Buenos Aires aprende
Agencia de Habilidades para el Futuro

Talento Tech

Iniciación con Python

Clase 07 - “Listas y tuplas”

Buenos Aires Ciudad

¡Les damos la bienvenida!

 Vamos a comenzar a grabar la clase

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Buenos Aires Ciudad

Talento Tech

Clase 06.

Bucles while

1. Control de flujo: bucles while.
2. Concepto y usos de los contadores.
3. Concepto y usos de los acumuladores.

Clase 07.

Listas y tuplas

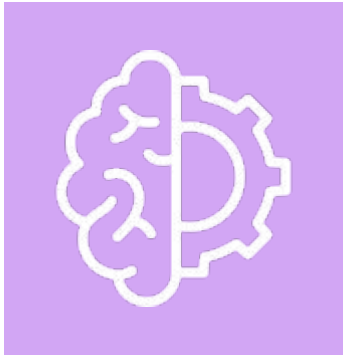
1. Listas y tuplas: creación y manipulación.
2. Uso de subíndices.
3. Métodos de listas y tuplas.
4. Recorrer una lista con while.

Clase 08.

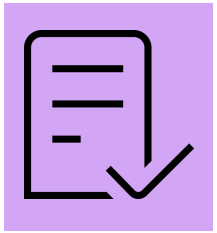
Ruta de avance

1. Pre-Entrega del Proyecto Final Integrador.
2. Menú de opciones.
3. Pedir, procesar y mostrar datos.

Pero antes...



¡Resolvamos los “**Ejercicios prácticos**” de la clase anterior!



Listas y tuplas

Organizando datos en Python

Imaginate que tenés que hacer las compras del mes. No basta con memorizar todo lo que tenés que comprar, y si lo intentás, probablemente olvides algo importante. Entonces, ¿qué hacés? Armás una **lista de compras**.

Python te ofrece una herramienta, llamada listas, qué es más o menos lo mismo pero aplicado a la programación: en lugar de recordar cada dato por separado, te proporciona una estructura donde podés almacenar toda esa información junta y organizada en lugar de manejar cada dato individualmente.

¿Qué es una lista en Python?

En Python, una lista es una estructura que te permite almacenar varios valores dentro de una sola variable. A diferencia de una simple variable que almacena un único valor, una lista puede contener varios elementos, como si fuera una caja que guarda distintas cosas, una al lado de la otra. Una lista se crea encerrando los elementos entre corchetes [] y separándolos por comas:

```
compras = ["manzanas", "pan", "leche", "queso"]
```

La lista llamada compras contiene cuatro productos. Cada uno de estos productos tiene una posición específica dentro de la lista, lo que nos permite acceder a ellos cuando lo necesitemos.

Creación y acceso a listas

Como vimos, crear una lista en Python es muy sencillo: sólo necesitás usar corchetes y separar los elementos con comas. Pero, ¿cómo hacés para acceder a un elemento específico dentro de la lista? Acá es donde entra el concepto de **índices**.

Cada elemento dentro de una lista tiene una posición, y **esa posición se llama índice**. El primer elemento de la lista tiene el **índice 0**, el segundo tiene el **índice 1**, y así sucesivamente.

Creación y acceso a listas

Podés acceder a un elemento de la lista simplemente usando su índice:

```
compras = ["manzanas", "pan", "leche", "queso"]
print(compras[0])  # Imprime "manzanas"
print(compras[2])  # Imprime "leche"
```

El índice 0 nos da el primer producto, que es "manzanas", y el índice 2 nos da el tercero, que es "leche". Y si querés cambiar el valor de un elemento, simplemente lo asignás de nuevo:

```
compras[1] = "yogur"
print(compras)  # Ahora la lista es ["manzanas", "yogur", "leche", "queso"]
```

Agregar elementos a la lista

Ya vimos que las listas se pueden modificar, por ejemplo, cambiando uno de sus elementos. Pero también podés podés agregar nuevos elementos a la lista usando el método append():

```
compras.append("cereales")
print(compras)
# Ahora la lista es ["manzanas", "yogur", "leche", "queso", "cereales"]
```

Esto es clave cuando trabajás en proyectos como el PFI, donde el inventario de productos cambia constantemente. Cada vez que se agrega un nuevo producto, lo podés añadir a la lista de forma sencilla y seguir adelante. ¡Y todavía hay más!

Quitar elementos de la lista

Cuando en el programa de tu Proyecto Final Integrador un producto se vende o se da de baja en el inventario, necesitás eliminarlo de la lista. Para esto, podemos usar el método remove(). **Este método busca el primer valor que coincida con el que indicás y lo elimina de la lista.**

```
productos = ["manzanas", "pan", "leche"]
productos.remove("pan")
print(productos)
 # Imprime ["manzanas", "leche"]
```

Por último, el método len() te permite conocer la cantidad de elementos que tiene una lista.

```
productos = ["manzanas", "pan", "leche"]
cantidad productos = len(productos)
print("Cantidad de productos:",
cantidad productos)
# Imprime "Cantidad de productos: 3"
```

Esto es útil para controlar bucles, donde querés recorrer toda la lista sin ir más allá de su límite.

Tuplas

Una **tupla** es como una lista, pero con una característica importante: **es inmutable**. Una vez que creás una tupla, no podés cambiar sus elementos. Son útiles cuando querés asegurarte de que ciertos valores no cambian.

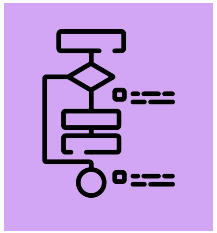
```
mi_tupla = ("manzanas", "pan", "leche")
```

Python sabe que se trata de una tupla y no de una lista porqué está delimitada por paréntesis en lugar de corchetes.

Tené cuidado: si intentás cambiar un elemento de una tupla, Python te va a decir que no se puede:

```
# Esto va a dar un error porque las tuplas
# no se pueden modificar
mi_tupla[1] = "yogur"
```

Las tuplas son útiles cuando querés proteger datos que no deberían cambiar, como precios fijos o información de configuración en el sistema. ¡Tenelo en cuenta!



Listas, tuplas y bucles

Listas y bucles while

Supongamos que querés mostrar la lista de productos disponibles en tu inventario. Lo podés hacer fácilmente usando un bucle while que recorra la lista desde el principio hasta el final, utilizando un contador como índice que comience en 0 y se incrementa en cada iteración.

```
productos = ["manzanas", "pan", "leche"]
indice = 0

while indice < len(productos):
    print("Producto", indice + 1, ":", productos[indice])
    indice = indice + 1
```

Probá este código.

```
Producto 1 : manzanas
Producto 2 : pan
Producto 3 : leche
```

Verás esto en la terminal.

Tuplas y bucles while

Al igual que las listas, podés recorrer una tupla con un bucle while. Si bien no podés modificar los elementos de una tupla, nada impide que puedas acceder a ellos de la misma forma que con una lista:

```
configuracion_tienda = ("nombre_tienda", "calle falsa 123", "Buenos Aires")
indice = 0

while indice < len(configuracion tienda):
   print("Dato", indice + 1, ":", configuracion_tienda[indice])
   indice = indice + 1
```

Probá este código.

Verás esto en la terminal:

```
Dato 1 : nombre tienda
Dato 2 : calle falsa 123
Dato 3 : Buenos Aires
```

Lista de listas

Lista de listas

Vimos que en una lista podemos almacenar datos numéricos y cadenas de caracteres. También se pueden almacenar datos booleanos, y cómo no, ¡otras listas! Cuando esto ocurre, decimos que tenemos una “lista de listas”.

Lista de listas

Una lista de listas es simplemente una lista donde cada uno de sus elementos es, a su vez, otra lista. Podés pensar en esto como una tabla, donde cada fila representa una lista con múltiples datos relacionados. Por ejemplo, en el contexto del Proyecto Final Integrador, podríamos tener una lista que almacene productos junto con su precio y cantidad disponible en el inventario. Cada "producto" sería una lista con esos tres elementos: nombre, precio y cantidad.

Inventario de productos

Ejemplo de lista de listas: Inventario de productos

```
# Lista de productos: cada producto tiene nombre, precio y cantidad
inventario = [ ["manzanas", 100, 50],
               ["pan", 50, 20],
               ["leche", 60, 30]  ]
# Recorrer el inventario y mostrar los datos de cada producto
indice = 0
while indice < len(inventario):
   producto = inventario[indice]
   print("Producto:", producto[0])
   print("Precio: $", producto[1])
   print("Cantidad disponible:", producto[2])
   print("-------------------------")
   indice = indice + 1
```

Inventario de productos

En el ejemplo anterior, cada producto es una lista que contiene tres elementos: el **nombre** del producto, su **precio** y la **cantidad** disponible. **La lista inventario contiene tres listas**, una por cada producto. Para acceder a un producto en particular, usamos dos niveles de índice.

- El primer índice nos dice qué producto queremos (por ejemplo, inventario[0] para las "manzanas").
- El segundo índice nos dice qué dato de ese producto queremos: inventario[0][0] es el nombre ("manzanas"), inventario[0][1] es el precio (100), y inventario[0][2] es la cantidad (50).

Y usamos un bucle while para recorrer cada producto en el inventario. Dentro del bucle, accedemos a los elementos individuales de cada producto con un segundo nivel de índices.

Ingreso masivo de datos

Ejemplo de lista de listas: Ingreso masivo de productos al inventario

Ahora, vamos a analizar un ejemplo que nos será útil para el TFI: usaremos un bucle while y una lista de listas para ingresar datos de varios productos al inventario.

El script comienza creando una lista vacía llamada productos, que va a almacenar los datos de los productos ingresados. Luego, pide al usuario que ingrese el código del producto, que es un número entero. El bucle while se utiliza para controlar la repetición del proceso de entrada de datos: mientras el usuario no ingrese un código igual a 0, el ciclo sigue ejecutándose.

Cada vez que el usuario ingresa un código válido, el script pide la descripción del producto, su precio y la cantidad que está siendo ingresada. Estos cuatro datos se almacenan como una lista dentro de la lista productos, lo que permite agrupar todos los datos de un mismo producto.

Ingreso masivo de datos

```python
productos = [] # Lista que almacenará los productos

# Solicitamos el primer código de producto
codigo = int(input("Ingresá el código del producto (0 para finalizar): "))

# Usamos el bucle while para ingresar los datos mientras el código no sea 0
while codigo != 0:
    descripcion = input("Ingresá la descripción del producto: ")
    precio = float(input("Ingresá el precio del producto: "))
    cantidad = int(input("Ingresá la cantidad del producto: "))

    # Agregamos los datos del producto como una lista dentro de la lista de productos
    productos.append([codigo, descripcion, precio, cantidad])

    # Solicitamos nuevamente el código del siguiente producto
    codigo = int(input("Ingresá el código del siguiente producto (0 para finalizar): "))
```

(Sigue en la próxima diapositiva)

Ingreso masivo de datos

```
# Mostramos la lista final de productos
print("Productos ingresados:")
for producto in productos:
    print("Código:", producto[0], "Descripción:", producto[1], "Precio:", producto[2],
"Cantidad:", producto[3])
```

(Viene de la diapositiva anterior)

El ciclo continúa solicitando productos hasta que el usuario ingresa el código "0". En ese momento, el bucle se detiene. Finalmente, el programa muestra todos los productos almacenados en la lista, accediendo a los datos de cada producto usando los índices correspondientes (por ejemplo, el código está en la posición 0 de la sublista, la descripción en la 1, y así sucesivamente).

De esta manera, la persona que usa el programa puede ingresar varios productos, cada uno con sus datos asociados, y almacenarlos en una estructura clara y organizada.

¡Vamos a la práctica! 

Ejercicios prácticos

Optativos | No entregables

Registro de productos en un inventario

Vas a implementar un sistema básico para registrar productos en el inventario de una tienda. El programa debe permitir que el usuario agregue productos a una lista hasta que decida no agregar más. Luego, deberás mostrar todos los productos ingresados al inventario.

Tips:

- Usá una lista para almacenar los productos. Diseña la lista pensando en el TFI.

Ejercicios prácticos

Optativos | No entregables

Consultar el stock de productos

Tu programa debe permitir al usuario consultar el inventario de una tienda para verificar si un producto está en stock. Si el producto está en la lista, el programa debe informarlo, si no, debe mostrar un mensaje indicando que no está disponible.

Tips:

- Usá una lista para almacenar los productos en stock.
- Permití que el usuario ingrese el nombre de un producto a consultar.
- Recorré la lista con un bucle while para verificar si el producto está en stock.