1B. EJERCICIO. ESTRUCTURAS DE CONTROL Y TIPOS DE DATOS

Unidad 1. Introducción a Python y estructuras de control

Datos
Nombre: Keith Alexis Gutiérrez Ibarra
Docente: Silvia Guadalupe Victorio Aguilar
Fecha de elaboración: 18/09/2025

Introducción

En este trabajo se desarrollarán dos ejercicios un sistema de cajero automático de retiro de dinero y un sistema de inicio de sesión en Python, donde el primero es un cajero automático para retiros que entrega la menor cantidad de billetes, mantiene un inventario por denominaciones y no dispensa billetes si no cuenta con los suficientes, el segundo al ser un sistema de inicio de sesión busca validar credenciales predefinidas y controla la cantidad de intentos permitidos, el propósito de estos ejercicios es desarrollar la lógica de programación usando estructuras de control y algoritmos.

Desarrollo

Ejercicio 1

a) Análisis del problema: Para el primer ejercicio se nos piden las siguientes condiciones que debe tener el algoritmo

- Entregar siempre la menor cantidad de billetes posible.
- Mantener un inventario que indique cuál es la cantidad de billetes disponible por cada denominación.
- Las denominaciones disponibles son: 50, 100, 200, 500 y 1000.
- Si el inventario no cuenta con una combinación de billetes suficientes para satisfacer el importe solicitado, no dispensará ningún billete.
- Siempre que inicie el algoritmo, deberá haber en inventario 10 billetes de cada denominación.

b) Diseño del algoritmo

- Se establecieron las variables de los billetes en inventario con la cantidad de 10 cada uno y las denominaciones que pedía el ejercicio
- Se solicitó al usuario el monto que desea retirar.
- Se programaron las condiciones para validar el monto: Si el monto es menor o igual a 0 o si el monto no es múltiplo de 50 se mostrará mensaje de error para cualquiera de los dos casos.
- Se establecieron las variables para el numero de billetes a entregar junto a la variable que permite el descuento de las cantidades de los billetes

- Se calcularon los billetes a entregar, comenzando desde la mayor denominación hasta la menor, considerando la cantidad disponible en inventario.
- Se establecieron las condiciones de entrega: Donde si el monto puede ser entregado con los billetes disponibles, mostrará los billetes entregados y actualizará el inventario o si el monto no puede ser entregado por falta de billetes se mostrará mensaje de error indicando que no hay suficientes billetes.
- Se programo una condición para los errores de entrada, mostrando mensaje en caso de que el usuario ingrese un valor que no cumpla alguna de las condiciones esperadas.

c) Código en Python

Ejercicio 1. Cajero automático.py > ...

```python
# Variable para contar billetes
billetes_1000 = 10
billetes_500 = 10
billetes_200 = 10
billetes_100 = 10
billetes_50 = 10

try:
    #  Iniciamos el sistema
    print("\n--- Dispen12sadora de Billetes")
    # Solicitar monto
    entrada = input("Ingresa la cantidad de billetes\n")
    # Transformamos el valor dato a un tipo de dato entero
    monto = int(entrada)
    #Condiciones segun la entrada de la cantidad de billetes a retirar
    if monto <= 0:
        # Si el numero es negativo
        print("Error: Ingrese un monto positivo.")
    elif monto % 50 != 0:
        # Si el numero no es multiplo de 50.
        print("Error: El monto debe ser un múltiplo de $50.")
    else:
        # Variable para billetes a entregar
        entregar_1000 = 0
        entregar_500 = 0
        entregar_200 = 0
        entregar_100 = 0
        entregar_50 = 0
        monto_restante = monto
        # Calculo de billetes a entregar por denominacion y restando de mayor a menor
        entregar_1000 = min(monto_restante // 1000, billetes_1000)
        monto_restante -= entregar_1000 * 1000
```

```python
Ejercicio 1. Cajero automático.py > ...
            entregar_500 = min(monto_restante // 500, billetes_500)
            monto_restante -= entregar_500 * 500
            entregar_200 = min(monto_restante // 200, billetes_200)
            monto_restante-= entregar_200 * 200
            entregar_100 = min(monto_restante // 100, billetes_100)
            monto_restante -= entregar_100 * 100
            entregar_50 = min(monto_restante // 50, billetes_50)
            monto_restante -= entregar_50 * 50
            # Condicion de entrega de billtes y actualizacion de inventario
            if monto_restante == 0:
                print(f"\nTransacción exitosa por ${monto}")
                if entregar_1000 > 0: print(f"\nse entregaron {entregar_1000} billete(s) de $1000")
                if entregar_500 > 0: print(f"\nse entregaron {entregar_500} billete(s) de $500")
                if entregar_200 > 0: print(f"\nse entregaron {entregar_200} billete(s) de $200")
                if entregar_100 > 0: print(f"\nse entregaron {entregar_100} billete(s) de $100")
                if entregar_50 > 0: print(f"\nse entregaron {entregar_50} billete(s) de $50")
                # Actualizar el inventario
                billetes_1000 -= entregar_1000
                billetes_500 -= entregar_500
                billetes_200 -= entregar_200
                billetes_100 -= entregar_100
                billetes_50 -= entregar_50
            # Condicion cuando no hay suficientes billetes
            else:
                print("\nError: No hay suficientes billetes para completar la transacción.")
    # Condicion cuando la entrada es difrente a las esperadas
    except ValueError:
        print("Error: Por favor, ingrese un número válido.")
```

d) Ejecución y pruebas

Ejemplos del funcionamiento del código en diversos escenarios:

- Transacción exitosa:

```
ercicio 1. Cajero automático.py"

--- Dispensadora de Billetes
Ingresa la cantidad de billetes
6850

Transacción exitosa por $6850

se entregaron 6 billete(s) de $1000

se entregaron 1 billete(s) de $500

se entregaron 1 billete(s) de $200

se entregaron 1 billete(s) de $100

se entregaron 1 billete(s) de $50
PS C:\Users\keith\OneDrive\Documentos\Pyrhon>
```

- Error por monto negativo o cero:

```
PS C:\Users\keith\OneDrive\Documentos\Pyrhon> &
ercicio 1. Cajero automático.py"

--- Dispensadora de Billetes
Ingresa la cantidad de billetes
-2650
Error: Ingrese un monto positivo.
PS C:\Users\keith\OneDrive\Documentos\Pyrhon>
```

- Error por monto no múltiplo de 50:

```
PS C:\Users\keith\OneDrive\Documentos\Pyrhon> &
ercicio 1. Cajero automático.py"

--- Dispensadora de Billetes
Ingresa la cantidad de billetes
7856
Error: El monto debe ser un múltiplo de $50.
PS C:\Users\keith\OneDrive\Documentos\Pyrhon>
```

- Error por falta de billetes:

```
PS C:\Users\keith\OneDrive\Documentos\Pyrhon> & C:/Users/keith/App
ercicio 1. Cajero automático.py"

--- Dispensadora de Billetes
Ingresa la cantidad de billetes
25650

Error: No hay suficientes billetes para completar la transacción.
PS C:\Users\keith\OneDrive\Documentos\Pyrhon>
```

- Error por entrada no esperada:

```
PS C:\Users\keith\OneDrive\Documentos\Pyrhon> &
ercicio 1. Cajero automático.py"

--- Dispensadora de Billetes
Ingresa la cantidad de billetes
hshshsh
Error: Por favor, ingrese un número válido.
PS C:\Users\keith\OneDrive\Documentos\Pyrhon>
```

Ejercicio 2

a) Análisis del problema: Para el primer ejercicio se nos piden las siguientes condiciones que debe tener el algoritmo

- Se permite un máximo de 3 intentos.
- Mostrar un error al usuario cuando no exista uno de los datos (usuario y contraseña).
- Si uno de los campos (usuario y contraseña) está vacío deberá mostrar un error de autentificación.
- Únicamente usar estructuras de control.

b) Diseño del algoritmo

- Se establecieron las variables para las credenciales correctas para el usuario y contraseña, también se agregó la variable para la cantidad máxima de intentos permitidos.
- Se inició un bucle while que controla los intentos restantes.
- Se solicitó al usuario que ingresara su nombre de usuario y contraseña.
- Se programaron las condiciones para validar los campos ingresados:
  - Si los campos están vacíos se muestra un mensaje de advertencia sin descontar intentos,
  - Si el usuario y la contraseña coinciden con las credenciales correctas se muestra mensaje de “Inicio de sesión correcto” y se termina el bucle.
  - Si las credenciales no coinciden se muestra mensaje de “Credenciales inválidas” y se descuenta un intento.
- Cuando el número de intentos llega a cero se muestra un mensaje de “Te has quedado sin intentos”

c) Implementación en Python

Ejercicio 2. Inicio de sesión.py > ...

```python
# Credenciales validas (como variables individuales)
usuario_correcto= "admin"
clave_correcta= "1234"
# Variable de cantidad maxima de intentos
intentos = 3
# Bucle de inicio de sesion con un maximo de intentos hasta llegar a 0
while intentos > 0:
    # Incio del sistema
    print("\n--- Sistema de Inicio de Sesion ---")
    print(f"Intentos restantes: {intentos}")
    # Solicitar credenciales
    print("Usuario: ")
    usuario = input()
    print("Contraseña: ")
    clave = input()
    # Condiciones de inicio de sesion correcto
    if usuario == usuario_correcto and clave == clave_correcta:
        print("Inicio de sesion correcto")
        # Rompe el bucle12
        break
    # Condicion cuando los campos estan vacios
    elif not usuario or not clave:
        print("Error de autenticacion")
        continue
    # Condicion cuando las credenciales son incorrectas
    else:
        print("Credenciales invalidas")
        # Operacion que descuenta los intentos
        intentos -= 1
# Cuando en contador llega a 0 se cierra el bucle
else:
    print("Te has quedado sin intentos")
```

d) Ejecución y pruebas

Capturas de pantalla donde se muestre:

- Inicio de sesión correcto con las credenciales correctas

```
PS C:\Users\keith\OneDrive\Documentos\Pyrhon> & C:/
ercicio 2. Inicio de sesión.py"

--- Sistema de Inicio de Sesion ---
Intentos restantes: 3
Usuario:
admin
Contraseña:
1234
Inicio de sesion correcto
PS C:\Users\keith\OneDrive\Documentos\Pyrhon>
```

- Error por credenciales incorrectas.

```
PS C:\Users\keith\OneDrive\Documentos\Pyr
ercicio 2. Inicio de sesión.py"

--- Sistema de Inicio de Sesion ---
Intentos restantes: 3
Usuario:
user
Contraseña:
1234
Credenciales invalidas

--- Sistema de Inicio de Sesion ---
Intentos restantes: 2
Usuario:
```

- Error por campos vacíos.

```
--- Sistema de Inicio de Sesion ---
Intentos restantes: 2
Usuario:

Contraseña:

Error de autenticacion
```

- Mensaje final cuando se acaban los intentos.

```
--- Sistema de Inicio de Sesion ---
Intentos restantes: 1
Usuario:
kkkj
Contraseña:
jkasjk
Credenciales invalidas
Te has quedado sin intentos
PS C:\Users\keith\OneDrive\Documentos\Pyrhon>
```

Conclusiones

En este trabajo se desarrollaron dos sistemas en Python un cajero automático y un sistema de inicio de sesión donde el primero permitió aplicar lógica condicional y ciclos para entregar la menor cantidad de billetes posible, manejar un inventario por denominación y garantizar que no se realicen transacciones cuando no hay billetes suficientes, el segundo permite hacer uso de bucles y condicionales para validar credenciales, controlar la cantidad máxima de intentos y manejar entradas vacías o incorrectas.

Estos ejercicios permiten ver la importancia de ver la lógica antes de implementar el código, para así elaborar de mejor manera una aplicación, además sirve para consolidar conceptos que vimos en la unidad acerca de la estructura de control y algoritmos