«Talento Tech»

Iniciación a la Programación con Python

CLASE 12

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Buenos Aires Ciudad

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Clase N° 12 | funciones y diccionarios

Temario:

- Diccionarios: uso y métodos esenciales.
- Uso de diccionarios como medio de almacenamiento temporal de datos

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Trabajo con funciones y diccionarios.

En esta clase vamos a poner en práctica gran parte de lo que venimos aprendiendo, integrando varios conceptos importantes como las funciones y los diccionarios. Recordemos que una función es una forma de organizar y reutilizar nuestro código agrupando instrucciones bajo un nombre que dé cuenta cómo ésta opera, mientras que los diccionarios son una estructura poderosa que nos permite almacenar y manipular datos de forma eficiente.

El objetivo es que empieces a visualizar cómo podés darle vida a tu Proyecto Final Integrador. En esta clase vas a trabajar con un sistema básico para gestionar un inventario de productos. Vamos a crear un conjunto de funciones que nos permitan agregar productos, actualizarlos, eliminarlos y mostrarlos. Toda esta información la vamos a almacenar en un diccionario, que será nuestra estructura principal de datos.

¿Te acordás del menú de opciones que ya creamos antes? Bueno, ahora vamos a ir conectando ese menú con funciones que añaden distintas utilidades. Pero lo más importante es que, al final, vas a tener las herramientas para empezar a pensar cómo armar tu propio sistema de gestión de inventarios para el PFI.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Uso de un diccionario para almacenar los datos.

Un diccionario en Python es una estructura de datos muy útil cuando necesitamos almacenar información de forma organizada, permitiendo asociar un valor a una clave. Esto lo convierte en la opción perfecta para guardar información relacionada con los productos de nuestro inventario.

Imaginá un diccionario como un "cajón" donde guardás objetos (los datos de cada producto), y cada cajón tiene una etiqueta (la clave) que te dice qué contiene. A diferencia de una lista, en la que accedés a los elementos por su posición, en un diccionario accedés a los valores utilizando la clave que les hayas asignado.

Diccionario de inventario.

En nuestro caso vamos a usar un diccionario para representar el inventario de productos de la tienda. Cada producto se almacenará dentro del diccionario usando un código numérico único (clave), y los detalles del producto (nombre, descripción, cantidad, etc.) estarán asociados a esa clave en forma de otro diccionario.

Estructura del diccionario:

```
inventario = {
    1: {
        "nombre": "Manzana",
        "descripcion": "Fruta fresca y deliciosa",
        "cantidad": 50,
        "precio": 0.5,
        "categoria": "Frutas"
    },
    2: {
        "nombre": "Pan",
```

```
        "descripcion": "Pan casero recién horneado",
        "cantidad": 20,
        "precio": 1.0,
        "categoria": "Panadería"
    }
}
```

Cómo ves, el diccionario inventario contiene nuestros productos. A su vez, cada producto tiene su propio código (en este caso, 1 para la manzana y 2 para el pan), y cada código está asociado a otro diccionario que guarda la información sobre este producto.

Claves del diccionario de cada producto.

Cada producto tiene una serie de claves (como "nombre", "cantidad", etc.) que nos permiten organizar toda la información asociada. Vamos a describir cada una:

nombre: Es el nombre del producto. Es un texto (cadena de caracteres) que describe de manera general lo que es el producto. Ejemplo: "Manzana", "Pan".

descripcion: Esta clave guarda una descripción más detallada del producto. Es también un texto que permite especificar características o particularidades del mismo. Ejemplo: "Fruta fresca y deliciosa", "Pan casero recién horneado".

cantidad: Indica la cantidad disponible del producto en el inventario. Es un número entero que nos dice cuántas unidades tenemos del producto. Ejemplo: 50 manzanas o 20 panes.

precio: Representa el precio del producto. Es un número decimal (tipo float) que indica el costo de una unidad del producto. Ejemplo: 0.5 para una manzana, o 1.0 para un pan.

categoria: Indica la categoría a la que pertenece el producto. Esto nos permite organizar los productos en grupos según su tipo. Ejemplo: "Frutas", "Panadería", "Lácteos".

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Cuando se registra un producto, creamos una entrada nueva en el diccionario inventario, donde el código del producto es la clave y la información sobre ese producto (nombre, descripción, cantidad, etc.) se almacena en un diccionario dentro de esa clave. Así, toda la información queda organizada y fácilmente accesible. Más adelante, reemplazaremos este diccionario por una base de datos, pero mientras tanto, nos servirá para seguir avanzando.

El menú interactivo de nuestro script.

Como hemos visto, el menú es una herramienta clave en cualquier programa interactivo. Nos permite presentar al usuario o usuaria una lista de opciones, donde puede seleccionar la acción que desea realizar. Vamos a implementar las funciones que ya hemos visto, y el menú se encargará de conectarlas con el flujo de interacción.

Nuestro menú principal mostrará las distintas acciones disponibles, como registrar productos, mostrar el inventario, actualizar productos, eliminarlos, buscarlos y generar reportes de bajo stock. Se seleccionará la acción escribiendo el número de la opción que se desea ejecutar. El programa entonces procede a realizar la función correspondiente.

Un ejemplo simple de cómo se vería el menú en el código:

```
def mostrar_menu():
    print("Menú de Gestión de Inventario:")
    print("1. Registrar producto")
    print("2. Mostrar productos")
    print("3. Actualizar producto")
    print("4. Eliminar producto")
    print("5. Buscar producto")
    print("6. Reporte de Bajo Stock")
    print("7. Salir")
```

Este fragmento define una función que imprime las opciones disponibles en pantalla. Cada vez que el menú se muestre, se verá esta lista con las acciones que se pueden tomar.

A continuación, necesitamos un bucle que mantenga el programa en ejecución, permitiendo que se seleccionen varias opciones hasta que se decida salir. El bucle continuará ejecutándose mientras no se elija la opción de salir. Para seleccionar una opción, la usuaria o usuario ingresará el número correspondiente, y el programa se encargará de ejecutar la función adecuada usando *if-elif* para manejar cada opción.

Por ejemplo:

```
while True:
    mostrar_menu()
    opcion = int(input("Seleccione una opción: "))

    if opcion == 1:
        registrar_producto()
    elif opcion == 2:
        mostrar_productos()
    elif opcion == 3:
        actualizar_producto()
    elif opcion == 4:
        eliminar_producto()
    elif opcion == 5:
        buscar_producto()
    elif opcion == 6:
        reporte_bajo_stock()
    elif opcion == 7:
        print("Saliendo del programa...")
        break
    else:
        print("Opción inválida. Intente nuevamente.")
```

El bucle comienza mostrando el menú y luego solicita que se seleccione una opción. Dependiendo de aquello que se ingrese, el programa verifica con *if-elif* cuál fue la opción seleccionada y ejecuta la función correspondiente. Si se selecciona la opción 7, que es "Salir", el programa imprime un mensaje de salida y utiliza break para salir del bucle, finalizando el programa.

Al ingresar una opción inválida (es decir, un número que no está en el menú), el programa muestra un mensaje de error y vuelve a mostrar el menú para que se intente de nuevo.

Este menú organiza todas las acciones posibles y facilita la interacción entre las personas y el programa. A través de las funciones que implementaremos a continuación, este menú permite registrar, actualizar, eliminar y buscar productos, así como generar reportes de bajo stock, todo de manera ordenada y clara.

La función registrar_producto.

La función registrar_producto() es la encargada de permitir que se agreguen nuevos productos al inventario. Cada vez que se registra un producto, este se guarda en el diccionario inventario con un código único y sus respectivos datos, como el nombre, la descripción, la cantidad disponible, el precio y la categoría.

Para crear esta función, primero necesitamos una variable que actúe como un contador para los códigos de los productos. De esta manera, cada vez que se registra un producto, se le asigna automáticamente un código nuevo. En este caso, el diccionario inventario usará el código del producto como clave, mientras que los valores asociados a esa clave serán otro diccionario que contendrá toda la información del producto.

Al ejecutar la función, el programa pedirá al usuario o usuaria que ingrese los detalles del producto: el nombre, la descripción, la cantidad, el precio y la categoría. Estos datos se guardarán en un diccionario que luego se añadirá al diccionario inventario, utilizando el código del producto como clave.

Aquí está el código de la función:

Buenos Aires aprende
Agencia de Habilidades para el Futuro

```
inventario = {}
codigo_actual = 1  # Contador para generar códigos únicos

def registrar_producto():
    global codigo_actual  # Para modificar el contador en cada registro
    nombre = input("Ingrese el nombre del producto: ")
    descripcion = input("Ingrese una breve descripción: ")
    cantidad = int(input("Ingrese la cantidad disponible: "))
    precio = float(input("Ingrese el precio del producto: "))
    categoria = input("Ingrese la categoría del producto: ")

    # Agregar el producto al diccionario 'inventario'
    inventario[codigo_actual] = {
        "nombre": nombre,
        "descripcion": descripcion,
        "cantidad": cantidad,
        "precio": precio,
        "categoria": categoria
    }

    print("Producto registrado con el código",codigo_actual)
    codigo_actual += 1
```

Primero, definimos la variable codigo_actual, que comenzará con el valor de 1. Cada vez que se registre un producto, este valor se usará como el código del nuevo producto, y luego se incrementará para que el siguiente producto tenga un código diferente.

Dentro de la función, usamos input() para solicitar al usuario que ingrese los detalles del producto. El nombre y la descripción se ingresan como texto, la cantidad se convierte en un número entero usando int(), y el precio se convierte en un número decimal usando float(). La categoría también se ingresa como texto.

Toda esta información se agrupa en un diccionario, donde cada clave representa un atributo del producto (nombre, descripción, cantidad, precio, categoría) y los valores correspondientes son los datos ingresados. Este diccionario se almacena en inventario, utilizando codigo_actual como la clave principal.

Finalmente, la función imprime un mensaje indicando que el producto ha sido registrado con éxito, mostrando su código único, y luego incrementa codigo_actual para asegurarse de que el próximo producto tenga un código diferente. Cada vez que se ejecuta, se actualiza el inventario con un nuevo producto, asignando un código único automáticamente.

La función mostrar_productos()

Ahora vamos a analizar cómo funciona la función mostrar_productos(), que es la encargada de mostrar todos los productos que están registrados en el diccionario inventario.

El objetivo de esta función es recorrer todo el inventario y mostrar la información de cada producto de manera clara, incluyendo su código, nombre, descripción, cantidad, precio y categoría. Si el inventario está vacío, la función deberá informar que no hay productos registrados aún.

Para lograr esto, usamos un bucle que recorre el diccionario inventario y, para cada producto, imprimimos sus detalles. A continuación vemos cómo sería el código y su funcionamiento:

```
def mostrar_productos():
    if not inventario:
        print("El inventario está vacío.")
    else:
        for codigo, datos in inventario.items():
            print(f"Código: {codigo}")
            print(f"Nombre: {datos['nombre']}")
            print(f"Descripción: {datos['descripcion']}")
```

```
        print(f"Cantidad: {datos['cantidad']}")
        print(f"Precio: {datos['precio']}")
        print(f"Categoría: {datos['categoria']}")
        print("-" * 30)
```

Primero, la función comienza verificando si el inventario está vacío. Para esto, utilizamos la expresión if not inventario. Esta condición evalúa si el diccionario no tiene productos registrados. En caso de que el inventario esté vacío, se muestra un mensaje diciendo "El inventario está vacío", y la función finaliza.

Si hay productos en el inventario, la función continúa recorriendo el diccionario utilizando un bucle for. En este bucle, codigo representa la clave de cada producto (el código único) y datos es el diccionario que contiene toda la información del producto (nombre, descripción, cantidad, precio, categoría).

Dentro del bucle, la función imprime el código del producto y accede a cada una de las claves del diccionario de datos para mostrar su valor. Utilizamos datos['nombre'] para obtener el nombre, datos['descripcion'] para la descripción, y así sucesivamente para cada atributo.

Entre cada producto, agregamos una línea de separación utilizando print("-" * 30) para hacer la salida más legible. De esta manera, se puede ver fácilmente qué productos están registrados y los detalles de cada uno.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

La función actualizar_producto.

La función actualizar_producto() es indispensable para que tu usuaria o usuario pueda cambiar la cantidad disponible de un producto que ya está registrado en el inventario. Se va a ingresar el código del producto que se necesita modificar, después la nueva cantidad, y si el producto está en el inventario, se actualizará. Si el producto no existe, le avisaremos al usuario.

El código de la función queda así:

```
def actualizar_producto():
    codigo = int(input("Ingrese el código del producto que desea actualizar: "))
    if codigo in inventario:
        nueva_cantidad = int(input("Ingrese la nueva cantidad: "))
        inventario[codigo]["cantidad"] = nueva_cantidad
        print("Producto", codigo, "actualizado con éxito.")
    else:
        print("Producto no encontrado.")
```

Lo primero que hace la función es solicitar el código del producto que se quiere actualizar. Usamos input() para que la persona ingrese el código, y lo convertimos en número entero con int(), ya que los códigos son números. Guardamos ese código en la variable codigo.

Después, el programa verifica si ese código existe en el diccionario inventario. Para esto usamos if codigo in inventario. Si el código está registrado, avanzamos y pedimos que se ingrese la nueva cantidad disponible del producto. Nuevamente usamos input() para capturar ese dato, y lo convertimos a número entero con int().

El siguiente paso es actualizar la cantidad en el diccionario. Como ya sabemos el código del producto, simplemente accedemos a su cantidad usando inventario[codigo]["cantidad"] y le asignamos el nuevo valor ingresado.

Finalmente, usamos print() para mostrar un mensaje que confirme que el producto fue actualizado correctamente. En este caso, concatenamos el mensaje con el número de código de esta manera: print("Producto", codigo, "actualizado con éxito."). De esta forma, no utilizamos f-strings ni otras técnicas avanzadas, solo unimos texto y variables con comas.

En el caso de que el código ingresado no exista en el inventario, la función entra en el bloque else y muestra un mensaje que dice "Producto no encontrado". Así, evitamos que intente actualizar un producto que no está registrado.

La función eliminar_producto.

Esta es la función que le permitirá borrar un producto del inventario. Lo que haremos es solicitar el código del producto que se quiere eliminar, verificar si existe en el diccionario del inventario, y si es así, eliminarlo. En caso de que no esté en el inventario, le vamos a avisar a quien ingresó el código que no se encontró el producto.

Aquí está el código de la función:

```
def eliminar_producto():
    codigo = int(input("Ingrese el código del producto que desea
eliminar: "))
    if codigo in inventario:
        del inventario[codigo]
        print("Producto con código", codigo, "eliminado con éxito.")
    else:
        print("Producto no encontrado.")
```

Primero, indicamos que se ingrese el código del producto que se desea eliminar usando input(). Convertimos esa entrada en un número entero con int(), ya que los códigos de los productos son números, y guardamos ese valor en la variable codigo.

Luego, el programa verifica si ese código existe en el diccionario inventario. Esto lo hacemos con la condición if codigo in inventario. Si el código existe, entonces usamos la instrucción del para eliminar el producto del diccionario. La línea del inventario[codigo] se encarga de borrar la entrada completa asociada a ese código del inventario.

Una vez que se elimina el producto, mostramos un mensaje para informar que el producto fue eliminado correctamente, concatenando el texto con el número de código para mayor claridad.

Si el código que ingresó el usuario no está en el inventario, la función entra en la parte de else y muestra el mensaje "Producto no encontrado", para que sepa que ese código no está registrado.

¿Por qué es importante esta función?

La función eliminar_producto() es de vital importancia porque en cualquier sistema de inventario puede surgir la necesidad de quitar productos, ya sea porque se dejaron de vender, porque hay un error en el registro, o por cualquier otro motivo. Con esta función, es factible eliminar productos del inventario de forma rápida y simple, manteniendo la base de datos actualizada y limpia de productos que ya no se necesiten.

La función buscar_producto.

Esta es la función que permitirá iniciar una búsqueda de un producto en el inventario a partir de su código. La idea es que el programa le pida a la persona que opera el programa ingresar el código del producto que está buscando, y si el producto existe en el inventario, mostrará toda su información. Si el código no está registrado, le avisamos que no se encontró el producto.

Aquí está el código de la función:

```
def buscar_producto():
    codigo = int(input("Ingrese el código del producto a buscar: "))
    if codigo in inventario:
        datos = inventario[codigo]
        print("Nombre:", datos['nombre'])
        print("Descripción:", datos['descripcion'])
        print("Cantidad:", datos['cantidad'])
        print("Precio:", datos['precio'])
        print("Categoría:", datos['categoria'])
    else:
        print("Producto no encontrado.")
```

En primer lugar la función demanda el código del producto que se quiere buscar. Usamos input() para que la usuaria o usuario pueda escribir ese código, y lo convertimos a un número entero con int(), ya que los códigos de producto son números. Ese valor se guarda en la variable codigo.

Luego, el programa verifica si ese código está registrado en el inventario utilizando if codigo in inventario. Si el código está en el diccionario, accedemos a los datos del producto usando inventario[codigo], y guardamos toda la información del producto en la variable datos.

Después, simplemente mostramos en pantalla los detalles del producto, como el nombre, la descripción, la cantidad disponible, el precio y la categoría. Usamos print() para mostrar toda esa información de manera clara y ordenada.

Si el código ingresado no existe en el inventario, la función entra en el bloque else y muestra un mensaje que da el siguiente aviso: "Producto no encontrado".

Esta función es muy útil para consultar rápidamente la información de cualquier producto que esté registrado en el sistema. Si en algún momento necesita saber detalles de un

producto específico, como su cantidad disponible o el precio, solo se necesita ingresar su código y el sistema le devolverá toda la información que necesita.

La función reporte_bajo_stock.

Esta función permite generar un reporte de productos que tengan una cantidad baja en el inventario. Lo que vamos a hacer es solicitar que se ingrese un límite de stock, y luego el programa buscará todos los productos cuya cantidad sea igual o inferior a ese límite. Finalmente, vamos a mostrar esos productos en pantalla.

Aquí está el código de la función:

```
def reporte_bajo_stock():
   limite = int(input("Ingrese el límite de stock para generar el
reporte: "))
   print("Productos con stock igual o inferior a", limite, ":")
   for codigo, datos in inventario.items():
       if datos["cantidad"] <= limite:
           print("Código:", codigo)
           print("Nombre:", datos['nombre'])
           print("Cantidad:", datos['cantidad'])
```

El primer paso de esta función es que se ingrese un límite de stock. Usamos input() para que se escriba ese límite, y lo convertimos en número entero con int() porque estamos trabajando con cantidades. Ese valor lo guardamos en la variable limite.

Luego, el programa imprime un mensaje inicial para indicar que se va a mostrar los productos que tienen una cantidad igual o menor a ese límite que ingresó.

El siguiente paso es recorrer el diccionario inventario usando un bucle for. En cada iteración, codigo representa la clave del producto (el código), y datos es el diccionario que contiene toda la información del producto.

Dentro del bucle, usamos if para verificar si la cantidad del producto es menor o igual al límite que ingresó el usuario. Esta verificación se hace con if datos["cantidad"] <= limite.

Si la condición se cumple, mostramos en pantalla el código del producto, su nombre, y la cantidad disponible.

De esta manera, el reporte va mostrando uno por uno los productos que tienen un stock bajo, según el criterio que definió la persona que está usando el programa.

Esta es una función importante para la gestión eficiente de un inventario. Saber qué productos tienen un stock bajo permite tomar decisiones informadas, como reponer mercadería antes de que se agote. Tener un reporte rápido y claro de los productos con bajo stock ayuda a evitar que la tienda se quede sin productos importantes para la venta.

Lo que se viene…

Para cerrar, adelantaremos un tema clave que vamos a abordar en las próximas clases: **el uso de una base de datos.** Hasta ahora, hemos estado guardando nuestros productos en un diccionario. Si bien es muy útil para aprender y manejar información de manera rápida, tiene una limitación importante: los datos no se guardan de forma permanente. Cuando apagás el programa, todo lo que cargaste en el diccionario desaparece. ¡Nos espera mucho por aprender y construir juntos!

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Ejercicios prácticos:

Registro de productos con validaciones

Escribí una función que permita registrar un nuevo producto en el inventario, pero con una condición: la cantidad de productos debe ser mayor que 0 y el precio también debe ser un valor positivo. Si se ingresa una cantidad o precio no válido, debe mostrarse un mensaje de error y pedir los datos nuevamente hasta que sean correctos.

Visualización personalizada de productos

Agregá una función al sistema que permita simular la venta de un producto. El usuario o usuaria deberá ingresar el código del producto y la cantidad a vender. Si la cantidad en stock es suficiente, la función debe restar esa cantidad del inventario. Si la cantidad solicitada es mayor a la disponible, debe mostrar un mensaje de error.

Buenos Aires aprende

Agencia de Habilidades para el Futuro

Buenos Aires Ciudad