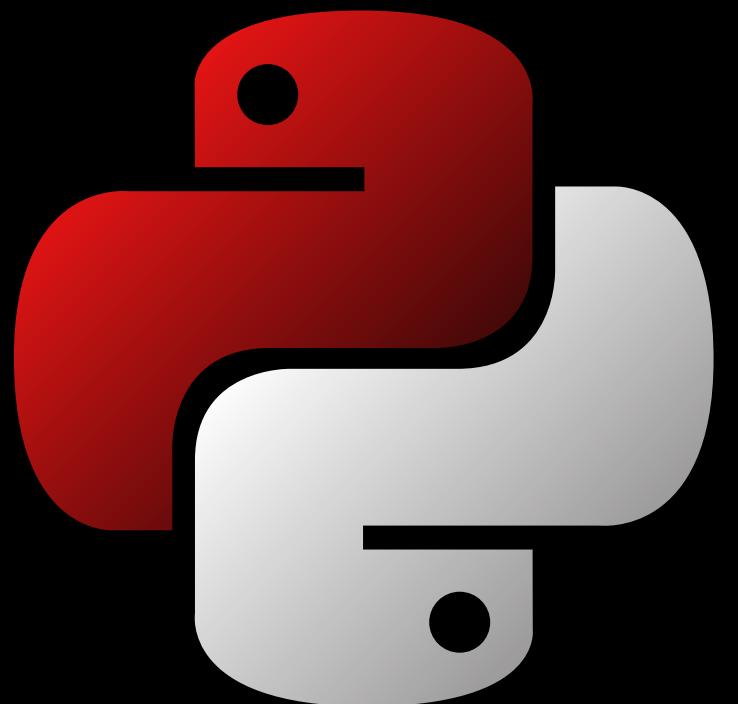


DEEPSEC ACADEMY

DeepSec Pentesting Web Python Course



DEEPSEC ACADEMY

Instructor



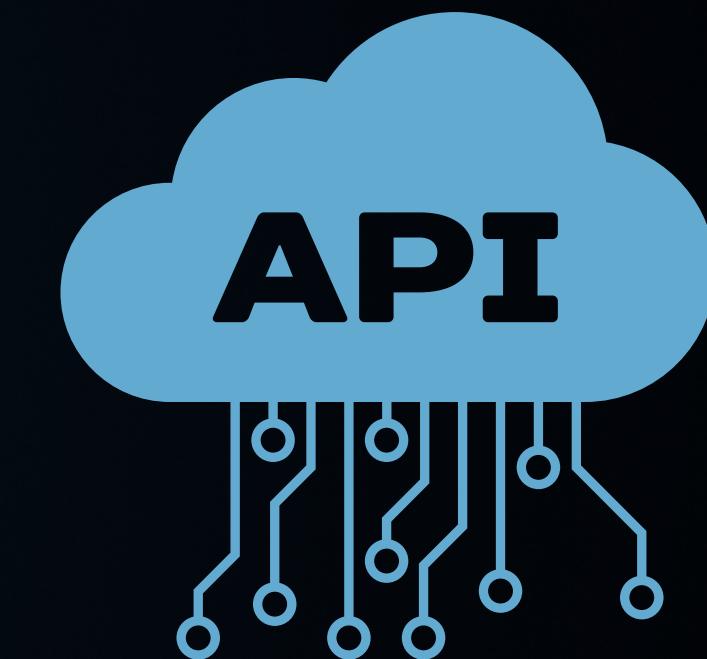
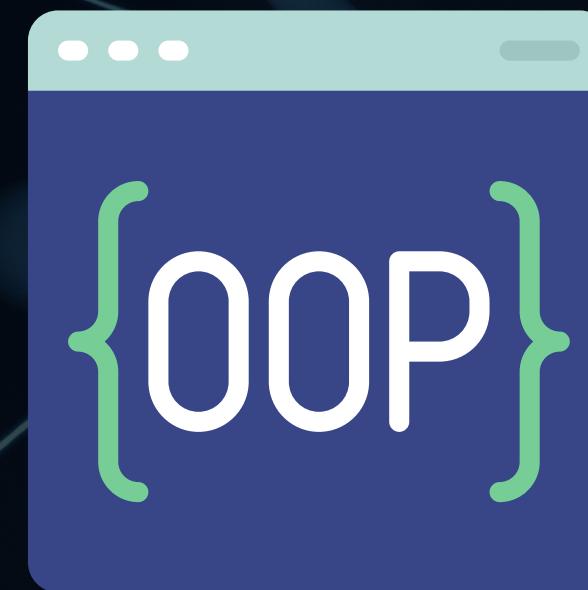
Diego Condori

Objetivos del curso

Introducirse en el Hacking etico



Aprender OOP (POO) y uso de API's



Reglamento



Puntualidad



**Microfono
Apagado**



Respeto

Material de Trabajo



Git-Hub

Todos los scripts los
almacenaremos en un
repositorio

**diegojoel301/
Curso_DSPWPC**

Repository de la clase DSPWPC

Contributors: 0 | Issues: 0 | Stars: 0 | Forks: 0

diegojoel301/Curso_DSPWPC: Repositorio de la clase DSPWPC

Repository de la clase DSPWPC. Contribute to diegojoel301/Curso_DSPWPC development by creating an account on GitHub.

[GitHub](#)

Material de Trabajo



Telegram

Las clases se subiran al
grupo de telegram

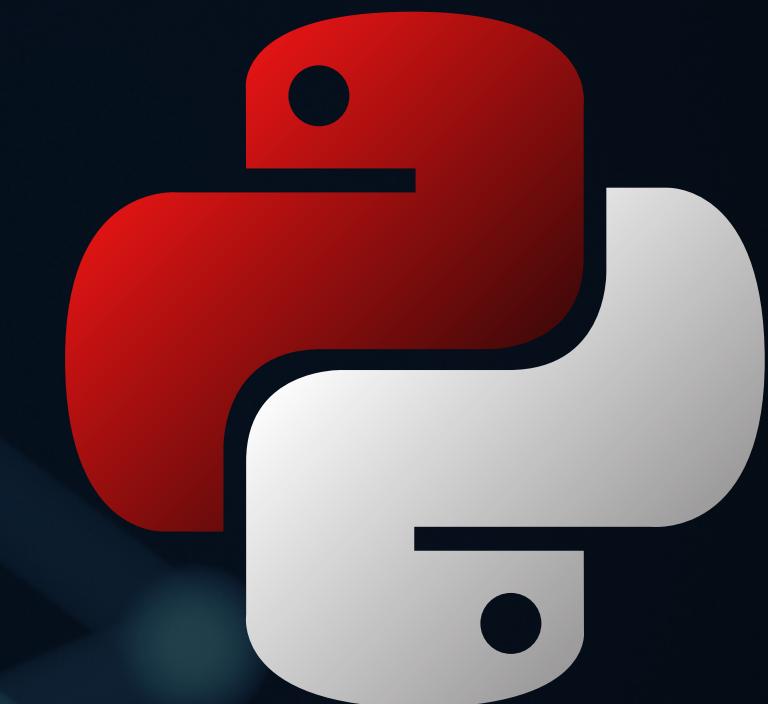


Curso DeepSec Pentesting Web ...

DEEPSEC ACADEMY

DEEPSEC ACADEMY

¿Porque Python es para hackers?



¿Porque Python es para hackers?



Librerias



Sintaxis sencilla

¿Porque Python es para hackers?

Los exploits suelen ser
creados en python por
su gran cantidad de
librerias



```
#!/usr/bin/python3

import requests
import sys
import warnings
from bs4 import BeautifulSoup
import json

warnings.filterwarnings("ignore", category=UserWarning, module='bs4')

if len(sys.argv) < 6:
    print("Usage: ./exploit.py http(s)://url username password listenerIP listenerPort")
    exit()

url = sys.argv[1]
username = sys.argv[2]
password = sys.argv[3]
ip = sys.argv[4]
port = sys.argv[5]

req = requests.session()
login_creds = {
    "username":username,
    "password":password,
    "mode":"normal"
}

print("[+] Sending login request...")
login = req.post(url+"/api/core/auth", json = login_creds)

/opt/exploitdb/exploits/python/webapps/48929.py
```

Exploit para RCE(Authenticated) Ajenti 2.1.36

Repasso de Python

Tipos de variables

Los tipos de variables que tenemos son:

Enteros: ..., -2, -1, 0, 1, 2, 3, 4, 5, 6, ...

Cadenas: "hola", "como", "estas",.....

Caracteres: 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h',...

Flotantes: 1.234, 1.235, 2.00, 2.3,.....

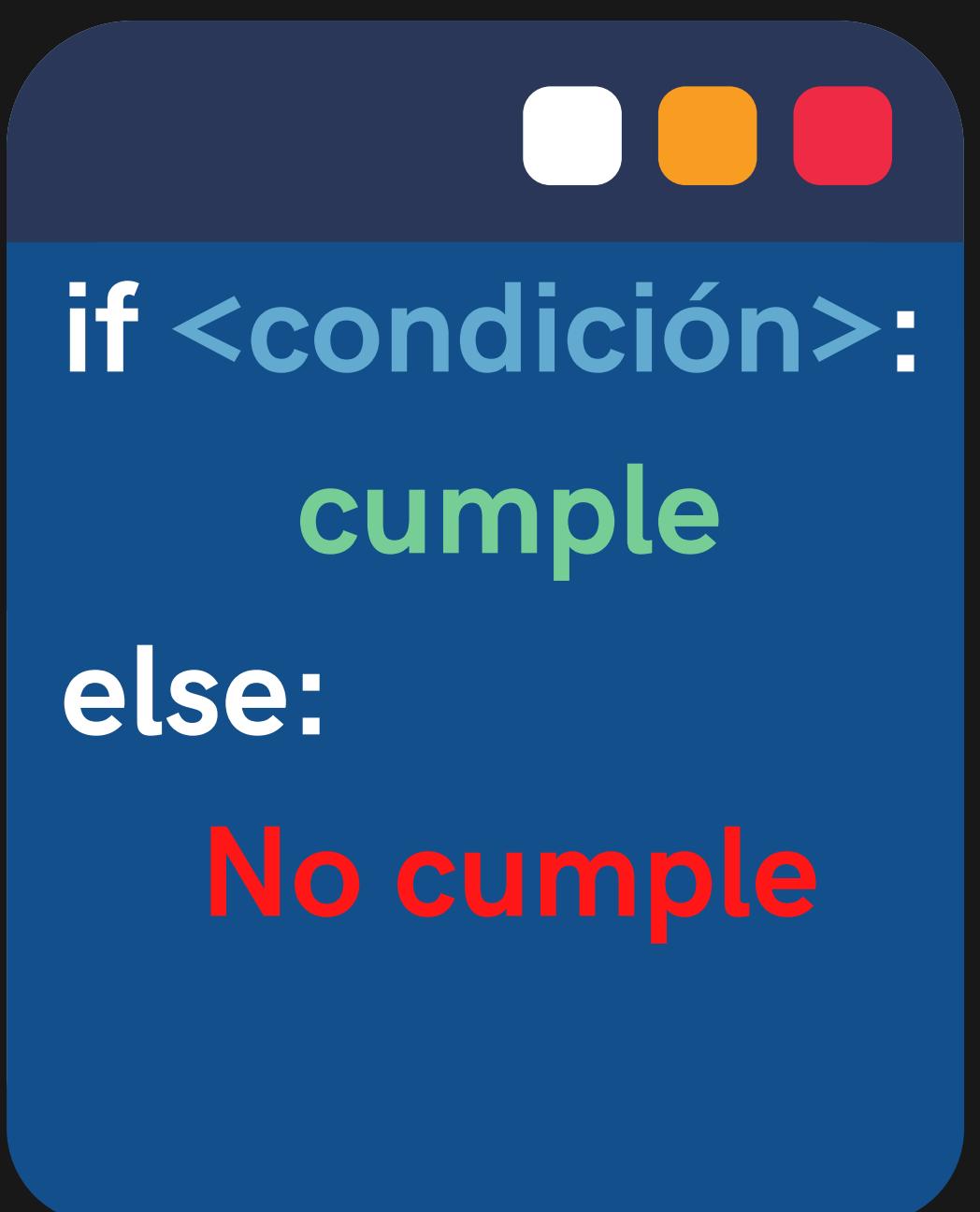
```
>>> entero = 1
>>> cadena = "string"
>>> caracter = 'c'
>>> flotante = 1.234567
```

Repasso de Python

Estructuras condicionales:

Las estructuras condicionales permiten decidir por cual alternativa seguira el flujo de un programa

```
>>> if entero == 1:  
...     print("Si es igual a uno")  
... else:  
...     print("No es igual a uno")  
...  
Si es igual a uno  
>>>
```



Repaso de Python

Estructuras Repetitivas:

Se utiliza cuando se quiere repetir un conjunto de sentencias un numero determinado de veces



```
for i in <conjunto, rango, vector lista,...>:  
    codigo
```

```
while <condicion>:  
    codigo
```

Repaso de Python

Estructuras Repetitivas:

```
>>> for i in range(0, 10):  
...     print(i, end = ' ')  
...  
0 1 2 3 4 5 6 7 8 9 >>>
```

Ejemplo de for

```
>>> n = 0  
>>> while n <= 10:  
...     print(n, end = ' ')  
...     n += 1  
...  
0 1 2 3 4 5 6 7 8 9 10 >>>
```

Ejemplo de while

Repaso de Python

Vectores y Matrices:

Un vector o array de elementos, es una estructura de datos que almacena datos de un solo tipo, una matriz es un array de arrays



[1][2][3][4]

0 1 2 3

Array de numeros



[a][b][3][4]

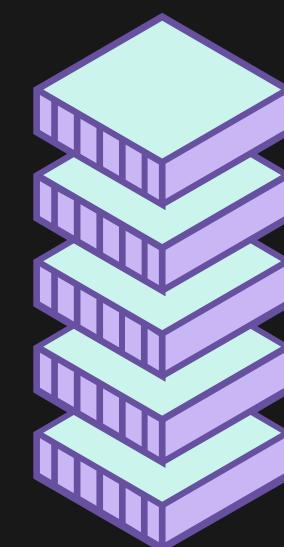
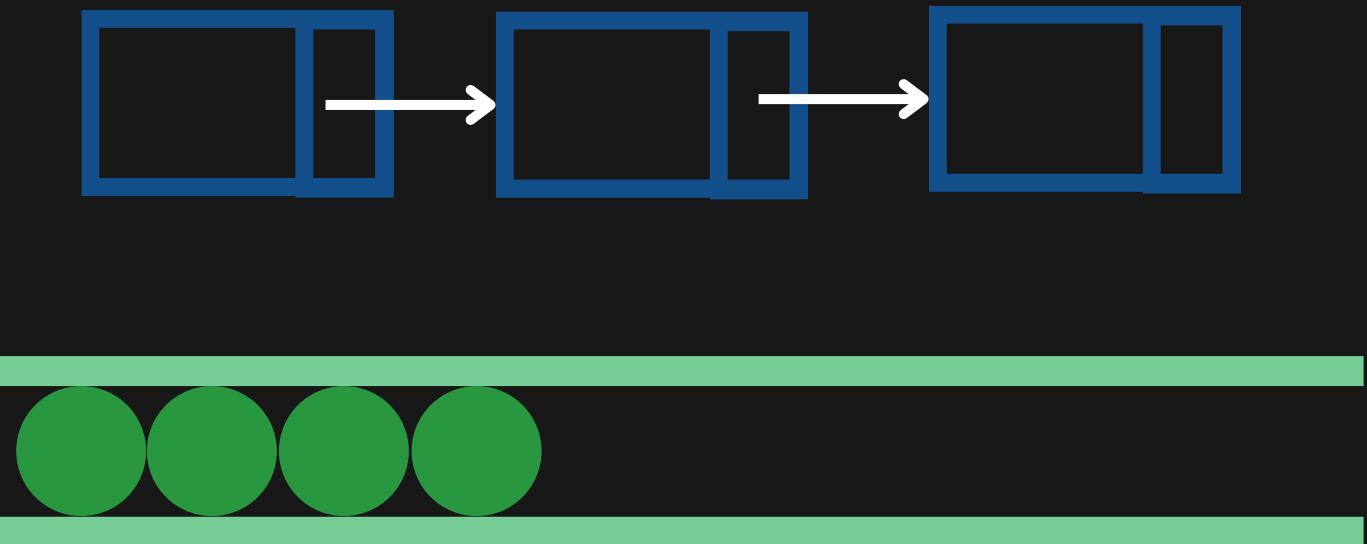
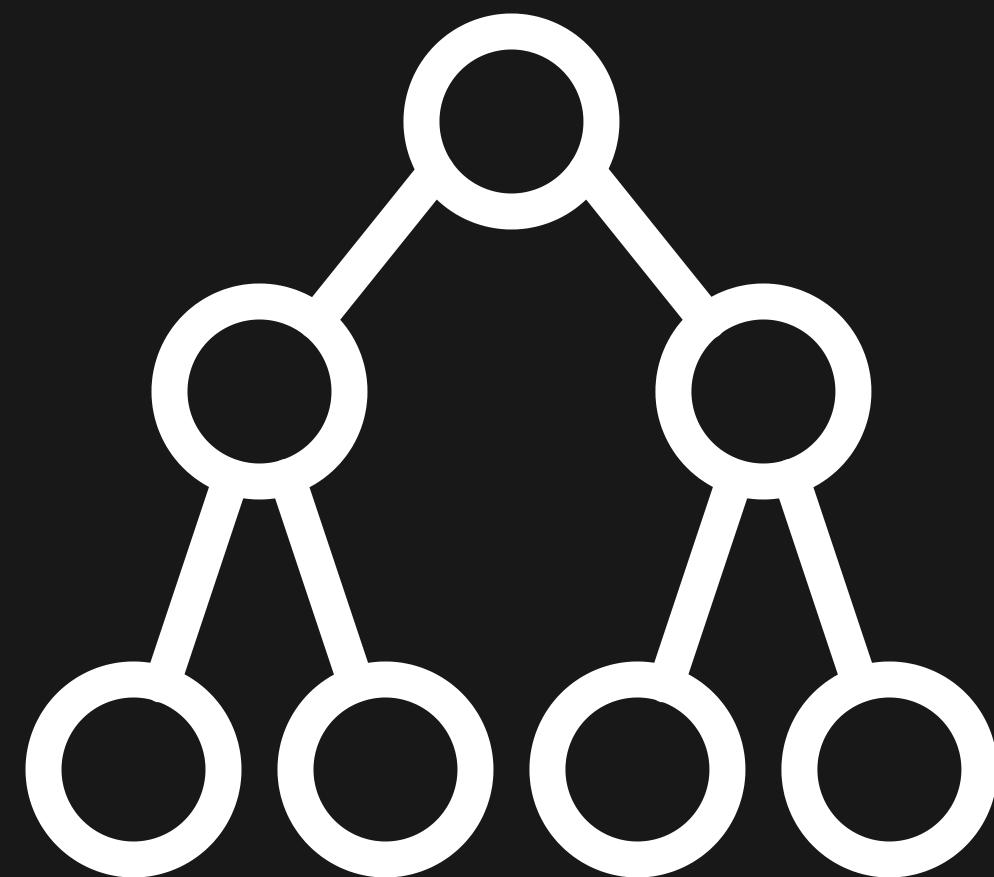
0 1 2 3

Array de ??????????

	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100

Repaso de Python

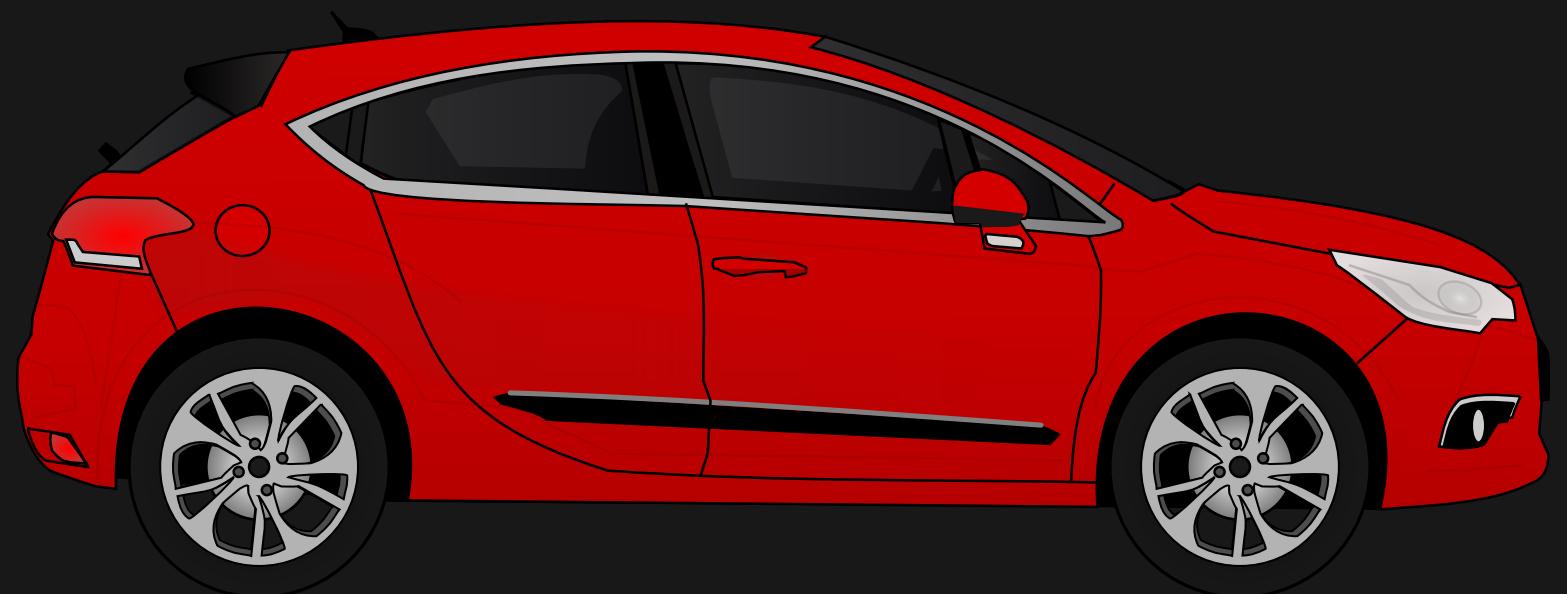
Otras estructuras de datos.....



```
CES = {  
    "company": "Sony",  
    "industry": "Automotive",  
    "model": "Vision S",  
    "year": 2020  
}  
  
data = CES.items()  
print(data)
```

Repasso de Python

Programacion Orientada a Objetos



Atributos:

- Color
- Modelo
- Placa
- Kilometraje
- Velocidad

.....

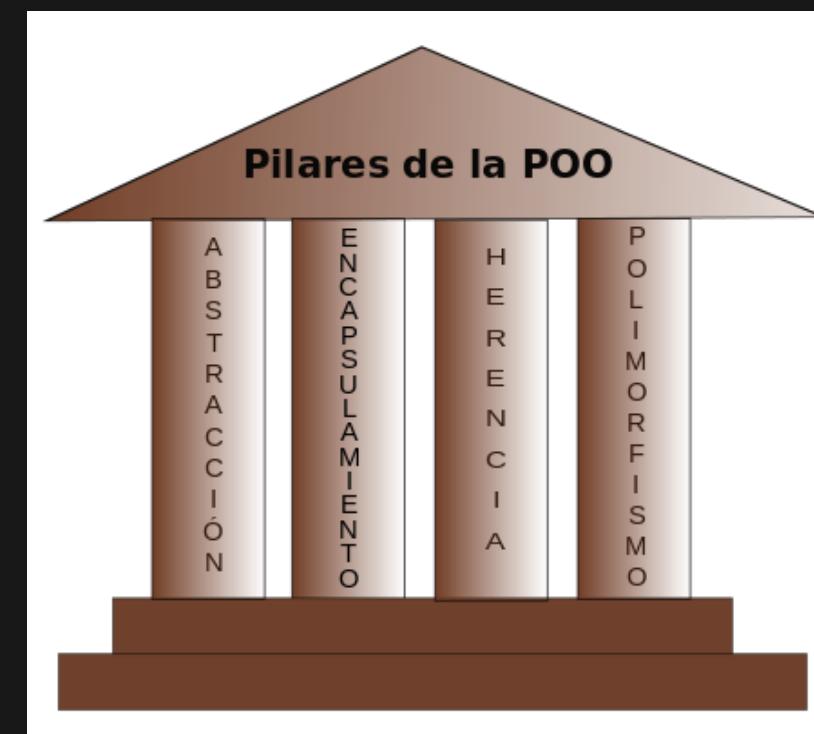
Metodos:

- Acelerar
- Frenar
- Abrir puerta
- Cerrar puerta

.....

Repaso de Python

Programacion Orientada a Objetos



- Herencia se refiere a usando la estructura y el comportamiento de una superclase en una subclase.
- Polimorfismo se refiere a cambiar el comportamiento de una superclase en la subclase.

POO

¿Como creamos una clase?

```
class Clase: # clase
    def __init__(self, valor_1, valor_2, valor_3, ....): # constructor
        # Atributos:
        self.atributo_1 = .....
        self.atributo_2 = .....
        .....
    # Metodos
    def metodo_1(self, valor_1, valor_2, valor_3, ....):
        .....
    def metodo_2(self, valor_1, valor_2, valor_3, ....):
        .....
        .....
```

POO

¿Como instanciamos una clase?

```
# Instanciar Clases  
# Si es que el constructor no necesita de valores iniciales:  
objeto = Clase()  
  
# Si es que el constructor necesita de valores iniciales:  
objeto = Clase(valor_1, valor_2, ....)  
  
# Ejecutar un metodo de un objeto  
objeto.metodo_1(valor_1, valor_2, ....)
```

Diccionario

¿Que es un diccionario?

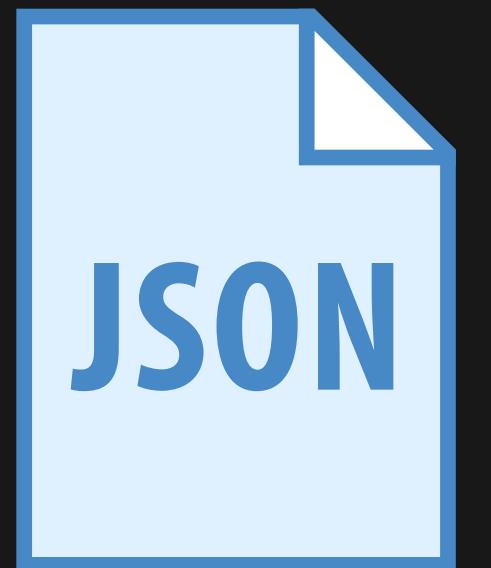
Es un tipo de mapa que asocia claves a valores, a comparacion de un vector que es indexado por un indice numerico, el diccionario es indexado por claves

{**key: value**}

```
CES = {  
    "company": "Sony",  
    "industry": "Automotive",  
    "model": "Vision S",  
    "year": 2020  
}  
  
data = CES.items()  
print(data)
```

Diccionario

¿Que es un diccionario?



```
{  
    "Id": 1,  
    "Name": "John",  
    "Age": 24  
},  
{  
    "Id": 2,  
    "Name": "Paul",  
    "Age": 19  
},  
{  
    "Id": 3,  
    "Name": "Rogger",  
    "Age": 21  
}
```



¡Muchas gracias!