In [1]:

```
#Para mostrar hasta 5
l = []
for i in range(6):
    l.append(i)
    print(l)
```

```
[0]
[0, 1]
[0, 1, 2]
[0, 1, 2, 3]
[0, 1, 2, 3, 4]
[0, 1, 2, 3, 4, 5]
```

In [3]:

```
#Para mostrar hasta 5 en list comprehension
lc = [i for i in range(6)]
lc
```

Out[3]:

```
[0, 1, 2, 3, 4, 5]
```

In [ ]:

```
#List comprehension siempre crea una lista vacía que se va a rellenar con la list compreh
ension
```

In [4]:

```
def fn(k=None):
    if k: #Aqui estamos comparando que el valor de k sea True, 1 == 1 o cualquier valor
que no sea cero. No es True porque el valor de k es None.
        return []
    else: #None, False, 0
        return [i for i in range(6)]

p = fn() #No nencesita porque ya tiene un valor (None)
print(p)
```

```
[0, 1, 2, 3, 4, 5]
```

In [8]:

```
def ct (val):
    if val > 5:
        return True
    else:
        return False
lc = [x for x in range(10) if ct(val=x)] #Es lo mismo que poner [x for x in range(10) if
x > 5]
lc
```

Out[8]:

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

In [7]:

```
for i in range (6):
    if ct(val=i): #Esto es false
        print("Mayor que 5")
    else:
        print("Menor que 5")
```

```
Menor que 5
Menor que 5
Menor que 5
```

```
Menor que 5
Menor que 5
Menor que 5
```

In [20]:

```
def ct(val, limite):
    a_retornar = False #Es otra forma de poner else: return False
    if val > limite:
        return True
    return a_retornar
```

In [21]:

```
def main(lista, limite):
    for x in lista:
        if ct(val=x, limite=limite):
            return [i for i in range(x)] #Esto es lo mismo que poner list(range(9))

l = list(range(10))
limite = 8
print(main(lista=l, limite=limite))
```

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

In [22]:

```
def main(lista, limite):
    for x in lista:
        if ct(val=x, limite=limite):
            return [i for i in range(x) if i<5]

l = list(range(10))
limite = 8
print(main(lista=l, limite=limite))
```

```
[0, 1, 2, 3, 4]
```

In [4]:

```
#Para comprobar los tipos de todos los elementos y agregar los números a una lista

l = ["st", 3, [0.1, 1], (8, 2), {2:"valor"}, {9, "s"}]
lista_numerica = []

for x in l:
    if isinstance(x, (int, float)): #Si es int o float
        lista_numerica.append(x)
    else:
        if isinstance(x, dict):
            for e in list (x.items()[0]): #Esto es una tupla de una lista
```

```
  File "<ipython-input-4-012bf1be40c5>", line 11
    for e in list (x.items()[0]): #Esto es una tupla de una lista
                                                                 ^
IndentationError: expected an indented block
```

In [5]:

```
#Para comprobar los tipos de todos los elementos y agregar los números a una lista

l = ["st", 3, [0.1, 1], (8, 2), {2:"valor"}, {9, "s"}]
lista_numerica = []

for x in l:
    if isinstance(x, (int, float)): #Si es int o float
        lista_numerica.append(x)
    else:
        if isinstance(x, dict):
            for k, v in x.items(): #Esto es una tupla de una lista
                if isinstance(k, (int, float)):
```

```
                    lista_numerica.append(k)
                if isinstance(v, (int,float)):
                    lista_numerica.append(v, (int, float))
        else: #Es una colección (diccionario no es una colección, es un diccionario)
            if isinstance(x, str):
                continue
            else: #Colección no str
                for elem in x:
                    if isinstance (elem, (int, float)):
                        lista_numerica.append(elem)

lista_numerica
```

Out[5]:

```
[3, 0.1, 1, 8, 2, 2, 9]
```

In [ ]:

```
#Para hacerlo más fácil utilizar una función - COGER DE LOS APUNTES DE GABRIEL

l = ["st", 2, [0.1, 1], (8, 2), {2:"valor"}, {9, "s"}]
lista_numerica = []

def check_and_types(elem, types, lista):
    if isinstance(elem,types):
        lista.append(elem)

for x in l:
    check_and_types(elem=x, types=(int, float), lista=lista_numerica):
        lista_numerica.append(x)
```

In [30]:

```
#LAMBDA#

def nombre_f(d, a):
    return d + 2

nombre_f(d=2, a=0)
```

Out[30]:

```
4
```

In [31]:

```
nombre_g = lambda d, a: d+2 #Despues de los los dos puntos equivale a retornar. Es el lis
t comprehension de las funciones
nombre_g(d=2, a=0)
```

Out[31]:

```
4
```

In [32]:

```
def nombre_f(k):
    return k

nombre_f = lambda k:k

nombre_f(k=2)
```

Out[32]:

```
2
```

In [37]:

```
def nombre_f(k):
    if k > 0:
        return k
```

```
    else:
        return 2

nombre_f = lambda k: print(k) if k > 0 else 2 #Si escribimos una condición en una funció
n lambda es obligatorio el if y else
r = nombre_f(k=4)
print(r)
```

```
4
None
```

In [39]:

```
def nombre_f(k):
    if k > 0:
        return k
    else:
        return 2

def f1(d):
    print(d+2)
    return d

nombre_f = lambda k: f1(d=k) if k > 0 else 2
r = nombre_f(k=4)
print(r)
```

```
6
4
```

In [ ]:

```
def is_higher_5 (elem):
    if elem > 5:
        return elem

l = [i for i in range(10)]
```