Iterables

Python Básico | 17 minutos

Aprende como recorrer colecciones de elementos de distintas maneras.

Que son?

Un `iterable` es una colección de elementos que podemos recorrer (como una `lista`).

Una `iteración` es cuando recorremos un `iterable` elemento por elemento.

> Un `iterador` es un *objeto* interno de Python que guarda cual es la iteración actual de un iterable, pero esos son tecnicismos que no son necesarios en este punto.

Como lo podemos lograr?

Ya vimos la manera mas común de hacer una iteración: con un ciclo `for`.

Sin embargo, no es la única manera que existe.

Iteración directa

Esta es la manera más sencilla y elegante.

Es muy util cuando necesitamos leer los elementos de la lista.

Copiar

```
my_favorite_records = [
  'Dark Side Of The Moon',
  'Fear of a Blank Planet',
  'Signify',
]

for record in my_favorite_records:
  print(f'Record: {record}')
```

```
Record: Dark Side Of The Moon
Record: Fear of a Blank Planet
```

Record: Signify

Iteración por índice

Esta es la manera clásica.

Es muy util cuando necesitamos hacer cambios en la lista (como mover elementos de una posición a otra).

La función `range` nos permite crear una lista temporal de un rango de números.

Es muy util al hacer una iteración por índice.

```
my_favorite_records = [
  'Dark Side Of The Moon',
  'Fear of a Blank Planet',
  'Signify',
]

for index in range(0, len(my_favorite_records)):
  record = my_favorite_records[index]
  print(f'Record {index}: {record}')
```

Copiar

Record 0: Dark Side Of The Moon
Record 1: Fear of a Blank Planet
Record 2: Signify

Aunque también podemos hacerlo usando un ciclo while (*no es lo recomendado, pero es posible*).

```python
my_favorite_records = [
  'Dark Side Of The Moon',
  'Fear of a Blank Planet',
  'Signify',
]

index = 0
while (index < len(my_favorite_records)):
  record = my_favorite_records[index]
  print(f'Record {index}: {record}')
  index += 1
```

Copiar

```
Record 0: Dark Side Of The Moon
Record 1: Fear of a Blank Planet
Record 2: Signify
```

Iteración directa incluyendo índice

La función `enumerate` nos permite combinar ambos métodos en uno.

Esta se ve elegante y es igual de flexible que la clásica.

```python
my_favorite_records = [
  'Dark Side Of The Moon',
  'Fear of a Blank Planet',
  'Signify',
]

for index, record in enumerate(my_favorite_records):
  print(f'Record {index}: {record}')
```

Copiar

```
Record 0: Dark Side Of The Moon
Record 1: Fear of a Blank Planet
Record 2: Signify
```

Strings

Los strings son algo similares a las listas por dentro.

Ya que son una *colección* de caracteres.

Esto significa que también podemos iterarlos.

Veamos un ejemplo:

```
my_string = 'Star Wars'

for char in my_string:
  print(char)
```

Copiar

```
S
t
a
r
W
a
r
s
```

Rompiendo ciclos

A veces queremos que un ciclo se detenga antes lo esperado.

La instrucción `break` nos permite lograr esto.

```
colors = [
  'black',
  'yellow',
  'red',
  'blue',
]

for color in colors:
  print(color)
  if color == 'yellow':
    break
```

Copiar

```
black
yellow
```

Tambien funciona para detener ciclos infinitos.

```python
counter = 0
while (True):
  print(counter)
  if counter == 6:
    break

  counter += 1
```

Copiar

```
0
1
2
3
4
5
6
```

Adelantando ciclos

A veces queremos saltarnos una iteración antes de que todas sus instrucciones se completen.

La instrucción `continue` nos permite lograr esto.

Copiar

```
colors = [
  'black',
  'yellow',
  'red',
  'blue',
]

for color in colors:
  if color == 'yellow':
    continue

  print(color)
```

black
red
blue