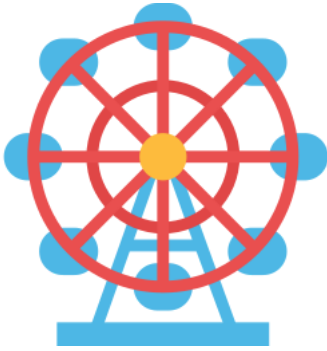


## ▼ Bucles



Cuando queremos hacer algo más de una vez (*iterar*) necesitamos un **bucle** y Python nos ofrece dos opciones para ello: `while` y `for`.

### ▼ Iterar con `for`

Python hace uso frecuentemente de **iteradores**.

Esto hace posible *recorrer* estructuras de datos sin conocer el tamaño que tienen o cómo están implementadas. Incluso es posible iterar sobre datos que se crean sobre la marcha, permitiendo el acceso a flujos de datos (*data streams*) que, de otra manera, no cabrían de una vez en la memoria de la máquina.

Para mostrar una iteración necesitamos algo sobre lo que iterar: **iterables**. Veamos un ejemplo con las cadenas de texto:

```
1 palabra = "abcdc"

1 for letra in palabra:# la orden "for" la palabra letra podría ser cualquiera la orden "in" nombre de la lista
2     print(letra)

a
b
c
d
c
```

### ▼ Generar secuencias de números

#### ▼ Bucles anidados

Es posible escribir un bucle dentro de otro. Esto se conoce como **bucles anidados**.

```
1 for i in range(3):
2     for j in range(2):
3         print(i, j)
4     print('---')

0 0
0 1
---
1 0
1 1
---
2 0
2 1
---
```

## Cuestionario sobre Bucles

Escribe un bucle `for` que imprima todos los elementos entre **-5** y **5** usando la función de `rango`.

```
1 # Escribe tu código debajo y presiona Shift+Enter para ejecutar
2
```

Imprime los elementos de la siguiente lista: Genres=[ 'rock', 'R&B', 'Soundtrack', 'R&B', 'soul', 'pop'] Asegúrate de seguir las convenciones de Python.

```
1 # Escribe tu código debajo y presiona Shift+Enter para ejecutar
2 Genres = ["rock","R&B","Soundtrack","R&B","soul","pop"]# una lista con sus elementos dentro de corchetes
3 for palabra in Genres:# la orden "for" una palabra para denominar la nueva lista "in" y el nombre de la lista de donde sacamos los ele
4     print(palabra)

rock
R&B
Soundtrack
R&B
soul
pop
```

---

Escribe un bucle for que imprima la siguiente lista: squares=['red', 'yellow', 'green', 'purple', 'blue']

```
1 # Escribe tu código debajo y presiona Shift+Enter para ejecutar
2 squares=["red","yellow","green","purple","blue"]
3 for colores in squares:
4     print(colores)
5

red
yellow
green
purple
blue
```

---

Escribe un bucle para mostrar los valores del Rating de una lista de reproducción de un álbum almacenada en la lista `PlayListRatings`. Si la puntuación es inferior a 6, sal del bucle. La lista `PlayListRatings` está dada por: `PlayListRatings = [10, 9.5, 10, 8, 7.5, 5, 10, 10]`

```
1 # Escribe tu código debajo y presiona Shift+Enter para ejecutar
2 PlayListRatings=[10,9.5,10,8,7.5,5,10,10]
3 for numeros in PlayListRatings:
4     if numeros < 6:# si "if" numero es menor que 6
5         print(numeros)
6

5
```

Escribe un bucle para copiar los string 'orange' de la lista `squares` a la lista `new_squares`. Para y sal del bucle si el valor de la lista no es 'orange':

```
1 # Escribe tu código debajo y presiona Shift+Enter para ejecutar
2 #Busca los colores en la lista squares y coloca los orange en la lista nex_squares
3 squares = ['orange', 'orange', 'purple', 'blue ', 'orange']# lista con elementos
4 new_squares = []# nueva lista vacia
5 for color in squares:#busca entre los elementos de la lista squares
6     if color ==("orange"):#si encuentra orange
7         new_squares.append("orange")#coloca en esta lista los orange
8
9 new_squares

['orange', 'orange', 'orange']
```

## ▼ Ejercicio

Dada una cadena de texto, indique el número de vocales que contiene:

Ejemplo:

 holaestoesunapruebaenpython

 12

```
1 # Escriba aquí su solución
2 texto = 'holaestoesunapruebaenpython'
3 vocales=["a","e","i","o","u"]# creamos una list. vocales para que tenga los elementos que debe buscar
```

```

4 suma_vocales=0# creo una variable que se llama suma_vocales que es igual a 0
5 for letras in texto:# una lista "letras" sus elementos van a ser las letras de la lista "texto"
6   if letras in vocales:#la comparativa if: si las letras son vocales se colocan en el list suma_vocales
7     suma_vocales = suma_vocales+1# revisa suma_vocales y les sumamos 1
8
9 print(suma_vocales)#así lo vemos #por ultimo muestra el valor de suma_vocales
12

```

La función `range()` devuelve un flujo de números en el rango especificado, sin necesidad de crear y almacenar previamente una larga estructura de datos. Esto permite generar rangos enormes sin consumir toda la memoria del sistema.

El uso de `range()` es similar a los *slices*: `range(start, stop, step)`. Podemos omitir `start` y el rango empezaría en 0. El único valor requerido es `stop` y el último valor generado será el justo anterior a este. El valor por defecto de `step` es 1, pero se puede ir "hacia detrás" con -1.

`range()` devuelve un objeto *iterable*, así que necesitamos obtener los valores paso a paso con una sentencia `for ... in` (o convertir el objeto a una secuencia como una lista).

Veamos un ejemplo generando el rango `[0, 1, 2]`

```

1 for x in range(0, 3):
2   print(x)

0
1
2

1 list(range(0, 3))

[0, 1, 2]

```

## ▼ Ejercicio (Opcional)

Imprima los 100 primeros números de la secuencia de Fibonacci: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...

## ▼ Pista: \_


Hay situaciones en las que no necesitamos usar la variable que toma valores en el rango, únicamente queremos *repetir una acción un número de veces*.

Para estos casos se suele recomendar usar el **subguión** (*guión bajo*) como nombre de variable, que da a entender que no estamos usando esta variable de forma explícita:

```

1 for _ in range(10):
2   print('Hello world!')

```

 Hello world!  
Hello world!  
Hello world!  
Hello world!  
Hello world!  
Hello world!  
Hello world!  
Hello world!  
Hello world!  
Hello world!

Simplemente hemos mostrado 10 veces el mensaje `Hello world!` sin necesidad usar un contador.

```

1 # Escriba aquí su solución
2 # Partiendo del 0 y el 1
3 a = 0
4 b = 1
5 #Bucle:
6

```

---

✓ 0 s completado a las 17:23

● ×