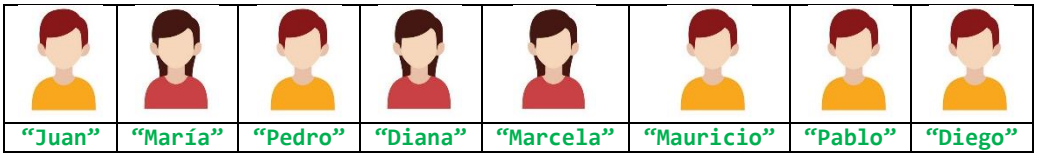
Nivel 3 – Archivos, listas, instrucciones repetitivas

Preparación: Listas

Las listas son una estructura que nos permite almacenar información de manera lineal; es decir un elemento después de otro. Suponga que queremos almacenar los nombres de los estudiantes de una clase, una forma de hacerlo es usando una lista.

clase



Crear listas

Supongamos que la lista se llama **clase**. Existen dos formas en que podemos crear la lista del ejemplo en Python. Tenga en cuenta que ambas formas son válidas.

Forma 1	Forma 2
Crear una lista vacía y posteriormente agregar los elementos a la lista; tenga en cuenta que el orden en que se agreguen los elementos será el orden en que estos se almacenen en la lista.	Crear una lista que desde su creación contenga los elementos; tenga en cuenta que el orden en que se agreguen los elementos será el orden en que estos se almacenen en la lista.

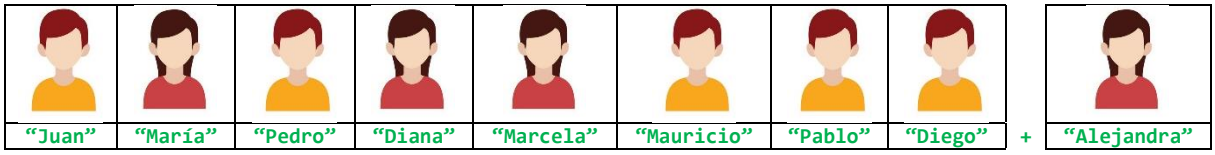
```
clase = []
clase.append("Juan")
clase.append("Maria")
clase.append("Pedro")
clase.append("Diana")
clase.append("Marcela")
clase.append("Mauricio")
clase.append("Pablo")
clase.append("Diego")
```

```
clase = ["Juan", "Maria", "Pedro",
"Diana", "Marcela", "Mauricio", "Pablo",
"Diego"]
```

Agregar nuevos elementos a una lista

A una lista que ya hemos creado podemos agregar nuevos elementos usando la operación **append**. Esta operación agregará el nuevo elemento al final de la lista.

clase



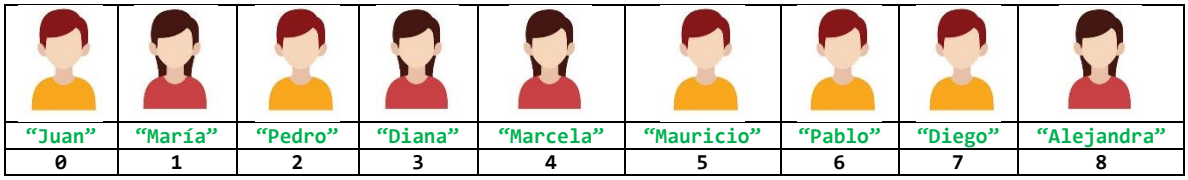
Esta operación la escribimos en código como:

```
clase.append("Alejandra")
```

Consultar elementos de una lista

Para acceder y consultar los elementos de una lista debemos usar la posición en que se encuentran, siempre teniendo en cuenta que la primera posición será la posición **0**. Esto quiere decir que si tenemos una lista de 9 elementos el ultimo elemento estará en la posición 8 (cantidad de elementos menos 1).

clase



Por ejemplo, si quisiéramos conocer el nombre del segundo alumno en la lista escribiremos

```
clase[1]
```

Si quisiera almacenar este nombre en alguna variable para después poder hacer alguna operación con él, podemos escribir:

```
nombre = clase[1]
```

Cantidad de elementos en una lista

Si queremos conocer el tamaño de una lista podemos usar la función **len** así:

```
len(clase)
```

Conociendo estas operaciones básicas podemos, por ejemplo, imprimir todos los elementos de una lista. Esto podemos hacerlo de al menos 3 formas usando las instrucciones repetitivas que conocemos

Forma 1	Forma 2	Forma 3
Usando la instrucción while	Usando la instrucción for in range	Usando la instrucción for; *esta forma la revisaremos en siguientes clases.*

`i = 0`
`while i < len(clase):`
`  print(clase[i])`
`  i+=1`
	`for i in range(len(clase)):`
	`  print(clase[i])`
		`for nombre in clase:`
		`  print(nombre)`

Actividad – *shows* de Netflix

Objetivos

1. Practicar las funciones asociadas al manejo de archivos.
2. Practicar la creación, manipulación y uso de las funciones asociadas a listas.
3. Practicar la construcción de instrucciones repetitivas o iterativas

Conociendo Netflix

En esta actividad el objetivo es abrir un archivo excel separado por comas (csv) para cargar la información contenida en él y almacenarla en una lista. El archivo de texto contiene información de 997 *shows* que se encuentran en Netflix.

Cada línea del archivo contiene la siguiente información de un *show*: `titulo`, `clasificación`, `nivel de la clasificación`, `descripción de la clasificación`, `año de estreno`, `puntaje de los usuarios`, `tamaño del puntaje de los usuarios`.

La primera línea del archivo contiene los encabezados separados por punto y coma ";".

```
title;rating;ratingLevel;ratingDescription;release year;user rating score;user rating
size
```

De la segunda línea en adelante, cada línea tiene la información de un *show*, por ejemplo:

```
White Chicks;PG-13;crude and sexual humor, language and some drug content;80;2004;82;80
```

Actividad 1 - cargar

En esta actividad el objetivo es abrir un archivo de texto para leer la información de los *shows* de Netflix.

Usted debe completar la función **`cargar()-> list`** que se encuentra en el archivo **`administrador_shows.py`**. La función debe retornar una lista con la información de todos los *shows* en el archivo.

El valor de retorno de la función es una lista de diccionarios. Es decir, en cada posición de la lista se encuentra un diccionario con la información de un show. El diccionario tiene la siguiente estructura:

```
{'title': 'Back to the Secret Garden',
'rating': 'G',
'release year': 2002}
```

Note que, aunque el archivo trae más información, solo construimos los diccionarios con 3 datos de cada película.

A continuación, se presenta un ejemplo de la lista con 4 shows:

```
[
  {'title': 'Balto 2: Wolf Quest', 'rating': 'G', 'release year': 2001},
  {'title': 'Balto', 'rating': 'G', 'release year': 1995},
  {'title': 'Back to the Secret Garden', 'rating': 'G', 'release year': 2002},
  {'title': 'Heavyweights', 'rating': 'PG', 'release year': 1995}
]
```

Paso a paso

Realice el siguiente paso a paso para completar la actividad 1.

`shows = [ ]`	Cree una lista vacía
`archivo = open("Netflix Shows.csv", "r")`	Abra, para lectura, el archivo que contiene la información de shows.
`encabezados = archivo.readline().split(";")`	Lea la primera línea del archivo la cual, contiene los encabezados. Use la función **`split`** para obtener una lista a partir de la línea, creando un elemento cada vez que se encuentra un punto y coma; *posteriormente profundizaremos en esta operación.*
`linea = archivo.readline()`	Lea la segunda línea del archivo.
`while linea != "":`	Mientras no haya llegado a la última línea del archivo (línea vacía) siga procesando el archivo
`    datos = linea.split(";")`	A partir de la línea con los datos del show cree una lista dividiendo la información en cada punto y coma.

`    show = {encabezados[0]: datos[0],`
`            encabezados[1]: datos[1],`
`            encabezados[4]: int(datos[4]) }`
	Cree un diccionario con la información relevante del show.

`    shows.append(show)`	Agregue el diccionario a la lista.
`    linea = archivo.readline()`	Lea la siguiente línea para continuar procesando el archivo.
`archivo.close()`	Cierre el archivo.
`return shows`	Devuelva la lista de shows.

Para probar la carga descomente la línea **`print(cargar())`**.

Actividad 2 – dar_titulos_anios

En esta actividad usted debe completar la función **`dar_titulos_anio(shows:list, anio:int)->list`** que se encuentra en el archivo **`administrador_shows.py`**, está función crea y retorna una lista con los títulos de los shows estrenados en el año que llega como parámetro. Si ningún *show* fue estrenado en el año dado la función retorna la lista vacía.

Paso a paso

Realice el siguiente paso a paso para completar la actividad 2. **Se recomienda que intente hacer la misma actividad usando la instrucción for – in – range.**

`titulos = [ ]`	Cree una lista vacía donde guardará los títulos de los *shows* que se estrenaron en el año dado.
`i = 0`	Inicie a revisar desde el primer *show*.
`while i < len(shows):`	Repita hasta llegar al último *show*.

`if shows[i]["release year"] == anio:`	Revise si el año de estreno del *show* corresponden al año dado.
`titulos.append(shows[i]["title"])`	Si el año es el de interés, agregue el título del *show* a la respuesta.
`i+=1`	Avance al siguiente *show*.
`return titulos`	Devuelva la lista de títulos.

Reto

Evite que la solución tenga elementos repetidos. *Ayuda: la instrucción* ***`x in lista`*** *devolverá* ***`True`*** *si* ***`x`*** *se encuentra en la lista o* ***`False`*** *en caso contrario.*

Actividad 3 – dar_titulos_clasificacion

En esta actividad usted debe completar la función **`dar_titulos_clasificacion(shows:list, clasificacion:str)->list`** que se encuentra en el archivo **`administrador_shows.py`**, esta función crea y retorna una lista con los títulos de los shows con la clasificación que llega como parámetro. Si ningún show pertenece a la clasificación dada la función retorna la lista vacía.

Actividad 4 – dar_titulo_anio_clasificacion

En esta actividad usted debe completar la función **`dar_titulo_anio_clasificacion(shows:list, anio:int, clasificacion: str)->list:`** que se encuentra en el archivo **`administrador_shows.py`**, esta función crea y retorna una lista con los títulos de los shows con la clasificación que llega como parámetro estrenadas en el año que lega por parámetro. Si ningún *show* pertenece a la clasificación dada y estrenada en el año dados la función retorna la lista vacía.

Actividad 5 – dar_show_mayor_anio

En esta actividad usted debe completar la función **`dar_show_mayor_anio(shows:list)->dict`** que se encuentra en el archivo **`administrador_shows.py`**, esta función retorna el *show* (diccionario) que tiene un mayor año de estreno. Si hay más de un show con el año mayor, retorna el primer show encontrado.

Actividad 6 – dar_show_nombre_corto

En esta actividad usted debe completar la función **`dar_show_nombre_corto(shows:list)->dict`** que se encuentra en el archivo **`administrador_shows.py`**, esta función retorna el *show* (diccionario) con el show nombre más corto. Si hay varios *shows* cuyo nombre tiene la misma longitud devuelve el primero encontrado.

Actividad 7 – dar_shows_nombre_largo

En esta actividad usted debe completar la función **`dar_shows_nombre_largo(shows:list)->dict`** que se encuentra en el archivo **`administrador_shows.py`**, esta función retorna un diccionario cuyas llaves son los años registrados en el archivo y los valores corresponden al título del show con el nombre más largo de cada año. Un fragmento del diccionario obtenido es:

```
{ …
  1982: 'The Last Unicorn',
  1990: 'The Real Ghostbusters',
  1940: 'Fantasia',
  1986: 'An American Tail' …}
```