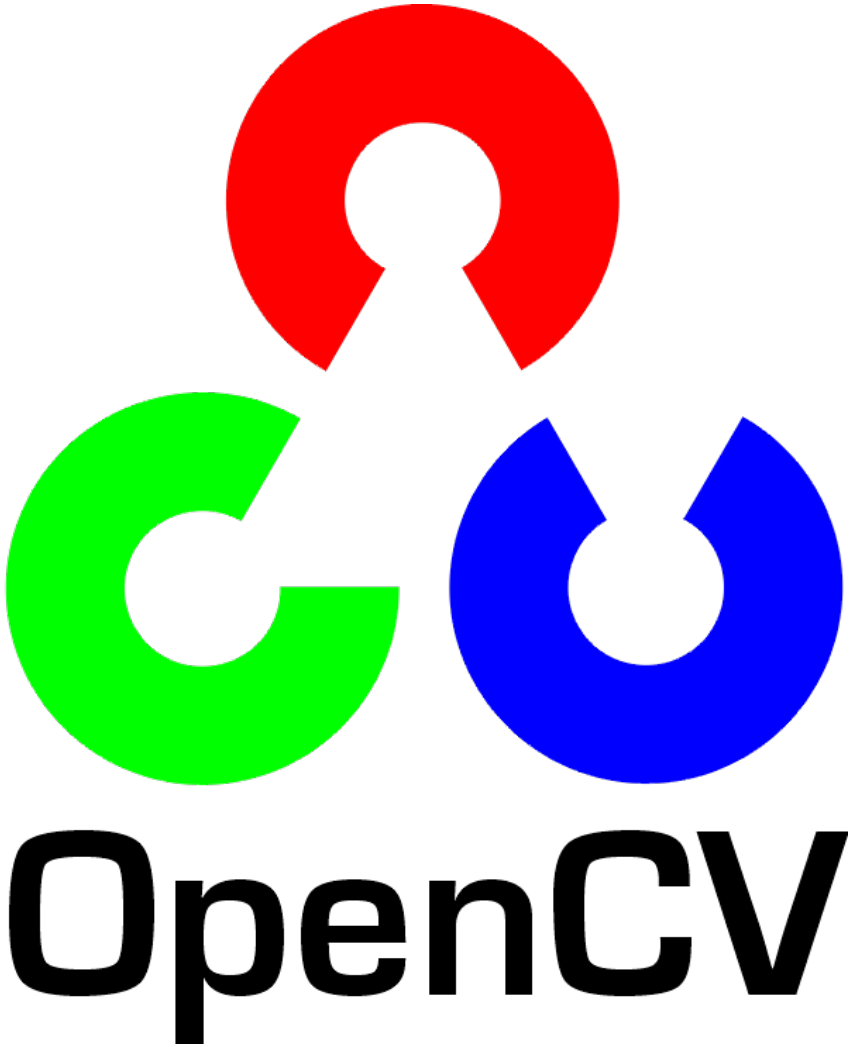


1. PYTHON EN DATA SCIENCE

- NLP -

1. PYTHON EN DATA SCIENCE

- Computer Vision -



1. PYTHON EN DATA SCIENCE

- Deep Learning -

Keras

PyTorch

TensorFlow

1. PYTHON EN DATA SCIENCE

- Big Data -

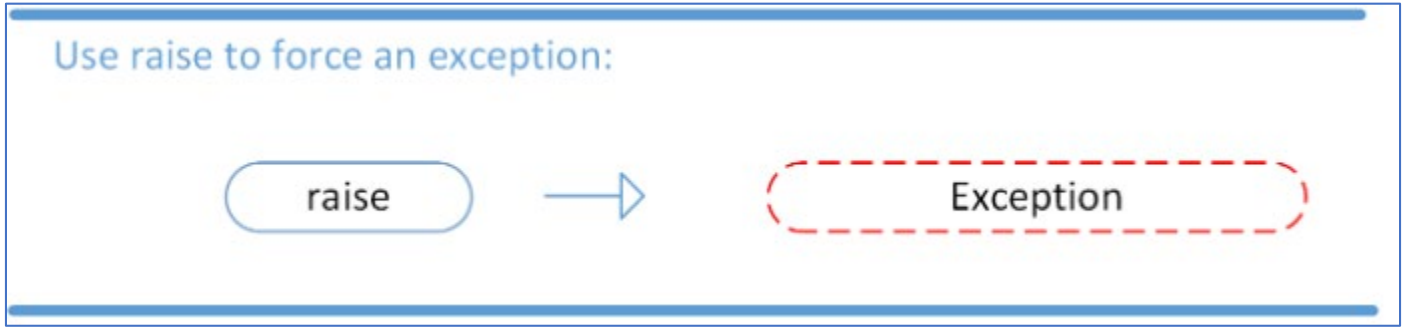
2. OOP

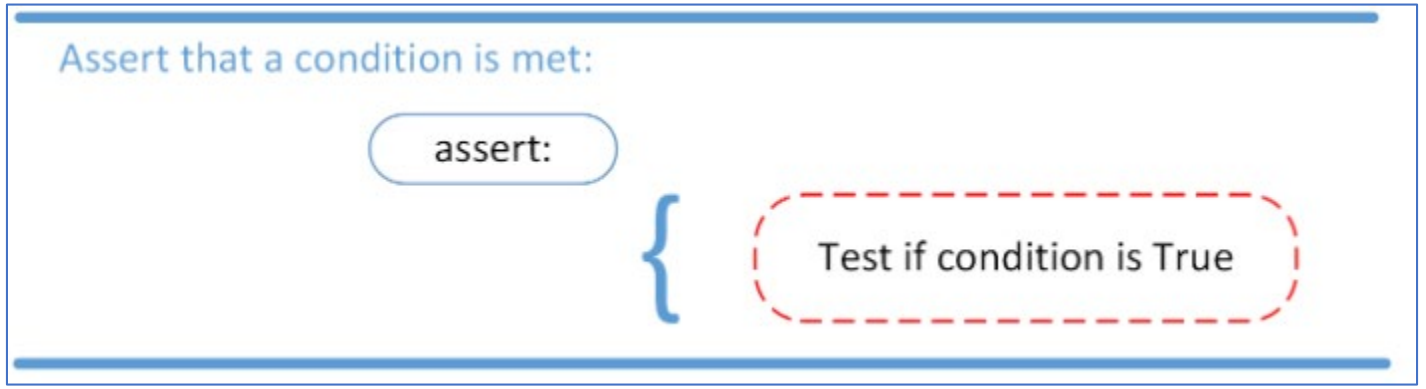
- Programación orientada a objetos.
- Paradigma:
  - Modelar objetos
  - Propiedades
  - Comportamiento

https://realpython.com/python3-object-oriented-programming/

```
class Dog:

    # Class Attribute
    species = 'mammal'

    # Initializer / Instance Attributes
    def __init__(self, name, age):
        self.name = name
        self.age = age


# Instantiate the Dog object
philo = Dog("Philo", 5)
mikey = Dog("Mikey", 6)

# Access the instance attributes
print("{} is {} and {} is {}.".format(
    philo.name, philo.age, mikey.name, mikey.age))

# Is Philo a mammal?
if philo.species == "mammal":
    print("{0} is a {1}!".format(philo.name, philo.species))
```

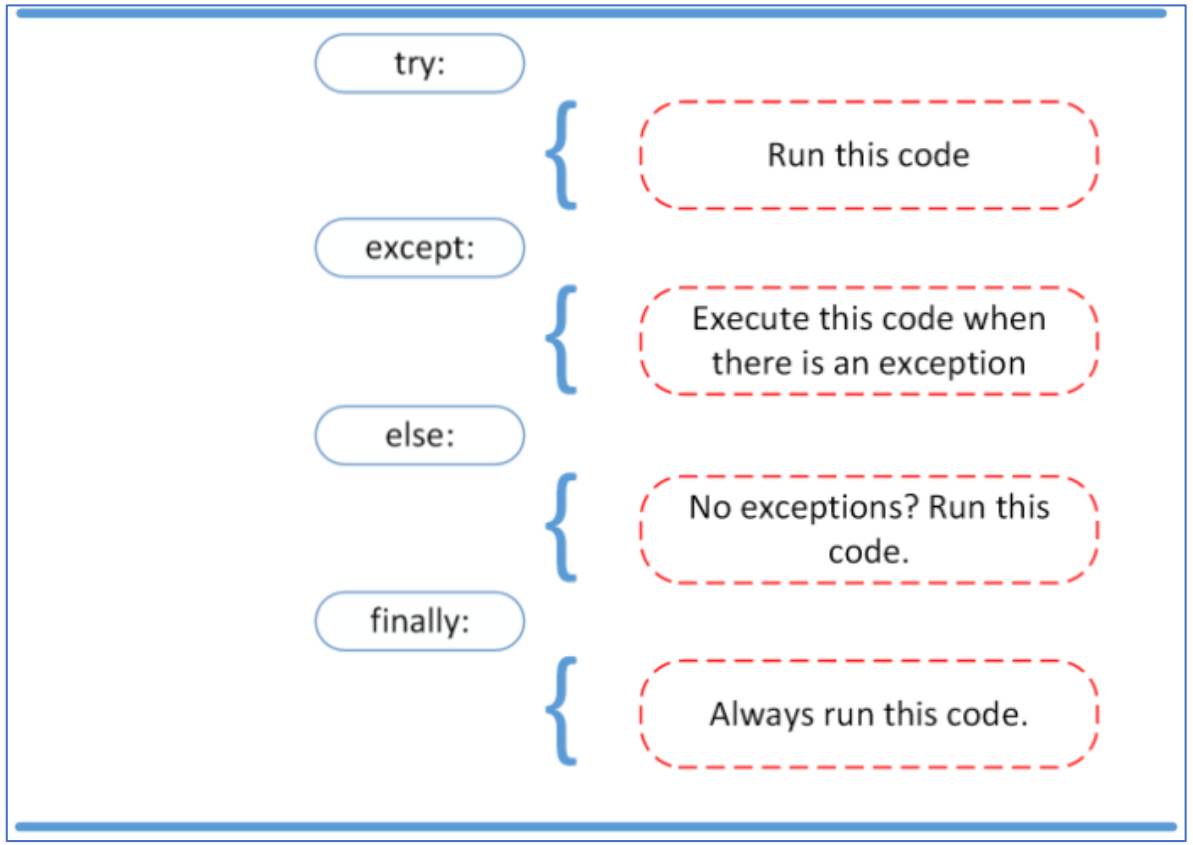
3. MANEJO DE ERRORES





https://realpython.com/python-exceptions/

3. MANEJO DE ERRORES



https://realpython.com/python-exceptions/

4. LIST COMPREHENSIONS

- Herramienta poderosa.
- Reduce líneas de código.
- Componentes:
  - Lista
  - Bucle
  - Condición

```
new_list = [expression for member in iterable (if conditional)]
```

https://realpython.com/list-comprehension-python/

5. FUNCIONES LAMBDA

- Origen: Lambda calculus (1930).
- Lenguaje funcional.
- No precisamente el paradigma de Python, pero se importó.
- También se conoce como función anónima.

```
lambda x, y: x + y
```

https://realpython.com/python-lambda/

6. MAP/REDUCE/FILTER

- **Map** aplica una misma función en distintos elementos.

```
my_pets = ['alfred', 'tabitha', 'william', 'arla']

uppered_pets = list(map(str.upper, my_pets))

print(uppered_pets)
```

https://www.learnpython.org/en/Map,_Filter,_Reduce

6. MAP/REDUCE/FILTER

- **Filter** aplica un filtro a aquellos elementos que se evalúan como False.

```
scores = [66, 90, 68, 59, 76, 60, 88, 74, 81, 65]

def is_A_student(score):
    return score > 75

over_75 = list(filter(is_A_student, scores))

print(over_75)
```

https://www.learnpython.org/en/Map,_Filter,_Reduce

6. MAP/REDUCE/FILTER

- **Reduce** es una función que agrega y reduce el número de elementos en una colección
- Se debe importar.

```
from functools import reduce
numbers = [3, 4, 6, 9, 34, 12]
def custom_sum(first, second):
    return first + second
result = reduce(custom_sum, numbers)
print(result)
```

https://www.learnpython.org/en/Map,_Filter,_Reduce

7. GENERADORES

- Generadores son iteradores.
- Ahorra espacio.
- Requerimiento para manejar grandes cantidades de datos.
- Elementos:
  - Componentes del list comprehension, pero usando ().
  - Yield.
  - Comando next.

```
file_name = "techcrunch.csv"
lines = (line for line in open(file_name))
list_line = (s.rstrip()split(",") for s in lines)
cols = next(list_line)
company_dicts = (dict(zip(cols, data)) for data in list_line)
funding = (
    int(company_dict["raisedAmt"])
    for company_dict in company_dicts
    if company_dict["round"] == "A"
)
total_series_a = sum(funding)
print(f"Total series A fundraising: ${total_series_a}")
```

https://realpython.com/introduction-to-python-generators/

8. DECORADORES

- Es una función que extiende la funcionalidad de otra función son modificarla.
- Se distinguen por @.

```
def my_decorator(func):
    def wrapper():
        print("Something is happening before the function is called.")
        func()
        print("Something is happening after the function is called.")
    return wrapper

@my_decorator
def say_whee():
    print("Whee!")
```

https://realpython.com/primer-on-python-decorators/

LABORATORIO Nº 2: PROFUNDIZAR EN PYTHON

Al finalizar el laboratorio, el alumno logrará:

- Ejecutar código de Python en un Notebook Jupyter.
- Profundizar en Python.

TAREA Nº 2: PROFUNDIZAR EN PYTHON

- Resolver los ejercicios en el Notebook Jupyter compartido.
- Enviar en **Notebook Jupyter** por correo al instructor.

RESUMEN

En este capítulo, usted aprendió:

- Python ofrece numerosas herramientas para potenciar la programación.

BIBLIOGRAFÍA

- Python. Python for beginners. https://www.python.org/doc/
- Scikit-learn. Biblioteca de aprendizaje automático. https://scikit-learn.org/stable/
- TensorFlow. Crea modelos de aprendizaje automático. https://www.tensorflow.org/?hl=es-419
- Kaggle. Comunidad de científicos de datos del aprendizaje automático. https://www.kaggle.com/


CIBERTEC