



# Clase 12 – Listas de listas

IIC1103 Sección 9 – 2019-2

Profesor: Felipe López

10-10-2019

# Resumen de la clase

- Las listas son variables en Python que guardan una secuencia de valores. Estos valores se denominan *elementos*.
- Por ejemplo:

```
lista1 = [1,2,3]
```

- Podemos crear listas vacías también, de la siguiente forma:

```
lista2 = []
```

- Para poder obtener algún elemento de una posición determinada en una lista, se hace de la siguiente forma:

```
elemento_en_posición_x = variable_de_tipo_lista[x]
```

Donde `elemento_en_posición_x` es la variable que almacenará el elemento que está en la posición `x` de la lista que está almacenada en la `variable_de_tipo_lista`.

# Resumen de la clase

- **Slice:** Es la acción de obtener una nueva lista desde otra lista. Esta nueva lista tendrá ciertos elementos que dependerá de cómo hicimos el *slice*. Si  $l$  es una lista, se hace de la siguiente manera:

$l[i:j]$

Donde obtenemos los elementos desde la posición  $i$  hasta  $j-1$ .

- **Edición de elementos:** Podemos editar elementos de una lista. Si  $l$  es una lista y  $k$  es una posición de la lista:

$l[k]=\text{nuevo elemento}$

Donde la información que estaba en la posición  $k$  se sobrescribió con el nuevo elemento. Recordemos que este nuevo elemento puede ser de cualquier tipo.

# Resumen de la clase

- **Agregar elementos:** Se pueden agregar nuevos elementos a una lista. Si `l` es una lista, se hace de la siguiente manera:

```
l.append(x)
```

Donde se agrega el elemento `x` a la lista.

- **Cantidad de elementos de una lista:** Se puede obtener la cantidad de elementos de una lista. Si `l` es una lista, se hace de la siguiente manera:

```
len(l)
```

- **for sobre una lista:** Podemos “recorrer” todos los elementos de una lista mediante un `for`. Si `l` es una lista, se hace de la siguiente manera:

```
for i in s1:
```

```
    i #i representa a cada uno  
    #de los elementos de l.
```

# Resumen de la clase

- **split()**: Si tenemos un *string* *s* y un *string* *x*:

`s.split(x)`

Esta operación separa al *string* *s* según el *string* *x*. Es decir, busca la o las ocurrencia(s) de *x* en *s* y cada vez que encuentra una, “corta” a *s*.

- **join()**: Si tenemos una lista de *strings* llamada *l* y un *string* *x*:

```
s = x.join(l)
```

Donde *s* es una variable que almacenará el *string* resultante al juntar los elementos de la lista *l* por medio del *string* *x*.

# Resumen de la clase

- **insert()**: Si `l` es una lista, `i` es un índice entre `0` y `len(l)` y `x` es un elemento que queremos agregar a esta lista, podemos ejecutar:

```
l.insert(i,x)
```

De esta forma, se agrega `x` en la posición `i`

¿Y si hay elementos después de la posición `i`? Estos se “desplazan” para dejar espacio para `x`.

- **pop()**: Si `l` es una lista, e `i` es un índice entre `0` y `len(l)`:

```
elem_i = l.pop(i)
```

`elem_i` es una variable que contiene al elemento de la lista `l` en la posición `i` antes de hacer el `pop()`. Después de este, la lista ya no contiene este elemento.

# Resumen de la clase

- **in:** Si `l` es una lista, y `x` es un elemento que podría estar en `l`

`x in l`

Devolverá `True` si `x` efectivamente está dentro de la lista `l`, y `False` en caso contrario.

- **+:** Si `l1` y `l2` son listas:

`nueva_lista = l1 + l2`

`nueva_lista` es una lista cuyos elementos son los de `l1` y `l2`.

# Contenidos

1. Listas de listas
2. Creación de listas de listas
3. Obtener elementos
4. Editar elementos
5. Largo de listas de listas
6. Ejercicios



# Listas de listas

- Recordemos que las listas pueden almacenar datos de cualquier tipo.

# Listas de listas

- Recordemos que las listas pueden almacenar datos de cualquier tipo.
- Además, pueden tener cualquier largo.

# Listas de listas

- Recordemos que las listas pueden almacenar datos de cualquier tipo.
- Además, pueden tener cualquier largo.
- Vimos además que listas es en realidad un tipo de dato.

# Listas de listas

- Recordemos que las listas pueden almacenar datos de cualquier tipo.
- Además, pueden tener cualquier largo.
- Vimos además que listas es en realidad un tipo de dato.
- Por lo tanto ¿Podríamos crear una lista *de listas*?

# Listas de listas

- Recordemos que las listas pueden almacenar datos de cualquier tipo.
- Además, pueden tener cualquier largo.
- Vimos además que listas es en realidad un tipo de dato.
- Por lo tanto ¿Podríamos crear una lista *de listas*?
- **¡Sí!**

# Listas de listas

Podemos definir una lista de listas de la siguiente manera:

```
matriz = [[“a”,”b”,”c”],[“d”,”e”,”f”],[“g”,”h”,”i”]]
```

La variable `matriz` es una lista que tiene 3 elementos. Estos tres elementos son listas. Cada una de estas listas tiene a su vez tres elementos:

- El primer elemento es una lista que contiene los elementos a,b, y c.
- El segundo elemento es una lista que contiene los elementos d,e, y f.
- El tercer elemento es una lista que contiene los elementos g,h, y i.

# Listas de listas

Detengámonos un poco a analizar la variable matriz.

El nombre de esta variable no es azaroso. Esta lista de listas puede representarse por la siguiente figura:

a	b	c
d	e	f
g	h	i

Es decir, una lista de listas se asemeja a una matriz. Esto significa que tendrá **filas y columnas**.

# Listas de listas

¿Qué significa que una matriz tenga filas y columnas? Que es posible encontrar elementos de esta matriz sabiendo en qué fila y en qué columna están. En este caso podemos enumerar las filas y columnas del 0 al 2.

	0	1	2
0	a	b	c
1	d	e	f
2	g	h	i

Se enumeran las filas y columnas del 0 al 2 ya que recordemos que en Python se empieza a contar desde el 0.



# Listas de listas

Por ejemplo, si quisiéramos saber obtener el elemento “f”, sabemos que está en la fila 1 y la columna 2.

	0	1	2
0	a	b	c
1	d	e	f
2	g	h	i

# Obtener elementos

Por ejemplo, si quisiéramos obtener el elemento “f”, sabemos que está en la fila 1 y la columna 2.

Podemos hacer esto en Python de la siguiente forma:

	0	1	2
0	a	b	c
1	d	e	f
2	g	h	i

```
matriz = [["a", "b", "c"], ["d", "e", "f"], ["g", "h", "i"]]  
print(matriz[1][2])
```

# Obtener elementos

De forma general, podemos obtener un elemento de una variable  $M$  que representa a una lista de listas de la siguiente manera:

$$M[i][j]$$

Donde  $i$  y  $j$  representa a una fila y a una columna específica de la lista de listas (recordemos que las filas y columnas se empiezan a contar desde 0).

# Crear lista de listas

- ¿Podemos crear una lista de  $n \times m$  con ciertos valores en cada celda?
- Digamos que el valor que queremos que esté en cada celda es  $k$ . Para hacerlo, podemos ocupar el siguiente comando:

`M = [[k for x in range(m)] for x in range(n)]`

- Por ejemplo:

```
M = [[0 for x in range(4)] for x in range(3)]  
print(M)
```

```
[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
```

# Edición de elementos

¿Y si quisiéramos cambiar el elemento que tiene el valor “h” por “z”?

	0	1	2
0	a	b	c
1	d	e	f
2	g	h	i

“h” está en la fila 2 y la columna 1. Para cambiarlo podemos hacerlo de la siguiente manera:

```
matriz = [["a", "b", "c"], ["d", "e", "f"], ["g", "h", "i"]]  
matriz[2][1] = "z"  
print(matriz)
```

```
[['a', 'b', 'c'], ['d', 'e', 'f'], ['g', 'z', 'i']]
```

# Edición de elementos

De forma general, podemos modificar un elemento de una variable  $M$  que representa a una lista de listas de la siguiente manera:

$$M[i][j] = k$$

Donde  $i$  y  $j$  representa a una fila y a una columna específica de la lista de listas (recordemos que las filas y columnas se empiezan a contar desde 0).  $k$  representa el nuevo elemento que modificará lo que estaba en la fila  $i$  y columna  $j$ .

# Largo de listas de listas

Imaginemos que recibimos una matriz  $M$  ya creada. No obstante, no sabemos a priori cuántas filas o columnas tiene. En forma general, si tenemos una matriz  $M$  con la misma cantidad de columnas por fila:

- Para poder saber el número de filas:  $\text{len}(M)$
- Para poder saber el número de columnas:  $\text{len}(M[0])$

# Largo de listas de listas

Veamos un ejemplo:

```
matriz2 = [["a","b","c"],["d","e","f"],["g","h","i"],["j","k","l"]]  
print("Filas:",len(matriz2))  
print("Columnas:",len(matriz2[0]))
```

```
[['a', 'b', 'c'], ['d', 'e', 'f'], ['g', 'z', 'i']]  
Filas: 4  
Columnas: 3
```



# Largo de listas de listas

¿Y si la cantidad de columnas por fila no es igual? Puede pasar.

En este caso, cada lista dentro de la lista principal tendría un largo distinto.

Por ejemplo:

```
matriz2 = [ ["a", "b", "c"], ["d", "e", "f", "z"], ["g", "h"], ["j"] ]  
print(len(matriz2[1]))  
print(len(matriz2[3]))
```

```
4
```

```
1
```

# Ejercicio

- El usuario ingresará la información de varios estudiantes (no se sabe cuántos). Los atributos de cada persona son:
  - Nombre
  - Fecha de nacimiento (en formato día/mes/año)
  - Casado (representado con True o False)
  - PPA (como float)
  - Ramos aprobados (lista de strings de largo distinto para cada persona)
- Cada persona viene en el siguiente formato:  
Juan Andres Pérez Rojas;31/07/1990;True;5.9;["Cálculo I","Cálculo II","Cálculo III","Algebra Lineal"]

# Ejercicio

1. Agregue estos datos a una lista de listas. El usuario ingresara END cuando quiera terminar de ingresar datos de estudiantes.

# Ejercicio

1. Agregue estos datos a una lista de listas. El usuario ingresara END cuando quiera terminar de ingresar datos de estudiantes.

```
M = []

datos_estudiante = input()

while datos_estudiante != "END":
    sListAux = datos_estudiante.split(";")
    print(sListAux)
    nombre = sListAux[0]
    fecha_nac = sListAux[1]
    casado = bool(sListAux[2])
    ppa = float(sListAux[3])
    ramos = sListAux[4].strip("[").strip("]").split(",")
    l_aux = [nombre, fecha_nac, casado, ppa, ramos]
    M.append(l_aux)

    datos_estudiante = input()
```

# Ejercicio

2. Cree una columna después de la fecha de nacimiento que represente a la edad de cada persona. Puede asumir que M ya está creada (de la parte 1).

# Ejercicio

2. Cree una columna después de la fecha de nacimiento que represente a la edad de cada persona. Puede asumir que M ya está creada (de la parte 1).

```
for estudiante in M:  
    fecha_nac = estudiante[1]  
    año = int(fecha_nac.split("/")[2])  
    edad = 2018-año  
    estudiante.insert(2, edad)
```

# Bibliografía

- <http://runest.ing.puc.cl/list.html#listas-de-listas>

# Links

- <https://repl.it/@FelipeLopez/IIC1103Listasdelistas> que contiene todos los ejemplos de la clase.



# Resumen de la clase

Podemos definir una lista de listas de la siguiente manera:

`matriz =`

```
[[“a”, “b”, “c”], [“d”, “e”, “f”], [“g”, “h”, “i”]]
```

La variable `matriz` es una lista que tiene 3 elementos. Estos tres elementos son listas. Cada una de estas listas tiene a su vez tres elementos:

El primer elemento es una lista que contiene los elementos `a`, `b`, y `c`.

El segundo elemento es una lista que contiene los elementos `d`, `e`, y `f`.

El tercer elemento es una lista que contiene los elementos `g`, `h`, y `i`.

	0	1	2
0	a	b	c
1	d	e	f
2	g	h	i

De forma general, podemos obtener un elemento de una variable `M` que representa a una lista de listas de la siguiente manera:

$$M[i][j]$$

Donde `i` y `j` representa a una fila y a una columna específica de la lista de listas (recordemos que las filas y columnas se empiezan a contar desde 0).

# Resumen de la clase

¿Podemos crear una lista de  $n \times m$  con ciertos valores en cada celda?

Digamos que el valor que queremos que esté en cada celda es  $k$ . Para hacerlo, podemos ocupar el siguiente comando:

```
M = [[k for x in range(m)] for x in range(n)]
```

De forma general, podemos modificar un elemento de una variable  $M$  que representa a una lista de listas de la siguiente manera:

$$M[i][j] = k$$

Donde  $i$  y  $j$  representa a una fila y a una columna específica de la lista de listas (recordemos que las filas y columnas se empiezan a contar desde 0).  $k$  representa el nuevo elemento que modificará lo que estaba en la fila  $i$  y columna  $j$ .

Imaginemos que recibimos una matriz  $M$  ya creada. No obstante, no sabemos a priori cuántas filas o columnas tiene. En forma general, si tenemos una matriz  $M$  con la misma cantidad de columnas por fila:

Para poder saber el número de filas: `len(M)`

Para poder saber el número de columnas: `len(M[0])`



# Clase 12 – Listas de listas

IIC1103 Sección 9 – 2019-2

Profesor: Felipe López

10-10-2019