Fundamentos de Python

Unidad 1. Introducción a Python y estructuras de control

PROJECTOS

1. Simulador de cajero automático
2. Sistema de inicio de sesión

ESTUDIANTE

Armando Gabriel Jacinto López

INSTITUCION

INFOTEC

FECHA

10/09/2025

Introducción

El presente informe analiza dos proyectos de desarrollo en Python, enfocados en las estructuras de control y tipos de datos. El primer proyecto es un **simulador básico de cajero automático**, diseñado para dispensar billetes de manera eficiente y gestionar un inventario limitado. El segundo proyecto consiste en un **sistema de inicio de sesión** que autentica usuarios con un número restringido de intentos.

El objetivo de estos proyectos es evaluar la funcionalidad, la estructura del código y las mejores prácticas aplicadas en ambos proyectos. Se examinarán tanto los aciertos en la implementación de la lógica, como el uso adecuado de las estructuras de control y datos, con el fin de poner en práctica los conceptos aprendidos durante esta primera unidad. Los proyectos, aunque resuelven sus problemas de manera funcional, sirven como una excelente base para discutir principios de diseño de software más avanzados.

Desarrollo

Proyecto 1: Simulador de Cajero Automático

El proyecto de cajero automático cumple con sus requisitos principales de manera directa. El algoritmo utiliza un **ciclo while** para continuar la operación de dispensar billetes hasta que el monto solicitado se reduzca a cero o no se pueda completar.

```
# Variables para contar billetes
FIFTY_BILL = 10
ONE_HUNDRED_BILL = 10
TWO_HUNDRED_BILL = 10
FIVE_HUNDRED_BILL = 10
THOUSAND_BILL = 10

# Variables para entregar billetes
FIFTY_BILL_COUNT = 0
ONE_HUNDRED_BILL_COUNT = 0
TWO_HUNDRED_BILL_COUNT = 0
FIVE_HUNDRED_BILL_COUNT = 0
THOUSAND_BILL_COUNT = 0

# Variable para ejecutar el ciclo principal
IS_RUNNING = True
```

Ilustración 1 Variables globales

La lógica principal reside en la función calculate, que usa un ciclo for para iterar a través de las denominaciones de billetes en orden descendente (de $1000 a $50). Este enfoque garantiza que se entregue la menor cantidad de billetes posible, como se especifica en los requisitos del proyecto.

```
# Funcion para calcular el numero de billetes a entregar
def calculate(amount):
    # Crear un diccionario para asignar un valor al nombre de las variables
    denominations = {
        1000: 'THOUSAND_BILL',
        500: 'FIVE_HUNDRED_BILL',
        200: 'TWO_HUNDRED_BILL',
        100: 'ONE_HUNDRED_BILL',
        50: 'FIFTY_BILL'
    }

    # Iterar en los valores en orden decendente
    for bill, var_name in denominations.items():
        if amount >= bill and globals()[var_name] > 0:
            amount -= bill
            # globals() para modificar las variables globales
            globals()[var_name] -= 1
            globals()[var_name + "_COUNT"] += 1
            return amount
```

Ilustración 2 Función principal

La gestión del inventario y el conteo de billetes dispensados se realiza a través de variables globales. El código utiliza la función globals() de Python para modificar estas variables desde dentro de la función calculate. Las variables FIFTY_BILL, ONE_HUNDRED_BILL, etc., representan el inventario disponible, mientras que sus contrapartes con el sufijo _COUNT almacenan los billetes entregados.

```python
# Inicio del programa
# Pedir un monto al usuario
print("ATM MACHINE")
answer = input("Insert the total amount (0 to close the program): ")
# Transformar la entrada a un numero entero
amount = int(answer)

# Loop principal
# se ejecutara hasta que el monto sea menor que el valor del billete con menos valor
while IS_RUNNING:
    # Si el monto total supera la suma de todos los billetes, no retornar ninguno
    if amount > 18500:
        print('Invalid Amount')
        IS_RUNNING = False
    # Si el monto es mayor o igual a 50, ejecutar la funcion
    elif amount >= 50:
        amount = calculate(amount)
    # Terminar el programa
    else:
        IS_RUNNING = False

# Mostrar resultados
print(f"Here's your change {amount}")
print("YOUR BILLS:")
print(f"THOUSAND_BILL: {THOUSAND_BILL_COUNT}")
print(f"FIVE HUNDRED BILL: {FIVE_HUNDRED_BILL_COUNT}")
print(f"TWO HUNDRED BILL: {TWO_HUNDRED_BILL_COUNT}")
print(f"ONE HUNDRED BILL: {ONE_HUNDRED_BILL_COUNT}")
print(f"FYFTY BILL: {FIFTY_BILL_COUNT}")
```

Ilustración 3 Ciclo while e inicio del programa

```
ATM MACHINE
Insert the total amount (0 to close the program): 15700
Here's your change 0
YOUR BILLS:
THOUSAND_BILL: 10
FIVE HUNDRED BILL: 10
TWO HUNDRED BILL: 3
ONE HUNDRED BILL: 1
FYFTY BILL: 0
```

Ilustración 4 Ejecución del sistema de cajero

Proyecto 2: Sistema de Inicio de Sesión

El sistema de inicio de sesión cumple con sus objetivos de forma sencilla y eficaz. Utiliza un **ciclo while** controlado por dos condiciones: una variable booleana VALID_CREDENTIALS y un contador ATTEMPS_LEFT. Este enfoque es un patrón de diseño común para bucles con un número limitado de iteraciones.

```python
# Credenciales validas
CORRECT_USER = "admin"
CORRECT_PASSWORD = "1234"
# Numero de intentos
ATTEMPS_LEFT = 3
# Variable para controlar el ciclo
VALID_CREDENTIALS = False

# Funcion para reducir el numero de intentos
def subtract_attemp():
    globals()['ATTEMPS_LEFT'] -= 1
```

Ilustración 5 Variables globales y la función para restar los intentos

La lógica de autenticación se maneja a través de una serie de **estructuras if-elif-else** que evalúan las entradas del usuario. El código es capaz de proporcionar mensajes de error específicos para casos como "Usuario inválido", "Contraseña inválida" o "Credenciales inválidas" (ambas incorrectas), lo que mejora la experiencia del usuario.

```python
# Menu principal
print("LOG IN PROGRAM")

# Ciclo while para iterar el numero de intentos
while not VALID_CREDENTIALS and ATTEMPS_LEFT > 0:
    # Mostrar intentos restantes
    print(f"Attempts left {ATTEMPS_LEFT}")
    # Entradas del usuario
    user = input("Insert the user name: ")
    password = input("Insert the password: ")
    # Validar ambas credenciales
    if user == CORRECT_USER and password == CORRECT_PASSWORD:
        VALID_CREDENTIALS = True
        print("Valid Credentials. You log in successfully")
    # Validar si las dos credenciales son erroneas
    elif user != CORRECT_USER and password != CORRECT_PASSWORD:
        subtract_attemp()
        print("Invalid Credentials (User and Password)")
    # Validar si el usuario es correcto
    elif user != CORRECT_USER:
        subtract_attemp()
        print("Invalid User")
    # Validar si la contrasena es correcto
    elif password != CORRECT_PASSWORD :
        subtract_attemp()
        print("Invalid Password")
    # Cualquier otro valor sera invalido
    else:
        subtract_attemp()
        print("Insert valid credentials")
```

Ilustración 6 Inicio del programa y ciclo while

```
LOG IN PROGRAM
Attempts left 3
Insert the user name: armando
Insert the password: 1234
Invalid User
Attempts left 2
Insert the user name: admin
Insert the password: 1234
Valid Credentials. You log in successfully
```

Ilustración 7 Ejecución del sistema de sesión

Conclusión

Ambos proyectos demuestran una comprensión sólida de los conceptos fundamentales de las estructuras de control y tipos de datos en el lenguaje Python, como el uso de estructuras de control (if, elif, else) y ciclos (while, for). Cumplen de manera efectiva con los requisitos establecidos, resolviendo los problemas planteados con soluciones directas.

Sin embargo, se tienen áreas de mejora en ambos proyectos, pudiendo evolucionar a diseños más robustos y estructurados. Una posible mejora podría ser aplicar conceptos de la programación orientada a objetos para gestionar los sistemas de manera más efectiva. Al igual, la seguridad en ambos proyectos podría ser mejor, como en el caso del sistema de inicio de sesión, no escribiendo las credenciales directamente en el código para evitar que sean vistas por usuarios con acceso al código.

En conclusión, estos dos proyectos sirvieron como una base fundamental para en un futuro embarcarse en problemas con mayor complejidad, de momento, se ha adoptado una buena base de conocimientos para seguir progresando en el programa e indagar en conceptos con mayor complejidad.