**UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS**

(Universidad del Perú, Decana de América)

**Facultad de Ciencias Matemáticas**

Imagen que contiene Logotipo

Descripción generada automáticamente

**Computación Científica – Estructura de Datos**

**Grupo 11 – Integrantes:**

**- Pozo Velarde, Luis Felipe**

**- Luis Enrique Huayta Huillcahuare**

17.6 Escriba un programa para concatenar dos objetos de lista enlazada de caracteres. La clase Concatenar Lista debe incluir un método llamado concatenar que tome referencias a ambos objetos lista como argumentos y que concateen la segunda lista con la primera.

***PSEUDOCÓDIGO:***

INICIO EL PROGRAMA

Clase Nodo:

Función Inicializar(dato):

nodo.dato = dato

nodo.siguiente = Nulo

Clase ListaEnlazada:

Función Inicializar():

lista.encabezado = Nulo

Función insertar\_al\_final(dato):

nuevo\_nodo = Nodo(dato)

Si lista.encabezado es Nulo:

lista.encabezado = nuevo\_nodo

Sino:

actual = lista.encabezado

Mientras actual.siguiente no sea Nulo:

actual = actual.siguiente

actual.siguiente = nuevo\_nodo

Función obtener\_cadena():

cadena = ""

actual = lista.encabezado

Mientras actual no sea Nulo:

cadena += Convertir a cadena actual.dato + " "

actual = actual.siguiente

Devolver cadena.strip()

Clase ConcatenarLista:

Función concatenar(lista1, lista2):

Si lista1.encabezado es Nulo:

lista1.encabezado = lista2.encabezado

Sino:

actual = lista1.encabezado

Mientras actual.siguiente no sea Nulo:

actual = actual.siguiente

actual.siguiente = lista2.encabezado

Función obtener\_lista():

lista = ListaEnlazada()

elementos = Obtener entrada del usuario("Ingrese los elementos de la lista separados por espacios").split()

Para cada elemento en elementos:

lista.insertar\_al\_final(elemento)

Devolver lista

Inicio del programa:

Imprimir "Ingrese los elementos de la primera lista:"

lista1 = obtener\_lista()

Imprimir "Ingrese los elementos de la segunda lista:"

lista2 = obtener\_lista()

ConcatenarListas.concatenar(lista1, lista2)

cadena\_concatenada = lista1.obtener\_cadena()

Imprimir "Cadena concatenada:", cadena\_concatenada

**PRUEBA DE ESCRITORIO:**

Ingrese los elementos de la primera lista: 1 2 3 4 5 6 7 8 9

Ingrese los elementos de la segunda lista: A B C D E F G H I

Cadena concatenada: 1 2 3 4 5 6 7 8 9 A B C D F G H I

***Código Fuente:***

class Nodo:

def \_\_init\_\_(self, dato):

self.dato = dato

self.siguiente = None

class ListaEnlazada:

def \_\_init\_\_(self):

self.encabezado = None

def insertar\_al\_final(self, dato):

nuevo\_nodo = Nodo(dato)

if not self.encabezado:

self.encabezado = nuevo\_nodo

else:

actual = self.encabezado

while actual.siguiente:

actual = actual.siguiente

actual.siguiente = nuevo\_nodo

def obtener\_cadena(self):

cadena = ""

actual = self.encabezado

while actual:

cadena += str(actual.dato) + " "

actual = actual.siguiente

return cadena.strip()

class ConcatenarLista:

def concatenar(lista1, lista2):

if not lista1.encabezado:

lista1.encabezado = lista2.encabezado

else:

actual = lista1.encabezado

while actual.siguiente:

actual = actual.siguiente

actual.siguiente = lista2.encabezado

def obtener\_lista():

lista = ListaEnlazada()

elementos = input("Ingrese los elementos de la lista separados por espacios: ").split()

for elemento in elementos:

lista.insertar\_al\_final(elemento)

return lista

# Obtener listas enlazadas del usuario

print("Ingrese los elementos de la primera lista:")

lista1 = obtener\_lista()

print("Ingrese los elementos de la segunda lista:")

lista2 = obtener\_lista()

# Concatenar listas

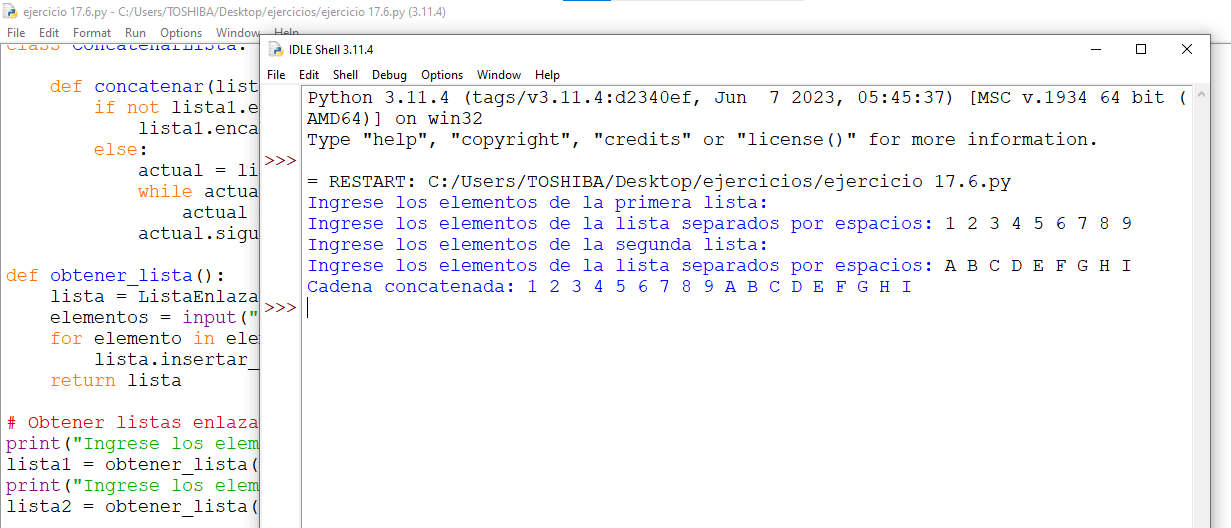
ConcatenarLista.concatenar(lista1, lista2)

# Obtener cadena resultante

cadena\_concatenada = lista1.obtener\_cadena()

print("Cadena concatenada:", cadena\_concatenada)

**Resultado:**



17.7 Escriba un programa para fusionar dos objetos de lista ordenada de enteros en un solo objeto de lista ordenada de enteros. El método fusionar de la clase FusionarLista debe recibir referencias a cada uno de los objetos lista que se van a fusionar, y debe devolver una referencia al objeto lista fusionado.

PSEUDOCODIGO:

Clase Nodo:

Atributos:

- valor

- siguiente

Constructor(valor, siguiente):

self.valor = valor

self.siguiente = siguiente

Función fusionarListas(lista1, lista2):

cabeza = Nodo() # Nodo ficticio para la cabeza

actual = cabeza

Mientras lista1 no sea nulo y lista2 no sea nulo:

Si lista1.valor < lista2.valor:

actual.siguiente = lista1

lista1 = lista1.siguiente

Sino:

actual.siguiente = lista2

lista2 = lista2.siguiente

actual = actual.siguiente

# Agregar los elementos restantes de ambas listas, si los hay

Si lista1 no es nulo:

actual.siguiente = lista1

Sino si lista2 no es nulo:

actual.siguiente = lista2

Devolver cabeza.siguiente

Función crearListaDesdeInput():

Escribir "Ingrese los valores separados por espacios:"

Leer valores

valores = Convertir valores a lista de enteros

nodo = nulo

Para cada valor en valores al revés:

nodo = Nodo(valor, nodo)

Devolver nodo

Procedimiento imprimirListaEnlazada(nodo):

Mientras nodo no sea nulo:

Imprimir nodo.valor + " "

nodo = nodo.siguiente

Imprimir nueva línea

# Ingresar las listas desde el usuario

Escribir "Ingrese la primera lista:"

lista1 = crearListaDesdeInput()

Escribir "Ingrese la segunda lista:"

lista2 = crearListaDesdeInput()

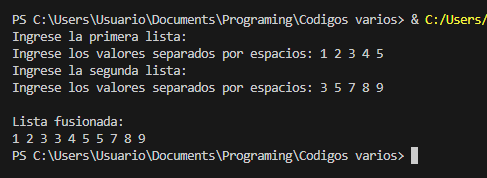
# Fusionar las listas y mostrar el resultado

listaFusionada = fusionarListas(lista1, lista2)

Escribir "\nLista fusionada:"

imprimirListaEnlazada(listaFusionada)

RESULTADO:



CÓDIGO FUENTE:

class Nodo:

    def \_\_init\_\_(self, valor=None, siguiente=None):

        self.valor = valor

        self.siguiente = siguiente

def fusionar\_listas(lista1, lista2):

    cabeza = Nodo()  # Nodo ficticio para la cabeza

    actual = cabeza

    while lista1 is not None and lista2 is not None:

        if lista1.valor < lista2.valor:

            actual.siguiente = lista1

            lista1 = lista1.siguiente

        else:

            actual.siguiente = lista2

            lista2 = lista2.siguiente

        actual = actual.siguiente

    # Agregar los elementos restantes de ambas listas, si los hay

    if lista1 is not None:

        actual.siguiente = lista1

    elif lista2 is not None:

        actual.siguiente = lista2

    return cabeza.siguiente

def crear\_lista\_desde\_input():

    valores = input("Ingrese los valores separados por espacios: ")

    valores = [int(valor) for valor in valores.split()]

    nodo = None

    for valor in reversed(valores):

        nodo = Nodo(valor, nodo)

    return nodo

def imprimir\_lista\_enlazada(nodo):

    while nodo is not None:

        print(nodo.valor, end=" ")

        nodo = nodo.siguiente

    print()

# Ingresar las listas desde el usuario

print("Ingrese la primera lista:")

lista1 = crear\_lista\_desde\_input()

print("Ingrese la segunda lista:")

lista2 = crear\_lista\_desde\_input()

# Fusionar las listas y mostrar el resultado

lista\_fusionada = fusionar\_listas(lista1, lista2)

print("\nLista fusionada:")

imprimir\_lista\_enlazada(lista\_fusionada)

17.8 Escriba un programa para insertar 25 enteros aleatorios de 0 a 100 en orden, en un objeto lista enlazada. El programa deberá calcular la suma de los elementos y el promedio de punto flotante de los elementos.

***PSEUDOCÓDIGO:***

Inicio

Clase Nodo

Función Inicializar(valor):

nodo.valor = valor

nodo.siguiente = Nulo

Clase ListaEnlazada

Función Inicializar():

lista.encabeasdo = Nulo

Función insertar\_ordenado(valor):

nuevo\_nodo = Nodo(valor)

Si lista.encabezado es Nulo:

lista. encabezado = nuevo\_nodo

Retornar

Si valor <= lista. encabezado.valor:

nuevo\_nodo.siguiente = lista.cabeza

lista.cabeza = nuevo\_nodo

Retornar

actual = lista. encabezado

Mientras actual.siguiente y actual.siguiente.valor < valor:

actual = actual.siguiente

nuevo\_nodo.siguiente = actual.siguiente

actual.siguiente = nuevo\_nodo

Función imprimir():

actual = lista.cabeza

Mientras actual sea diferente de Nulo:

Imprimir actual.valor

actual = actual.siguiente

Imprimir "terminado"

Función sumar\_elementos():

suma = 0

actual = lista. encabezado

Mientras actual sea diferente de Nulo:

suma = suma + actual.valor

actual = actual.siguiente

Retornar suma

Función promedio\_elementos():

cantidad = 0

suma = 0

actual = lista. encabezado

Mientras actual sea diferente de Nulo:

suma = suma + actual.valor

cantidad = cantidad + 1

actual = actual.siguiente

Si cantidad es igual a 0:

Retornar 0

Retornar suma / cantidad

Crear lista

Para cada x en el rango de 25:

lista.insertar\_ordenado(generar\_numero\_aleatorio())

Imprimir "Lista enlazada:"

lista.imprimir()

suma = lista.sumar\_elementos()

Imprimir "Suma de los elementos:", suma

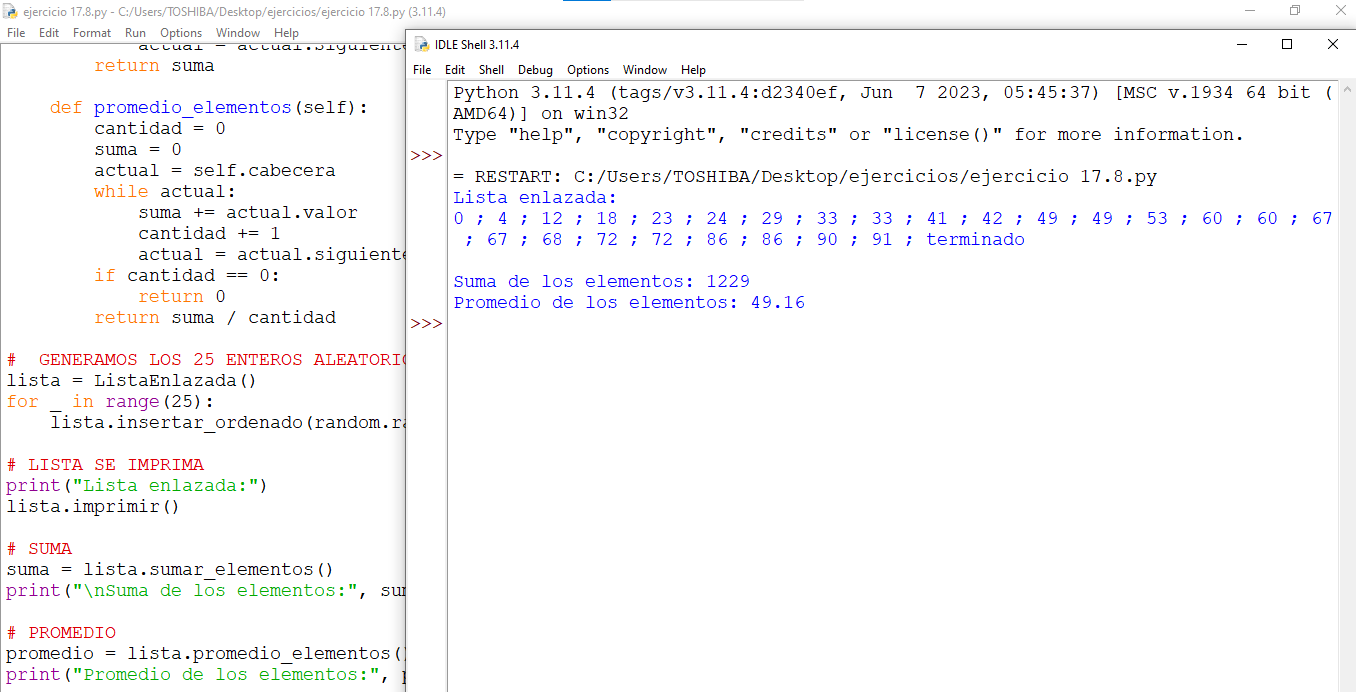
promedio = lista.promedio\_elementos()

Imprimir "Promedio de los elementos:", promedio

Fin

**PRUEBA DE ESCRITORIO:**

1. **Inicialización**:
   * Se crea una lista enlazada vacía.
2. **Inserción de elementos**:
   * Se generan 25 números enteros aleatorios en el rango de 0 a 100.
   * Cada número se inserta en la lista enlazada de manera ordenada.
3. **Impresión de la lista**:
   * Se imprime la lista enlazada, mostrando todos los elementos en orden.
4. **Cálculo de la suma de los elementos**:
   * Se recorren todos los elementos de la lista y se suman sus valores.
5. **Cálculo del promedio de los elementos**:
   * Se recorren todos los elementos de la lista y se suman sus valores, al mismo tiempo que se cuenta cuántos elementos hay.
   * Se calcula el promedio dividiendo la suma total entre el número de elementos.
6. **Impresión de la suma y el promedio**:
   * Se muestra en la salida la suma de los elementos y su promedio.

**Resultado:** 

***Código Fuente:***

import random

class Nodo:

def \_\_init\_\_(self, valor):

self.valor = valor

self.siguiente = None

class ListaEnlazada:

def \_\_init\_\_(self):

self.cabecera = None

def insertar\_ordenado(self, valor):

nuevo\_nodo = Nodo(valor)

if self.cabecera == None:

self.cabecera = nuevo\_nodo

return

if valor <= self.cabecera.valor:

nuevo\_nodo.siguiente = self.cabecera

self.cabecera = nuevo\_nodo

return

actual = self.cabecera

while actual.siguiente and actual.siguiente.valor < valor:

actual = actual.siguiente

nuevo\_nodo.siguiente = actual.siguiente

actual.siguiente = nuevo\_nodo

def imprimir(self):

actual = self.cabecera

while actual:

print(actual.valor, end=' ; ')

actual = actual.siguiente

print("terminado")

def sumar\_elementos(self):

suma = 0

actual = self.cabecera

while actual:

suma += actual.valor

actual = actual.siguiente

return suma

def promedio\_elementos(self):

cantidad = 0

suma = 0

actual = self.cabecera

while actual:

suma += actual.valor

cantidad += 1

actual = actual.siguiente

if cantidad == 0:

return 0

return suma / cantidad

# GENERAMOS LOS 25 ENTEROS ALEATORIOS

lista = ListaEnlazada()

for \_ in range(25):

lista.insertar\_ordenado(random.randint(0, 100))

# LISTA SE IMPRIMA

print("Lista enlazada:")

lista.imprimir()

# SUMA

suma = lista.sumar\_elementos()

print("\nSuma de los elementos:", suma)

# PROMEDIO

promedio = lista.promedio\_elementos()

print("Promedio de los elementos:", promedio)

17.9 Escriba un programa para crear un objeto lista enlazada de 10 caracteres, y que luego cree un segundo objeto lista que contenga una copia de la primera lista, pero en orden inverso.

PSEUDOCODIGO:

Clase Nodo:

Atributos:

- valor

- siguiente

Constructor(valor, siguiente):

self.valor = valor

self.siguiente = siguiente

Función crearListaEnlazada(valores):

cabeza = Nodo() # Nodo ficticio para la cabeza

actual = cabeza

Para cada valor en valores:

actual.siguiente = Nodo(valor)

actual = actual.siguiente

Devolver cabeza.siguiente

Función imprimirListaEnlazada(nodo):

Mientras nodo no sea nulo:

Imprimir nodo.valor + " "

nodo = nodo.siguiente

Imprimir nueva línea

Función invertirLista(lista):

cabezaInvertida = nulo

Mientras lista no sea nulo:

nuevoNodo = Nodo(lista.valor, cabezaInvertida)

cabezaInvertida = nuevoNodo

lista = lista.siguiente

Devolver cabezaInvertida

Procedimiento principal:

Escribir "Ingrese 10 caracteres separados por espacios:"

Leer caracteresOriginales

caracteresOriginales = caracteresOriginales.split()[:10]

listaOriginal = crearListaEnlazada(caracteresOriginales)

Escribir "\nLista original:"

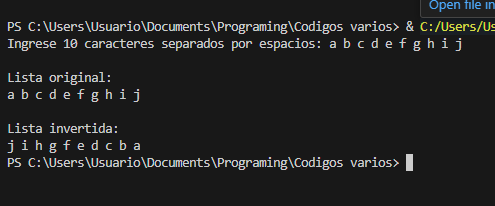
imprimirListaEnlazada(listaOriginal)

listaInvertida = invertirLista(listaOriginal)

Escribir "\nLista invertida:"

imprimirListaEnlazada(listaInvertida)

RESULTADO:



CÓDIGO FUENTE:

class Nodo:

    def \_\_init\_\_(self, valor=None, siguiente=None):

        self.valor = valor

        self.siguiente = siguiente

def crear\_lista\_enlazada(valores):

    cabeza = Nodo()  # Nodo ficticio para la cabeza

    actual = cabeza

    for valor in valores:

        actual.siguiente = Nodo(valor)

        actual = actual.siguiente

    return cabeza.siguiente

def imprimir\_lista\_enlazada(nodo):

    while nodo is not None:

        print(nodo.valor, end=" ")

        nodo = nodo.siguiente

    print()

def invertir\_lista(lista):

    cabeza\_invertida = None

    while lista is not None:

        nuevo\_nodo = Nodo(lista.valor, cabeza\_invertida)

        cabeza\_invertida = nuevo\_nodo

        lista = lista.siguiente

    return cabeza\_invertida

# Ingresar caracteres para la lista original desde el usuario

caracteres\_originales = input("Ingrese 10 caracteres separados por espacios: ").split()[:10]

lista\_original = crear\_lista\_enlazada(caracteres\_originales)

print("\nLista original:")

imprimir\_lista\_enlazada(lista\_original)

# Invertir la lista y crear una nueva lista enlazada

lista\_invertida = invertir\_lista(lista\_original)

print("\nLista invertida:")

imprimir\_lista\_enlazada(lista\_invertida)

17.10 Escriba un programa que reciba una línea de texto como entrada y que utilice un objeto pila para imprimir las palabras de la línea en orden inverso.

***PSEUDOCÓDIGO:***

Inicio

Función imprimir\_en\_orden\_inverso(texto):

palabras = dividir\_texto\_en\_palabras(texto)

pila = crear\_pila()

// Apilar las palabras en la pila

Para cada palabra en palabras:

apilar(pila, palabra)

// Desapilar y mostrar las palabras en orden inverso

Mientras pila no esté vacía:

palabra = desapilar(pila)

imprimir palabra

Función dividir\_texto\_en\_palabras(texto):

retorno texto.split()

Función crear\_pila():

retorno []

Función apilar(pila, elemento):

agregar elemento a pila

Función desapilar(pila):

retorno pila.pop()

texto = leer\_input("Ingrese una línea de texto: ")

imprimir\_en\_orden\_inverso(texto)

Fin

**PRUEBA DE ESCRITORIO:**

1. **División del texto en palabras**:
   * Utilizando **split()**, dividimos la línea de texto en palabras:

palabras = ["Super", "Patrulla", "Delta", "SPD "]

1. **Inicialización de la "pila"**:
   * Creamos una lista vacía llamada **pila**.
2. **Apilado de las palabras en la "pila"**:

Pila = ["Super", "Patrulla", "Delta", "SPD "]

1. **Impresión del mensaje**:
   * Imprimimos un mensaje indicando que estamos mostrando las palabras en orden inverso.
2. **Muestra de las palabras en orden inverso**:

“SPD Delta Patrulla Super “Principio del formulario

***Código Fuente:***

def imprimir\_en\_orden\_inverso(texto):

# divide el texto

palabras = texto.split()

# Inicializa una lista que actuará como pila

pila = []

# Apila cada palabra en la "pila"

for palabra in palabras:

pila.append(palabra)

# Desapila y muestra cada palabra en orden inverso

print("Palabras en orden inverso:")

while pila:

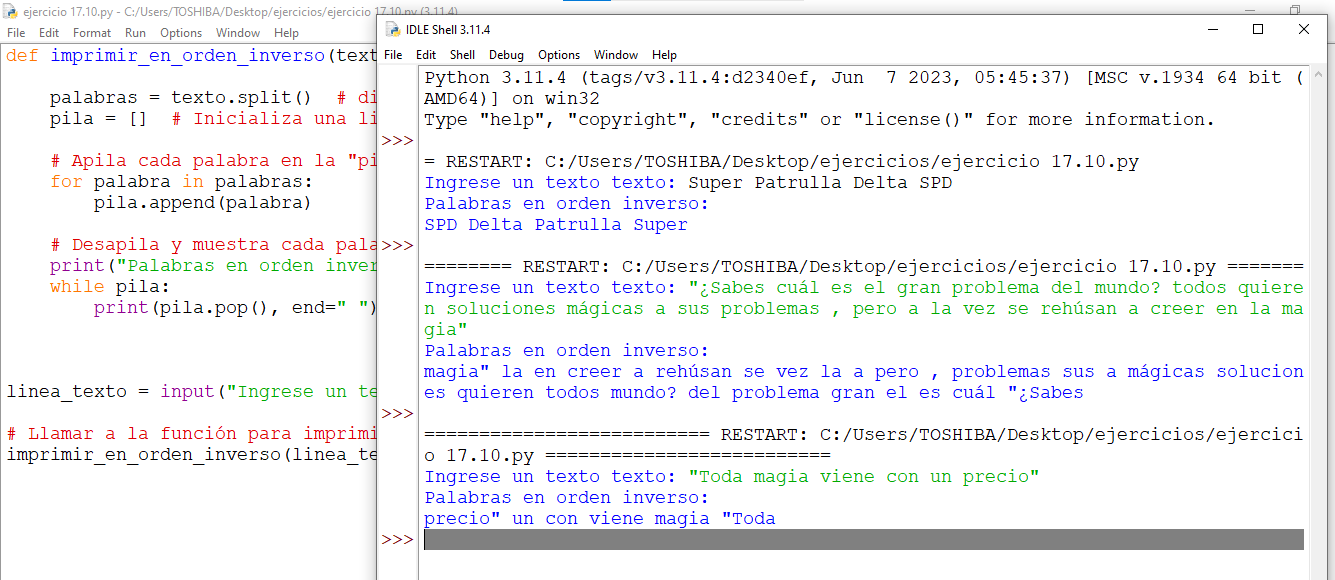
print(pila.pop(), end=" ")

linea\_texto = input("Ingrese un texto texto: ")

# Llamar a la función para imprimir en inverso

imprimir\_en\_orden\_inverso(linea\_texto)

**Resultado:**

******

17.11 Escriba un programa que utilice una pila para determinar si una cadena es un palíndromo (es decir, que la cadena se deletree en forma idéntica, tanto al revés como al derecho). El programa debe ignorar espacios y puntuación.

PSEUDOCODIGO:

Función limpiarCadena(cadena):

cadenaLimpia = ""

Para cada caracter en cadena:

Si el caracter es alfanumérico:

Agregar el caracter convertido a minúsculas a cadenaLimpia

Devolver cadenaLimpia

Función esPalindromo(cadena):

cadenaLimpia = limpiarCadena(cadena)

pila = Pila()

Para cada caracter en cadenaLimpia:

Apilar el caracter en la pila

cadenaInvertida = ""

Mientras la pila no esté vacía:

Concatenar el caracter desapilado a cadenaInvertida

Devolver cadenaLimpia igual a cadenaInvertida

Procedimiento Principal:

Escribir "Ingrese una cadena para verificar si es un palíndromo:"

Leer cadenaUsuario

resultado = esPalindromo(cadenaUsuario)

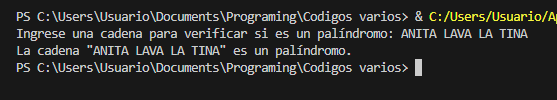
Si resultado es Verdadero:

Escribir "La cadena '", cadenaUsuario, "' es un palíndromo."

Sino:

Escribir "La cadena '", cadenaUsuario, "' no es un palíndromo."

RESULTADO:



CÓDIGO FUENTE:

def es\_palindromo(cadena):

    # Función para limpiar la cadena de espacios y puntuación

    def limpiar\_cadena(cadena):

        return ''.join(caracter.lower() for caracter in cadena if caracter.isalnum())

    cadena\_limpiada = limpiar\_cadena(cadena)

    pila = []

    for caracter in cadena\_limpiada:

        pila.append(caracter)

    cadena\_invertida = ''.join(pila[::-1])

    return cadena\_limpiada == cadena\_invertida

# Solicitar al usuario que ingrese la cadena

cadena\_usuario = input("Ingrese una cadena para verificar si es un palíndromo: ")

resultado = es\_palindromo(cadena\_usuario)

if resultado:

    print(f'La cadena "{cadena\_usuario}" es un palíndromo.')

else:

    print(f'La cadena "{cadena\_usuario}" no es un palíndromo.')