**Instituto Superior del Milagro**

**Tecnicatura Superior en Análisis de Sistemas**

**Profesora: Ramiro, Medina**

**PROGRAMACIÓN III**

**MANEJO DE VARIABLES, SENTENCIAS CONDICIONALES Y BUCLES**

Texto

Descripción generada automáticamente

**Alumna:**

**Castillo Sonia Verónica**

**Ejercicio 18:** Definir una función que calcule la suma y multiplicación de los elementos de una lista.

>>> def suma\_y\_multiplicacion(lista):

... suma = 0

... multiplicacion = 1

... for elemento in lista:

... suma += elemento

... multiplicacion \*= elemento

... return suma, multiplicacion

...

>>> def main():

... lista\_numeros = [3, 13, 23, 33, 43, 53, 63, 73, 83, 93]

... suma, multiplicacion = suma\_y\_multiplicacion(lista\_numeros)

... print(f"La suma de los elementos de la lista es: {suma}")

... print(f"La multiplicacion de los elementos de la lista es: {multiplicacion}")

...

>>> if \_\_name\_\_ == "\_\_main\_\_":

... main()

...

La suma de los elementos de la lista es: 480

La multiplicacion de los elementos de la lista es: 2394832584543399

>>>

**Ejercicio 19:** Escribir una función que reciba una lista de números y un exponente y devuelva otra lista donde sus elementos correspondan a la lista original elevados al exponente dado.

>>> def elevar\_lista\_a\_exponente(lista, exponente):

... lista\_elevada = [elemento \*\* exponente for elemento in lista]

... return lista\_elevada

...

>>> def main():

... lista\_numeros = [12, 23, 34, 45, 56, 67, 78, 89]

... exponente = 3

... lista\_resultado = elevar\_lista\_a\_exponente(lista\_numeros, exponente)

... print(f"La lista original es: {lista\_numeros}")

... print(f"La lista con los elementos elevados al exponente {exponente} es: {lista\_resultado}")

...

>>> if \_\_name\_\_ == "\_\_main\_\_":

... main()

...

La lista original es: [12, 23, 34, 45, 56, 67, 78, 89]

La lista con los elementos elevados al exponente 3 es: [1728, 12167, 39304, 91125, 175616, 300763, 474552, 704969]

>>>

**Ejercicio 20:** Escribir una función que convierta un número entero expresado en base decimal a binario y otra función que convierta un número entero binario a base decimal.

>>> def decimal\_a\_binario(numero\_decimal):

... if numero\_decimal == 0:

... return "0"

... binario = ""

... while numero\_decimal > 0:

... binario = str(numero\_decimal % 2) + binario

... numero\_decimal = numero\_decimal // 2

... return binario

...

>>> def binario\_a\_decimal(numero\_binario):

... numero\_decimal = 0

... potencia = 0

... for digito in reversed(numero\_binario):

... if digito == '1':

... numero\_decimal += 2 \*\* potencia

... potencia += 1

... return numero\_decimal

...

>>> def main():

... numero\_decimal = 10

... binario = decimal\_a\_binario(numero\_decimal)

... print(f"El número decimal {numero\_decimal} en binario es: {binario}")

... numero\_binario = "1010"

... decimal = binario\_a\_decimal(numero\_binario)

... print(f"El número binario {numero\_binario} en decimal es: {decimal}")

...

>>> if \_\_name\_\_ == "\_\_main\_\_":

... main()

...

El número decimal 10 en binario es: 1010

El número binario 1010 en decimal es: 10

>>>

**Ejercicio 21:** Definir una función llamada separar() que tome una lista de números enteros y devuelva dos listas ordenadas de menor a mayor: la primera con los números pares y la segunda con los números impares.

>>> def separar(lista):

... pares = []

... impares = []

... for numero in lista:

... if numero % 2 == 0:

... pares.append(numero)

... else:

... impares.append(numero)

... pares.sort()

... impares.sort()

... return pares, impares

...

>>> def main():

... lista\_numeros = [3, 14, 25, 36, 47, 58, 69, 80]

... lista\_pares, lista\_impares = separar(lista\_numeros)

... print(f"Lista de números pares: {lista\_pares}")

... print(f"Lista de números impares: {lista\_impares}")

...

>>> if \_\_name\_\_ == "\_\_main\_\_":

... main()

...

Lista de números pares: [14, 36, 58, 80]

Lista de números impares: [3, 25, 47, 69]

>>>

**Ejercicio 22:** Definir una función que genere una cantidad finita de cadenas con números enteros que puedan servir como contraseñas, las cadenas tendrán una longitud definida. La longitud de la cadena será un valor pasado por defecto en 8.

>>> import random

>>> import string

>>> def generar\_contraseñas(cantidad, longitud=8):

... contraseñas = []

... for \_ in range(cantidad):

... contraseña = ''.join(random.choices(string.digits, k=longitud))

... contraseñas.append(contraseña)

... return contraseñas

...

>>> def main():

... cantidad\_contraseñas = 7

... contraseñas = generar\_contraseñas(cantidad\_contraseñas)

... print("Contraseñas generadas:")

... for contraseña in contraseñas:

... print(contraseña)

...

>>> if \_\_name\_\_ == "\_\_main\_\_":

... main()

...

Contraseñas generadas:

45583893

72339050

43871721

39747536

39541699

44578433

88852560

>>>

**Ayuda:** El método random.choice(cad) devuelve un carácter elegido al azar de la secuencia cad proporcionada. El método string.join(iterable) une los elementos del iterable (un ciclo for, por ejemplo) y devuelve la cadena resultante como salida.

**Ejercicio 23:** Definir una función que concatena las cadenas que se reciba como parámetros no importa la cantidad de estos, de forma que devuelva una oración.

>>> def concatenar\_cadenas(\*cadenas):

... resultado = ' '.join(cadenas)

... return resultado

...

>>> def main():

... cadena1 = "Hola"

... cadena2 = "este es un"

... cadena3 = "calido dia de verano"

... resultado = concatenar\_cadenas(cadena1, cadena2, cadena3)

... print(f"La oración concatenada es: '{resultado}'")

...

>>> if \_\_name\_\_ == "\_\_main\_\_":

... main()

...

La oración concatenada es: 'Hola este es un calido dia de verano'

>>>

**Ejercicio 24:** Definir una función que reciba un diccionario que contenga nombre completo y correo electrónico de una lista de personas, la cantidad de personas puede variar y debe mostrarse de forma ordenada. Un ejemplo de los datos sería: Cáceres, Pedro usuario1@gmail.com Gómez, Gustavo usuario2@hotmail.com Lamas, María [usuario3@hotmail.com](mailto:usuario3@hotmail.com)

>>> def mostrar\_ordenado(diccionario):

... nombres\_ordenados = sorted(diccionario.keys())

... for nombre in nombres\_ordenados:

... print(f"{nombre}: {diccionario[nombre]}")

...

>>> def main():

... personas = {

... "Caceres, Pedro": "usuario1@gmail.com",

... "Gomez, Gustavo": "usuario2@hotmail.com",

... "Lamas, Maria": "usuario3@hotmail.com",

... "Rodriguez, Ana": "usuario4@yahoo.com",

... "Martinez, Juan": "usuario5@gmail.com",

... "Perez, Laura": "usuario6@hotmail.com",

... "Sanchez, Diego": "usuario7@yahoo.com",

... "Fernandez, Carla": "usuario8@gmail.com",

... "Lopez, Mario": "usuario9@hotmail.com",

... "Diaz, Sofia": "usuario10@yahoo.com"

... }

... mostrar\_ordenado(personas)

...

>>> if \_\_name\_\_ == "\_\_main\_\_":

... main()

...

Caceres, Pedro: usuario1@gmail.com

Diaz, Sofia: usuario10@yahoo.com

Fernandez, Carla: usuario8@gmail.com

Gomez, Gustavo: usuario2@hotmail.com

Lamas, Maria: usuario3@hotmail.com

Lopez, Mario: usuario9@hotmail.com

Martinez, Juan: usuario5@gmail.com

Perez, Laura: usuario6@hotmail.com

Rodriguez, Ana: usuario4@yahoo.com

Sanchez, Diego: usuario7@yahoo.com

>>>

**Ejercicio 25:** Definir una función que reciba un parámetro por referencia y otra que sea un parámetro por valor.

>>> def por\_referencia(lista):

... lista.append(4)

...

>>> mi\_lista = [5, 6, 7, 8, 9, 10]

>>> por\_referencia(mi\_lista)

>>> print(mi\_lista)

[5, 6, 7, 8, 9, 10, 4]

>>>

>>> mi\_numero = 5

>>> por\_valor(mi\_numero)

>>> print(mi\_numero)

5

>>>

**ARGUMENTOS INDETERMINADOS Y FUNCIONES INTEGRADAS. MANEJO DE EXCEPCIONES.**

**Ejercicio 26:** Escribir una función que solicite una cadena de texto al usuario e indique cuántas letras mayúsculas tiene.

>>> def contar\_mayusculas():

... cadena = input("Ingrese una cadena de texto: ")

... contador = 0

... for caracter in cadena:

... if caracter.isupper():

... contador += 1

... return contador

...

>>> def main():

... cantidad\_mayusculas = contar\_mayusculas()

... print(f"La cadena tiene {cantidad\_mayusculas} letras mayúsculas.")

...

>>> if \_\_name\_\_ == "\_\_main\_\_":

... main()

...

Ingrese una cadena de texto: '"Hola Mundo"'

La cadena tiene 2 letras mayúsculas.

>>>

>>> if \_\_name\_\_ == "\_\_main\_\_":

... main()

...

Ingrese una cadena de texto: '"Python es Genial"'

La cadena tiene 2 letras mayúsculas.

>>> if \_\_name\_\_ == "\_\_main\_\_":

... main()

...

Ingrese una cadena de texto: '"Esta Es Una Frase"'

La cadena tiene 4 letras mayúsculas.

>>>

**Ejercicio 27:** Escribir un programa que pida dos o más palabras separadas por una coma e indique si cada par de palabras riman o no. Si coinciden las tres últimas letras tiene que indicar que riman. Si coinciden sólo las dos últimas tiene que indicar que riman un poco y si no, que no riman.

>>> def verificar\_rima(palabras):

... palabras = [palabra.strip() for palabra in palabras.split(",")]

... resultados = []

... for i in range(len(palabras) - 1):

... palabra1 = palabras[i]

... palabra2 = palabras[i + 1]

... if len(palabra1) >= 3 and len(palabra2) >= 3:

... if palabra1[-3:].lower() == palabra2[-3:].lower():

... resultados.append("riman")

... elif palabra1[-2:].lower() == palabra2[-2:].lower():

... resultados.append("riman un poco")

... else:

... resultados.append("no riman")

... else:

... resultados.append("no riman")

... return resultados

...

>>> def main():

... entrada = input("Ingrese dos o más palabras separadas por coma: ")

... resultados = verificar\_rima(entrada)

... for i, resultado in enumerate(resultados, start=1):

... print(f"Par {i}: {resultado}")

...

>>> if \_\_name\_\_ == "\_\_main\_\_":

... main()

...

Ingrese dos o más palabras separadas por coma: gato, perro, lago, rio, loma, cerro

Par 1: no riman

Par 2: no riman

Par 3: no riman

Par 4: no riman

Par 5: no riman

>>> if \_\_name\_\_ == "\_\_main\_\_":

... main()

...

Ingrese dos o más palabras separadas por coma: camión, avión, casa, masa, perro, gato

Par 1: riman

Par 2: no riman

Par 3: riman

Par 4: no riman

Par 5: no riman

>>>

**Ejercicio 28:** Realizar un programa que muestre al usuario una lista de productos, permita ingresar el código del producto elegido y la cantidad de unidades que desea comprar. Finalmente, el programa debe calcular el total a pagar simulando una factura. Diseñe todas las funciones que considere necesarias para el correcto funcionamiento del programa.

>>> productos = [

... {"código": "001", "nombre": "Manzanas", "precio": 120.0},

... {"codigo": "002", "nombre": "Plátanos", "precio": 80.0},

... {"codigo": "003", "nombre": "Papas", "precio": 150.0},

... {"codigo": "004", "nombre": "Zanahorias", "precio": 100.0},

... {"codigo": "005", "nombre": "Lechuga", "precio": 70.0},

... {"codigo": "006", "nombre": "Tomates", "precio": 200.0},

... {"codigo": "007", "nombre": "Pan", "precio": 90.0},

... {"codigo": "008", "nombre": "Huevos", "precio": 180.0},

... {"codigo": "009", "nombre": "Queso", "precio": 250.0},

... {"codigo": "010", "nombre": "Yogurt", "precio": 120.0}

... ]

>>> def mostrar\_productos():

... print("Lista de productos disponibles:")

... print("--------------------------------")

... for producto in productos:

... print(f"Código: {producto['codigo']} - Nombre: {producto['nombre']} - Precio: ${producto['precio']}")

... print("--------------------------------")

...

>>> def realizar\_compra():

... codigo = input("Ingrese el código del producto que desea comprar: ")

... cantidad = int(input("Ingrese la cantidad de unidades que desea comprar: "))

... producto\_encontrado = None

... for producto in productos:

... if producto['codigo'] == codigo:

... producto\_encontrado = producto

... break

... if producto\_encontrado:

... total\_pagar = producto\_encontrado['precio'] \* cantidad

... return producto\_encontrado, cantidad, total\_pagar

... else:

... return None, 0, 0.0

...

>>> def generar\_factura(producto, cantidad, total\_pagar):

... print("Factura de compra")

... print("-----------------")

... print(f"Producto: {producto['nombre']}")

... print(f"Cantidad: {cantidad}")

... print(f"Precio unitario: ${producto['precio']}")

... print(f"Total a pagar: ${total\_pagar}")

... print("-----------------")

...

>>> def main():

... mostrar\_productos()

... producto, cantidad, total\_pagar = realizar\_compra()

... if producto:

... generar\_factura(producto, cantidad, total\_pagar)

... else:

... print("El producto ingresado no está disponible.")

...

>>> if \_\_name\_\_ == "\_\_main\_\_":

... main()

...

Lista de productos disponibles:

--------------------------------

Código: 001 - Nombre: Manzanas - Precio: $120.0

Código: 002 - Nombre: Plátanos - Precio: $80.0

Código: 003 - Nombre: Papas - Precio: $150.0

Código: 004 - Nombre: Zanahorias - Precio: $100.0

Código: 005 - Nombre: Lechuga - Precio: $70.0

Código: 006 - Nombre: Tomates - Precio: $200.0

Código: 007 - Nombre: Pan - Precio: $90.0

Código: 008 - Nombre: Huevos - Precio: $180.0

Código: 009 - Nombre: Queso - Precio: $250.0

Código: 010 - Nombre: Yogurt - Precio: $120.0

--------------------------------

Ingrese el código del producto que desea comprar: 008

Ingrese la cantidad de unidades que desea comprar: 12

Factura de compra

-----------------

Producto: Huevos

Cantidad: 12

Precio unitario: $180.0

Total a pagar: $2160.0

-----------------

>>>

>>> if \_\_name\_\_ == "\_\_main\_\_":

... main()

...

Lista de productos disponibles:

--------------------------------

Código: 001 - Nombre: Manzanas - Precio: $120.0

Código: 002 - Nombre: Plátanos - Precio: $80.0

Código: 003 - Nombre: Papas - Precio: $150.0

Código: 004 - Nombre: Zanahorias - Precio: $100.0

Código: 005 - Nombre: Lechuga - Precio: $70.0

Código: 006 - Nombre: Tomates - Precio: $200.0

Código: 007 - Nombre: Pan - Precio: $90.0

Código: 008 - Nombre: Huevos - Precio: $180.0

Código: 009 - Nombre: Queso - Precio: $250.0

Código: 010 - Nombre: Yogurt - Precio: $120.0

--------------------------------

Ingrese el código del producto que desea comprar: 007

Ingrese la cantidad de unidades que desea comprar: 6

Factura de compra

-----------------

Producto: Pan

Cantidad: 6

Precio unitario: $90.0

Total a pagar: $540.0

-----------------

>>>

>>> if \_\_name\_\_ == "\_\_main\_\_":

... main()

...

Lista de productos disponibles:

--------------------------------

Código: 001 - Nombre: Manzanas - Precio: $120.0

Código: 002 - Nombre: Plátanos - Precio: $80.0

Código: 003 - Nombre: Papas - Precio: $150.0

Código: 004 - Nombre: Zanahorias - Precio: $100.0

Código: 005 - Nombre: Lechuga - Precio: $70.0

Código: 006 - Nombre: Tomates - Precio: $200.0

Código: 007 - Nombre: Pan - Precio: $90.0

Código: 008 - Nombre: Huevos - Precio: $180.0

Código: 009 - Nombre: Queso - Precio: $250.0

Código: 010 - Nombre: Yogurt - Precio: $120.0

--------------------------------

Ingrese el código del producto que desea comprar: 001

Ingrese la cantidad de unidades que desea comprar: 9

Factura de compra

-----------------

Producto: Manzanas

Cantidad: 9

Precio unitario: $120.0

Total a pagar: $1080.0

-----------------

>>>

**Ejercicio 29:** Las reglas de cierto juego con dados son las siguientes: El jugador debe hacer una apuesta de dinero y elegir lanzar con 1, 2 o 3 dados. Diseñe una función que trabaje con cualquier cantidad de dados y retorne la suma de sus valores. El script debe solicitar el monto de la apuesta, preguntar al usuario con cuántos dados jugará, generar aleatoriamente los valores de los dados e invocar adecuadamente a la función. Luego decidir el resultado de la apuesta de la siguiente manera:

● Si jugó con 1 dado, el jugador gana el 10% de lo apostado si la suma fue mayor a 4.

● Si jugó con 2 dados, el jugador gana el 50% de lo apostado si la suma fue mayor a 8.

● Si jugó con 3 dados, el jugador gana el 300% de lo apostado si la suma es 18.

>>> import random

>>> def lanzar\_dados(num\_dados):

... valores\_dados = [random.randint(1, 6) for \_ in range(num\_dados)]

... suma = sum(valores\_dados)

... return suma

...

>>> def jugar\_juego():

... apuesta = float(input("Ingrese el monto de la apuesta: "))

... num\_dados = int(input("¿Con cuántos dados desea jugar? (1, 2 o 3): "))

... if num\_dados < 1 or num\_dados > 3:

... print("Cantidad de dados no válida. Debe ser 1, 2 o 3.")

... return

... suma\_dados = lanzar\_dados(num\_dados)

... print(f"Resultado del lanzamiento de {num\_dados} dados: {suma\_dados}")

... if num\_dados == 1:

... if suma\_dados > 4:

... ganancia = 0.1 \* apuesta

... print(f"Ganaste el 10% de tu apuesta: {ganancia}")

... else:

... print("No ganaste en esta jugada.")

... elif num\_dados == 2:

... if suma\_dados > 8:

... ganancia = 0.5 \* apuesta

... print(f"Ganaste el 50% de tu apuesta: {ganancia}")

... else:

... print("No ganaste en esta jugada.")

... elif num\_dados == 3:

... if suma\_dados == 18:

... ganancia = 3.0 \* apuesta

... print(f"Ganaste el 300% de tu apuesta: {ganancia}")

... else:

... print("No ganaste en esta jugada.")

...

>>> jugar\_juego()

Ingrese el monto de la apuesta: 100

¿Con cuántos dados desea jugar? (1, 2 o 3): 3

Resultado del lanzamiento de 3 dados: 8

No ganaste en esta jugada.

>>> jugar\_juego()

Ingrese el monto de la apuesta: 1500

¿Con cuántos dados desea jugar? (1, 2 o 3): 2

Resultado del lanzamiento de 2 dados: 6

No ganaste en esta jugada.

>>> jugar\_juego()

Ingrese el monto de la apuesta: 500

¿Con cuántos dados desea jugar? (1, 2 o 3): 1

Resultado del lanzamiento de 1 dados: 5

Ganaste el 10% de tu apuesta: 50.0

>>>

>>> jugar\_juego()

Ingrese el monto de la apuesta: 300

¿Con cuántos dados desea jugar? (1, 2 o 3): 1

Resultado del lanzamiento de 1 dados: 6

Ganaste el 10% de tu apuesta: 30.0

>>>

**Ejercicio 30:** Escribir la función parrafos(), la cual debe generar un texto que contenga la cantidad de párrafos que especifique el usuario. Deberá generar las palabras aleatoriamente.

>>> import random

>>> palabras = ["casa", "perro", "gato", "mesa", "silla", "ordenador", "ventana", "puerta", "coche", "jardín", "flor", "árbol", "luz", "luna", "sol", "nube", "agua", "playa", "arena", "mar", "río", "montaña", "ciudad", "parque", "camino", "verano", "invierno", "primavera", "otoño", "calor", "frío", "comida", "bebida", "café", "té", "leche", "pan", "fruta", "verdura", "carne", "pescado", "pollo", "arroz", "pasta", "ensalada", "postre", "amor", "familia", "amistad", "alegría", "tristeza", "miedo", "sorpresa", "odio", "paz", "guerra", "trabajo", "estudio", "dinero", "tiempo", "esperanza", "sueño", "vida", "muerte", "corazón", "mente", "cuerpo", "alma", "belleza", "fealdad", "éxito", "fracaso", "fuerza", "debilidad", "poder", "humildad", "paciencia", "impaciencia", "bondad", "maldad", "verdad", "mentira", "justicia", "injusticia", "libertad", "esclavitud", "día", "noche", "mañana", "tarde", "hora", "minuto", "segundo", "mundo", "universo", "galaxia", "planeta", "sol", "luna", "estrella", "naturaleza", "arte", "ciencia", "cultura", "historia", "geografía", "matemáticas", "literatura", "música", "deporte", "viaje", "aventura", "felicidad", "éxito", "pasión", "energía", "creatividad", "inspiración", "pasado", "presente", "futuro", "alegría", "tristeza", "emoción", "sentimiento", "palabra", "silencio", "ruido", "vida", "muerte", "amor", "odio", "soledad", "compañía", "sabiduría", "ignorancia", "virtud", "vicio", "risa", "lágrima", "sueño", "realidad", "esperanza", "desesperación", "sonrisa", "dolor", "paz", "conflicto", "triunfo", "derrota", "fortaleza", "debilidad", "poder", "impotencia", "belleza", "fealdad", "calma", "tormenta", "amistad", "enemistad", "inicio", "fin", "comienzo", "final", "camino", "meta", "libertad", "prisión", "ganancia", "pérdida", "riqueza", "pobreza", "esperanza", "desesperanza", "victoria", "derrota", "héroe", "villano", "amor", "desamor", "dicha", "desdicha", "esperanza", "desesperanza", "propósito", "destino", "causa", "efecto", "realidad", "ilusión", "creación", "destrucción", "construcción", "destrucción", "construcción", "sueño", "pesadilla", "regalo", "castigo", "alegría", "tristeza", "felicidad", "infelicidad", "placer", "dolor", "amor", "odio", "paz", "guerra", "fuerza", "debilidad", "riqueza", "pobreza", "verdad", "mentira", "esperanza", "desesperanza", "vida", "muerte", "éxito", "fracaso", "crecimiento", "decadencia", "bien", "mal", "claridad", "oscuridad", "confianza", "duda", "compasión", "indiferencia", "avance", "retroceso", "evolución", "involución", "determinación", "indecisión", "triunfo", "derrota", "resistencia", "rendición", "victoria", "derrota", "esperanza", "desesperanza", "belleza", "fealdad", "soledad", "compañía", "alegría", "tristeza", "calma", "tormenta", "corazón", "mente", "cuerpo", "alma", "vida", "muerte", "amor", "odio", "paz", "guerra", "fuerza", "debilidad", "riqueza", "pobreza", "verdad", "mentira", "esperanza", "desesperanza", "vida", "muerte", "éxito", "pez", "insecto", "mano", "pie", "boca", "nariz", "ojo", "oreja", "cabeza", "cuerpo", "corazón", "sangre", "piel", "hueso", "músculo"]

>>> def parrafos(num\_parrafos):

... texto = ""

... for \_ in range(num\_parrafos):

... num\_palabras = random.randint(5, 15)

... parrafo = " ".join(random.sample(palabras, num\_palabras))

... texto += parrafo + "\n\n"

... return texto

...

>>> def main():

... num\_parrafos = int(input("Ingrese la cantidad de párrafos que desea generar: "))

... texto\_generado = parrafos(num\_parrafos)

... print(texto\_generado)

...

>>> if \_\_name\_\_ == "\_\_main\_\_":

... main()

...

Ingrese la cantidad de párrafos que desea generar: 5

triunfo confianza verdad creatividad nariz fuerza crecimiento felicidad

soledad cabeza tristeza inicio derrota verdura luna

naturaleza montaña odio mesa desesperanza pesadilla castigo sueño compasión victoria muerte perro indecisión

esperanza fuerza deporte río propósito desamor ganancia pobreza odio bondad gato amistad pie desesperanza derrota

ciudad paciencia leche presente nariz tristeza verano efecto amor debilidad río ensalada futuro

>>>

**Ejercicio 31:** Definir una función que extraiga el título de una página web. Investigar sobre las librerías integradas requests y BeautifulSoup.

>>> import requests

>>> from bs4 import BeautifulSoup

>>> def extraer\_titulo(url):

... try:

... response = requests.get(url)

... if response.status\_code == 200:

... soup = BeautifulSoup(response.content, 'html.parser')

... titulo = soup.title.text.strip()

... return titulo

... else:

... print(f"Error al acceder a la página: {response.status\_code}")

... return None

... except Exception as e:

... print(f"Error durante la solicitud HTTP: {e}")

... return None

...

>>> def main():

... url = input("Ingrese la URL de la página web: ")

... titulo = extraer\_titulo(url)

... if titulo:

... print(f"El título de la página '{url}' es: '{titulo}'")

... else:

... print("No se pudo extraer el título de la página.")

...

Ingrese la URL de la página web: https://www.cnn.com

El título de la página 'https://www.cnn.com' es: 'Breaking News, Latest News and Videos | CNN'

>>>

>>> if \_\_name\_\_ == "\_\_main\_\_":

... main()

...

Ingrese la URL de la página web: https://www.google.com

El título de la página 'https://www.google.com' es: 'Google'

>>>

>>> if \_\_name\_\_ == "\_\_main\_\_":

... main()

...

Ingrese la URL de la página web: https://www.wikipedia.org

El título de la página 'https://www.wikipedia.org' es: 'Wikipedia'

>>>

>>> if \_\_name\_\_ == "\_\_main\_\_":

... main()

...

Ingrese la URL de la página web: https://www.bbc.com

El título de la página 'https://www.bbc.com' es: 'BBC Home - Breaking News, World News, US News, Sports, Business, Innovation, Climate, Culture, Travel, Video & Audio'

>>>