# Rangos

En Python 3, range es un tipo de datos. El tipo range es una lista **inmutable** de números enteros en **sucesión aritmética.**

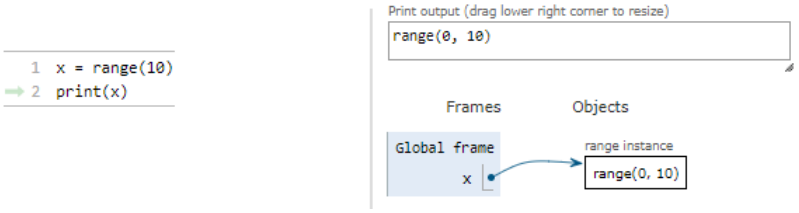
> **Inmutable significa que, a diferencia de las listas, los range no se pueden modificar.**
>
> **Una sucesión artimética es una sucesión en la que la diferencia entre dos términos consecutivos es siempre la misma.**

Un range se crea llamando al tipo de datos con uno, dos o tres argumentos numéricos, como si fuera una función. El tipo range() con un único argumento se escribe range(n) y crea una lista inmutable de n números enteros consecutivos que empieza en 0 y acaba en n - 1

Para ver los valores del range(), es necesario convertirlo a lista mediante la función list()

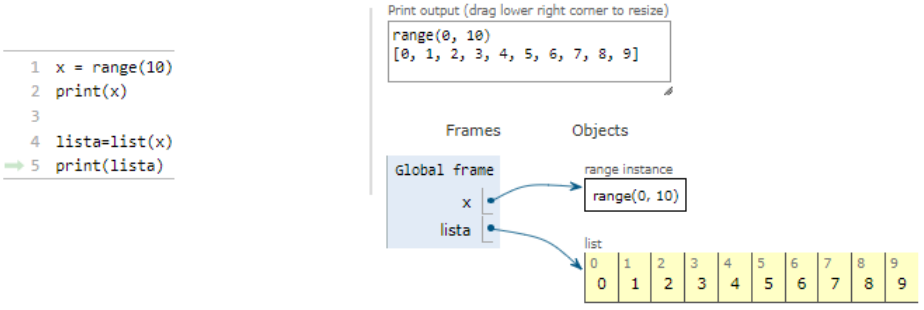
In [1]:
```
x = range(10)
x
```

Out[1]: `range(0, 10)`




In [2]:
```
list(x)
```

Out[2]: `[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]`




In [3]:
```
range(7)
```

Out[3]: `range(0, 7)`

In [4]:
```
list(range(7))
```

Out[4]: `[0, 1, 2, 3, 4, 5, 6]`

**Si n no es positivo, se crea un range vacío:**

In [5]:
```
x = range(-2)
x
```

Out[5]: `range(0, -2)`

In [6]:
```
list(x)
```

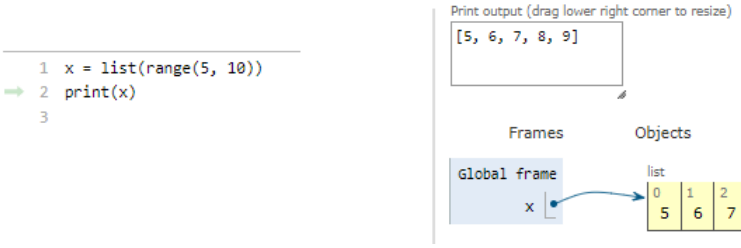
Out[6]: `[]`

In [7]:
```
list(range(0))
```

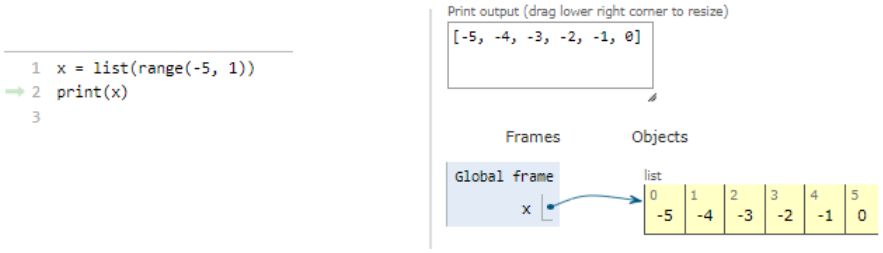
Out[7]: `[]`

El tipo range con dos argumentos se escribe range(m, n) y crea una lista inmutable de enteros consecutivos que empieza en m y termina en n - 1.

```
In [8]: x = list(range(5, 10))
        x
Out[8]: [5, 6, 7, 8, 9]
```




```
In [9]: x =list(range(-5, 1))
        x
Out[9]: [-5, -4, -3, -2, -1, 0]
```



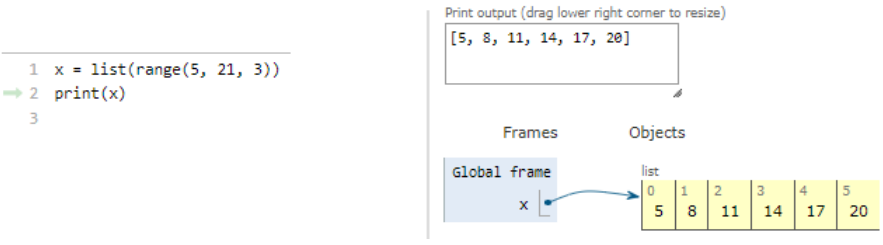

**Si n es menor o igual que m, se crea un range vacío.**

```
In [10]: list(range(5, 1))
Out[10]: []
```

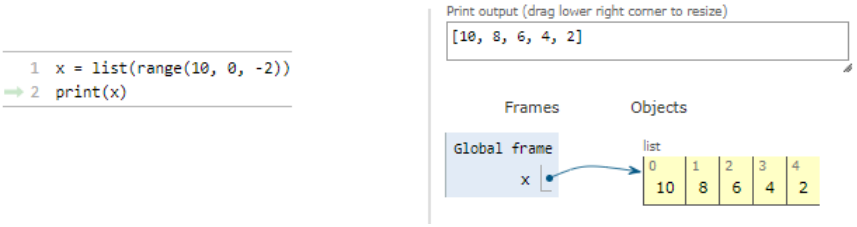
```
In [11]: list(range(3, 3))
Out[11]: []
```

El tipo range con tres argumentos se escribe range(m, n, p) y crea una lista inmutable de enteros que empieza en m y termina justo antes de superar o igualar a n, aumentando los valores de p en p. Si p es negativo, los valores van disminuyendo de p en p.

```
In [12]: x = list(range(5, 21, 3))
         x
Out[12]: [5, 8, 11, 14, 17, 20]
```




```
In [13]: list(range(10, 0, -2))
Out[13]: [10, 8, 6, 4, 2]
```



**p no puede ser 0, Si p es positivo y n menor o igual que m, o si p es negativo y n mayor o igual que m, se crea un range vacío.**

In [14]:
```
range(4,18,0)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [14], in <cell line: 1>()
----> 1 range(4,18,0)

ValueError: range() arg 3 must not be zero
```

In [15]:
```
list(range(25, 20, 2))
```

Out[15]: []

In [16]:
```
list(range(20, 25, -2))
```

Out[16]: []

En los range( m, n, p), se pueden escribir p range distintos que generan el mismo resultado. Por ejemplo:

In [17]:
```
list(range(10, 20, 3))
```

Out[17]: [10, 13, 16, 19]

In [18]:
```
list(range(10, 21, 3))
```

Out[18]: [10, 13, 16, 19]

In [19]:
```
list(range(10, 22, 3))
```

Out[19]: [10, 13, 16, 19]

**En resumen, los tres argumentos del tipo range(m, n, p) son:**

> **m: el valor inicial**

> **n: el valor final (que no se alcanza nunca)**

> **p: el paso (la cantidad que se avanza cada vez)**

- **Si se escriben sólo dos argumentos, Python le asigna a p el valor 1. Es decir range(m, n) es lo mismo que range(m, n, 1)**
- **Si se escribe sólo un argumento, Python, le asigna a m el valor 0 y a p el valor 1. Es decir range(n) es lo mismo que range(0, n, 1).**

El tipo range() sólo admite argumentos enteros. Si se utilizan argumentos decimales, se produce un error:

In [20]:
```
range(3.5, 10, 2)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [20], in <cell line: 1>()
----> 1 range(3.5, 10, 2)

TypeError: 'float' object cannot be interpreted as an integer
```

No se pueden concatenar tipos range(), ya que el resultado de la concatenación puede no ser un tipo range()):

In [21]:
```
x = range(3) + range(5)
```

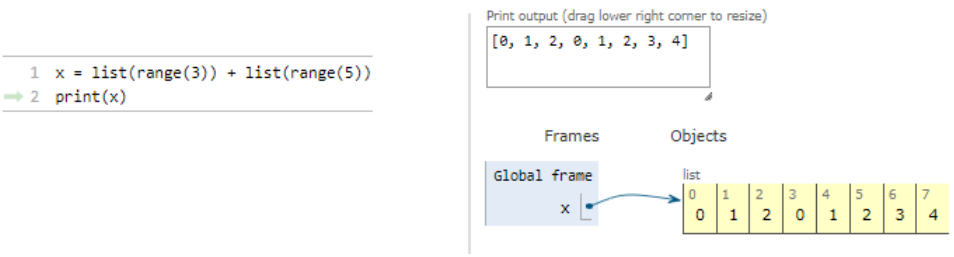
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [21], in <cell line: 1>()
----> 1 x = range(3) + range(5)

TypeError: unsupported operand type(s) for +: 'range' and 'range'
```

Pero sí se pueden concatenar tipos range() previamente convertidos en listas. El resultado es lógicamente una lista, que no se puede convertir a tipo range():

In [22]:
```
x = list(range(3)) + list(range(5))
x
```

Out[22]:
```
[0, 1, 2, 0, 1, 2, 3, 4]
```




No se pueden concatenar tipos range(), ni aunque el resultado sea una lista de números enteros en sucesión aritmética.

In [23]:
```
x = range(1, 3) + range(3, 5)
x
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [23], in <cell line: 1>()
----> 1 x = range(1, 3) + range(3, 5)
      2 x

TypeError: unsupported operand type(s) for +: 'range' and 'range'
```

In [24]:
```
x = list(range(1, 3)) + list(range(3, 5))
x
```

Out[24]:
```
[1, 2, 3, 4]
```

## La función len()

La función len() devuelve la longitud de una cadena de caracteres o el número de elementos de una lista:

El argumento de la función len() es la lista o cadena que queremos "medir

In [25]:
```
len("mensaje secreto")
```

Out[25]:
```
15
```

In [26]:
```
len(["a","b","c"])
```

Out[26]:
```
3
```

In [27]:
```
len(range(1, 100, 7))
```

Out[27]:
```
15
```

El valor devuelto por la función len() se puede usar como parámetro de range():

In [28]:
```
list(range(len("mensaje secreto")))
```

Out[28]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
```

In [29]:
```
list(range(len(["a","b","c"])))
```

Out[29]:
```
[0, 1, 2]
```

In [30]:
```
list(range(len(range(1, 100, 7))))
```

Out[30]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
```

In [31]:
```
len([0,3,5,7,8,9])
```

Out[31]:
```
6
```