Cristina Cumplido Huertas
Estructuras de datos y de control

Sprint 7 Data Analytics

Resolverás algunos problemas de la vida cotidiana aplicando las estructuras de datos y control en Python.

Un cliente de la empresa en la cual trabajas pide una lista de programas muy sencillos que le facilitarían muchos procesos. Sin embargo, el departamento TU está muy complicado con los tiempos, por lo cual te piden que hagas la programación.

Cristina Cumplido Huertas
Estructuras de datos y de control

Nivel 1

- Ejercicio 1

Calculadora del índice de masa corporal.

Escribe una función que calcule el IMC ingresado por el usuario/a, es decir, quién lo ejecute tendrá que ingresar estos datos. Puedes obtener más información de su cálculo en:
La función tiene que clasificar el resultado en sus respectivas categorías.

Al iniciar el ejercicio se nos revela información importante para proceder con los cálculos y categorías: "Para calcular el IMC es necesario dividir el peso en kg de un paciente entre el cuadrado de su altura en metros. Con esto obtendremos un valor que se situará en uno de los siguientes rangos:

- Bajo peso: menos de 18,5.
- Peso normal: 18,5-24,9.
- Sobrepeso: 25-29,9.
- Obesidad: más de 30."

En primer lugar, definimos la función calcular_imc con los parámetros peso y altura: **def calcular_imc(peso, altura):**

Luego, calculamos el IMC (Índice de Masa Corporal) utilizando la fórmula mencionada anteriormente: **imc = peso / (altura ** 2)**

A continuación, indicamos que se debe imprimir el resultado del IMC y clasificar el valor según las categorías de "bajo peso", "peso normal", "sobrepeso" u "obesidad", utilizando condicionales. Por ejemplo, si el IMC es menor a 18.5, se imprimirá "Bajo peso"; si esto no se cumple, pasará a la siguiente condición. De esta manera, el programa evaluará las diferentes categorías:

```
print("Tu IMC es:", imc)

if imc < 18.5:
    print("Clasificación: Bajo peso")
elif 18.5 <= imc < 24.9:
    print("Clasificación: Peso normal")
elif 25 <= imc < 29.9:
    print("Clasificación: Sobrepeso")
else:
    print("Clasificación: Obesidad")
```

Más tarde, fuera de la función, utilizamos input() para pedir a los usuarios su peso y altura. También empleamos el método .replace(',' , '.') para convertir comas en puntos decimales en caso de que los usuarios introduzcan el número con comas. Es importante convertir el texto devuelto por input() a un valor de tipo decimal con float().

peso = input("Indica tu peso en kg: ").replace(',', '.')
peso = float(peso)

altura = input("Indica tu altura en metros: ").replace(',', '.')
altura = float(altura)

Finalmente, hacemos un llamado a la función calcular_imc(peso, altura) para ejecutar el cálculo con los valores proporcionados por los usuarios: **calcular_imc(peso, altura)**

Código:

```
def calcular_imc(peso, altura):
    imc = peso / (altura ** 2)
    print("Tu IMC es:", imc)

    if imc < 18.5:
        print("Clasificación: Bajo peso")
    elif 18.5 <= imc < 24.9:
        print("Clasificación: Peso normal")
    elif 25 <= imc < 29.9:
        print("Clasificación: Sobrepeso")
    else:
        print("Clasificación: Obesidad")

peso = input("Indica tu peso en kg: ").replace(',', '.')
peso = float(peso)

altura = input("Indica tu altura en metros: ").replace(',', '.')
altura = float(altura)

calcular_imc(peso, altura)
```

✓ 7.7s

```
Tu IMC es: 22.229061933586404
Clasificación: Peso normal
```

Cristina Cumplido Huertas
Estructuras de datos y de control

- Ejercicio 2

Convertidor de temperaturas.

Existen varias unidades de temperatura utilizadas en diferentes contextos y regiones. Las más comunes son Celsius (°C), Fahrenheit (°F) y Kelvin (K). También existen otras unidades como Rankine (°Ra) y Réaumur (°Re). Selecciona al menos 2 conversores, de tal manera que al introducir una temperatura devuelva, como mínimo, dos conversiones.

En primer lugar, definimos la función convertir_temperatura que toma un único parámetro (celsius), que representa la temperatura en grados Celsius introducida por los usuarios:
def convertir_temperatura(celsius)

Dentro de la función, realizamos dos conversiones de temperaturas:

1. De Celsius a Fahrenheit: **fahrenheit = (celsius * 9/5) + 32**

2. De Celsius a Kelvin: **kelvin = celsius + 273.15**

Después, imprimimos el valor que han introducido los usuarios junto a las conversiones correspondientes:
print("Has introducido una temperatura en Celsius de:", celsius)
print("La temperatura en Fahrenheit es:", fahrenheit)
print("La temperatura en Kelvin es:", kelvin)

Antes de llamar a la función, pedimos a los usuarios que introduzcan la temperatura en grados Celsius con un input(). Y con el método .replace si al introducir la temperatura los usuarios han utilizado una coma se convierte directamente en un punto:
celsius = input("Introduce temperatura en Celsius: ").replace(',', '.')

Luego, convertimos el input, que es un texto (str), en un número de tipo decimal (float):
celsius = float(celsius)

Finalmente, llamamos a la función **convertir_temperatura(celsius)** y le pasamos el valor de temperatura que los usuarios han introducido.

Cristina Cumplido Huertas
Estructuras de datos y de control

Código:

```
def convertir_temperatura(celsius):
    fahrenheit = (celsius * 9/5) + 32
    kelvin = celsius + 273.15

    print("Has introducido una temperatura en Celsius de:", celsius)
    print("La temperatura en Fahrenheit es:", fahrenheit)
    print("La temperatura en Kelvin es:", kelvin)

celsius = input("Introduce temperatura en Celsius: ").replace(',','.')
celsius = float(celsius)

convertir_temperatura(celsius)
```

✓ 3.8s

```
Has introducido una temperatura en Celsius de: 56.9
La temperatura en Fahrenheit es: 134.42000000000002
La temperatura en Kelvin es: 330.04999999999995
```

Cristina Cumplido Huertas
Estructuras de datos y de control

- Ejercicio 3

Contador de palabras de un texto.

Escribe una función que, dado un texto muestre las veces que aparece cada palabra.

En primer lugar, definimos la a función **def contar_palabras_en_texto()** que se encargará de contar las palabras en el texto introducido por los usuarios.

Después se solicita a los usuarios con un input() a introducir un texto:
texto = input("Introduce un texto: ")

Luego, para que no haya inconvenientes convertimos el texto en minúsculas con el método .lower(). También eliminamos todos los signos de puntuación del texto introducido por los usuarios.
texto = texto.lower()
texto = texto.translate(str.maketrans("", "", string.punctuation))

Al inicio del código, fuera de la función, se ha de importar string: **Import string.**
Lo estoy explicando en este punto ya que esta importación es necesaria para acceder a la lista de caracteres de puntuación, sin ello no podríamos acceder a esta lista y no podríamos eliminar la puntuación de manera tan sencilla.

Más tarde, para poder dividir el texto en una lista de palabras, utilizando los espacios como separadores usaremos split(): **palabras = texto.split()**

A continuación damos orden a la consola de que se imprima el texto en pantalla:
print("Has introducido el siguiente texto:", texto)

Para poder contar y mostrar la frecuencia de cada palabra utilizaremos un bucle for para recorrer cada palabra única en el conjunto de palabras. Usamos set para eliminar las palabras duplicadas y trabajar solo con palabras únicas. Así, podemos contar cuántas veces aparece cada palabra sin tener que contarla varias veces. Dentro del bucle, se imprime cuántas veces aparece cada palabra utilizando count(), que cuenta la frecuencia de cada palabra en la lista original.

```
for palabra in set(palabras):
    print("'" + palabra + "' aparece " + str(palabras.count(palabra)) + " vez/veces.")
```

Por último, hacemos un llamado a la función y así iniciamos el proceso de contar palabras cuando se ejecuta el programa: **contar_palabras_en_texto()**

Código:

```
import string

def contar_palabras_en_texto():
    texto = input("Introduce un texto: ")
    texto = texto.lower()
    texto = texto.translate(str.maketrans("", "", string.punctuation))
    palabras = texto.split()
    print("Has introducido el siguiente texto:", texto)

    for palabra in set(palabras):
        print("'" + palabra + "' aparece " + str(palabras.count(palabra)) + " vez/veces.")

contar_palabras_en_texto()
```

✓ 22.0s

```
Has introducido el siguiente texto: la casa de la esquina es una casa triste y solitaria
'triste' aparece 1 vez/veces.
'casa' aparece 2 vez/veces.
'una' aparece 1 vez/veces.
'la' aparece 2 vez/veces.
'esquina' aparece 1 vez/veces.
'de' aparece 1 vez/veces.
'y' aparece 1 vez/veces.
'solitaria' aparece 1 vez/veces.
'es' aparece 1 vez/veces.
```

- Ejercicio 4

Diccionario inverso.

Resulta que el cliente tiene una encuesta muy antigua que se almacena en un diccionario y los resultados los necesita al revés, es decir, intercambiados las claves y los valores. Los valores y claves en el diccionario original son únicos; si este no es el caso, la función tendría que imprimir un mensaje de advertencia.

En primer lugar, definimos los diccionarios. Aclaro que son dos para obtener resultados diferentes.

Diccionario = {
'a': 1,
'b': 2,
'c': 3,
'd': 4,
'e': 5
}

Diccionario2 = {
'a': 1,
'b': 2,
'c': 3,
'd': 4,
'e': 2
}

Ahora, definimos la función que invertirá las claves y valores del diccionario:
def reverse_dictionary(diccionario)

Luego, creamos un nuevo diccionario vacío para almacenar el resultado, donde los valores originales se convertirán en claves y las claves originales se convertirán en valores:
diccionario_invertido = {}

A continuación recorremos el diccionario original con el método .items(), que nos devuelve una lista de pares (clave, valor) del diccionario: **for clave, valor in diccionario.items()**

Dentro del bucle, verificamos si el valor ya existe como clave en el diccionario invertido En este caso, imprimimos un mensaje de error (múltiples llaves para un valor) y terminamos la función con return:

```
if valor in diccionario_invertido:
    print("Error: multiple keys for one value")
    return
```

No obstante, si el valor no existe en el diccionario invertido, lo añadimos. De este modo, el valor del diccionario original se convierte en la clave del nuevo diccionario, y la clave del diccionario original se convierte en el valor:
diccionario_invertido[valor] = clave

Llegados a este punto debemos imprimir el diccionario invertido:
print(diccionario_invertido)

Finalmente, llamamos a la función con el primer diccionario:
reverse_dictionary(diccionario)

Cristina Cumplido Huertas
Estructuras de datos y de control

Resultados

- Con diccionario (sin duplicados): Al llamar a la función con el primer diccionario, se invertirá correctamente, tendremos un diccionario invertido {1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e'}:

Diccionario

```
# Diccionario sin duplicados
diccionario = {
    'a': 1,
    'b': 2,
    'c': 3,
    'd': 4,
    'e': 5
}


# En este caso dependiendo de si eliges el diccionario  o el diccionario2 te saldrá un resultado u otro.

def reverse_dictionary(diccionario):
    diccionario_invertido = {}
    for clave, valor in diccionario.items():
        if valor in diccionario_invertido:
            print("Error: multiple keys for one value")
            return
        diccionario_invertido[valor] = clave
    print(diccionario_invertido)

reverse_dictionary(diccionario)
```

```
{1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e'}
```

- Con diccionario2 (con duplicados): Al llamar a la función con el segundo diccionario dará el mensaje de error múltiples llaves para un valor:

Diccionario2

```python
# Diccionario 2 con duplicados

diccionario2 = {
    'a': 1,
    'b': 2,
    'c': 3,
    'd': 4,
    'e': 2
}

# En este caso dependiendo de si eliges el diccionario  o el diccionario2 te saldrá un resultado u otro.

def reverse_dictionary(diccionario2):
    diccionario_invertido = {}
    for clave, valor in diccionario2.items():
        if valor in diccionario_invertido:
            print("Error: multiple keys for one value")
            return
        diccionario_invertido[valor] = clave
    print(diccionario_invertido)

reverse_dictionary(diccionario2)
```

✓ 0.0s

```
Error: multiple keys for one value
```