ESTRUCTURA DE DATOS Y PROGRAMACIÓN ORIENTADA A OBJETOS

Tema Nº3:Cadenas y métodos de lista

Indicador de logro Nº3:Utiliza los diferentes métodos de cuerda a través de un lenguaje de programación.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº3:**

Excepciones

**Subtema 3.1:**

Caracteres y cadenas.

* El primer problema a mostrar es el cifrado César.
* El segundo es la transformación inversa ahora debe ser clara.
* El tercer programa muestra un método simple que le permite ingresar una línea llena de números y procesarlos fácilmente.
* El cuarto programa implementa (en una forma ligeramente simplificada) un algoritmo utilizado por los bancos europeos para especificar los números de cuenta. El estándar denominado IBAN (Número de cuenta bancaria internacional).
* **CIFRADO CÉSAR**

Este código fue (probablemente) inventado y utilizado por Gaius Julius Caesar y sus tropas durante las Guerras Gálicas. La idea es bastante simple: cada letra del mensaje es reemplazada por su consecuente más cercano (A se convierte en B, B se convierte en C, etc.). La única excepción es Z, que se convierte en A.

Este programa es una implementación muy simple (pero funcional) del algoritmo. Lo hemos escrito usando los siguientes supuestos:

* Solo acepta letras latinas (nota: los romanos no utilizaron espacios en blanco ni dígitos).
* Todas las letras del mensaje están en mayúsculas (nota: los romanos solo sabían mayúsculas).

PROGRAMA:

text = input("INGRESAR MENSAJE: ")

cipher = ""

for char in text:

if not char.isalpha():

continue

char = char.upper()

code = ord(char) + 1

if code > ord("Z"):

code = ord("A")

cipher += chr(code)

print(cipher)

INTERPRETACIÓN:

* **Línea 01**: solicite al usuario que ingrese el mensaje de una línea abierto (sin cifrar);
* **Línea 02**: preparar una cadena para un mensaje cifrado (vacío por ahora);
* **Línea 03**: comienza la iteración a través del mensaje;
* **Línea 04**: si el carácter actual no es alfabético . . .
* **Línea 05**: . . . ignóralo;
* **Línea 06**: convierte la letra a mayúsculas (es preferible hacerlo a ciegas, en lugar de verificar si es necesario o no);
* **Línea 07**: obtener el código de la letra e incrementarlo en uno;
* **Línea 08**: si el código resultante ha "dejado" el alfabeto latino (si es mayor que el código Z). . .
* **Línea 09**: . . . cambiarlo al código A;
* **Línea 10**: añadir el carácter recibido al final del mensaje cifrado;
* **Línea 11**: imprimir el cifrado.

LA SALIDA SERÁ:

INGRESAR MENSAJE: IDAT

JEBU

**LA TRANSFORMACIÓN INVERSA**

Usa el criptograma del programa anterior y devuelve las letras que están antes de las ingresadas.

PROGRAMA:

cipher = input("Entrar criptograma: ")

text = ""

for char in cipher:

if not char.isalpha():

continue

char = char.upper()

code = ord(char) - 1

if code < ord("A"):

code = ord("Z")

text += chr(code)

print(text)

LA SALIDA SERÁ:

Entrar criptograma: HOLA

GNKZ

* **EXTRAER NÚMEROS DE UNA LÍNEA DE TEXTO**

El tercer programa muestra un método simple que le permite ingresar una línea llena de números y procesarlos fácilmente.

Nota: la función de entrada de **rutina()**, combinada con las funciones **int()** o **float()**, no es adecuada para este propósito.

El procesamiento será extremadamente fácil: queremos que los números se sumen.

line = input("Introduzca línea llena de números - sepárelos con espacios: ")

strings = line.split()

sum = 0

try:

for substr in strings:

sum += float(substr)

print("Sum = ", sum)

except:

print("No es un número: ", substr)

INTERPRETACIÓN:

* **Línea 01**: pida al usuario que ingrese una línea llena con cualquier número de números (los números pueden ser flotantes)
* **Línea 02**: divide la línea que recibe una lista de subcadenas;
* **Línea 03**: iniciar la suma total a cero;
* **Línea 04**: como la conversión de cadena → flotante puede generar una excepción, es mejor continuar con la protección del bloque try-except;
* **Línea 05**: iterar a través de la lista. . .
* **Línea 06**: . . . y tratar de convertir todos sus elementos en números flotantes; si funciona, aumentar la suma;
* **Línea 07**: todo está bien hasta ahora, así que imprime la suma;
* **Línea 08**: el programa termina aquí en caso de error;
* **Línea 09**: imprima un mensaje de diagnóstico que muestre al usuario el motivo del error.

LA SALIDA SERÁ:

Introduzca línea llena de números - sepárelos con espacios: 4 5 6 7 8 9

Sum = 39.0

* **COMPROBANDO EL IBAN**

El cuarto programa implementa (en una forma ligeramente simplificada) un algoritmo utilizado por los bancos europeos para especificar los números de cuenta. Proporciona un método simple y bastante confiable para validar los números de cuenta contra los errores tipográficos simples que pueden ocurrir durante la reescritura del número, por ejemplo, desde documentos en papel, como facturas o facturas, hasta computadoras.

iban = input("Entrar IBAN, por favor: ")

iban = iban.replace(' ','')

if not iban.isalnum():

print("Caracteres inválidos dentro de IBAN - lo siento!")

elif len(iban) < 15:

print("IBAN demasiado corto")

elif len(iban) > 31:

print("IBAN demasiado largo")

else:

iban = (iban[4:] + iban[0:4]).upper()

iban2 = ""

for ch in iban:

if ch.isdigit():

iban2 += ch

else:

iban2 += str(10 + ord(ch) - ord("A"))

ibann = int(iban2)

if ibann % 97 == 1:

print("¡Parece legítimo!")

else:

print("No creo que sea un IBAN válido, lo siento")

LA SALIDA SERÁ:

Entrar IBAN, por favor: GRUPO UTP

IBAN demasiado corto

**Actividad:**

**CASO PRÁCTICO 1:**

Introducir un texto e indicar cuantas palabras tiene y cuáles son:

SOLUCIÓN:

# Introducir un texto

Texto = input("Texto a introducir: ")

# Convertir un texto a una lista de palabras

Palabra\_cadenas = Texto.split()

# Visualizar para las palabras del texto introducido

print("Listado de palabras")

for i, Listado\_palabra in enumerate(Palabra\_cadenas):

print("La palabra: ", Listado\_palabra)

# Visualizar el número de palabras introducidas

print("El texto tiene: ", i + 1, "palabras")

LA SALIDA ES:

Texto a introducir: DESARROLLO DE SISTEMAS

Listado de palabras

La palabra: DESARROLLO

La palabra: DE

La palabra: SISTEMAS

El texto tiene: 3 palabras

**CASO PRÁCTICO 2:**

Introducir un texto, y visualizarlo al revés e indicar si es un palíndromo.

SOLUCIÓN:

# Introducir un texto

texto1 = input("Dame una cadena: ")

# Crear la palabra al revés

texto1\_al\_reves = texto1[::-1]

# Visualizar la palabra al revés

print(texto1\_al\_reves)

# Verificar si la palabra introducida es igual que al revés

if (texto1 == texto1\_al\_reves):

print("Es palíndromo")

else:

print("No es palíndromo")

SU SALIDA ES:

Dame una cadena: AMA

AMA

Es palíndromo

**CASO PRÁCTICO 3:**

Un grupo de amigos desea codificar los mensajes secretos de tal forma que no puedan ser interpretados con una lectura directa, para lo cual se han establecido las siguientes reglas:

* Todo mensaje debe estar sus letras en mayúsculas.
* El mensaje solamente contiene letras, números y/o espacios.
* Reemplazar cada letra por la que sigue según abecedario, excepto Z que se deberá reemplazar con la letra A.
* Reemplazar cada dígito encontrado por el siguiente número excepto el 9 que deberá ser reemplazado por el 0.
* Los espacios en blanco se mantienen.
* Debe mostrar el mensaje codificado

SOLUCIÓN:

# Introducir texto a codificar, debe contener letras, números o espacios

nuevo\_codigo= input("Texto a codificar: ")

ve = list(nuevo\_codigo.upper())

# Inicializar las variables

Num\_letras = len(ve)

# Calcula la longitud del texto

letra = ""

valor\_inicial = 0

cantidad\_validos = 0

# Verificar caracter a caracter si cumple con las condiciones establecidas

while valor\_inicial < Num\_letras:

# Va verificando letra a letra

letra = ve[valor\_inicial]

valor\_inicial += 1

if letra.isspace() == True:

# Saber si el caracter es un espacio

cantidad\_validos = cantidad\_validos + 1

# Saber si hay letras o números

if letra.isdigit() == True or letra.isupper() == True:

cantidad\_validos =cantidad\_validos + 1

if cantidad\_validos == Num\_letras:

# Cumple con los requisitos exigidos

print("Eltexto introducido fue correcto")

nuevo\_codigo = []

# Crea el código oculto

for i in ve:

if ord(i) == 90:

nuevo\_codigo.append("A")

else:

if ord(i) == 57:

nuevo\_codigo.append("0")

else:

if i == "":

nuevo\_codigo.append("")

else:

nuevo\_codigo.append(chr(ord(i) + 1))

# Imprime el nuevo código generado

print("El código correcto es: ", ''.join(nuevo\_codigo))

else:

# Uno o más requisitos letras, números y no alfanuméricos no se cumple

print("El texto tecleado no es correcto porque tiene caracteres distintos de espacios, letras y números")

**SU SALIDA ES:**

Texto a codificar: t 9 ze

El texto introducido fue correcto

El código correcto es: U 0 AF