ESCUELA DE ESTUDIOS SUPERIORES CALMECAC
LICENCIATURA EN SISTEMAS COMPUTACIONALES ADMINISTRATIVOS
JOSE DANIEL MORA ORTIZ

ACTIVIDAD IA 802

Actividad 1: Programación en Python

A continuación, se presentan ejercicios para practicar estructuras de control (if, for, while, switch, funciones). Cada ejercicio incluye el código y su explicación para que puedas entender el **por qué** y **para qué** de su uso. Tu tarea es **replicar, modificar y comentar** cada fragmento de código.

Estructura condicional if

Ejercicio 1. Verifica si un número es positivo, negativo o cero.

```python
edad = int(input("Introduce tu edad: "))

if edad < 13:
    print("Eres un niño.")
elif edad < 18:
    print("Eres un adolescente.")
elif edad < 60:
    print("Eres un adulto.")
else:
    print("Eres un adulto mayor.")
```

¿Por qué if?
La estructura if-elif-else permite clasificar casos mutuamente excluyentes. Aquí usamos rangos para decidir en qué etapa de la vida se encuentra una persona.

Preguntas de lógica:

1. ¿Qué sucedería si no incluyéramos el else?

   El programa no imprimiría nada si el número es cero

2. ¿Cómo cambiarías el código para incluir una categoría “adulto joven” de 18 a 29 años?

   Se agrega un elif con la condición >=18 <= 30

3. ¿Qué pasaría si alguien introduce una edad negativa?

   Se debería manejar con otra condición para evitar errores

ESCUELA DE ESTUDIOS SUPERIORES CALMECAC
LICENCIATURA EN SISTEMAS COMPUTACIONALES ADMINISTRATIVOS
JOSE DANIEL MORA ORTIZ

Ejercicio 2: Validación de contraseña con advertencia

```python
usuario = input("Usuario: ")
password = input("Contraseña: ")

if usuario == "admin" and password == "secreto123":
    print("Acceso concedido.")
elif usuario != "admin":
    print("Usuario incorrecto.")
else:
    print("Contraseña incorrecta.")
```

¿Por qué if?
Permite validar condiciones combinadas con operadores lógicos (and, or). Aquí se usa para diferenciar fallas de autenticación.

Preguntas de lógica:

1. ¿Qué operadores lógicos se usan en el código?

   Se usa 'and' para validar ambas condiciones y '!=' para comparar valores

2. ¿Por qué es importante validar el usuario antes que la contraseña?

   Por seguridad y eficiencia, evitando intentos innecesarios

3. ¿Cómo implementarías un intento fallido limitado a 3 veces?

   Usando un contador y un bucle while para controlar los intentos

Ejercicio 3: Mostrar tabla de multiplicar de un número

```python
numero = int(input("Introduce un número: "))

for i in range(1, 11):
    print(f"{numero} x {i} = {numero * i}")

```

¿Por qué for?
El for se usa cuando conoces de antemano el número de repeticiones. Aquí se usa para imprimir del 1 al 10.

Preguntas de lógica:

1. ¿Qué hace range(1, 11) exactamente?

   Genera números del 1 al 10

2. ¿Cómo podrías modificar el código para imprimir la tabla al revés?

   Usando range(10, 0, -1)

3. ¿Cómo harías que imprima todas las tablas del 1 al 10?

   Usando un bucle anidado para recorrer los números del 1 al 10

Ejercicio 4: Contar letras en una palabra (sin espacios)

```python
palabra = input("Escribe una frase: ")
contador = 0

for letra in palabra:
    if letra != " ":
        contador += 1

print("Número de letras (sin espacios):", contador)

```

¿Por qué for con if?
El for recorre carácter por carácter, y el if filtra los espacios. Esto combina lógica de control con análisis de datos.

Preguntas de lógica:

1. ¿Qué otros caracteres podrías querer ignorar además de espacios?

   Signos de puntuación, números, tabulaciones y saltos de línea

2. ¿Cómo podrías contar solo vocales?

   Filtrando solo caracteres en "aeiouAEIOU"

3. ¿Qué pasa si no usas if?

   Se contarían todos los caracteres sin distinción

Ejercicio 5: Validar entrada hasta que sea correcta

```python
respuesta = ""

while respuesta != "python":
    respuesta = input("¿Cuál es el mejor lenguaje de programación?: ")

print("¡Correcto!")

```

¿Por qué while?
Se utiliza cuando no sabes cuántas veces se va a repetir el ciclo. El programa sigue preguntando hasta que se cumpla la condición.

Preguntas de lógica:

1. ¿Qué pasaría si el usuario nunca escribe la palabra correcta?

   El programa entraría en un bucle infinito

2. ¿Cómo agregarías un número máximo de intentos?

   Usando una variable de contador

3. ¿Puedes cambiar el programa para que no importe si escriben “Python” con mayúscula?

   Convirtiendo la entrada a minúsculas antes de compararla

Ejercicio 6: Cajero automático simulado

```python
saldo = 1000
opcion = ""

while opcion != "salir":
    print("\n1. Consultar saldo\n2. Retirar\n3. Salir")
    opcion = input("Elige una opción: ")

    if opcion == "1":
        print("Saldo actual:", saldo)
    elif opcion == "2":
        cantidad = int(input("¿Cuánto deseas retirar?: "))
        if cantidad <= saldo:
            saldo -= cantidad
            print("Retiro exitoso. Nuevo saldo:", saldo)
        else:
            print("Fondos insuficientes.")
    elif opcion == "3":
        opcion = "salir"
    else:
        print("Opción no válida.")

```

¿Por qué match-case?

Permite manejar múltiples casos claramente, como un switch. Es más legible que varios if.

Preguntas de lógica:

1. ¿Qué ventaja tiene match sobre if en este caso?

   Sintaxis más clara y legible

2. ¿Qué pasa si el usuario pone otro símbolo?

   Se mostrará "Opción no válida"

3. ¿Qué deberías validar antes de dividir?

   Que el divisor no sea cero, aunque en este código en particular no se realiza ninguna división

Ejercicio 7: Menú de operaciones matemáticas

```python
op = input("Elige una operación (+, -, *, /): ")
a = int(input("Primer número: "))
b = int(input("Segundo número: "))

match op:
    case "+":
        print("Resultado:", a + b)
    case "-":
        print("Resultado:", a - b)
    case "*":
        print("Resultado:", a * b)
    case "/":
        print("Resultado:", a / b if b != 0 else "No se puede dividir entre cero")
    case _:
        print("Operación no válida.")

```

¿Por qué match-case?
Permite manejar múltiples casos claramente, como un switch. Es más legible que varios if.

Preguntas de lógica:

1. ¿Qué ventaja tiene match sobre if en este caso?

   Permite una sintaxis más clara y legible

2. ¿Qué pasa si el usuario pone otro símbolo?

   Si el usuario ingresa un símbolo o un valor que no es uno de los esperados ("+", "-", "*", "/"), el programa imprimirá "operación no valida."

3. ¿Qué deberías validar antes de dividir?

   Que el divisor no sea cero

Ejercicio 8: Función que devuelve si un número es primo

```python
def es_primo(n):
    if n < 2:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

print(es_primo(13))  # True

```

¿Por qué función?

Porque encapsulamos una tarea reutilizable. Y se usa la raíz cuadrada como optimización.

Preguntas de lógica:

1. ¿Qué hace exactamente el bucle for dentro de la función?

   Verifica si el número es divisible por algún otro número

2. ¿Por qué se empieza desde 2 y no desde 1?

   Porque todos los números son divisibles por 1

3. ¿Qué pasaría si no usáramos return dentro del if?

   El bucle continuaría ejecutándose, incluso después de encontrar un divisor