Repositorio de palabras

1. False

- **Definición**: Se refiere al valor booleano falso, que indica una condición incorrecta o que no se cumple.
- **Ejemplo**: activo = False

```python
# Sistema de autenticación
usuario_logueado = False
if usuario_logueado == False:
    print("Debe iniciar sesión")
    print("Acceso denegado")
```

2. None

- **Definición**: Representa la ausencia de un valor o la falta de contenido en una variable
- **Ejemplo**: resultado = None

```python
# Función que puede no retornar valor
def buscar_usuario(id):
    if id == 1:
        return "Juan"
    return None

resultado = buscar_usuario(5)
if resultado is None:
    print("Usuario no encontrado")
```

3. True

- **Definición**: Representa el valor booleano verdadero

- **Ejemplo**: confirmado = True

```python
# Sistema de configuración
modo_desarrollo = True
if modo_desarrollo == True:
    print("Ejecutando en modo desarrollo")
    print("Logs detallados activados")
```

4. and

- **Definición**: un operador lógico que devuelve verdadero solo si todas las condiciones que evalúa son verdaderas. Si alguna de las condiciones es falsa, el resultado de la operación AND será falso.
- **Ejemplo**: if edad >= 18 and tiene_licencia:

```python
# Validación de acceso
edad = 25
es_miembro = True

if edad >= 18 and es_miembro:
    print("Bienvenido al club")
else:
    print("Acceso restringido")
```

5. as

- **Definición**: Actuando generalmente como un alias para un objeto o para definir tipos
- **Ejemplo**: import numpy as np

```python
# Importar con alias
import datetime as dt

# Usar el alias
ahora = dt.datetime.now()
print(f"Fecha actual: {ahora.strftime('%Y-%m-%d')}")
```

6. assert

- **Definición**: Verifica que una condición sea verdadera
- **Ejemplo**: assert x > 0, "x debe ser positivo"

```python
# Validación con assert
def calcular_promedio(notas):
    assert len(notas) > 0, "La lista no puede estar vacía"
    assert all(0 <= nota <= 100 for nota in notas), "Notas deben estar entre
    return sum(notas) / len(notas)

# Probar la función
notas = [85, 90, 78]
promedio = calcular_promedio(notas)
print(f"Promedio: {promedio}")
```

7. break

- **Definición**: Sale de un bucle
- **Ejemplo**: if condicion: break

```python
# Validación con assert
def calcular_promedio(notas):
    assert len(notas) > 0, "La lista no puede estar vacía"
    assert all(0 <= nota <= 100 for nota in notas), "Notas deben estar entre
    return sum(notas) / len(notas)

# Probar la función
notas = [85, 90, 78]
promedio = calcular_promedio(notas)
print(f"Promedio: {promedio}")
```

8. class

- **Definición**: Define una clase
- **Ejemplo**: class Persona:

```python
# Definir una clase simple
class Estudiante:
    def __init__(self, nombre, edad):
        self.nombre = nombre
        self.edad = edad

    def presentarse(self):
        return f"Hola, soy {self.nombre} y tengo {self.edad} años"

# Usar la clase
alumno = Estudiante("Ana", 20)
print(alumno.presentarse())
```

9. continue

- **Definición**: para saltarse la iteración actual y pasar inmediatamente a la siguiente, sin ejecutar el resto del código de esa iteración
- **Ejemplo**: if x % 2 == 0: continue

```python
# Definir una clase simple
class Estudiante:
    def __init__(self, nombre, edad):
        self.nombre = nombre
        self.edad = edad

    def presentarse(self):
        return f"Hola, soy {self.nombre} y tengo {self.edad} años"

# Usar la clase
alumno = Estudiante("Ana", 20)
print(alumno.presentarse())
```

10. def

- **Definición**: Define una función
- **Ejemplo**: def saludar():

```python
# Definir función para calcular área
def calcular_area_rectangulo(largo, ancho):
    area = largo * ancho
    return area

# Usar la función
resultado = calcular_area_rectangulo(5, 3)
print(f"El área del rectángulo es: {resultado}")
```

11. del

- **Definición**: Elimina una variable o elemento
- **Ejemplo**: del lista[0]

```python
# Gestión de memoria
lista_tareas = ["lavar", "cocinar", "estudiar", "limpiar"]
print(f"Tareas iniciales: {lista_tareas}")

del lista_tareas[1]  # Eliminar "cocinar"
print(f"Después de eliminar: {lista_tareas}")

# Eliminar variable completa
temp = "dato temporal"
del temp
# print(temp)  # Esto daría error
```

12. elif

- **Definición**: "Else if" - condición alternativa
- **Ejemplo**: elif nota >= 70:

```python
# Sistema de calificaciones
nota = 85

if nota >= 90:
    print("Calificación: A")
elif nota >= 80:
    print("Calificación: B")
elif nota >= 70:
    print("Calificación: C")
else:
    print("Calificación: F")
```

13. else

- **Definición**: Alternativa cuando la condición es falsa

- **Ejemplo**: else: print("No cumple")

```python
# Verificar si un número es par
numero = 7

if numero % 2 == 0:
    print(f"{numero} es par")
else:
    print(f"{numero} es impar")
```

14. except

- **Definición**: Maneja excepciones
- **Ejemplo**: except ValueError:

```python
# Manejo seguro de entrada del usuario
try:
    edad = int(input("Ingrese su edad: "))
    print(f"Usted tiene {edad} años")
except ValueError:
    print("Error: Debe ingresar un número válido")
except Exception as e:
    print(f"Error inesperado: {e}")
```

15. finally

- **Definición**: Bloque que siempre se ejecuta después de try/except
- **Ejemplo**: finally: archivo.close()

```python
# Manejo de archivo con cleanup
def procesar_archivo():
    archivo = None
    try:
        archivo = open("datos.txt", "r")
        contenido = archivo.read()
        print("Archivo procesado exitosamente")
    except FileNotFoundError:
        print("Archivo no encontrado")
    finally:
        if archivo:
            archivo.close()
            print("Archivo cerrado")

procesar_archivo()
```

16. for

- **Definición**: Crea bucles de iteración
- **Ejemplo**: for i in range(10):

```python
# Iterar sobre diferentes tipos de datos
frutas = ["manzana", "banana", "naranja"]

print("Lista de frutas:")
for i, fruta in enumerate(frutas):
    print(f"{i+1}. {fruta}")

print("\nTabla del 5:")
for i in range(1, 11):
    print(f"5 x {i} = {5*i}")
```

17. from

- **Definición**: Importa elementos específicos de un módulo
- **Ejemplo**: from math import sqrt

```python
# Importar funciones específicas
from math import sqrt, pi, pow

# Calcular área de un círculo
radio = 5
area = pi * pow(radio, 2)
print(f"Área del círculo: {area:.2f}")

# Calcular hipotenusa
a, b = 3, 4
hipotenusa = sqrt(a**2 + b**2)
print(f"Hipotenusa: {hipotenusa}")
```

18. global

- **Definición**: Declara una variable global
- **Ejemplo**: global contador

```python
# Contador global
contador = 0

def incrementar():
    global contador
    contador += 1
    print(f"Contador: {contador}")

def reset_contador():
    global contador
    contador = 0
    print("Contador reiniciado")

# Usar las funciones
incrementar()  # Contador: 1
incrementar()  # Contador: 2
reset_contador()  # Contador reiniciado
```

19. if

- **Definición**: Estructura condicional
- **Ejemplo**: if edad >= 18:

```python
# Sistema de seguridad simple
password = "python123"
intento = input("Ingrese la contraseña: ")

if intento == password:
    print("Acceso concedido")
    print("Bienvenido al sistema")
else:
    print("Contraseña incorrecta")
```

20. import

- **Definición**: Importa módulos
- **Ejemplo**: import math

```python
# Importar módulos completos
import random
import os

# Usar funciones del módulo
numero_aleatorio = random.randint(1, 100)
print(f"Número aleatorio: {numero_aleatorio}")

directorio_actual = os.getcwd()
print(f"Directorio actual: {directorio_actual}")
```

21. in

- **Definición**: Verifica si un elemento está en una secuencia
- **Ejemplo**: if 'a' in palabra:

```python
# Verificar membresía
colores_disponibles = ["rojo", "azul", "verde", "amarillo"]
color_elegido = "azul"

if color_elegido in colores_disponibles:
    print(f"Color {color_elegido} disponible")
else:
    print("Color no disponible")

# Buscar en texto
texto = "Python es genial"
if "Python" in texto:
    print("Encontré Python en el texto")
```

22. is

- **Definición**: Compara identidad de objetos
- **Ejemplo**: if x is None:

```python
# Comparar identidad vs igualdad
lista1 = [1, 2, 3]
lista2 = [1, 2, 3]
lista3 = lista1

print(f"lista1 == lista2: {lista1 == lista2}")  # True (mismo contenido)
print(f"lista1 is lista2: {lista1 is lista2}")  # False (diferentes objet
print(f"lista1 is lista3: {lista1 is lista3}")  # True (mismo objeto)

# Uso típico con None
valor = None
if valor is None:
    print("El valor es None")
```

23. lambda

- **Definición**: Crea funciones anónimas
- **Ejemplo**: cuadrado = lambda x: x**2

```python
# Función lambda simple
cuadrado = lambda x: x ** 2
print(f"5 al cuadrado: {cuadrado(5)}")

# Usar lambda con map y filter
numeros = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
cuadrados = list(map(lambda x: x**2, numeros))
pares = list(filter(lambda x: x % 2 == 0, numeros))

print(f"Cuadrados: {cuadrados}")
print(f"Números pares: {pares}")
```

24. nonlocal

- **Definición**: Declara variable en ámbito no local
- **Ejemplo**: nonlocal contador

```python
# Función anidada con nonlocal
def crear_contador():
    count = 0

    def incrementar():
        nonlocal count
        count += 1
        return count

    def obtener_valor():
        return count

    return incrementar, obtener_valor

# Usar el contador
inc, get = crear_contador()
print(f"Valor inicial: {get()}")  # 0
print(f"Después de incrementar: {inc()}")  # 1
print(f"Después de incrementar: {inc()}")  # 2
```

25. not

- **Definición**: Operador lógico de negación
- **Ejemplo**: if not activo:

```python
# Sistema de verificación
usuario_bloqueado = False
sesion_activa = True

if not usuario_bloqueado:
    print("Usuario permitido")

if not sesion_activa:
    print("Debe iniciar sesión")
else:
    print("Sesión activa")

# Verificar lista vacía
lista = []
if not lista:
    print("La lista está vacía")
```

26. or

- **Definición**: Operador lógico "o"
- **Ejemplo**: if lluvia or nieve:

```python
# Sistema de alertas meteorológicas
lluvia = False
nieve = True
granizo = False

if lluvia or nieve or granizo:
    print("Alerta meteorológica activa")
    print("Se recomienda precaución al salir")
else:
    print("Condiciones climáticas normales")
```

27. pass

- **Definición**: No hace nada, placeholder
- **Ejemplo**: def funcion_pendiente(): pass

```python
# Estructura de programa en desarrollo
class Vehiculo:
    def __init__(self, marca):
        self.marca = marca

    def acelerar(self):
        pass  # TODO: implementar más tarde

    def frenar(self):
        pass  # TODO: implementar más tarde

# Función placeholder
def procesar_datos():
    pass  # Implementación pendiente

# Manejo de excepción sin acción
try:
    resultado = 10 / 2
except ZeroDivisionError:
    pass  # Ignorar error por ahora
```

28. raise

- **Definición**: Lanza una excepción
- **Ejemplo**: raise ValueError("Error")

```python
# Validación personalizada
def validar_edad(edad):
    if not isinstance(edad, int):
        raise TypeError("La edad debe ser un número entero")

    if edad < 0:
        raise ValueError("La edad no puede ser negativa")

    if edad > 150:
        raise ValueError("Edad no realista")

    return True

# Probar la validación
try:
    validar_edad(25)
    print("Edad válida")
except (TypeError, ValueError) as e:
    print(f"Error: {e}")
```

29. return

- **Definición**: Retorna un valor de una función
- **Ejemplo**: return resultado

```python
# Función con múltiples returns
def clasificar_numero(num):
    if num > 0:
        return "positivo"
    elif num < 0:
        return "negativo"
    else:
        return "cero"

# Función que retorna múltiples valores
def operaciones_basicas(a, b):
    suma = a + b
    resta = a - b
    multiplicacion = a * b
    return suma, resta, multiplicacion

# Usar las funciones
print(clasificar_numero(-5))
resultado = operaciones_basicas(10, 3)
print(f"Resultados: {resultado}")
```

30. try

- **Definición:** Inicia manejo de excepciones
- **Ejemplo:** try: resultado = 10/x

```python
# Calculadora segura
def calculadora():
    try:
        num1 = float(input("Primer número: "))
        operacion = input("Operación (+, -, *, /): ")
        num2 = float(input("Segundo número: "))

        if operacion == "+":
            resultado = num1 + num2
        elif operacion == "-":
            resultado = num1 - num2
        elif operacion == "*":
            resultado = num1 * num2
        elif operacion == "/":
            resultado = num1 / num2
        else:
            print("Operación no válida")
            return

        print(f"Resultado: {resultado}")

    except ValueError:
        print("Error: Ingrese números válidos")
    except ZeroDivisionError:
        print("Error: No se puede dividir por cero")

calculadora()
```

31. while

- **Definición**: Crea bucles condicionales
- **Ejemplo**: while contador < 10:

```python
# Calculadora segura
def calculadora():
    try:
        num1 = float(input("Primer número: "))
        operacion = input("Operación (+, -, *, /): ")
        num2 = float(input("Segundo número: "))

        if operacion == "+":
            resultado = num1 + num2
        elif operacion == "-":
            resultado = num1 - num2
        elif operacion == "*":
            resultado = num1 * num2
        elif operacion == "/":
            resultado = num1 / num2
        else:
            print("Operación no válida")
            return

        print(f"Resultado: {resultado}")

    except ValueError:
        print("Error: Ingrese números válidos")
    except ZeroDivisionError:
        print("Error: No se puede dividir por cero")

calculadora()
```

32. with

- **Definición**: Maneja context managers
- **Ejemplo**: with open("archivo.txt") as f:

```python
# Manejo automático de archivos
def escribir_log(mensaje):
    with open("log.txt", "a") as archivo:
        from datetime import datetime
        timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        archivo.write(f"[{timestamp}] {mensaje}\n")
    # El archivo se cierra automáticamente

# Múltiples archivos
def copiar_archivo():
    try:
        with open("origen.txt", "r") as origen, open("destino.txt", "w") as desti
            contenido = origen.read()
            destino.write(contenido.upper())
        print("Archivo copiado y convertido a mayúsculas")
    except FileNotFoundError:
        print("Archivo origen no encontrado")

escribir_log("Sistema iniciado")
escribir_log("Usuario conectado")
```

33. yield

- **Definición**: Crea generadores
- **Ejemplo**: yield valor

```python
# Generador de números Fibonacci
def fibonacci_generator(limite):
    a, b = 0, 1
    count = 0
    while count < limite:
        yield a
        a, b = b, a + b
        count += 1

# Generador de números pares
def numeros_pares(maximo):
    num = 0
    while num <= maximo:
        if num % 2 == 0:
            yield num
        num += 1

# Usar los generadores
print("Primeros 10 números Fibonacci:")
for num in fibonacci_generator(10):
    print(num, end=" ")

print("\n\nNúmeros pares hasta 20:")
for par in numeros_pares(20):
    print(par, end=" ")
```

34. async

- **Definición**: Define funciones asíncronas
- **Ejemplo**: async def funcion_asincrona():

```python
import asyncio

# Función asíncrona simple
async def saludar_async(nombre, delay):
    print(f"Iniciando saludo para {nombre}")
    await asyncio.sleep(delay)  # Simula operación que toma tiempo
    print(f"¡Hola {nombre}!")
    return f"Saludo completado para {nombre}"

# Función principal asíncrona
async def main():
    # Ejecutar múltiples saludos concurrentemente
    tareas = [
        saludar_async("Ana", 1),
        saludar_async("Luis", 2),
        saludar_async("María", 1.5)
    ]

    resultados = await asyncio.gather(*tareas)

    for resultado in resultados:
        print(resultado)

# Ejecutar el programa asíncrono
# asyncio.run(main())  # Descomenta para ejecutar
```

35. await

- **Definición**: Espera resultado de función asíncrona
- **Ejemplo**: resultado = await funcion_asincrona()

```python
import asyncio
import random

# Simular descarga de archivo
async def descargar_archivo(nombre_archivo):
    print(f"Iniciando descarga de {nombre_archivo}")

    # Simular tiempo de descarga variable
    tiempo_descarga = random.uniform(1, 3)
    await asyncio.sleep(tiempo_descarga)

    print(f"Descarga de {nombre_archivo} completada")
    return f"Archivo {nombre_archivo} descargado en {tiempo_descarga:.2f}s"

# Función que usa await
async def procesar_descargas():
    archivos = ["imagen1.jpg", "video.mp4", "documento.pdf"]

    for archivo in archivos:
        resultado = await descargar_archivo(archivo)
        print(f"Resultado: {resultado}")

# Programa principal
async def programa_principal():
    print("Iniciando descarga de archivos...")
    await procesar_descargas()
    print("Todas las descargas completadas")
```