

### **3. Ejercicio de Programación**

### **Intento de Fuerza Bruta**

**ESTUDIANTES:**

Juan Sebastian Londoño Mosquera – 1065871157

Keimer Duvan Moreno Murillo – 1077436457

**MATERIA:**

Herramientas de Programación I

**PROFESOR:**

JUAN DAVID OSORIO SANCHEZ

**FECHA:**

6/09/2025

# Código En Virtual Studio ( Python )

```
#Juan Londoño
# Y
#Keimer Moreno

#Codigo Fuerza Bruta con validaciones -----
-----
# Paso 1: Solicitar numero de usuarios (validando que sea un numero)
while True:
    num_usuarios = input("Cuantos usuarios desea registrar: ").strip()
    if num_usuarios.isdigit() and int(num_usuarios) > 0:
        num_usuarios = int(num_usuarios)
        break
    else:
        print("Por favor, ingrese un numero valido mayor que 0.")

# Paso 2: Crear diccionario para almacenar usuarios
usuarios = {}

for i in range(num_usuarios):
    print(f"\nRegistro del usuario {i+1}")

    # Validacion nombre (solo letras y espacios)
    while True:
        nombre = input("Ingrese el nombre: ").strip()
        if nombre.replace(" ", "").isalpha():
            break
        print("Nombre invalido. Solo se permiten letras y espacios.")

    # Validacion numero de cuenta bancaria (solo digitos y minimo 10)
    while True:
        cuenta = input("Ingrese el numero de cuenta bancaria: ").strip()
        if cuenta.isdigit() and len(cuenta) >= 10:
            break
        print("Numero de cuenta invalido. Debe contener solo numeros y minimo 10 digitos.")

    # Validacion clave de 3 digitos
    while True:
        clave = input("Ingrese clave de 3 digitos: ").strip()
        if clave.isdigit() and len(clave) == 3:
            print("Clave valida")
            break
        print("Clave invalida, intente nuevamente")

    # Guardamos en el diccionario
    usuarios[nombre] = {
        "cuenta": cuenta,
        "clave": clave
    }
    print("\nUsuario registrado correctamente\n")

# Paso 3: Solicitar el usuario a consultar
usuario_buscar = input("\nIngrese el nombre del usuario a consultar: ").strip()
```

```

if usuario_buscar in usuarios:
    clave_real = usuarios[usuario_buscar]["clave"]

    # Paso 4: Simular fuerza bruta
    print("\nIniciando ataque de fuerza bruta para encontrar la clave...")
    encontrada = False
    intentos = 0

    for intento in range(0, 1000): # claves de 000 a 999
        intentos += 1
        clave_intento = str(intento).zfill(3) # asegura formato 3 digitos

        if clave_intento == clave_real:
            print(f"Intento {intento}: {clave_intento} <-- ESTA AQUI ")
            encontrada = True
            print(f"\nClave encontrada: {clave_intento}")
            print(f"Total de intentos realizados: {intento}")
            break
        else:
            print(f"Intento {intento}: {clave_intento}")

    if not encontrada:
        print("No se encontro la clave (error en los datos.)")

    # Paso 5: Mostrar la informacion completa
    print("\nInformacion del usuario consultado:")
    print(f"Nombre: {usuario_buscar}")
    print(f"N Cuenta: {usuarios[usuario_buscar]['cuenta']}")
    print(f"Clave: {usuarios[usuario_buscar]['clave']}")

else:
    print("\nEl usuario no esta registrado.")

```

## Pruebas de ejecución

### Intento 1: Error al ingresar letras en vez de número de usuarios

Entrada:

Cuantos usuarios desea registrar: tres

Salida esperada en consola:

Por favor, ingrese un numero valido mayor que 0.

### Intento 2: Error en número de cuenta bancaria demasiado corto

Entrada:

Ingresel número de cuenta bancaria: 12345

Salida esperada en consola:

Número de cuenta invalido. Debe contener solo numeros y minimo 10 digitos.

### Intento 3: Error en clave no numérica

Entrada:

Ingresar clave de 3 digitos: ab1

Salida esperada en consola:

Clave invalida, intente nuevamente

## OUTPUT

```
Registro del usuario 1
Ingresar el nombre: 123
Nombre invalido. Solo se permiten letras y espacios.
Ingresar el nombre: h 0 L a
Ingresar el numero de cuenta bancaria: Pedro
Número de cuenta invalido. Debe contener solo numeros y minimo 10 digitos.
Ingresar el numero de cuenta bancaria: 12345
Número de cuenta invalido. Debe contener solo numeros y minimo 10 digitos.
Ingresar el numero de cuenta bancaria: 1234577880
Ingresar clave de 3 digitos: 5554
Clave invalida, intente nuevamente
Ingresar clave de 3 digitos: dsf
Clave invalida, intente nuevamente
Ingresar clave de 3 digitos: 123
Clave valida

Usuario registrado correctamente

Registro del usuario 2
Ingresar el nombre: juanita
Ingresar el numero de cuenta bancaria: 123123
Número de cuenta invalido. Debe contener solo numeros y minimo 10 digitos.
Ingresar el numero de cuenta bancaria: 1231212312321
Ingresar clave de 3 digitos: 123
Clave valida

Usuario registrado correctamente

Ingresar el nombre del usuario a consultar: Pedro

El usuario no esta registrado.
Press any key to continue . . .
```

Como se logra apreciar en esta imagen, vemos que valida bien los errores por cada atributo/dato del sistema creado y verifica si lo ingresado es correcto, sino, vuelve a repetírtelo con un mensaje incluido. Al ingresar lo correcto, prosigue con su función y después tienes que escoger el usuario a atacar , si no ingresas el correcto el programa se termina.

```
Cuantos usuarios desea registrar: 1

Registro del usuario 1
Ingrrese el nombre: profesor
Ingrrese el numero de cuenta bancaria: 1234567890
Ingrrese clave de 3 digitos: 123
Clave valida

Usuario registrado correctamente

Ingrrese el nombre del usuario a consultar: profesor

Iniciando ataque de fuerza bruta para encontrar la clave...
Intento 1: 000
Intento 2: 001
Intento 3: 002
Intento 4: 003
Intento 5: 004
Intento 6: 005
Intento 7: 006
Intento 8: 007
Intento 9: 008
Intento 10: 009
Intento 11: 010
Intento 12: 011
Intento 13: 012
Intento 14: 013
Intento 15: 014
Intento 16: 015
Intento 17: 016
Intento 18: 017
Intento 19: 018
Intento 20: 019
Intento 21: 020
Intento 22: 021
```

Aquí vemos una versión con solo un usuario registrado en la que esta vez si ingresamos el nombre correcto que pusimos arriba "Profesor" y empezó el ataque en menos de un segundo ya tenía la clave encontrada.

```
Intento 121: 121
Intento 122: 121
Intento 123: 122
Intento 124: 123  <-- ESTA AQUI

Clave encontrada: 123
Total de intentos realizados: 124

Informacion del usuario consultado:
Nombre: profesor
N Cuenta: 1234567890
Clave: 123
Press any key to continue . . .
```

Por ultimo como logramos apreciar, en el intento 124 ya que el 1 es el 0, encontró la clave exitosamente y te lo remarca con un texto para lograr apreciar la clave correctamente y mas abajo se muestran los datos exactos exitosamente encontrados.

The screenshot shows a Visual Studio code editor window with a Python script and a terminal window below it.

**Python Script Content:**

```
7 # Paso 1: Solicitar numero de usuarios (validando que sea un numero)
8 while True:
9     num_usuarios = input("Cuantos usuarios desea registrar: ").strip()
10    if num_usuarios.isdigit() and int(num_usuarios) > 0:
11        num_usuarios = int(num_usuarios)
12        break
13    else:
14        print("Por favor, ingrese un numero valido mayor que 0.")
15
16 # Paso 2: Crear diccionario para almacenar usuarios
17 usuarios = {}
18
19 for i in range(num_usuarios):
20     print(f"\nRegistro del usuario {i+1}")
21
22     # Validacion nombre (solo letras y espacios)
23     while True:
24         nombre = input("Ingrese el nombre: ").strip()
25         if nombre.replace(" ", "").isalpha():
26             break
27         print("Nombre invalido. Solo se permiten letras y espacios.")
28
29     # Validacion numero de cuenta bancaria (solo digitos y minimo 10)
30     while True:
31         cuenta = input("Ingrese el numero de cuenta bancaria: ").strip()
32         if cuenta.isdigit() and len(cuenta) >= 10:
33             break
34         print("Numero de cuenta invalido. Debe contener solo numeros y minimo 10 digitos.")
```

**Terminal Output:**

```
Cuantos usuarios desea registrar: 1
Registro del usuario 1
Ingrese el nombre: juan
Ingrese el numero de cuenta bancaria: 1234567890
Ingrese clave de 3 digitos: 123
Clave valida

Usuario registrado correctamente
```

**Visual Studio Status Bar:**

Línea: 32 | Carácter: 51 | SPC | CRLF

## Visual Studio