Resumen Ayudantía 3

October 19, 2022

1 Ayudantía 3 FPI E-3

2 Listas en Python

Las listas en Python son una forma de agrupar elementos. Podemos realizar distintas operaciones con listas y los elementos de estas listas pueden ser Integers, Float, Booleanos, Strings e incluso sub-listas. **Debemos tener en consideración que una lista SIEMPRE se escribe entre paréntesis cuadrados ([]) y los elementos se separan utilizando comas (,)**

Lista	3	2	1	4
Posicion	0	1	2	3

Los elementos de una lista se cuentan desde la posición **0** hasta la posición **n-1**, donde **n** es la cantidad de elementos que contiene una lista (También conocido como el largo de una lista). Podemos utilizar la función **len(nombre_lista)** para poder obtener el largo de una lista, donde **nombre_lista** es el nombre de la variable que contiene a la lista.

2.1 Definir una lista vacía

En Python podemos definir una lista inicialmente vacía igualando la variable a utilizar a **[]**

[1]:
```
# En Python podemos definir una lista inicialmente vacia igualando la variable a␣
↪utilizar a []
lista_vacia = []
print("Una lista vacia es: ", lista_vacia)

# Tambien podemos definir una lista que ya viene con elementos
lista = [3,4,2]
print("Una lista con elementos es: ", lista)
```

```
Una lista vacia es:  []
Una lista con elementos es:  [3, 4, 2]
```

2.2 Unir dos listas y repetir los elementos de una lista

Podemos unir dos listas haciendo una "suma" (+) de ellas. Podemos repetir elementos de una lista utilizando la multiplicación.

[2]:
```
lista1 = [3,4,2]                          # Se crea una lista con 3, 4 y 2
lista2 = [9,5,6]                          # Se crea una lista con 9, 5 y 6
print("La lista 1 es: ", lista1)
print("La lista 2 es: ", lista2)
lista3 = lista1 + lista2                  # Se obtiene la union de ambas listas
print("La union de ambas listas es: ", lista3)
lista4 = lista1 * 3                       # Se repiten 3 veces los elementos de␣
 ↪lista 1
print("La lista que contiene los elementos de lista 1 repetidos 3 veces es: ",␣
 ↪lista4)
```

```
La lista 1 es:  [3, 4, 2]
La lista 2 es:  [9, 5, 6]
La union de ambas listas es:  [3, 4, 2, 9, 5, 6]
La lista que contiene los elementos de lista 1 repetidos 3 veces es:  [3, 4, 2,
3, 4, 2, 3, 4, 2]
```

2.3 Acceder a un elemento específico de una lista

Para acceder a un elemento de una lista, podemos utilizar **nombre_lista[x]**, donde **nombre_lista** corresponde a la variable que contiene a la lista y **x** corresponde a la posición del elemento al que queremos acceder.

[3]:
```
lista = [3,4,2]                           # Se crea una lista con 3, 4 y 2
print("La lista es: ", lista)
elemento = lista[1]                       # Se accede al elemento de la posicion 1␣
 ↪de la lista (4)
print("Accediendo al segundo elemento de la posicion 1 de la lista: ", elemento)
```

```
La lista es:  [3, 4, 2]
Accediendo al segundo elemento de la posicion 1 de la lista:  4
```

2.4 Modificar un elemento específico de una lista

Para modificar un elemento especifico de una lista, podemos utilizar la forma anterior de acceder a un elemento especifico de una lista, pero esta vez igualando al elemento al que queremos que se modifique.

[4]:
```
lista = [3,4,2]                           # Se crea una lista con 3, 4 y 2
print("La lista es: ", lista)
```

```
elemento = lista[1]                    # Se accede al elemento de la posicion 1␣
 ↪de la lista (4)
print("Accediendo al segundo elemento de la posicion 1 de la lista: ", elemento)
lista[1] = 99
print("La lista modificada es: ", lista)
elemento = lista[1]                    # Se accede al elemento de la posicion 1␣
 ↪de la lista (99)
print("Accediendo al segundo elemento de la posicion 1 de la lista: ", elemento)
```

```
La lista es:  [3, 4, 2]
Accediendo al segundo elemento de la posicion 1 de la lista:  4
La lista modificada es:  [3, 99, 2]
Accediendo al segundo elemento de la posicion 1 de la lista:  99
```

2.5 Agregar elementos a una lista

Podemos utilizar el método **nombre_lista.append(x)** para agregar un elemento a una lista, donde **nombre_lista** es el nombre de la variable que contiene a la lista y **x** es el elemento que queremos agregar a la lista. **Append** agrega el elemento automáticamente al final de la lista (O en otras palabras, en la última posición)

[5]:
```
# Utilizamos .append() para agregar elementos a una lista

lista = []                             # Utilizamos [] para definir una lista␣
 ↪vacia
print("La lista inicial es: ", lista)
lista.append(2)                        # Agregamos el numero 2 a la lista
print("La lista luego del primer append es: ", lista)
lista.append(3)                        # Agregamos el numero 3 a la lista
print("La lista luego del segundo append es: ", lista)
```

```
La lista inicial es:  []
La lista luego del primer append es:  [2]
La lista luego del segundo append es:  [2, 3]
```

También podemos utilizar el método **nombre_lista.insert(x,y)**, donde **nombre_lista** es el nombre de la variable que contiene la lista, **x** es la posición de la lista donde se quiere agregar un elemento e **y** corresponde al elemento a agregar.

[6]:
```
lista = [3,4,2]                              # Se define una lista con 3, 4 y 2
print("Lista antes de insert", lista)
lista.insert(2, 10)                          # Se agrega un 10 en la penultima␣
 ↪posicion de la lista
print("Lista despues de insert", lista)
```

```
Lista antes de insert [3, 4, 2]
Lista despues de insert [3, 4, 10, 2]
```

Notemos que con estas operaciones y la gran mayoría de las que vienen a continuación, no es necesario igualar a una nueva variable. Una vez se ejecuta la línea de código, Python hace el

cambio dentro de la variable original!

2.6 Quitar elementos de una lista

El método **nombre_lista.remove(x)** nos permite eliminar un elemento de una lista, donde **nombre_lista** es el nombre de la variable que contiene a la lista y **x** es el elemento a eliminar de la lista. **Remove** elimina solamente la primera instancia que encuentre de un elemento, por lo que no elimina todas las instancias inmediatamente!

[7]:
```
lista = [3,4,2,3,4,2]                          # Se define una lista con 3, 4,␣
 ↪2, 3, 4 y 2
print("Lista antes de los remove", lista)
lista.remove(3)                                # Se elimina el primer 3
print("Lista luego del primer remove", lista)
lista.remove(3)                                # Se elimina el segundo 3
print("Lista luego del segundo remove", lista)
```

```
Lista antes de los remove [3, 4, 2, 3, 4, 2]
Lista luego del primer remove [4, 2, 3, 4, 2]
Lista luego del segundo remove [4, 2, 4, 2]
```

También podemos utilizar el método **nombre_lista.pop(x)**, donde **nombre_lista** es el nombre de la variable que contiene a la lista y **x** es la posición del elemento que se quiere eliminar. **Si no se especifica un x, elimina al último elemento de la lista.**

[8]:
```
lista = [3,4,5,2,3]                            # Se crea una lista con 3, 4, 5, 2␣
 ↪y 3
print("Lista antes de los pop", lista)
lista.pop()                                    # Se elimina el ultimo elemento de␣
 ↪la lista (3)
print("Lista luego del primer pop (Elimina el último 3)", lista)
lista.pop(3)                                   # Se elimina el elemento de la␣
 ↪posicion 3 (2)
print("Lista luego del segundo pop (Elimina el 2)", lista)
```

```
Lista antes de los pop [3, 4, 5, 2, 3]
Lista luego del primer pop (Elimina el último 3) [3, 4, 5, 2]
Lista luego del segundo pop (Elimina el 2) [3, 4, 5]
```

2.7 Contar cuantas veces se repite un elemento de una lista

Para contar cuantas veces se repite un elemento en una lista, podemos utilizar el método **nombre_lista.count(x)**, donde **nombre_lista** es el nombre de la variable que contiene a la lista y x es el elemento que queremos saber cuántas veces se repite dentro de la lista

[9]:
```
lista = [3,4,2,2,2,2,2,2,1]                    # Se crea una lista con 3, 4, 2,␣
 ↪2, 2, 2, 2, 2 y 1
print("La lista es: ", lista)
```

```
contador = lista.count(3)                          # Se cuenta las veces que se␣
 ↪repite el 3
print("Veces que se repite el 3: ", contador)
contador = lista.count(2)                          # Se cuenta las veces que se␣
 ↪repite el 2
print("Veces que se repite el 2: ", contador)
```

```
La lista es:  [3, 4, 2, 2, 2, 2, 2, 2, 1]
Veces que se repite el 3:  1
Veces que se repite el 2:  6
```

2.8 Obtener la posición de un elemento de una lista

Podemos utilizar el método **nombre_lista.index(x)** para obtener la posición de un elemento de una lista, donde **nombre_lista** es el nombre de la variable que contiene a la lista y **x** es el elemento al cual le queremos averiguar la posición dentro de la lista. Si un elemento se repite, entonces entrega la primera posición donde se encuentra el elemento.

[10]:
```
lista = [3,4,2,4,2]                                # Se crea una lista con 3, 4, 2,␣
 ↪4 y 2
print("La lista es:", lista)
posicion = lista.index(2)                          # Se obtiene la posicion del␣
 ↪primer 2 (2)
print("Posicion de el 2 en la lista:", posicion)
```

```
La lista es: [3, 4, 2, 4, 2]
Posicion de el 2 en la lista: 2
```

2.9 Ordenar una lista, obtener su máximo, su mínimo y la suma de elementos

Podemos ordenar una lista de elementos utilizando el método **nombre_lista.sort()**, donde **nombre_lista** es el nombre de la variable que contiene a la lista. Además, podemos utilizar las funciones **max(nombre_lista)**, **min(nombre_lista)** y **sum(nombre_lista)** para poder obtener el máximo, mínimo y la suma de elementos correspondientes de una lista.

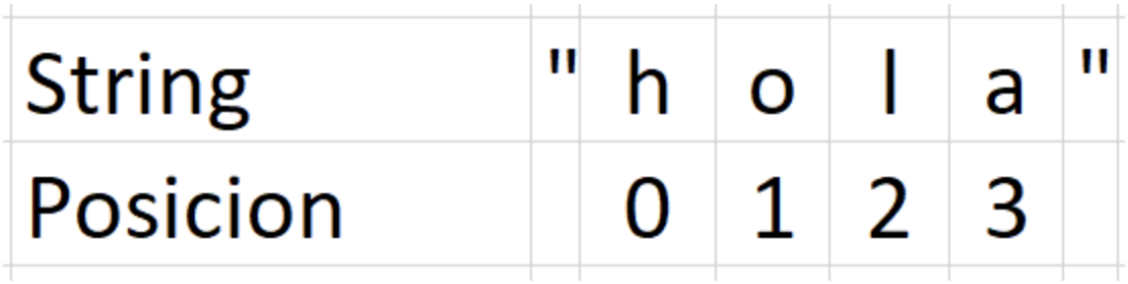
[11]:
```
lista = [9,6,3,1,2]                                # Se crea una lista con␣
 ↪9, 6, 3, 1 y 2
print("La lista original es: ", lista)
lista.sort()                                       # Se ordena la lista␣
 ↪utilizando sort()
print("La lista ordenada de menor a mayor es: ", lista)
maximo = max(lista)                                # Se obtiene el máximo␣
 ↪de la lista
print("El máximo número de la lista es: ", maximo)
minimo = min(lista)                                # Se obtiene el minimo␣
 ↪de la lista
print("El minimo numero de la lista es: ", minimo)
```

```
suma = sum(lista)                                    # Se obtiene la suma de␣
 ↪los elementos de la lista
print("La suma de los elementos de la lista es: ", suma)
```

```
La lista original es:  [9, 6, 3, 1, 2]
La lista ordenada de menor a mayor es:  [1, 2, 3, 6, 9]
El máximo número de la lista es:  9
El minimo numero de la lista es:  1
La suma de los elementos de la lista es:  21
```

3 Strings en Python

Los Strings corresponden a caracteres o cadenas de textos. En palabras simples, son letras o palabras. Se escriben utilizando comillas dobles o comillas simples. Comparten algunas características con las listas. También se cuentan desde 0 hasta **n-1** elementos y podemos obtener el largo del string utilizando **len(nombre_string)**, donde **nombre_string** es el nombre de la variable que contiene al string.



3.1 Unir dos strings

Similar a las listas, podemos utilizar la suma para unir dos strings. Los strings se unen exactamente en la última y primera posición del primer y segundo string respectivamente.

[12]:
```
string1 = "hola "                              # Se crea el string "hola "
print("El primer string es: ", string1)
string2 = "mundo"                              # Se crea el string "mundo"
print("El segundo string es: ", string2)
union = string1 + string2                      # Se crea la union de ambos strings␣
 ↪()"hola mundo")
print("La union de ambos strings es: ", union)
```

```
El primer string es:  hola
El segundo string es:  mundo
La union de ambos strings es:  hola mundo
```

Nótese que en el primer string el último elemento es un espacio para que asi haya una separación entre hola y mundo. De lo contrario, el resultado hubiera sido **holamundo** (Sin el espacio entremedio)

3.2 Repetir un string

Similar a las listas, utilizando la multiplicación, podemos repetir un string x veces.

[13]:
```
string1 = "hola!"                    # Se crea el string "hola!"
string2 = string1 * 5                # Se repite el string "hola!" 5 veces.
print("El string original es: ", string1)
print("El string repetido 5 veces es: ", string2)
```

```
El string original es:  hola!
El string repetido 5 veces es:  hola!hola!hola!hola!hola!
```

3.3 Acceder a un elemento específico de un string

Similar a las listas, podemos acceder a un elemento especifico de un string utilizando la forma **nombre_string[x]**, donde **nombre_string** es el nombre de la variable que contiene al string y **x** es la posición del elemento al que se quiere acceder.

[14]:
```
string1 = "hola mundo"               # Se crea el string "hola mundo"
elemento = string1[5]                # Se accede al elemento de la posicion 5 del␣
 ↪string ("m")
print("El string original es: ", string1)
print("El elemento en la posicion 5 del string es: ", elemento)
```

```
El string original es:  hola mundo
El elemento en la posicion 5 del string es:  m
```

A diferencias de las listas, no podemos utilizar lo anterior para modificar un elemento especifico de una lista. Una alternativa a eso es transformar el string a lista y luego transformarlo a String, algo que se mostrará más adelante

3.4 Transformar el string a minúsculas o a mayúsculas

Podemos utilizar los métodos **nombre_string.lower()** o **nombre_string.upper()** si queremos transformar el string a minúsculas o a mayúsculas. **nombre_string** corresponde al nombre de la variable que contiene el string.

[15]:
```
string1 = "Touhou SCARLET weather RhApSoDy"                    # Se crea el string␣
 ↪"Touhou SCARLET weather RhApSoDy"
print("El string original es:", string1)
print("El string con el metodo upper es: ", string1.upper()) # Se obtiene el␣
 ↪string "TOUHOU SCARLET WEATHER RHAPSODY"
print("El string con el metodo lower es: ", string1.lower()) # Se obtiene el␣
 ↪string "touhou scarlet weather rhapsody"
```

```
El string original es: Touhou SCARLET weather RhApSoDy
El string con el metodo upper es:  TOUHOU SCARLET WEATHER RHAPSODY
El string con el metodo lower es:  touhou scarlet weather rhapsody
```

3.5 Eliminar espacios y saltos de línea de un string

Podemos utilizar el método **nombre_string.strip()** para eliminar los saltos de línea (****) o los espacios al inicio y al final de un string.

[16]:
```
string1 = "    Plus    Ultra   \n"                    # Se crea el string "
 ↪Plus    Ultra   \n"
print("El string original es: ", string1)
string2 = string1.strip()                             # Se obtiene el string "Plus
 ↪   Ultra"
print("El string modificado con strip es: ", string2)
```

```
El string original es:      Plus    Ultra

El string modificado con strip es:  Plus    Ultra
```

Nótemos que **si es necesario guardar el resultado de strip en una nueva variable!**

3.6 Algunas verificaciones importantes con Strings

- **nombre_string.isupper()**: Retorna verdadero si un String está solo compuesto por mayúsculas. Falso en caso contrario.
- **nombre_string.islower()**: Retorna verdadero si un String está solo compuesto por minúsculas. Falso en caso contrario.
- **nombre_string.isalpha()**: Retorna verdadero si un String está solo compuesto por letras y/o caracteres (Sin números). Falso en caso contrario
- **nombre_string.isdigit()**: Retorna verdadero si un String está solo compuesto por números (dígitos). Falso en caso contrario

4 Consultas y Dudas que han tenido trabajando con Strings y Listas

4.1 1. ¿Como transformo un string a una lista?

Para esto podemos utilizar el método **nombre_lista.split(x)**, donde **nombre_string** es el nombre de la variable que contiene el string y **x** es el elemento donde se quiere que se separen los elementos para transformarlos a elementos de una lista. Si no se especifica un x, entonces se consideran los espacios como el elemento separador.

[17]:
```
string1 = "hola mundo"                                # Se crea el string "hola
 ↪mundo"
lista = string1.split()                               # Se transforma el string
 ↪a lista, separando los elementos por " "
print("El string 1 original es: ", string1)
print("El string 1 transformado a lista es: ", lista)

string2 = "19/10/2022"                                # Se crea el string "19/10/
 ↪2022"
lista2 = string2.split("/")                           # Se transforma el string
 ↪a lista, separando los elementos por "/"
```

```
print("El string 1 original es: ", string2)
print("El string 1 transformado a lista es: ", lista2)
```

```
El string 1 original es:  hola mundo
El string 1 transformado a lista es:  ['hola', 'mundo']
El string 1 original es:  19/10/2022
El string 1 transformado a lista es:  ['19', '10', '2022']
```

4.2 2. ¿Como transformo una lista a un string?

Para esto podemos utilizar el método **elemento.join(nombre_lista)**, donde **elemento** es un elemento que se unirá con cada elemento de la lista y **nombre_lista** es el nombre de la variable que contiene a la lista

[18]:
```
lista1 = ["hola " , "como ", "estas"]              # Se crea una lista con "hola␣
 ↪", "como ", "estas"
string1 = "".join(lista1)                          # Se transforma la lista a␣
 ↪string
print("La lista original es: ", lista1)
print("La lista transformada a string es: ", string1)
```

```
La lista original es:  ['hola ', 'como ', 'estas']
La lista transformada a string es:  hola como estas
```

4.3 3. ¿Para que funciona el eval()?

eval() funciona para que Python pueda identificar mediante como está escrita una entrada a que tipo de dato corresponde el elemento ingresado. Esto es esencial si queremos ingresar listas por entradas y ahorrarnos posibles problemas al transformar de strings a listas.

[19]:
```
entrada1 = eval(input("Ingrese un elemento: "))            # Se pido que se␣
 ↪ingrese un elemento por teclado
print("El tipo de dato de la entrada es: ", type(entrada1))
```

```
Ingrese un elemento: ["Python", "C", "Java"]
El tipo de dato de la entrada es:  <class 'list'>
```

Como Python ve que el elemento ingresado está escrito con paréntesis cuadrados al inicio y al final del elemento y además de que tenemos elementos separados por coma (,), entonces inmediatamente sabe que se trata de una lista y eval() retorna la lista escrita tal cual como la ingresamos.

4.4 4. ¿Que son las "banderas" (flags) y como funcionan?

Las banderas (o flags, como se le puede encontrar en algunos códigos) son solamente variables de tipo Booleano. Usualmente parten con un valor y a medida que se va ejecutando el código cambian a otro valor (Se sube o se baja la "bandera", dependiendo si es True o False. Por eso se les llama como bandera).

[21]:
```
lista = eval(input("Ingrese una lista por entrada: ")) # Se pide que se ingrese␣
 ↪un elemento por teclado
flag_dos = False                                       # Declaramos una flag␣
 ↪(Variable booleana) inicializada en False
i = 0                                                  # Se declara un contador␣
 ↪para recorrer
while i < len(lista):                                  # Mientras i sea menor al␣
 ↪largo de la lista
    elemento_actual = lista[i]                         # Se obtiene el elemento␣
 ↪en la posicion i de la lista
    if elemento_actual == 2:                           # Si el elemento es igual␣
 ↪a 2
        flag_dos = True                                # La bandera se vuelve␣
 ↪verdadera
    i += 1                                             # Se aumenta i para que␣
 ↪continue el ciclo While

if(flag_dos):                                          # Si se encontro 2
    print("Se pudo encontrar el numero dos")
else:                                                  # Si no se encontro 2
    print("No se encontro el numero dos")
```

```
Ingrese una lista por entrada: [1,3,5,6,11,4,5,677,3,2,1,6,3,13]
Se pudo encontrar el numero dos
```

Como podemos observar en el código anterior, la bandera comienza con el valor Falso y luego cuando se encuentra el número dos, cambia a verdadero. Es ahí donde "se sube la bandera". Flag, o Bandera, no es nada más que un nombre a como usualmente se le conocen a estas variables Booleanas. Pero debemos no asustarnos y considerar que solo son Variables que tiene el valor True o False inicialmente y que luego en algún punto del programa, cambian de valor. Nada más, y nada menos.

[ ]:
```

```