Control de Flujo

Antonio Espín Herranz

Estructuras de control: if

- Condicional: if
- Las sentencias quedan fuera o no del if por la indentación de esta.
- Los dos puntos al final de cada if / else son obligatorios y se extiende a todas las sentencias iterativas.

```
if a != 0:
    # dentro del if
else:
    # dentro del else.
Fuera del if y del else.
```

- Se puede comparar cadenas con == en un if.

Estructuras de control: if

- Puedo utilizar los operadores: and, or y not.

  if a > 5 and b < 4:

- Las condiciones se pueden realizar de la siguiente manera:

```
if 0 < dia < 16:
    print('primera quincena')
else:
    if 15 < dia < 31:
        print ('segunda quincena')
```

Evaluación de condiciones

- Las condiciones se evalúan **en corto circuito**.
- Si tenemos un and y el primero es falso ya no se evalúa la 2ª.
- Y de la misma forma si estamos en un or y el primero es True ya no se evalúa el resto.

Las condiciones anidadas

- Las condiciones múltiples anidadas del estilo:
  if / else if

- Se pueden agrupar en:
  if condición:
  elif otra condición:

- **Ejemplo**:

```
if numero < 0:
        print("Negativo")
elif numero > 0:
        print("Positivo")
else:
        print("Cero")
```

A if C else B

- También existe una construcción similar al operador **?:** de otros lenguajes (como Java o C), que no es más que una forma compacta de expresar un if else.

- En esta construcción se evalúa el predicado C y se devuelve A si se cumple o B si no se cumple: *A if C else B.*

- *Ejemplo:*
  - var = “par” if (num % 2 == 0) else “impar”

- En el caso de switch no existe en Python.

Sentencias iterativas

- **while condición:**
  - Sentencias (con un tabulador se encuentran dentro del bucle).

- El bucle while (mientras) ejecuta un fragmento de código mientras se cumpla una condición.

```
edad = 0
while edad < 18:
    edad = edad + 1
    print (“Felicidades, tienes “ + str(edad))
```

Sentencias iterativas

- Podemos utilizar la palabra **break** para romper el bucle:
  - El **parámetro** que pasamos a **input** es la cadena que se imprime de **prompt**.

```
while True:
    entrada = input(“> “)
    if entrada == “adios”:
        break
    else:
        print(entrada)
```

Sentencias iterativas

- La palabra **continue** fuerza otra iteración en el bucle. En este caso se salta los números pares.

```
i = 0
while i < 10:
    i = i+1
    if i%2==0:
        continue
    print i
```

Sentencias iterativas

- El while se puede combinar con **else**

```
i=20
while i < 20:
        print(i)
        i+=1
else:
        print (“fin del bucle“)
```

- El **else** lo ejecuta cuando no se cumple la condición.

Sentencias iterativas

- En Python for se utiliza como una forma genérica de iterar sobre una secuencia

- for variable in serie-valores:
  - Sentencias

- Muchas posibilidades:

  for nombre in [‘pepe’, ‘ana’, ‘ … ‘]:

  for variable in range(inicio, fin):

  for car in ‘micadena’:

- Recorrer rangos de valores, listas, cadenas, tuplas …

Sentencias iterativas

- Ejemplo con tuplas, con listas y cadenas se puede utilizar igual:

- Ejemplo:

```
t=(1,2,3,4,5,6,8)
for j in t:
        print (j, end=“ “)  # no hace salto de línea
```

Sentencias iterativas

- En el bucle for también se puede utilizar la sentencia else como en el bucle while.

- Se ejecuta cuando termina de recorrer la sentencia.

```
L=[3]
for k in L:
        print(k)
else:
        print (“fin del bucle“)
```

Sentencias iterativas

- Ejemplo: iterar sobre una lista de tuplas.

```
dic = [('a',1),('b',2),('c',3)]
for a,b in dic:
    print(a,b)
```

- En el for podemos recoger más de un valor cuando estamos iterando.

range

- Genera una lista de valores:
- range(8)
  - Genera los valores 0…7
- range(ini, fin)
  - Los valores entre inicio y fin-1.
- range(ini,fin,inc)
  - Los valores entre inicio y fin-1 con el incremento indicado.
  - range(8,1,-1) → [8,7,6,5,4,3,2]