«Talento Tech»

Iniciación a la Programación con Python

CLASE 4

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Buenos Aires Ciudad

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Clase N° 4

Tipos de datos

Temario:

- Definimos los requisitos del Proyecto Integrador
- Creamos un menú de opciones provisorio
- Creamos código para pedir, procesar y mostrar datos

Introducción al Proyecto Final Integrador (PFI)

Este es el momento donde todo lo que venimos aprendiendo empieza a tomar forma. A lo largo de las clases, fuimos construyendo las bases de algo mucho más grande: el Proyecto Final Integrador (PFI). Este proyecto es la pieza central del curso y la manera en la que vamos a consolidar todos los conceptos y habilidades que fuimos sumando. El objetivo final es que desarrolles una aplicación en Python que te permita gestionar el inventario de una pequeña tienda. Todo esto, claro, a través de la terminal o consola.

Quizás te estés preguntando, ¿qué tiene de especial? Bueno, la verdad es que este proyecto va mucho más allá de escribir algunas líneas de código. Te va a permitir integrar lo que aprendiste sobre la entrada y salida de datos, la manipulación de información, y las estructuras de control. Vas a ver cómo todo cobra sentido cuando desarrolles algo que realmente funcione, y que, además, pueda ser parte de tu portafolio profesional.

Alcance del Proyecto Final Integrador:

El desafío es claro: crear una aplicación que permita a nuestra clientela gestionar el inventario de la tienda desde la terminal. La idea es que puedas agregar productos, consultarlos, actualizarlos, eliminarlos y generar listados completos del inventario. Todo esto es clave para consolidar tu manejo de Python y los conceptos de programación que venimos trabajando.

Durante el curso te vamos a ir guiando paso a paso pero lo más importante es que al final vas a tener un sistema funcional, con el que cualquier persona podría gestionar el stock de una tienda de forma rápida y sencilla.

Objetivos del PFI:

Registro de productos: La aplicación va a permitir que la usuaria o usuario ingrese nuevos productos en el inventario. Esto incluye detalles como el nombre del producto, la cantidad disponible, el precio por unidad y cualquier otra información que sea relevante. Con el

tiempo, vamos a aprender a organizar y guardar estos datos usando estructuras que te permitan mantener todo en orden.

Consulta de productos: Nuestra clientela debe poder consultar el inventario, para ver información detallada de cada producto, como cuántos quedan en stock o cuál es el precio. Acá es donde la función **print()** se convierte en tu aliada, porque vas a aprender a mostrar los datos de manera clara y estructurada, para que quien use tu aplicación pueda entender fácilmente la información que está viendo.

Actualización de productos: Si un o una cliente compra productos o si llega nueva mercadería la aplicación va a necesitar reflejar esos cambios. La persona que use el sistema podrá actualizar la cantidad disponible de cada producto, lo que implica que vamos a desarrollar la lógica necesaria para buscar y modificar los datos en el sistema de manera segura.

Eliminación de productos: La tienda también puede decidir dejar de vender ciertos productos, y ahí es donde entrás vos: el sistema que vas a construir tiene que permitir eliminar productos del inventario sin que esto afecte el correcto funcionamiento de la aplicación. Vamos a aprender a manejar estas situaciones sin problemas.

Listado Completo y Reporte de Bajo Stock: Además de poder consultar productos de forma particular, la aplicación será capaz de generar listados completos del inventario y, algo muy útil, un reporte de productos con bajo stock. De esta manera, se podrá tomar decisiones informadas sobre qué productos se necesita reponer. La claridad a la hora de presentar la información va a ser fundamental acá.

El **PFI** no sólo se trata de desarrollar una aplicación que funcione, sino también de aprender a estructurar un proyecto de programación de manera ordenada y eficiente. Todo lo que venimos viendo hasta ahora va a cobrar más sentido mientras avanzamos en el desarrollo del PFI. Desde cómo ingresamos y procesamos los datos, hasta cómo los mostramos y manejamos la lógica del flujo del programa: todo está conectado.

Además, este proyecto no es únicamente un ejercicio académico: es una experiencia real en el desarrollo de software que te va a proporcionar una pieza valiosa para tu portafolio profesional. **¡Al final del curso vas a tener tu propia aplicación completa y funcional!**

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Definición del menú de opciones

Aunque recién estamos empezando a conocer Python, ya estamos en condiciones de crear la columna vertebral de nuestra aplicación: **el menú de opciones.** Este menú será fundamental, porque será la puerta de entrada a todas las funcionalidades que vamos a desarrollar en el Proyecto Final Integrador (PFI). ¿Qué hace un menú? Simple: permite que quien use tu programa elija qué quiere hacer, de manera clara y organizada.

Este menú es la **interfaz inicial** de nuestro sistema, donde se van a organizar todas las acciones que podrá realizar quien utilice nuestro software. Imaginate que estás en una tienda y querés registrar nuevos productos, consultar el inventario o eliminar algo que ya no se vende: **todo eso va a ser posible gracias al menú que vamos a construir.**

Objetivo del menú de opciones

El objetivo principal en esta etapa es que aprendamos a construir un menú interactivo utilizando dos herramientas que ya conocemos bien: **print() e input()**. Con print(), se mostrarán las opciones a elegir y, con input(), vamos a capturar la elección que haga. Más adelante, le daremos vida a cada una de estas opciones pero, por ahora, nos vamos a enfocar en la estructura del menú y en cómo capturamos la elección de nuestras usuarias y usuarios.

Código del menú de opciones

Vamos a ver un ejemplo de cómo se construye este menú básico:

```
# Menú de opciones
print("Menú de Gestión de Productos\n")
print("1. Alta de productos nuevos")
print("2. Consulta de datos de productos")
print("3. Modificar la cantidad en stock de un producto")
print("4. Dar de baja productos")
print("5. Listado completo de los productos")
print("6. Lista de productos con cantidad bajo mínimo")
print("7. Salir")

# Solicitar al usuario que seleccione una opción
opcion = int(input("Por favor, selecciona una opción (1-7):
"))

# Mostramos el nro de la opción seleccionada
print("Has seleccionado:", opcion)
```

Desglose y explicación del código

Lo primero que hacemos es presentar el menú. Esto lo logramos con una serie de **print()** que muestra las opciones disponibles. Esta función nos permite mostrar texto en la pantalla y, al añadir **\n**, agregamos un *salto de línea* para aumentar la legibilidad del menú. Cada opción se muestra en una línea separada con su respectivo número para que quien use la aplicación sepa qué ítem elegir.

Luego, con la función **input()**, solicitamos que se elija una de esas opciones por medio del ingreso de un número del 1 al 7. La entrada de la usuaria o usuario, que por defecto es una cadena de texto, **se convierte a un número entero usando int()** y se guarda en la variable "opcion". Esto es de suma importancia: pronto vamos a utilizar esa variable para tomar decisiones sobre qué acción ejecutar.

Finalmente, mostramos una confirmación de la opción seleccionada. En este caso, solo imprimimos el número que ingresó, pero en clases futuras vamos a ver cómo hacer que cada opción del menú esté conectada con una funcionalidad específica del sistema.

Propósito y ventajas del menú

El menú es esencial porque le permite a quien use tu aplicación interactuar de manera intuitiva con las diferentes funcionalidades que vayas desarrollando. En lugar de escribir una lista interminable de instrucciones el menú organiza todo en una estructura simple facilitando la usabilidad y navegabilidad. Esto mejora mucho la experiencia de uso, porque hace que la interacción sea mucho más fluida y accesible.

Además, el menú no solo es útil en esta etapa inicial. A medida que el sistema crezca, se convertirá en una herramienta central para integrar nuevas funcionalidades. Si en el futuro necesitás agregar más opciones o conectar el menú con nuevas tareas, el dispositivo ya va a estar listo para recibir dichas expansiones. Es como construir los cimientos de una casa: podés seguir agregando pisos y habitaciones sin tener que rehacer todo desde cero. Esto refiere, de alguna manera, al concepto de **"escalabilidad"** que tanto se utiliza en programación: poder reutilizar código y añadir nuevas utilidades y funciones a nuestro software sin mayor problema.

Por ahora, te invito a que pruebes este menú en tu computadora. Ingresá diferentes números y jugá con las opciones. Si querés, podés personalizarlo cambiando los textos o agregando nuevas opciones, lo que te va a ayudar a comprender mejor cómo funcionan **print() e input()** en conjunto.

En clases futuras vamos a empezar a darle vida a este menú conectándolo con estructuras de control como *if, else, y elif*, para que tu programa no solo muestre opciones sino que también responda de manera inteligente a cada una de ellas.

Pedir, procesar y mostrar datos

Ahora que ya estamos más familiarizados con Python vamos a poner manos a la obra con un ejercicio práctico que combina todo lo que venimos aprendiendo. **El objetivo es pedir datos, procesarlos y luego mostrar los resultados de manera clara.** Este ciclo es fundamental en cualquier aplicación real: recibir información, trabajar con ella y devolvérsela a quien interactúe de forma entendible.

La idea de esta práctica es que te sientas a gusto con el flujo básico de cualquier programa interactivo en Python. Primero, vas a recibir información que se ingrese, luego vas a hacer un cálculo o procesar esos datos de alguna forma y finalmente vas a mostrar los resultados en la terminal. Es un ciclo que vas a ver repetido una y otra vez en el desarrollo de aplicaciones y es esencial que lo domines, especialmente pensando en tu Proyecto Final Integrador (PFI).

Este ejercicio va a reforzar tu comprensión de cómo Python maneja la entrada, el procesamiento y la salida de datos. Además, vas a trabajar con diferentes tipos de datos: **cadenas de texto, números enteros y flotantes.** Y no sólo eso, también vas a practicar la conversión entre tipos cuando sea necesario. Al final, lo importante es que veas cómo Python puede manejar tareas cotidianas como gestionar inventarios y procesar datos de manera eficiente sentando las bases para proyectos más complejos.

¡Salimos a cenar!

Vamos a escribir un programa que calcule el costo total de una salida a cenar para un grupo de personas. En el mismo se pide la cantidad de personas, el precio promedio del plato y una propina fija del 10%.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

```
# Pedir la cantidad de personas
personas = int(input("¿Cuántas personas van a cenar?: "))

# Pedir el precio promedio del plato por persona
precio_plato = float(input("¿Cuál es el precio promedio del
plato?: "))

# Calcular el total de la cena (sin propina)
total_cena = personas * precio_plato

# Calcular la propina (10%)
propina = total_cena * 0.10

# Calcular el total final con propina
total_con_propina = total_cena + propina

# Mostrar los resultados
print("Total de la cena (sin propina): ", total_cena)
print("Propina (10%): ", propina)
print("Total a pagar (con propina): ", total_con_propina)
```

Este programa es bastante simple, pero muy útil para calcular el costo total de una cena para un grupo, teniendo en cuenta el precio promedio de los platos y la propina fija del 10%.

Primero, se pide que se ingrese cuántas personas van a cenar, usando **input()** y convirtiendo la respuesta en un número entero con int(). Luego, se le pide que ingrese el precio promedio del plato por persona, y esta entrada se convierte en un número decimal con float().

Con esos datos, el programa calcula el total de la cena sin propina multiplicando la cantidad de personas por el precio del plato. Para calcular la propina, que es un 10% del total de la cena, multiplica el total por 0.10. Luego, el programa suma el total de la cena y la propina para obtener el total final con la propina incluida.

Por último el programa muestra tres resultados con la función **print()**: el total de la cena sin propina, el valor de la propina y el total a pagar con la propina incluida.

Pero este programa puede mejorarse, y mucho, con algunos pequeños cambios:

```
# Pedir la cantidad de personas
personas = int(input("¿Cuántas personas van a cenar?: "))

# Pedir el precio promedio del plato por persona
precio_plato = float(input("¿Cuál es el precio promedio del
plato?: "))

# Calcular el total de la cena (sin propina)
total_cena = personas * precio_plato

# Calcular la propina (10%)
propina = total_cena * 0.10

# Calcular el total final con propina
total_con_propina = total_cena + propina

# Mostrar los resultados usando f-strings
print(f"Total de la cena (sin propina): {total_cena:.2f}")
print(f"Propina (10%): {propina:.2f}")
print(f"Total a pagar (con propina): {total_con_propina:.2f}")
```

Buenos Aires aprende
Agencia de Habilidades para el Futuro

¿Qué son las f-strings y cómo funcionan?

Las **f-strings** (o "formatted strings") son una forma súper cómoda y moderna de manejar la salida de datos en Python. Se introdujeron en Python 3.6 y permiten incrustar valores o expresiones directamente dentro de las cadenas de texto de una manera mucho más simple y legible.

Las **f-strings** te permiten escribir una cadena de texto precedida por la **letra f,** y dentro de esa cadena podés usar llaves “{ }” para **insertar variables o expresiones directamente**.

```
nombre = "Juan"
edad = 25

# Usamos f-string para incrustar las variables dentro del
texto
print(f"Hola, {nombre}. Tenés {edad} años.")
```

En este caso, la **f-string** toma el valor de las variables nombre y edad y las inserta directamente en la cadena de texto. El resultado que se muestra sería:

```
Hola, Juan. Tenés 25 años.
```

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Ventajas de las f-strings

Son de extrema utilidad para combinar texto y variables de manera súper intuitiva sin necesidad de concatenar o usar complicadas combinaciones de símbolos. Esto hace que el código sea más fácil de escribir y de leer.

Además de variables, podés insertar expresiones completas dentro de las llaves. Por ejemplo:

```
print(f"El doble de 5 es {5 * 2}")
```

Esto evalúa directamente la expresión 5 * 2 y muestra el resultado.

Las f-strings también te permiten formatear números de manera fácil, como mostramos en el código anterior con .2f. Esto le dice a Python que muestre el número con dos decimales, lo cual es muy útil cuando trabajás con dinero, por ejemplo.

En el código del ejemplo modificado:
Usamos f-strings para mejorar la forma en que mostramos los resultados:

```
print(f"Total de la cena (sin propina): {total_cena:.2f}")
print(f"Propina (10%): {propina:.2f}")
print(f"Total a pagar (con propina): {total_con_propina:.2f}")
```

Gracias a esto Python inserta el valor de total_cena, propina y total_con_propina directamente en la cadena de texto, y además los formatea con dos decimales para que el resultado sea más claro y profesional.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Ejercicios prácticos

Ticket de compra

¡Vamos a crear tu propio ticket de compra! El desafío es escribir un programa que le pida al usuario el nombre, la cantidad y el valor unitario de tres productos. Después, tu programa tiene que calcular el importe de IVA (21%) de cada uno y mostrar en la terminal un ticket de compra con todos los datos.

Código del ejemplo práctico

Este es un ejemplo del código que podrías haber escrito

```
# Solicitar el nombre, la cantidad y el valor unitario de tres productos
nombre1 = input("Ingresá el nombre del primer producto: ")
cantidad1 = int(input("Ingresá la cantidad: "))
valor_unitario1 = float(input("Ingresá el valor unitario: "))

nombre2 = input("Ingresá el nombre del segundo producto: ")
cantidad2 = int(input("Ingresá la cantidad: "))
valor_unitario2 = float(input("Ingresá el valor unitario: "))

nombre3 = input("Ingresá el nombre del tercer producto: ")
cantidad3 = int(input("Ingresá la cantidad: "))
valor_unitario3 = float(input("Ingresá el valor unitario: "))

# Calcular el importe de IVA (21%) para cada producto
iva1 = (valor_unitario1 * cantidad1) * 0.21
```

```
iva2 = (valor_unitario2 * cantidad2) * 0.21
iva3 = (valor_unitario3 * cantidad3) * 0.21

# Calcular el costo total de cada producto (sin IVA)
costo1 = valor_unitario1 * cantidad1
costo2 = valor_unitario2 * cantidad2
costo3 = valor_unitario3 * cantidad3

# Calcular el total con IVA
total_con_iva1 = costo1 + iva1
total_con_iva2 = costo2 + iva2
total_con_iva3 = costo3 + iva3

# Mostrar el ticket de compra simulado
print("\n--- TICKET DE COMPRA ---")
print("Cantidad\tDescripción\t\tPrecio Unitario\t\tTotal")
print(cantidad1, "\t\t", nombre1, "\t\t\t",
format(valor_unitario1, ".2f"), "\t\t\t",
format(total_con_iva1, ".2f"))
print(cantidad2, "\t\t", nombre2, "\t\t\t",
format(valor_unitario2, ".2f"), "\t\t\t",
format(total_con_iva2, ".2f"))
print(cantidad3, "\t\t", nombre3, "\t\t\t",
format(valor_unitario3, ".2f"), "\t\t\t",
format(total_con_iva3, ".2f"))
print("-----------------------------------------")

# Calcular el total neto y el IVA total
total_neto = costo1 + costo2 + costo3
```

```
iva_total = iva1 + iva2 + iva3
total_general = total_con_iva1 + total_con_iva2 +
total_con_iva3

# Mostrar el total neto, IVA y total general
print("\t\t\t\t\t\tTotal Neto:\t", format(total_neto, ".2f"))
print("\t\t\t\t\t\tIVA:\t\t", format(iva_total, ".2f"))
print("\t\t\t\t\t\tTotal:\t\t", format(total_general, ".2f"))
```

Este código te va a permitir simular un ticket de compra, con todos los detalles de los productos, el IVA y el total a pagar. ¡Probalo y fíjate cómo queda!

Consumo de combustible

En este ejercicio, vas a crear una aplicación en Python que calcule el consumo de combustible de un coche durante un viaje. El programa va a solicitar para funcionar que se ingrese cuántos litros de combustible gasta el auto por cada 100 km, el costo por litro del mismo y la distancia del viaje. Con estos datos, tu programa va a calcular cuánto consumió el vehículo y cuánto dinero se gastó en combustible. ¡Es súper útil!

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Resolución

```
# Entrada de datos
litros_por_100km = float(input("Ingresá la cantidad de litros que consume el coche por cada 100 km: "))
costo_por_litro = float(input("Ingresá el costo de cada litro de combustible: "))
longitud_viaje = float(input("Ingresá la longitud del viaje en kilómetros: "))

# Proceso: Calcular los litros consumidos y el dinero gastado
litros_consumidos = (litros_por_100km * longitud_viaje) / 100
dinero_gastado = litros_consumidos * costo_por_litro

# Salida: Mostrar los litros consumidos y el dinero gastado
print("Litros de combustible consumidos: ", litros_consumidos)
print("Dinero gastado en combustible: $", dinero_gastado)
```

Este programa es una interesante práctica para trabajar con operaciones matemáticas y la entrada de datos del usuario. Te permite ver cómo Python puede ser utilizado para resolver problemas cotidianos como calcular los costos de un viaje en auto.

Buenos Aires aprende

Agencia de Habilidades para el Futuro

Buenos Aires Ciudad