Python para Análisis de Datos

Módulo 01

Estructuras de control

Estructuras de control

Las estructuras de control permiten controlar el flujo de ejecución de un programa, que de otro modo es de arriba hacia abajo. Las más básicas son el condicional y los bucles, aunque funciones y clases también son estructuras de control.

Las instrucciones que pertenecen al bloque de código de la estructura se indica con sangría. Estas estructuras se pueden combinar de diversas maneras y anidar unas dentro de otras para obtener el resultado deseado.

Condicional

Permite decidir si se ejecuta un bloque de código o no en base a una condición. Se utilizan las palabras **`if`**, **`elif`** y **`else`** en distintas combinaciones. En Python no existe la estructura switch/case, pero se puede recrear con **`if/elif.`**

```
1  if num >= 0:
2      print("Positivo")
```

```
1  if num >= 0:
2      print("Positivo")
3  else:
4      print("Negativo")
```

```
1  if num == 1:
2      print("caso 1")
3  elif num == 2:
4      print("caso 2")
5  elif num == 3:
6      print("caso 3")
7  elif num == 3:
8      print("caso 3")
9  else:
10     print("otro caso")
```

Bucle while

El ***bucle while*** es un bucle condicional que ejecuta un bloque de código siempre que la condición se evalúe a `True`.

Remarquemos que la condición puede ser cualquier expresión, cuyo resultado se va a convertir a booleano. En general, todo tipo de dato tiene un único valor que se evalúa a `False` mientras todos los demás se evalúan a `True`.

Para números ese valor es cero. Para colecciones ese valor es la colección vacía.

```
1  n = 0
2  while n<5:
3      print(n)
4      n = n + 1
```

```
0
1
2
3
4
```

```
1  # se sale cuando la lista esté vacía
2  lista = [100, 200, 300]
3  while lista:
4      print(lista.pop())
```

```
300
200
100
```

Bucle for

El ***bucle for*** permite iterar sobre cada uno de los elementos de un iterable, como una lista, por ejemplo. El bucle recorre cada elemento y lo asigna a una variable definida en el propio bucle. Así, podemos hacer uso del elemento refiriéndonos a la variable.

Resulta ser un iterable todo objeto que implemente el protocolo de iteradores de Python a través de los métodos **`__iter__`** y **`__next__`**. Por ejemplo, además de usar el bucle *for* con las colecciones de Python, lo podemos hacer también con archivos, cursores de bases de datos, arrays de Numpy, series y dataframes de Pandas, etc.

Cuando se itera sobre una secuencia, se recorre cada uno de sus elementos. Sin embargo, cuando se itera sobre un diccionario se recorre cada una de las claves.

```
1 lista = [1,2,3,4]
2 for i in lista:
3     print(i)
```

```
1
2
3
4
```

```
1 dicc = {"A": 1, "B": 2, "C": 3, "D": 4}
2 for i in dicc:
3     print(i)
```

```
A
B
C
D
```

Ejemplo: Diccionario de frecuencias

Supongamos que tenemos una lista con varios elementos y queremos saber cuántas veces aparece cada uno. Podemos usar un diccionario para llevar este registro mientras recorremos la lista.

Comenzamos con un diccionario vacío y recorremos la lista con un bucle *for.* Cada vez que nos encontremos con un elemento que no está en la lista tenemos que iniciar la correspondiente entrada con una cuenta de 1. Si el elemento ya está en el diccionario hay que sumar 1 a la respectiva cuenta.

Ejemplo: Diccionario de frecuencias

```
1 letras = ['C', 'B', 'A', 'B', 'C', 'B', 'B', 'B', 'A', 'C']
2 frecuencias = {}
3 for letra in letras:
4     if letra in frecuencias:
5         frecuencias[letra] += 1
6     else:
7         frecuencias[letra] = 1
8 print(frecuencias)
```

```
{'C': 3, 'B': 5, 'A': 2}
```

Listas por comprensión

Existe una sintaxis que permite crear listas de una forma muy concisa: **listas por comprensión.** Consiste en especificar los elementos que va a contener la lista en la propia definición de la lista.

Consiste en colocar dentro de los corchetes una expresión y una cláusula *for* (que puede o no estar seguida de más cláusulas *for* o un *if*).

Se usa para generar elementos que siguen un patrón o aplicar transformaciones a los elementos de otra secuencia.

```python
[i**2 for i in range(10)]
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

```python
palabras = ["hola", "mundo", "python"]
mayusculas = [st.upper() for st in palabras]
mayusculas
```

```
['HOLA', 'MUNDO', 'PYTHON']
```

```python
numeros = [14, 5, 4, 10, 15, 4, 7, 6, 3, 14, 9, 17]
pares = [num for num in numeros if num%2==0]
pares
```

```
[14, 4, 10, 4, 6, 14]
```

¡Muchas gracias!

¡Sigamos trabajando!