**LISTAS EN PYTHON**

Una lista es un conjunto de elementos, variables (no necesariamente del mismo tipo). Son como los arrays en cualquier otro lenguaje de programación y se definen de esta manera:

names = ['juan', 'delfina', 'carlos', 'maria']

(son totalmente recorribles con un for (foreach))

for elemento in names:

    print(elemento)

Los elementos de cada lista se manejan con **índices** comenzandos estos por el 0.

Si accedisiemos al índice -1, se obtendría el último elemento y así también podrían manejarse los índices de una list.

**RECORRIENDO LISTAS CON INTERVALOS**

Además de correr una lista con un for, se puede hacer manejando intervalos

# recorre desde el primer indice hasta el segundo, sin incluirle

# si el indice inicial es 0, puede omitirse, de esta forma

# print(names[:2])

print(names[0:2])

# recorre desde el indice indicado hasta el final

print(names[1:])

En todos los casos, el primer índice siempre se incluye, mientras que no lo hace el ultimo

**MODIFICANDO ELEMENTOS**

Para modificar el valor de un elemento en específico, se procede igual que en js, accediendo al índice requerido y utilizando **el operador de asignación**

songs = ['best of you', 'DOA', 'free me',

         'skin and bones', 'everlong', 'long road to ruin']

songs[2] = 'arlandria'

**ITERANDO LISTA**

Una lista puede iterarse con un for. Notese lo que ocurre cuando durante una iteración se modifican valores de una lista

songs = ['best of you', 'DOA', 'free me',

         'skin and bones', 'everlong', 'long road to ruin']

i = 0

for e in songs:

    e = i

    print(e)

    i += 1

print(songs)

en este caso modificamos cada elemento y lo reemplazamos por la variable **i**. Lo que ocurre al imprimir la lista fuera del ciclo for, los elementos se mantienen tal cual como los delcaramos al incio, haciendo caso omiso a la asignación ocurrida dentro del ciclo.

**ADD ELEMENTOS A LA LISTA**

El método a usar es **.append()**:

songs.append('wheels')

puedo también insertar elementos en índices específicos, de la siguiente forma:

songs.insert(1, 'walk')

IMPORTANTE: si el índice especificado está ocupado, se desplazan todos los elementos de modo que el nuevo valor tome el índice querido

Si el índice es mayor al total de elementos, se agrega al final.

**ELIMINAR ELEMENTOS**

Para eliminar elementos ocupo el método **.remove().** Es importante entender que este método toma como parámetro el valor del elemento y no el índice.

songs.remove('walk')

el método **.pop()** elimina el ultimo elemento de una lista

la sentencia **del lista[indice]** elimina el índice deseado.

El método **.clear()** limpia los elementos de una lista completamente

del **lista** elimina la variable, la lista como tal

**TUPLAS**

A diferencia de la lista, la tupla es una lista inmutable. Pueden agregarse elementos pero no modificar su orden o eliminarles.

Los índices y rangos se manejan de igual modo que las listas, al igual que el método len().

Si una tupla contiene un solo elemento, se indica con coma al final.

tupla\_individual = ('atras',)

cuando se utiliza la funcion print(), puede pasarse como parámetro lo que se agrega al final de cada línea, para eliminar el salto de línea (\n) por defecto

for i in equipos:

    print(i, end=' ')

en este código se imprime cada elemento de la tupla equipos, sin un salto de línea.

**PARSEO DE TUPLA A LISTA**

lista\_individual = list(tupla\_individual)

se puede pasar de tupla a lista con el método **list()**

**PARSEO DE LISTA A TUPLA**

tupla\_individual = tuple(lista\_individual)

el parseo inverso se declara con el método **tuple()**

Cabe resaltar que los tipos o clases de estas variables, si se pasan por el método type() devuelven **list y tuple** respectivamente

Estas conversiones pueden servir para modificar los elementos de una tupla, ya que para hacerlo es necesario que la variable sea de clase list, para finalmente volver al tipo tuple.

**SET**

Un set, a diferencia de los anteriores, no mantiene un orden. Tampoco permite elementos duplicados ni la modificación de elementos existentes. Si podemos agregar/eliminar

Los set se declaran entre {}

planetas = {'mercurio', 'venus', 'tierra', 'marte',

            'jupiter', 'saturno', 'urano', 'neptuno'}

Al igual que las colecciones anteriores, pueden usarse los métodos **del nombre** para eliminar completamente el set y **nombre.clear()** para vaciarle.

Para añadir un elemento se utiliza **.add()** con el valor del elemento por parámetro

A la hora de eliminar pueden usarse **.remove()** o **.discard()**. Ambas reciben por parámetro el valor a eliminar en el set

la ventaja de la segunda opción (discard) es que no lanza un error en caso de no encontrar el elemento pasado por parámetro.

**DICCIONARIOS**

Los diccionarios son el equivalente a los arrays asociativos en php. Compuestos por pares **key-value**

Se declaran de esta manera:

diccionario1 = {

    'primero': 'enero',

    'segundo': 'febrero',

    'tercero': 'marzo'

}

Para trabajar con los mismos, no se accede al índice, sino a la llave. Ej:

print(diccionario1['primero'])

Otra alternativa es:

print(diccionario1.get('primero'))

Para realizar una modificación, se procede de la sig manera:

diccionario1['primero'] = 'diciembre'

**ITERANDO UN DICCIONARIO**

for llave, valor in diccionario1.items():

    print(llave, valor)

con este código se accede tanto a llave como a valor. Podría dejarse este ciclo e imprimir u operar solo con lo que sea necesario.

ITERANDO LLAVES

for llave in diccionario1.keys():

    print(llave)

ITERANDO VALUES

for valor in diccionario1.values():

    print(valor)

se puede comprobar si una llave existe en un diccionario con la sentencia **in:**

print('primero' in diccionario1)

el código anterior devuelve true

**REMOVIENDO UN ELEMENTO**

Se utiliza el método **.pop**(), pasando por param() el valor de la key

**LIMPIANDO Y ELIMINANDO EL DICCIONARIO**

Diccionario1.clear()

Del diccionario1