

In [10]:

```
#Pop
```

```
lista = [2 , 4 , 6 , "a", 6]  #Para borrar el segundo 6  
#Remove borra el primer elemento con el valor dado  
lista.remove(6)  
lista
```

Out[10]:

```
[2, 4, 'a', 6]
```

In [13]:

```
lista = [2 , 4 , 6 , "a", 6]  
lista.pop() #Por defecto borra el último valor, pero si le incluyes información borrará e  
l valor que le digas  
print (lista)
```

```
[2, 4, 6, 'a']
```

In [14]:

```
lista = [2 , 4 , 6 , "a", 6]  
lista.pop(4) #Elimina la posición 4  
print (lista)
```

```
[2, 4, 6, 'a']
```

In [4]:

```
def remove2(a):  
    x = a + 2  
y = remove2(6)  
print(y) #Devuelve none porque dentro de la funcion no tiene remove
```

```
None
```

In []:

```
def remove2(a):  
    x = a + 2  
y = remove2(6)  
print(y)
```

In [17]:

```
def f(s):  
    print(s)  
  
x = f(2)  
print(x)  
  
#Diferencia entre imprimir y retornar. Lo que retornas no se necesita imprimir en pa  
ntalla. Ves el dos por pantalla pero no retorna 2, es decir, la x vale none.
```

```
2
```

```
None
```

In [18]:

```
def f(s):  
    print(s)  
    return s #En este caso ya x vale 2 e imprime el 2
```

```
x = f(2)  
print(x)
```

2

In [20]:

```
#Diccionarios

hola = "saludo internacional"
#clave/key = valor (Todas las claves tienen un valor)

diccionario = {"key" : "valor"}
#Una key puede ser string o int pero no una lista
#Valor sí puede ser una lista
diccionario
```

Out[20]:

```
{'key': 'valor'}
```

In [21]:

```
print(diccionario["key"]) #NO SE PUEDEN REPETIR CLAVES, LAS CLAVES SON UNICAS. ej: en un
diccionario no se pueden repetir palabras, aquí igual. Ojo pero un diccionario puede tene
r una clave con varios valores, varios str, pero no puedes repetir la clave, es decir, gu
ardas los distintos valores en la misma clave
```

valor

In [22]:

```
diccionario2 = {2: "este es el valor", 2: "este es el valor"}
diccionario2
```

Out[22]:

```
{2: 'este es el valor'}
```

In [25]:

```
diccionario3 = {"gabvaztor@gmail.com": ["Password", "Gabvaztor", 648194794, "C/Pepito",
"Profesor"]} #esto puede contener varios correos con sus respectivos datos
diccionario3["gabvaztor@gmail.com"]
```

Out[25]:

```
['Password', 'Gabvaztor', 648194794, 'C/Pepito', 'Profesor']
```

In []:

```
diccionario4 = {"juan": 8, "silvia":10, "juan":9}
```

In []:

```
diccionario4 = {9.8: "s"} #Puede ser float
```

In [30]:

```
diccionario5 = {"k": "v", 8:[7, "y"], 6:{1.1 :[5]}} #Diccionario dentro de un diccionari
o con una lista de valor 5. ¿Cómo accedemos a ese 5?
```

```
# El 6 ""esto es una clave, para acceder al diccionario""
# El 1.1 ""esto es otra clave, para acceder a los valores del diccionario ""
# El 5 ""esto es un valor""
type(diccionario[6][1.1][0])
```

KeyError

Traceback (most recent call last)

<ipython-input-30-7dd8bce4b3ef> in <module>

4 # El 1.1 ""esto es otra clave, para acceder a los valores del diccionario ""

5 # El 5 ""esto es un valor""

----> 6 type(diccionario[6][1.1][0])

KeyError: 6

In [32]:

```
diccionario5.keys()
```

Out[32]:

```
dict_keys(['k', 8, 6])
```

In [34]:

```
list(diccionario5.values())
```

Out[34]:

```
['v', [7, 'y'], {1.1: [5]}]
```

In [35]:

```
for key in diccionario5.keys(): #Para acceder a las claves de un diccionario y recorrerlas
    print(key)
```

```
k
8
6
```

In [37]:

```
for value in diccionario5.values(): #Para acceder a los valores de un diccionario y recorrerlos
    print(value)
```

```
v
[7, 'y']
{1.1: [5]}
```

In [38]:

```
for key, value in diccionario5.items():
    print(key, ":", value)
```

```
k : v
8 : [7, 'y']
6 : {1.1: [5]}
```

In [40]:

```
for key, value in diccionario5.items():
    if type(key) == int and key > 5:
        print(key, ":", value)
```

```
8 : [7, 'y']
6 : {1.1: [5]}
```

In [47]:

```
for pos, (key, value) in enumerate(diccionario5.items()): #Para mostrar la posición, las claves y los valores
    print(pos)
    print(key, ":", value)
    print("-----")
```

```
0
k : v
-----
1
8 : [7, 'y']
-----
2
6 : {1.1: [5]}
-----
```

In [46]:

```
for pos, (key, value) in enumerate(diccionario5.items()): #Para mostrar la posición, las  
claves y los valores  
    print("Iteración número:", pos+1)  
    print(key, ":", value)  
    print("-----")
```

```
Iteración número: 1  
k : v  
-----  
Iteración número: 2  
8 : [7, 'y']  
-----  
Iteración número: 3  
6 : {1.1: [5]}  
-----
```

In [50]:

```
d = {"k": 2, "k2": [8, "p"]}  
  
#Acceder a un valor a traves de una key  
key_buscar = "k2"  
d[key_buscar]
```

Out[50]:

```
[8, 'p']
```

In [51]:

```
#Buscar/Mostrar todas las keys  
key_buscar = "k2"  
list(d.keys())
```

Out[51]:

```
['k', 'k2']
```

In [52]:

```
#Buscar/Mostrar todas los values  
key_buscar = "k2"  
list(d.values())
```

Out[52]:

```
[2, [8, 'p']]
```

In [53]:

```
#Buscar/Mostrar todas las keys/values - te lo devuelve como tuplas  
key_buscar = "k2"  
list(d.items())
```

Out[53]:

```
[('k', 2), ('k2', [8, 'p'])]
```

In [55]:

```
#Recorrer la lista value de la clave "k2"  
for key, values in d.items():  
    if key == "k2":  
        for x in values:  
            print (x)
```

```
8  
p
```

In [56]:

```
#Eliminar una clave de un diccionario borrar k, no se puede modificar solo borrar o modificar su valor
```

```
del d ["k"]  
d
```

Out[56]:

```
{'k2': [8, 'p']}
```

In [59]:

```
#Modificar el valor de una clave  
d["k2"] = [8, "a"]  
d
```

Out[59]:

```
{'k2': [8, 'a']}
```

In [60]:

```
d["k2"].append(3)  
d
```

Out[60]:

```
{'k2': [8, 'a', 3]}
```

In [61]:

```
#Añadir un diccionario a otro
```

```
d1 = {8: [8, 9]}  
d2 = {"a": "hola"}  
d1.update(d2)
```

```
print(d2)  
print(d1)
```

```
{'a': 'hola'}  
{8: [8, 9], 'a': 'hola'}
```

In [64]:

```
def add_to_list(lista, to_add):  
    lista.append (to_add)  
  
l = [2, 7]  
l = add_to_list(lista=l, to_add=6) #Muestra none porque no tiene return  
print(l)
```

None

In [65]:

```
d = {2: "h"}  
l = [5, "m", d]  
l[-1]
```

Out[65]:

```
{2: 'h'}
```

In []:

```
#Retornar más de un elemento  
k = 2, 7  
k
```

In [66]:

```
k, j = 2, 7  
print(k)
```

```
print(j)
```

```
2  
7
```

In [67]:

```
def f():  
    return 3, 6, 8  
  
def cualquiercosa():  
    return 5, 2, 8  
  
g, h = f(), cualquiercosa()  
print(g)  
print(h)
```

```
(3, 6, 8)  
(5, 2, 8)
```

In [70]:

```
#Función con parámetro  
def nombre_funcion(param):  
    return param  
  
x = nombre_funcion("argumento")  
print(x)  
g = type(x)  
g
```

argumento

Out[70]:

str

In [72]:

```
#Función con parámetro/valor por defecto  
  
def nombre_funcion_por_defecto(param=2):  
    return param  
x = nombre_funcion_por_defecto()  
x
```

Out[72]:

2

In [73]:

```
#Prioridad de argumento vs defecto  
def nombre_funcion_por_defecto(param=2):  
    return param  
x = nombre_funcion_por_defecto(param=7)  
x
```

Out[73]:

7

In [74]:

```
#Prioridad de argumento vs defecto  
def nombre_funcion_por_defecto(param=2, y=5):  
    return param  
x = nombre_funcion_por_defecto()  
x
```

Out[74]:

2

In [75]:

```
#Si un argumento de la función no tiene un valor por defecto, es obligatorio darle un valor al llamar a la función.
```

```
#Si le damos un valor a un argumento de la función al ser llamada, ese valor tiene prioridad al valor por defecto
```

```
def nombre_funcion_por_defecto(param=2, y=5):  
    return param + y  
x = nombre_funcion_por_defecto(4, 7)  
x
```

Out[75]:

11

In [79]:

```
def get_the_last_value_from_list(lista, position=-1):  
    if len(lista) > 0: #Si hay algun elemento  
        return lista[position]  
    else:  
        return "No hay elementos"
```

```
lista = ["m", 2]  
x = get_the_last_value_from_list(lista=lista)  
print(x)
```

2

In [82]:

```
lista = [2, "8", [6, "joven"], "casa", "coche", "pepino", [9, 8]]
```

```
def muestra_todos_excepto_joven_casa_9(lista):  
    for x in lista:  
        if x == lista[2]:  
            print(x[0])  
        elif x == lista[3]:  
            continue  
        elif x == lista[-1]:  
            print(x[1])  
        else:  
            print(x)
```

```
muestra_todos_excepto_joven_casa_9(lista=lista)
```

```
2  
8  
6  
coche  
pepino  
8
```

In [83]:

```
jupyter nbconvert --to pdf Day2.ipynb.ipynb
```

```
File "<ipython-input-83-e19c01c5917f>", line 1  
    jupyter nbconvert --to pdf Day2.ipynb.ipynb  
    ^
```

SyntaxError: invalid syntax

In []: