# Listas en Python

Materia: Redes Neuronales.

Profesora: Camacho Vázquez Vanessa Alejandra.

Ahora aprenderemos sobre las listas, que son uno de los objetos más útiles y flexibles en Python. Las listas son objetos iterables.

## Listas

Las listas son contenedores mutables y ordenados de objetos. Las listas se distinguen porque sus elementos están encerrados entre paréntesis cuadrados. Por ejemplo:

```
In [1]: lista1 = ['Alvaro', 'Daniel', 'Pilar', 'Beatriz']

for i in lista1:
    print(i, end=' ')
print('')
```

```
Alvaro Daniel Pilar Beatriz
```

Una lista puede contener elementos muy complejos como el siguiente ejemplo:

In [2]:
```
t1 = (1, 'Oleg', 24.5)
l1 = ['Maria', 'Bonita']
l2 = [t1, l1]
t2 = (l2,'manzana')

print('t1=', t1)
print('l1=', l1)
print('l2=', l2)
print('t2=', t2)
```

```
t1= (1, 'Oleg', 24.5)
l1= ['Maria', 'Bonita']
l2= [(1, 'Oleg', 24.5), ['Maria', 'Bonita']]
t2= ([(1, 'Oleg', 24.5), ['Maria', 'Bonita']], 'manzana')
```

Vamos a acceder a los elementos individuales de las listas y tuplas que hemos creado antes.

In [3]:
```
for i in t1:
    print(i)
print('\n')

for i in l1:
    print(i)
print('\n')

for i in l2:
    print(i)
print('\n')

for i in t2:
    print(i)
```

```
1
Oleg
24.5


Maria
Bonita


(1, 'Oleg', 24.5)
['Maria', 'Bonita']


[(1, 'Oleg', 24.5), ['Maria', 'Bonita']]
manzana
```

Ahora accedamos al interior de las estructuras complejas.

```
print(t2[0])
print(t2[1])
print('\n')

print(t2[0][0])
print(t2[0][1])
print('\n')

print(t2[0][0][0])
print(t2[0][0][1])
print(t2[0][0][2])
print('\n')

print(t2[0][1][0])
print(t2[0][1][1])
print(t2[0][0][2])
print('\n')
```

Tomen capturas de pantalla de sus resultados y expliquen en un párrafo la razón detrás de ellos*** → adjunten sus códigos.

Constructor de lista – *list(iterable)*

```
In [5]: vocalTupla = ('a', 'e','i','o','u')
        vocalLista = list(vocalTupla )

        print(vocalTupla)
        print(vocalLista)
```

```
('a', 'e', 'i', 'o', 'u')
['a', 'e', 'i', 'o', 'u']
```

Adiciona elementos a una lista - *lista.append(objeto)*

```
In [6]: alfabeto = 'abcdefghijklmnñopqrstuvwxyz'
        print( 'alfabeto es un objeto string: ',type(alfabeto))
```

```
alfabeto es un objeto string:  <class 'str'>
```

Vamos a crear una lista vacía y la vamos a llenar con cada uno de los elementos de *alfabeto*

```
In [7]: alfaL = []

        for i in alfabeto:
            alfaL.append(i)

        print(alfaL)
```

```
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'ñ', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w',
'x', 'y', 'z']
```

Concatenar 2 listas

In [8]:
```
N = len(alfaL)

numL = []

for i in range(N):
    numL.append(i)

print('alfaL = ',alfaL)
print('\n')
print('numL = ',numL)
print('\n')

alfanumL = alfaL + numL
alfanumLL = [alfaL, numL]

print('alfanumL = ',alfanumL)
print('\n')
print('alfanumLL = ',alfanumLL)
```

In [9]:
```
# Acceso a los elementos de la lista alfanumLL

for i in range(N):
    print(alfanumLL[0][i],alfanumLL[1][i] )
```

Eliminar elementos de una lista - *lista.remove(objeto)*

In [10]:
```
alfaL.remove('c')
print(alfaL)
```

Métodos para manipulación de listas

| Method | Description |
|---|---|
| append() | Agrega un elemento al final de la lista |
| clear() | Remueve todo los elementos de la lista |
| copy() | Regresa una copia de la lista |
| count() | Regresa el número de elementos con el valor especificado |
| extend() | Agrega elementos de una lista (o cualquier iterable) al final de esta lista |
| index() | Regresa el índice del primer elemento con el valor especificado |
| insert() | Adiciona un elemento en la posición especificada |
| pop() | Remueve un elemento en la posición especificada y puede retornarlo si se hce una asignación |
| remove() | Remueve el item con este valor específico |
| reverse() | Invierte el orden de la lista |
| sort() | Ordena la lista |

Prueben cada uno de los métodos mencionados e impriman los resultados***

Ejercicio. Declaren una lista vacía, modifiquen el siguiente bucle para que en lugar de imprimir la pareja especificada forme una tupla con ella y la agregue a la lista previamente declarada.

In [13]:
```
# Acceso a los elementos de la lista alfanumLL

for i in range(N):
    print(alfanumLL[0][i],alfanumLL[1][i] )
```

```
0 a
1 b
2 c
3 d
4 e
5 f
6 g
7 h
8 i
9 j
10 k
11 l
12 m
13 n
14 ñ
15 o
16 p
17 q
```

… llega hasta 26 z.

Tomen captura de pantalla de tus códigos y resultados***

## Comprensión de listas

Ya vimos que es posible crear listas a partir de objetos iterables. Aquí mostramos que es posible hacer nuevas listas con ciertas manipulaciones a partir de otras. Esto con un sintaxis simple y rápido.

Veamos un caso sin usar comprensión de listas

In [14]:
```
frutas = ["manzana", "kiwi", "guanabana", "limón", "níspero", "Pomelo", "Mango"]

lista2 = []

for x in frutas:
    if "a" in x:
        lista2.append(x)

print(lista2)
```

```
['manzana', 'guanabana', 'Mango']
```

Funciona, pero necesitamos varias líneas. Ahora usando comprensión

In [15]:
```
lista2 = [x for x in frutas if "a" in x]
print(lista2)
```

```
['manzana', 'guanabana', 'Mango']
```

Podemos usar cualquier operador lógico y operador de comparación en estos casos

In [16]:
```
lista2 = [x for x in frutas if "a" not in x]
print(lista2)
```

```
['kiwi', 'limón', 'níspero', 'Pomelo']
```

In [17]:
```
lista2 = [x for x in frutas if len(x)<6]
print(lista2)
```

```
['kiwi', 'limón', 'Mango']
```

Podemos hacer comprensión con cualquier tipo de iterable

In [18]:
```
lista2 = [x for x in range(10) if x % 2 ==0]
print(lista2)
```

```
[0, 2, 4, 6, 8]
```

In [19]:
```
lista2 = [x for x in t1  if type(x)==int or type(x)==float]
print(lista2)
```

```
[1, 24.5]
```

Ejercicio. Usando compresión de listas, cree una lista que solo incluya los elementos de **frutas** que tengan más de cinco letras. ***la lista debe contener mínimo 25 elementos.