**Ejercicios Prácticos: Operadores Lógicos**

**📝 Instrucciones**

* Resuelve los ejercicios en orden
* Intenta predecir el resultado antes de ejecutar el código
* Verifica tus respuestas con Python
* Si te equivocas, analiza por qué

**🟢 NIVEL 1: Básico (Operadores Fundamentales)**

**Ejercicio 1.1: Predice los Resultados**

# Evalúa sin ejecutar:

print(True and False)

print(True or False)

print(not True)

print(not False)

**Tu predicción**: FALSE, TRUE, FALSE, TRUE  
**Resultado real**: FALSE, TRUE, FALSE, TRUE  
**Explicación**: Leyes de tabla de verdad

**Ejercicio 1.2: Operadores Combinados**

a, b, c = True, False, True

print(a and b) # ?

print(a or b) # ?

print(b or c) # ?

print(a and c) # ?

**Tus predicciones**: FALSE, TRUE, TRUE, TRUE

**Ejercicio 1.3: Precedencia**

a, b, c = True, False, True

print(a and b or c) # ?

print(a or b and c) # ?

print(not a or b) # ?

print(not (a or b)) # ?

**Tus predicciones**: TRUE, TRUE, FALSE, FALSE

**Ejercicio 1.4: Comparaciones y Lógica**

x = 5

print(x > 3 and x < 10) # ?

print(x < 3 or x > 10) # ?

print(not x > 3) # ?

**Tus predicciones**: TRUE, FALSE, FALSE

**Ejercicio 1.5: Comparaciones Encadenadas**

x = 5

print(3 < x < 10) # ?

print(1 <= x <= 3) # ?

print(10 > x > 3) # ?

**Tus predicciones**: TRUE, FALSE, TRUE

**🟡 NIVEL 2: Intermedio (Valores y Cortocircuito)**

**Ejercicio 2.1: Valores Retornados**

print("hola" and "mundo") # ?

print("hola" and "") # ?

print("" and "mundo") # ?

print("hola" or "mundo") # ?

print("" or "mundo") # ?

**Tus predicciones**: mundo, “”, “”, hola, mundo

**Ejercicio 2.2: Truthy y Falsy**

print(bool(0)) # ?

print(bool("")) # ?

print(bool([])) # ?

print(bool([0])) # ?

print(bool(" ")) # ?

print(bool(None)) # ?

**Tus predicciones**: FALSE, FALSE, FALSE, TRUE, TRUE, FALSE

**Ejercicio 2.3: Evaluación de Cortocircuito**

Evalúa qué se imprime:

def f1():

print("f1 ejecutada")

return True

def f2():

print("f2 ejecutada")

return False

# Caso 1

print("Caso 1:")

resultado = f1() and f2()

print(f"Resultado: {resultado}")

# Caso 2

print("\nCaso 2:")

resultado = f2() and f1()

print(f"Resultado: {resultado}")

# Caso 3

print("\nCaso 3:")

resultado = f1() or f2()

print(f"Resultado: {resultado}")

**Tu predicción**: f1 f2 false, f2 false, f1 true

**Ejercicio 2.4: Operadores de Pertenencia**

nums = [1, 2, 3, 4, 5]

print(3 in nums) # ?

print(6 in nums) # ?

print(6 not in nums) # ?

word = "Python"

print("P" in word) # ?

print("p" in word) # ?

print("th" in word) # ?

**Tus predicciones**: TRUE, FALSE, TRUE, TRUE, FALSE, TRUE

**Ejercicio 2.5: Identidad vs Igualdad**

lista1 = [1, 2, 3]

lista2 = [1, 2, 3]

lista3 = lista1

print(lista1 == lista2) # ?

print(lista1 is lista2) # ?

print(lista1 == lista3) # ?

print(lista1 is lista3) # ?

**Tus predicciones**: TRUE, FALSE, TRUE, TRUE

**🔴 NIVEL 3: Avanzado (Aplicaciones Prácticas)**

**Ejercicio 3.1: Validación de Formulario**

Implementa la función validar\_datos que verifica si: - El nombre tiene entre 2 y 30 caracteres - El email contiene '@' - La edad es mayor o igual a 18 - La contraseña tiene al menos 8 caracteres

def validar\_datos(nombre, email, edad, password):

# Tu código aquí

pass

# Pruebas

print(validar\_datos("Ana", "ana@email.com", 25, "secreto123")) # Debe ser True

print(validar\_datos("", "no-email", 15, "123")) # Debe ser False

**Ejercicio 3.2: Sistema de Autorización**

Implementa un sistema que determine si un usuario puede acceder a un recurso basado en: - Debe estar autenticado - Debe ser administrador O tener el permiso específico - No debe estar en la lista negra

def puede\_acceder(usuario, permiso\_requerido, lista\_negra):

# Tu código aquí

pass

# Usuario ejemplo

admin = {

"id": 1,

"autenticado": True,

"admin": True,

"permisos": ["leer", "escribir"]

}

usuario\_normal = {

"id": 2,

"autenticado": True,

"admin": False,

"permisos": ["leer"]

}

usuario\_bloqueado = {

"id": 3,

"autenticado": True,

"admin": False,

"permisos": ["leer", "escribir"]

}

lista\_negra = [3, 4]

print(puede\_acceder(admin, "borrar", lista\_negra)) # True

print(puede\_acceder(usuario\_normal, "leer", lista\_negra)) # True

print(puede\_acceder(usuario\_normal, "escribir", lista\_negra)) # False

print(puede\_acceder(usuario\_bloqueado, "leer", lista\_negra)) # False

**Ejercicio 3.3: Acceso Seguro a Diccionario**

Implementa una función obtener\_valor\_seguro que retorne: - El valor de la clave si existe - Un valor predeterminado si la clave no existe

def obtener\_valor\_seguro(diccionario, clave, predeterminado=None):

# Tu código aquí (sin usar .get())

pass

config = {"timeout": 30, "retries": 3}

print(obtener\_valor\_seguro(config, "timeout")) # 30

print(obtener\_valor\_seguro(config, "cache")) # None

print(obtener\_valor\_seguro(config, "cache", 60)) # 60

**Ejercicio 3.4: Filtrar Lista**

Escribe una función para filtrar una lista de productos según criterios: - Precio dentro de un rango (min y max) - Opcionalmente filtrar por categoría - Solo productos disponibles

def filtrar\_productos(productos, precio\_min, precio\_max, categoria=None):

# Tu código aquí

pass

productos = [

{"nombre": "Laptop", "precio": 1200, "categoria": "Electrónica", "disponible": True},

{"nombre": "Teléfono", "precio": 800, "categoria": "Electrónica", "disponible": False},

{"nombre": "Libro", "precio": 15, "categoria": "Libros", "disponible": True},

{"nombre": "Audífonos", "precio": 200, "categoria": "Electrónica", "disponible": True},

]

print(filtrar\_productos(productos, 0, 500))

print(filtrar\_productos(productos, 100, 1000, "Electrónica"))

**Ejercicio 3.5: Evaluación de Riesgo**

Implementa un sistema de evaluación de riesgo crediticio:

def evaluar\_riesgo(cliente):

"""

Evalúa si un cliente tiene bajo riesgo crediticio.

Criterios:

- Score crediticio alto (>700) O

- Ingreso anual >50000 Y historial > 2 años O

- Cliente VIP Y sin deudas pendientes

"""

# Tu código aquí

pass

cliente1 = {

"nombre": "Ana García",

"score\_crediticio": 720,

"ingreso\_anual": 45000,

"años\_historial": 3,

"vip": False,

"deudas\_pendientes": False

}

cliente2 = {

"nombre": "Luis Pérez",

"score\_crediticio": 680,

"ingreso\_anual": 60000,

"años\_historial": 4,

"vip": False,

"deudas\_pendientes": False

}

cliente3 = {

"nombre": "Carmen Ruiz",

"score\_crediticio": 690,

"ingreso\_anual": 30000,

"años\_historial": 1,

"vip": True,

"deudas\_pendientes": False

}

print(evaluar\_riesgo(cliente1)) # True

print(evaluar\_riesgo(cliente2)) # True

print(evaluar\_riesgo(cliente3)) # True

**📊 Ejercicios de Debugging**

**Debug 1: Encuentra el Error**

# Este código debería verificar si el usuario tiene permisos

def verificar\_permisos(usuario, accion):

if usuario["permisos"] and accion in usuario["permisos"]:

return True

else:

return False

# Prueba

usuario = {"id": 1, "nombre": "Juan"}

print(verificar\_permisos(usuario, "leer"))

**¿Qué está mal?**: El diccionario no tienen ”permisos”

**Debug 2: Encuentra el Error**

# Este código debería filtrar estudiantes aprobados

estudiantes = [

{"nombre": "Ana", "nota": 85},

{"nombre": "Luis", "nota": None},

{"nombre": "Carmen", "nota": 92}

]

aprobados = [e for e in estudiantes if e["nota"] >= 60]

print(aprobados)

**¿Qué está mal?**: No puede comparar “nota” de Luis al ser “None”