

IA PUCP - Diplomatura de Desarrollo de Aplicaciones de Inteligencia Artificial  
**Python para Ciencia de Datos**



## **Introducción (rápida) a Python II**

# Repeticiones en Python

```
print("===")  
for i in range(3):  
    print(i)  
    print("===")  
print("fin")
```



# Repeticiones en Python

```
print("===")  
for i in range(3):  
    print(i)  
    print("===")  
print("fin")
```

```
===
```

# Repeticiones en Python

```
print("===")  
for i in range(3):  
    print(i)  
    print("===")  
print("fin")
```

Variables:

```
i = 0
```

```
===
```

# Repeticiones en Python

```
print("===")  
for i in range(3):  
    print(i)  
    print("===")  
print("fin")
```

Variables:

```
i = 0
```

```
===  
0
```

# Repeticiones en Python

```
print("===")  
for i in range(3):  
    print(i)  
    print("===")  
print("fin")
```

Variables:

```
i = 0
```

```
===  
0  
===
```

# Repeticiones en Python

```
print("===")  
for i in range(3):  
    print(i)  
    print("===")  
print("fin")
```

Variables:

```
i = 1
```

```
===  
0  
===
```

# Repeticiones en Python

```
print("===")  
for i in range(3):  
    print(i)  
    print("===")  
print("fin")
```

Variables:

```
i = 1
```

```
===  
0  
===  
1
```



# Repeticiones en Python

```
print("===")  
for i in range(3):  
    print(i)  
    print("===")  
print("fin")
```

Variables:

```
i = 1
```

```
===  
0  
===  
1  
===
```

# Repeticiones en Python

```
print("===")  
for i in range(3):  
    print(i)  
    print("===")  
print("fin")
```

Variables:

```
i = 2
```

```
===  
0  
===  
1  
===
```

# Repeticiones en Python

```
print("===")  
for i in range(3):  
    print(i)  
    print("===")  
print("fin")
```

Variables:

```
i = 2
```

```
===  
0  
===  
1  
===  
2
```

# Repeticiones en Python

```
print("===")  
for i in range(3):  
    print(i)  
    print("===")  
print("fin")
```

Variables:

```
i = 2
```

```
===  
0  
===  
1  
===  
2  
===
```

# Repeticiones en Python

```
print("===")
for i in range(3):
    print(i)
    print("===")
print("fin")
```

Variables:


~~i~~

Esta variable `i` solo pertenece al ámbito del bloque `for`

```
===
0
===
1
===
2
===
fin
```

# Repeticiones en Python

```
print("===")  
for i in range(3):  
    print(i)  
    print("===")  
print("fin")
```



El código continuará  
secuencialmente después

```
===  
0  
===  
1  
===  
2  
===  
fin
```

# Sintaxis de la instrucción `for`

```
for variable in range(numero) :  
    expresión  
    expresión
```

- Cada vez que se ejecuta el loop, la `variable` toma un valor del objeto `range`

Nota: no modificar el valor de la variable

## El objeto `range(stop)`

- Cuando instanciamos el objeto `range` de esta forma, nos entregará una secuencia de números
  - Inicia en 0 (`start = 0`)
  - Va en paso de 1 en 1 (`step = 1`)
  - Su último valor es `stop-1`



## El objeto `range(stop)`

¿Qué salida muestra el siguiente código?

```
for i in range(5):  
    print(i)
```

A

1  
2  
3  
4  
5

B

0  
1  
2  
3  
4  
5

C

0  
1  
2  
3  
4

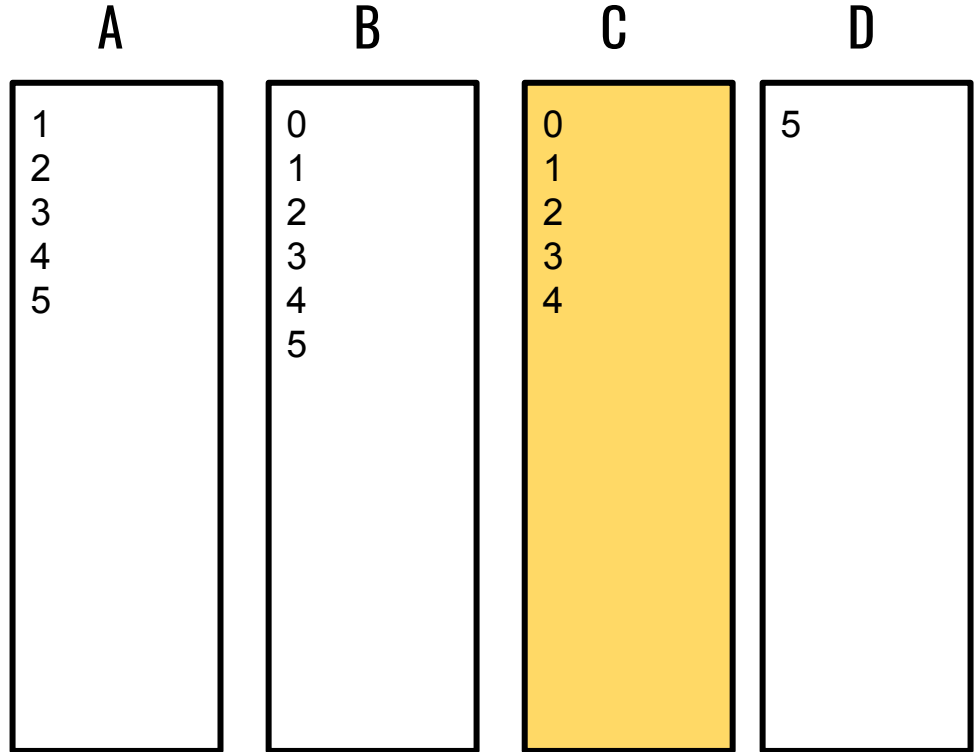
D

5

## El objeto `range(stop)`

¿Qué salida muestra el siguiente código?

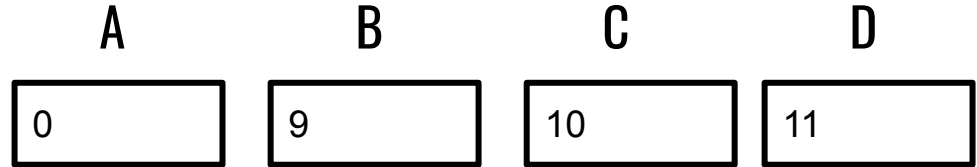
```
for i in range(5):  
    print(i)
```



## El objeto `range(stop)`

¿Cuántos valores genera  
el siguiente código?

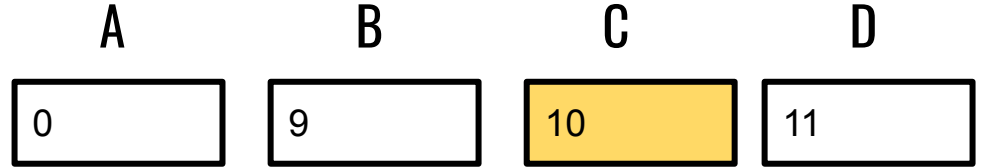
```
for i in range(10):  
    print(i)
```



## El objeto `range(stop)`

¿Cuántos valores genera  
el siguiente código?

```
for i in range(10):  
    print(i)
```



El objeto `range(start, stop)`

- Cuando instanciamos el objeto range de esta forma, nos entregará una secuencia de números
  - Inicia en `start`
  - Va en paso de 1 en 1 (`step = 1`)
  - Su último valor es `stop-1`

El objeto `range(start, stop)`

¿Qué salida muestra el siguiente código?

```
for i in range(1, 5):  
    print(i)
```

A

1  
2  
3  
4

B

0  
1  
2  
3  
4  
5

C

1  
2  
3  
4  
5

D

5

El objeto `range(start, stop)`

¿Qué salida muestra el siguiente código?

```
for i in range(1, 5):  
    print(i)
```

A

1  
2  
3  
4

B

0  
1  
2  
3  
4  
5

C

1  
2  
3  
4  
5

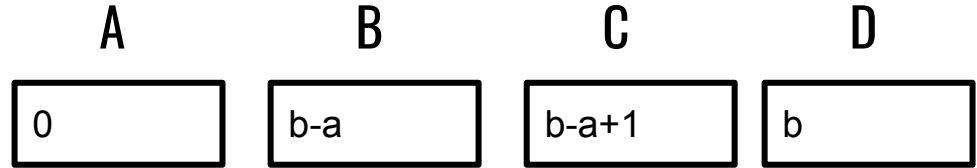
D

5

# El objeto `range(stop)`

¿Cuántos valores genera  
el siguiente código? si  $0 < a < b$

```
for i in range(a, b):  
    print(i)
```

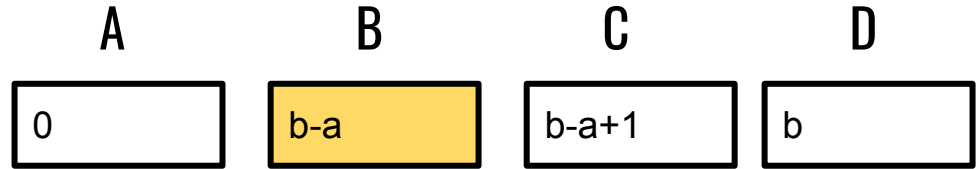




# El objeto `range(stop)`

¿Cuántos valores genera  
el siguiente código? si  $0 < a < b$

```
for i in range(a, b):  
    print(i)
```



## El objeto `range(start, stop, step)`

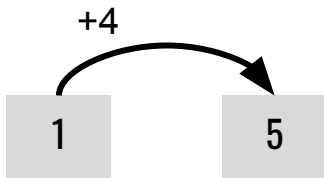
- Cuando instanciamos el objeto `range` de esta forma, nos entregará una secuencia de números
  - Inicia en `start`
  - Va en pasos de `step`
  - Su último valor posible es el anterior a `stop`

```
range(1, 17, 4)
```

## El objeto `range(start, stop, step)`

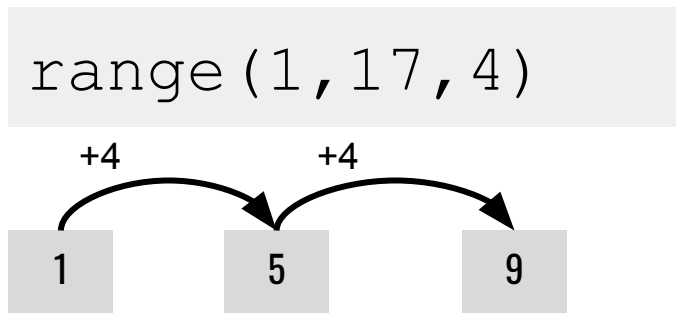
- Cuando instanciamos el objeto `range` de esta forma, nos entregará una secuencia de números
  - Inicia en **`start`**
  - Va en pasos de **`step`**
  - Su último valor posible es el anterior a **`stop`**

```
range(1, 17, 4)
```



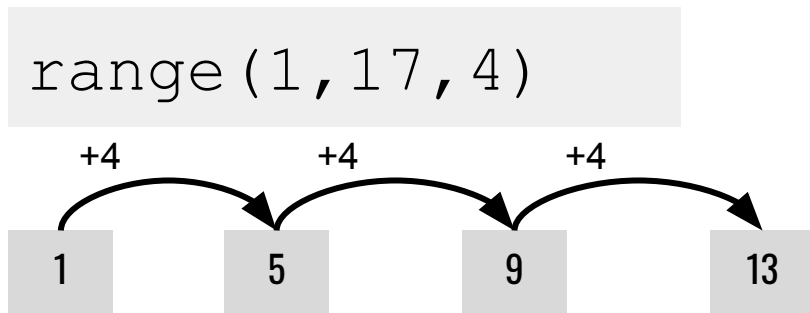
## El objeto `range(start, stop, step)`

- Cuando instanciamos el objeto `range` de esta forma, nos entregará una secuencia de números
  - Inicia en **`start`**
  - Va en pasos de **`step`**
  - Su último valor posible es el anterior a **`stop`**



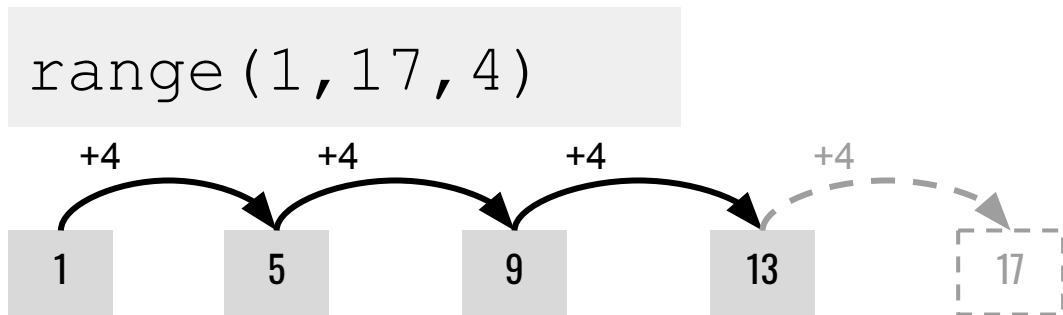
## El objeto `range(start, stop, step)`

- Cuando instanciamos el objeto `range` de esta forma, nos entregará una secuencia de números
  - Inicia en **`start`**
  - Va en pasos de **`step`**
  - Su último valor posible es el anterior a **`stop`**



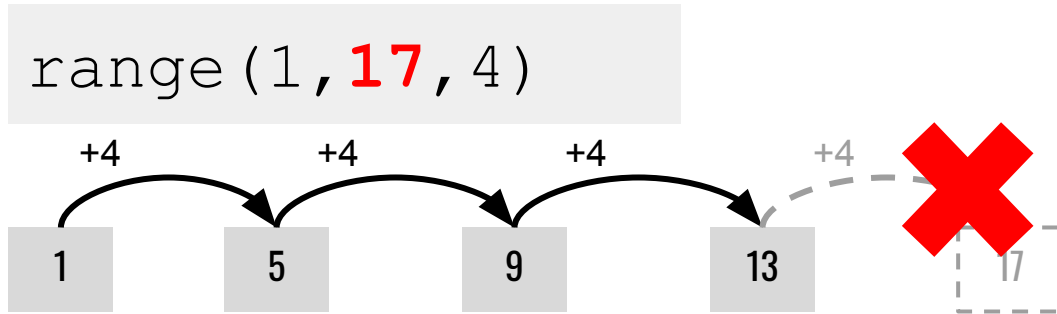
## El objeto `range(start, stop, step)`

- Cuando instanciamos el objeto `range` de esta forma, nos entregará una secuencia de números
  - Inicia en **`start`**
  - Va en pasos de **`step`**
  - Su último valor posible es el anterior a **`stop`**



# El objeto `range(start, stop, step)`

- Cuando instanciamos el objeto `range` de esta forma, nos entregará una secuencia de números
  - Inicia en **`start`**
  - Va en pasos de **`step`**
  - Su último valor posible es el anterior a **`stop`**



Siempre el  
máximo valor  
anterior a `stop`  
en la secuencia

El objeto `range(start, stop, step)`

¿Qué salida muestra el siguiente código?

```
for i in range(2, 18, 3):  
    print(i)
```

A

2  
5  
8  
11  
14

B

2  
5  
8  
11  
14  
17

C

2  
5  
8  
11  
14  
17  
20

D

2  
5  
8  
11  
14  
17  
20  
24



El objeto `range(start, stop, step)`

¿Qué salida muestra el siguiente código?

```
for i in range(2, 18, 3):  
    print(i)
```

A

2  
5  
8  
11  
14

B

2  
5  
8  
11  
14  
17

C

2  
5  
8  
11  
14  
17  
20

D

2  
5  
8  
11  
14  
17  
20  
24

El objeto `range(start, stop, step)`

A

A tall, empty rectangular box with a black border.

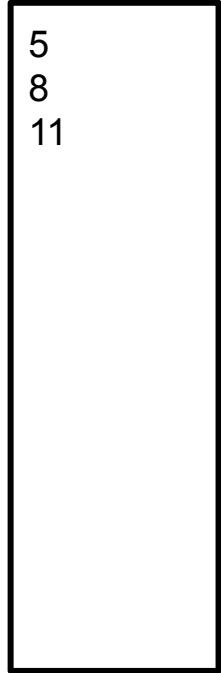
B

A tall rectangular box with a black border containing the numbers 10, 7, and 4 stacked vertically.

C

A tall rectangular box with a black border containing the numbers 5 and 8 stacked vertically.

D

A tall rectangular box with a black border containing the numbers 5, 8, and 11 stacked vertically.

¿Qué salida muestra el siguiente código?

```
for i in range(10, 5, 3):  
    print(i)
```

El objeto `range(start, stop, step)`

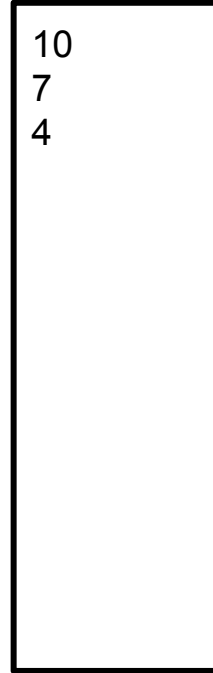
¿Qué salida muestra el siguiente código?

```
for i in range(10, 5, 3):  
    print(i)
```

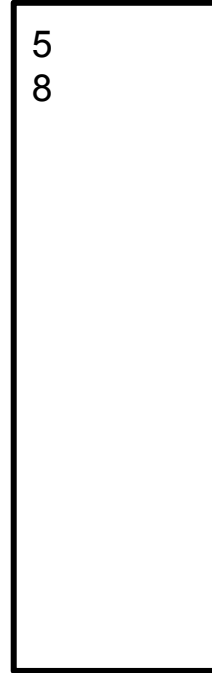
A



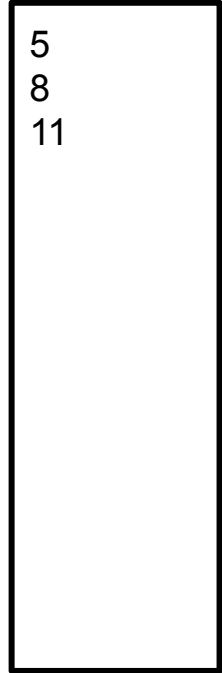
B



C



D



El objeto `range(start, stop, step)`

A

A tall, empty rectangular box with a black border.

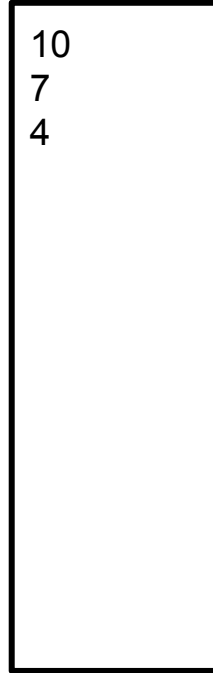
B

10  
7

A tall rectangular box with a black border containing the numbers 10 and 7 stacked vertically.

C

10  
7  
4

A tall rectangular box with a black border containing the numbers 10, 7, and 4 stacked vertically.

D

5  
8  
11

A tall rectangular box with a black border containing the numbers 5, 8, and 11 stacked vertically.

¿Qué salida muestra el siguiente código?

```
for i in range(10, 4, -3):  
    print(i)
```

El objeto `range(start, stop, step)`

¿Qué salida muestra el siguiente código?

```
for i in range(10, 4, -3):  
    print(i)
```

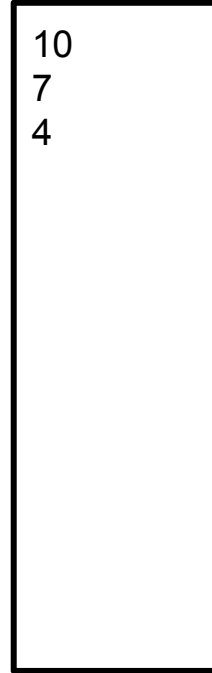
A

A tall, empty rectangular box with a black border, representing the output of the code for option A.

B

A tall rectangular box with a black border and a yellow background. It contains the numbers 10 and 7 stacked vertically, representing the output of the code for option B.

C

A tall, empty rectangular box with a black border, representing the output of the code for option C.

D

A tall, empty rectangular box with a black border, representing the output of the code for option D.

El objeto `range(start, stop, step)`

Con  
`step < 0`

- Cuando instanciamos el objeto `range` de esta forma, nos entregará una secuencia de números
  - Inicia en `start`
  - Va en pasos de `step`
  - Su último valor posible es el anterior a `stop`

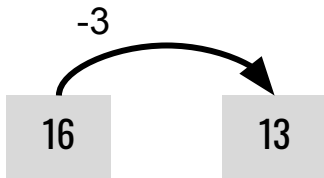
```
range(16, 4, -3)
```

El objeto `range(start, stop, step)`

Con  
`step < 0`

- Cuando instanciamos el objeto `range` de esta forma, nos entregará una secuencia de números
  - Inicia en **`start`**
  - Va en pasos de **`step`**
  - Su último valor posible es el anterior a **`stop`**

```
range(16, 4, -3)
```

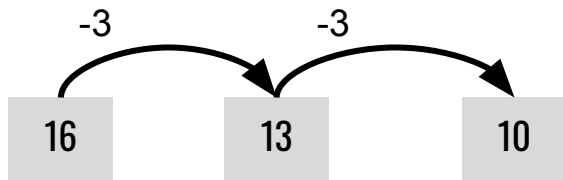


El objeto `range(start, stop, step)`

Con  
`step < 0`

- Cuando instanciamos el objeto `range` de esta forma, nos entregará una secuencia de números
  - Inicia en **`start`**
  - Va en pasos de **`step`**
  - Su último valor posible es el anterior a **`stop`**

```
range(16, 4, -3)
```

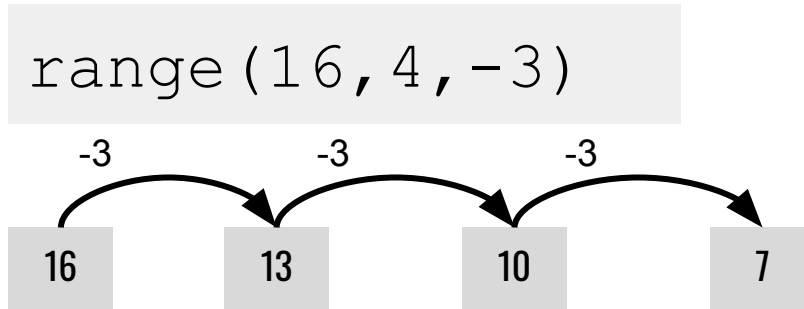




El objeto `range(start, stop, step)`

Con  
`step < 0`

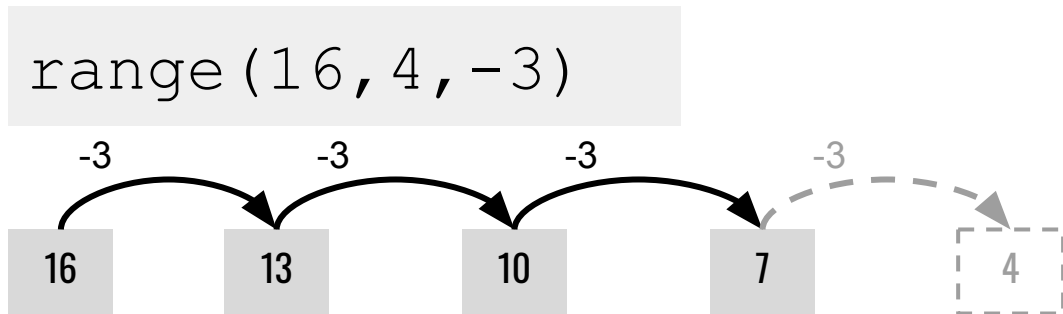
- Cuando instanciamos el objeto `range` de esta forma, nos entregará una secuencia de números
  - Inicia en **`start`**
  - Va en pasos de **`step`**
  - Su último valor posible es el anterior a **`stop`**



## El objeto `range(start, stop, step)`

Con  
`step < 0`

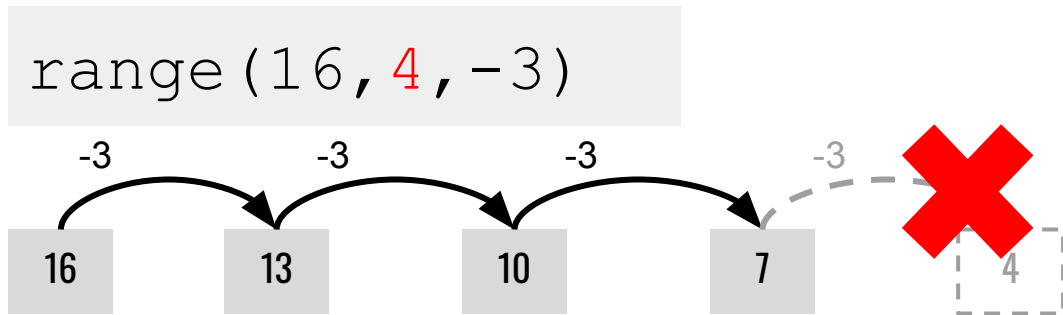
- Cuando instanciamos el objeto `range` de esta forma, nos entregará una secuencia de números
  - Inicia en **`start`**
  - Va en pasos de **`step`**
  - Su último valor posible es el anterior a **`stop`**



# El objeto `range(start, stop, step)`

Con  
`step < 0`

- Cuando instanciamos el objeto `range` de esta forma, nos entregará una secuencia de números
  - Inicia en **`start`**
  - Va en pasos de **`step`**
  - Su último valor posible es el anterior a **`stop`**



Siempre el  
máximo valor  
anterior a `stop`  
en la secuencia

## Valores de `range(10)`

¿Qué salida muestra el siguiente código?

```
for i in range(10):  
    print(i)
```

A

1  
2  
3  
4  
5  
6  
7  
8  
9  
10

B

0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10

C

0  
1  
2  
3  
4  
5  
6  
7  
8  
9

D

10

## Valores de `range(10)`

¿Qué salida muestra el siguiente código?

```
for i in range(10):  
    print(i)
```

A

1  
2  
3  
4  
5  
6  
7  
8  
9  
10

B

0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10

C

0  
1  
2  
3  
4  
5  
6  
7  
8  
9

D

10

## Valores de `range(10)`

A

0
---

B

9
---

C

10
----

D

11
----

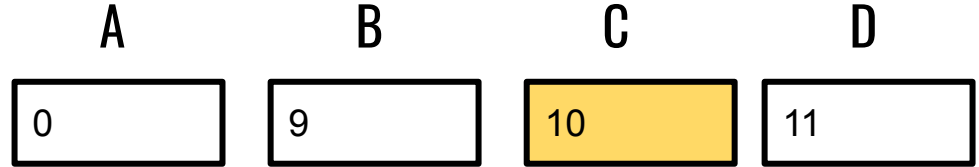
¿Cuántos valores genera  
el siguiente código?

```
for i in range(10):  
    print(i)
```

## Valores de `range(10)`

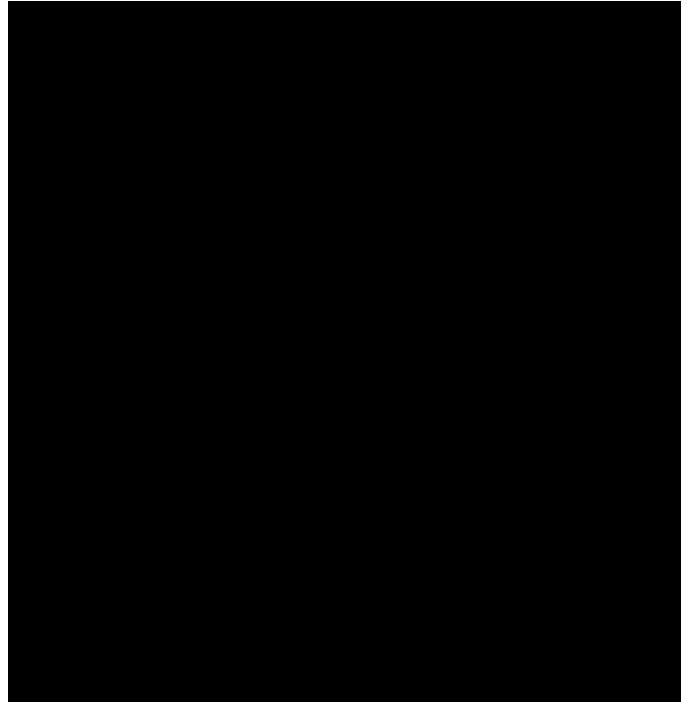
¿Cuántos valores genera  
el siguiente código?

```
for i in range(10):  
    print(i)
```



# Repeticiones en Python

```
print("===")  
for i in range(10):  
    print(i)  
    print("===")  
    if i==2:  
        break  
print("fin")
```





# Repeticiones en Python

```
print("===")  
for i in range(10):  
    print(i)  
    print("===")  
    if i==2:  
        break  
print("fin")
```

```
===
```

# Repeticiones en Python

```
print("===")  
for i in range(10):  
    print(i)  
    print("===")  
    if i==2:  
        break  
print("fin")
```

Variables:

```
i = 0
```

```
===
```

# Repeticiones en Python

```
print("===")  
for i in range(10):  
    print(i)  
    print("===")  
    if i==2:  
        break  
print("fin")
```

Variables:

```
i = 0
```

```
===  
0
```

# Repeticiones en Python

```
print("===")
for i in range(10):
    print(i)
    print("===")
    if i==2:
        break
print("fin")
```

Variables:

```
i = 0
```

```
===
0
===
```

# Repeticiones en Python

```
print("===")  
for i in range(10):  
    print(i)  
    print("===")  
    if i==2:  
        break  
print("fin")
```

Variables:

```
i = 0
```

```
===  
0  
===
```

# Repeticiones en Python

```
print("===")  
for i in range(10):  
    print(i)  
    print("===")  
    if i==2:  
        break  
print("fin")
```

Variables:

```
i = 1
```

```
===  
0  
===
```

# Repeticiones en Python

```
print("===")  
for i in range(10):  
    print(i)  
    print("===")  
    if i==2:  
        break  
print("fin")
```

Variables:

```
i = 1
```

```
===  
0  
===  
1
```

# Repeticiones en Python

```
print("===")  
for i in range(10):  
    print(i)  
    print("===")  
    if i==2:  
        break  
print("fin")
```

Variables:

```
i = 1
```

```
===  
0  
===  
1  
===
```



# Repeticiones en Python

```
print("===")  
for i in range(10):  
    print(i)  
    print("===")  
    if i==2:  
        break  
print("fin")
```

Variables:

```
i = 1
```

```
===  
0  
===  
1  
===
```

# Repeticiones en Python

```
print("===")  
for i in range(10):  
    print(i)  
    print("===")  
    if i==2:  
        break  
print("fin")
```

Variables:

```
i = 2
```

```
===  
0  
===  
1  
===
```

# Repeticiones en Python

```
print("===")  
for i in range(10):  
    print(i)  
    print("===")  
    if i==2:  
        break  
print("fin")
```

Variables:

```
i = 2
```

```
===  
0  
===  
1  
===  
2
```

# Repeticiones en Python

```
print("===")
for i in range(10):
    print(i)
    print("===")
    if i==2:
        break
print("fin")
```

Variables:

```
i = 2
```

```
===
0
===
1
===
2
===
```

# Repeticiones en Python

```
print("===")  
for i in range(10):  
    print(i)  
    print("===")  
    if i==2:  
        break  
print("fin")
```

Variables:

**i = 2**

```
===  
0  
===  
1  
===  
2  
===
```

# Repeticiones en Python

```
print("===")
for i in range(10):
    print(i)
    print("===")
    if i==2:
        break
print("fin")
```

Variables:

```
i = 2
```

```
===
0
===
1
===
2
===
```

# Repeticiones en Python

```
print("===")  
for i in range(10):  
    print(i)  
    print("===")  
    if i==2:  
        break  
print("fin")
```



La instrucción `break` detiene arbitrariamente el loop, saliendo del bloque

Variables:

```
i = 2
```

```
===  
0  
===  
1  
===  
2  
===
```

# Repeticiones en Python

```
print("===")
for i in range(10):
    print(i)
    print("===")
    if i==2:
        break
print("fin")
```

Variables:

```
i = 2
```

```
===
0
===
1
===
2
===
fin
```



# Repeticiones en Python

```
print("===")
for i in range(10):
    print(i)
    print("===")
    if i==2:
        break
print("fin")
```

Variables:

~~i~~

Esta variable `i` solo pertenece al ámbito del bloque `for`

```
===
0
===
1
===
2
===
fin
```

# Repeticiones en Python

```
print("===")  
for i in range(10):  
    print(i)  
    print("===")  
    if i==2:  
        break  
print("fin")
```



El código continuará  
secuencialmente después

```
===  
0  
===  
1  
===  
2  
===  
fin
```

# Valores de `range(10)` e instrucción `break`

¿Qué salida muestra el siguiente código?

```
for i in range(10):  
    if i==4:  
        break  
    print(i)
```

A

1  
2  
3  
4  
5  
6  
7  
8  
9

B

0  
1  
2  
3  
4  
5  
6  
7  
8  
9

C

0  
1  
2  
3

D

1  
2  
3

# Valores de `range(10)` e instrucción `break`

¿Qué salida muestra el siguiente código?

```
for i in range(10):  
    if i==4:  
        break  
    print(i)
```

A

1  
2  
3  
4  
5  
6  
7  
8  
9

B

0  
1  
2  
3  
4  
5  
6  
7  
8  
9

C

0  
1  
2  
3

D

1  
2  
3

# Repeticiones `while`

`while` **condición**:

    expresión

    expresión

...

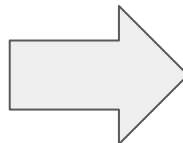
- La condición evalúa a `bool`
- Si la **condición** es **True**, se ejecutan todas las instrucciones del bloque
- Se vuelve a evaluar la condición
- Se repite hasta que la condición sea **False**

Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)

# Repeticiones en Python

```
i = 0
while i<9:
    print(i)
    i = i + 1
print("fin")
```



0  
1  
2  
3  
4  
5  
6  
6  
7  
8  
fin

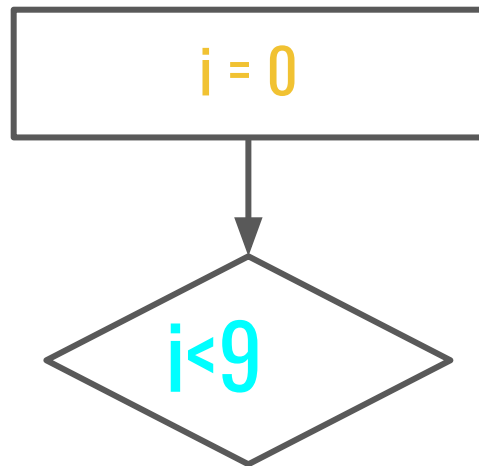
# Repeticiones en Python

i = 0

```
i = 0
while i<9:
    print(i)
    i = i + 1
print("fin")
```

# Repeticiones en Python

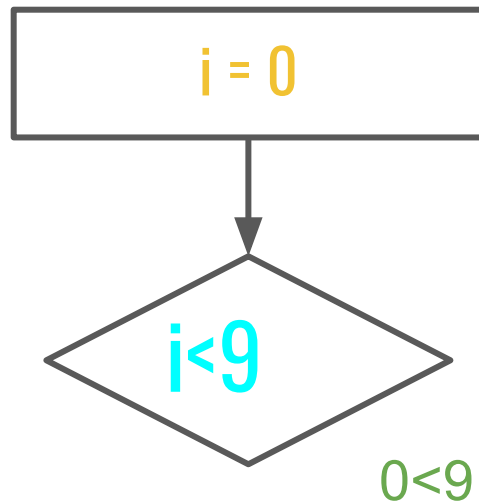
```
i = 0  
while i<9:  
    print(i)  
    i = i + 1  
print("fin")
```





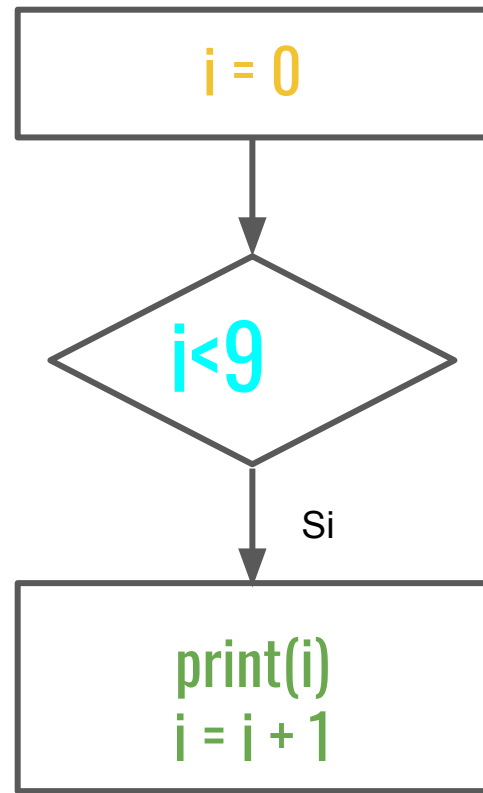
# Repeticiones en Python

```
i = 0  
while i<9:  
    print(i)  
    i = i + 1  
print("fin")
```



# Repeticiones en Python

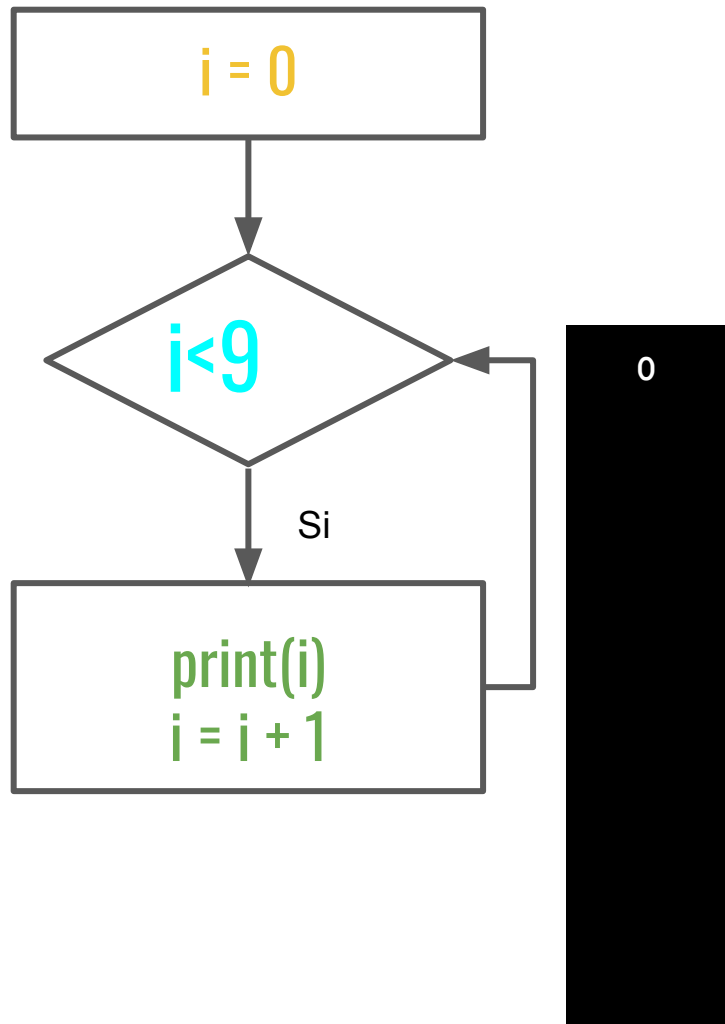
```
i = 0  
while i<9:  
    print(i)  
    i = i + 1  
print("fin")
```



0

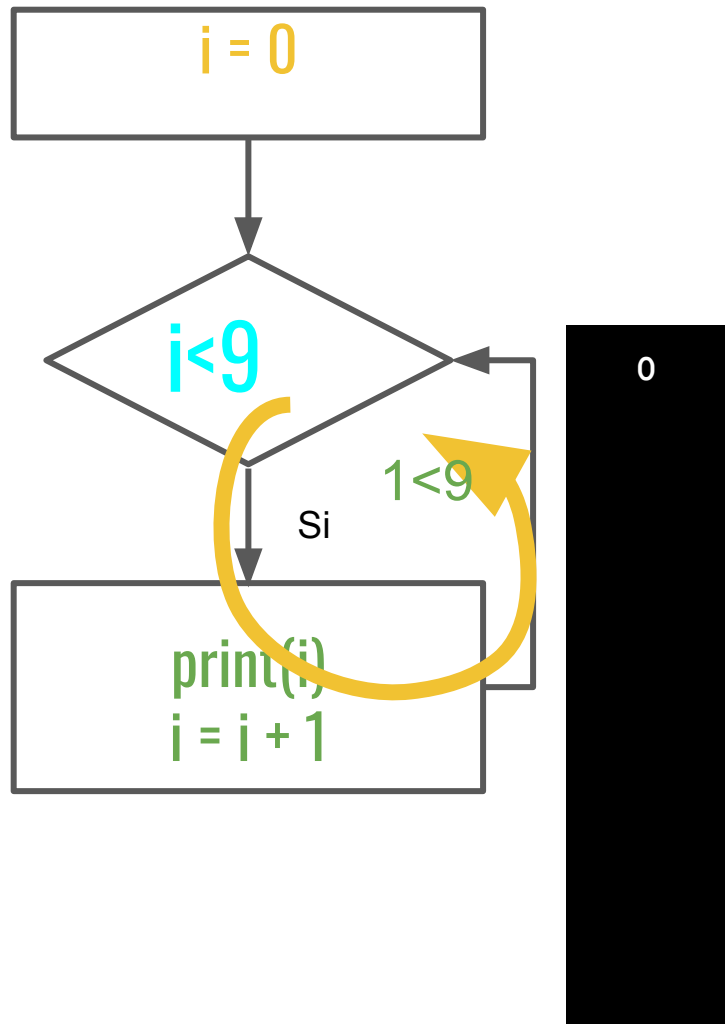
# Repeticiones en Python

```
i = 0
while i < 9:
    print(i)
    i = i + 1
print("fin")
```



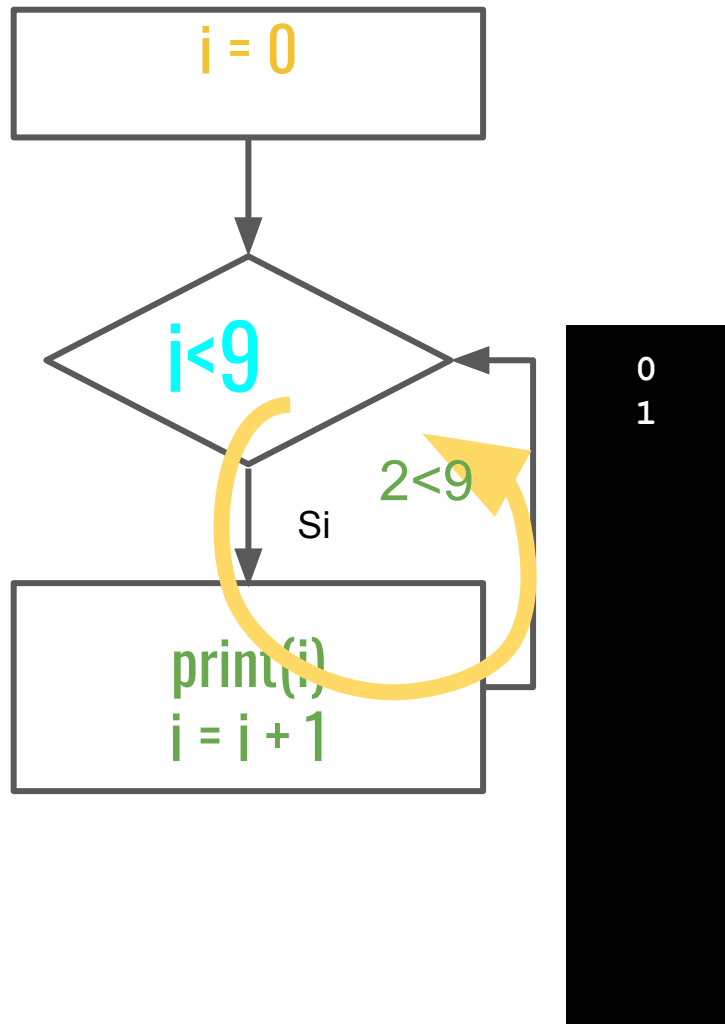
# Repeticiones en Python

```
i = 0
while i < 9:
    print(i)
    i = i + 1
print("fin")
```



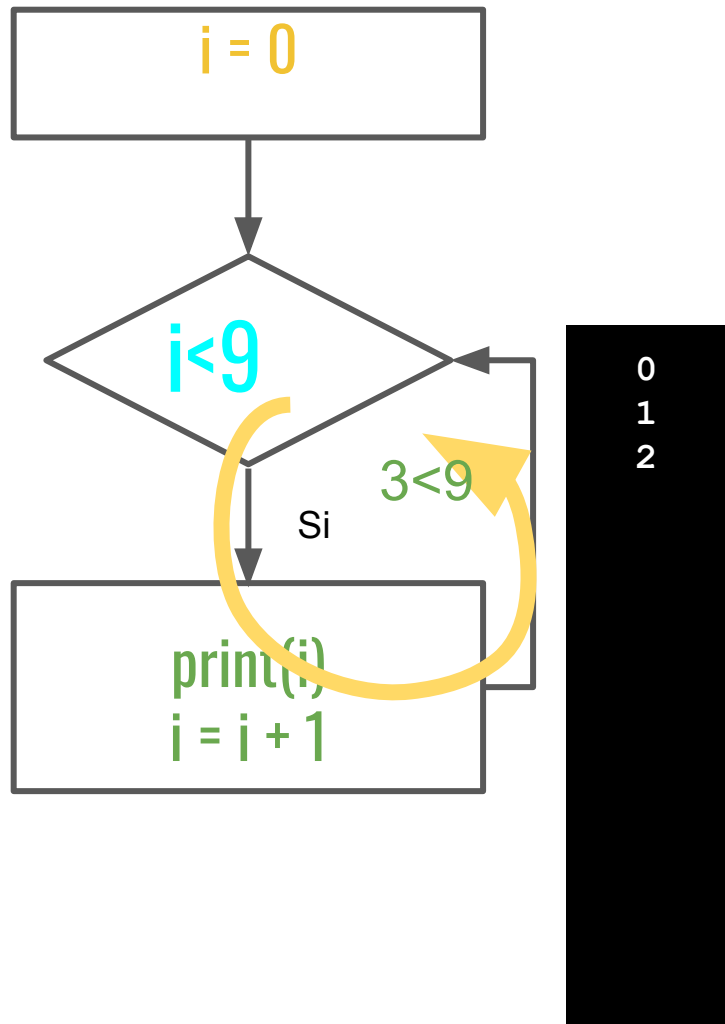
# Repeticiones en Python

```
i = 0  
while i < 9:  
    print(i)  
    i = i + 1  
print("fin")
```



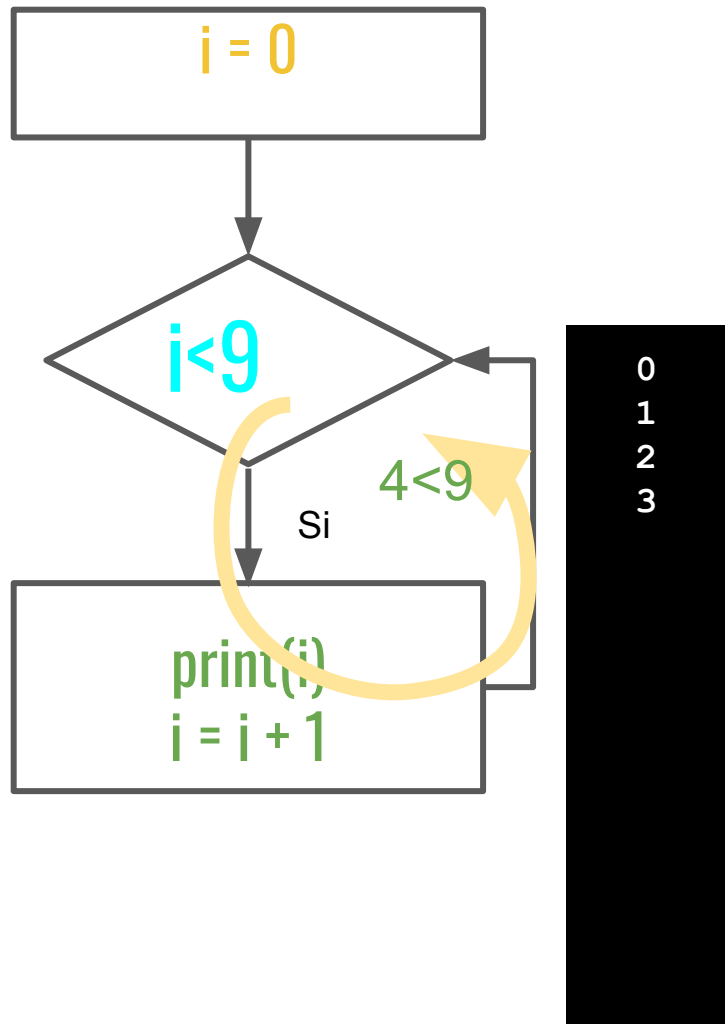
# Repeticiones en Python

```
i = 0  
while i<9:  
    print(i)  
    i = i + 1  
print("fin")
```



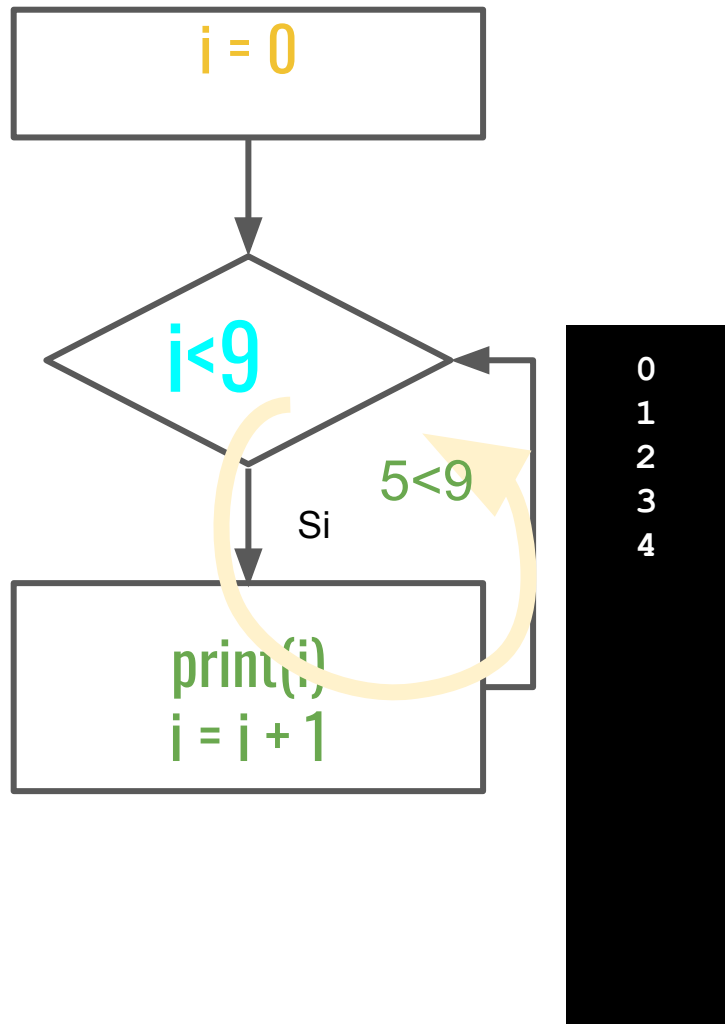
# Repeticiones en Python

```
i = 0
while i < 9:
    print(i)
    i = i + 1
print("fin")
```



# Repeticiones en Python

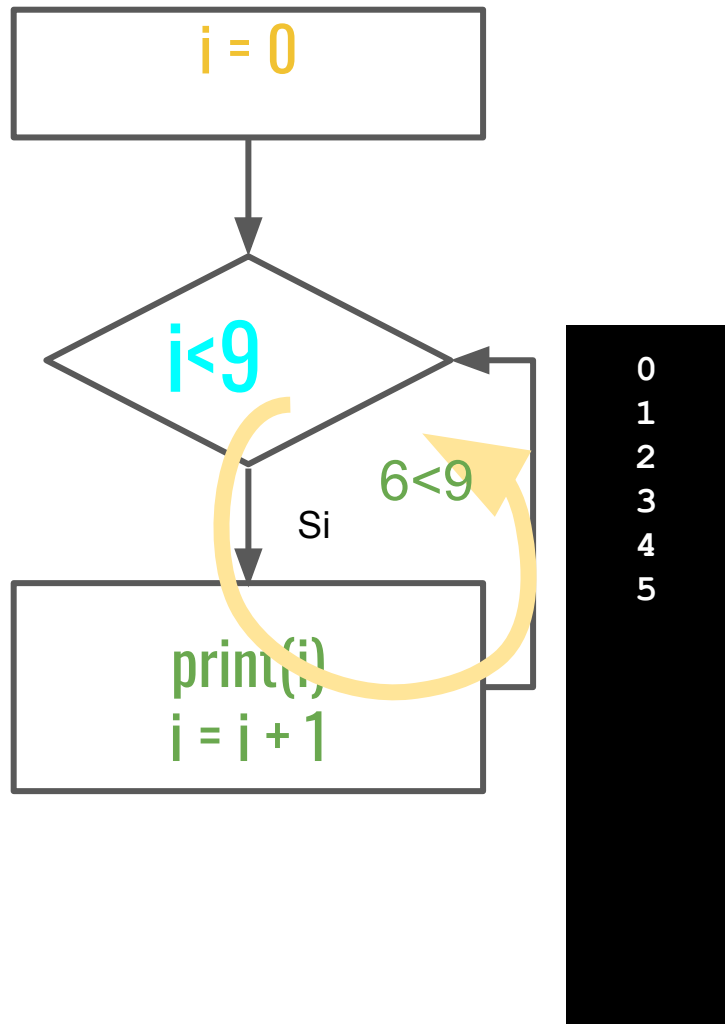
```
i = 0
while i < 9:
    print(i)
    i = i + 1
print("fin")
```





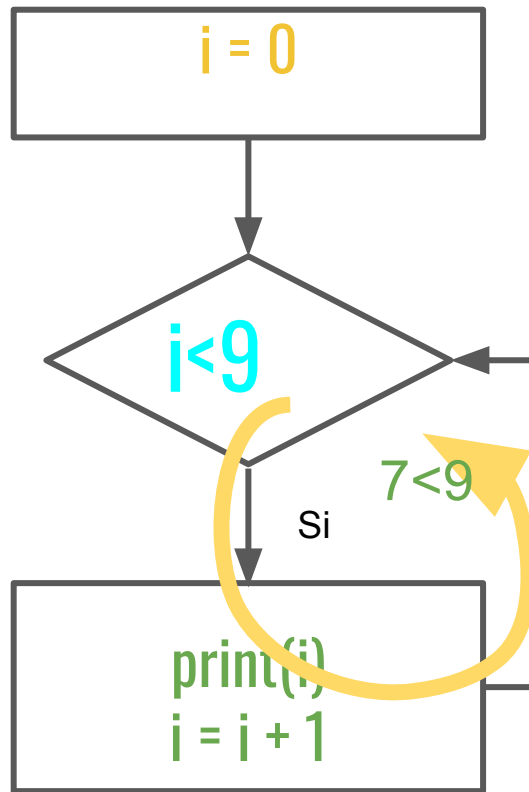
# Repeticiones en Python

```
i = 0
while i < 9:
    print(i)
    i = i + 1
print("fin")
```



# Repeticiones en Python

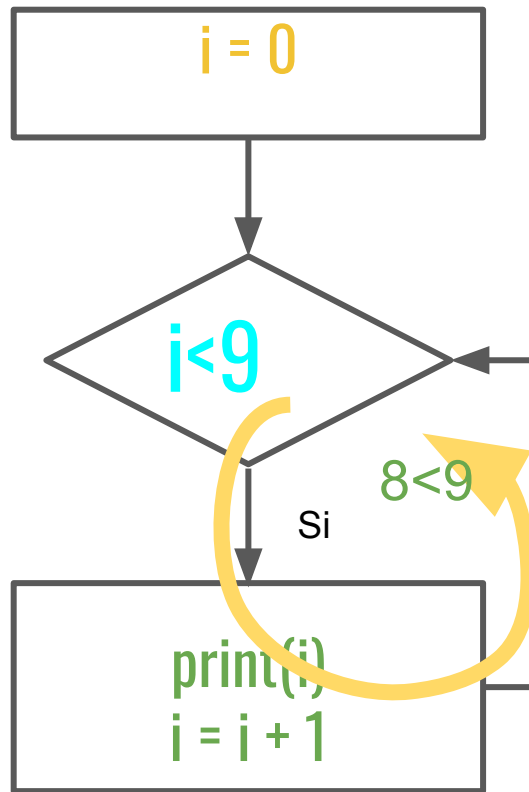
```
i = 0
while i < 9:
    print(i)
    i = i + 1
print("fin")
```



0  
1  
2  
3  
4  
5  
6

# Repeticiones en Python

```
i = 0  
while i < 9:  
    print(i)  
    i = i + 1  
print("fin")
```



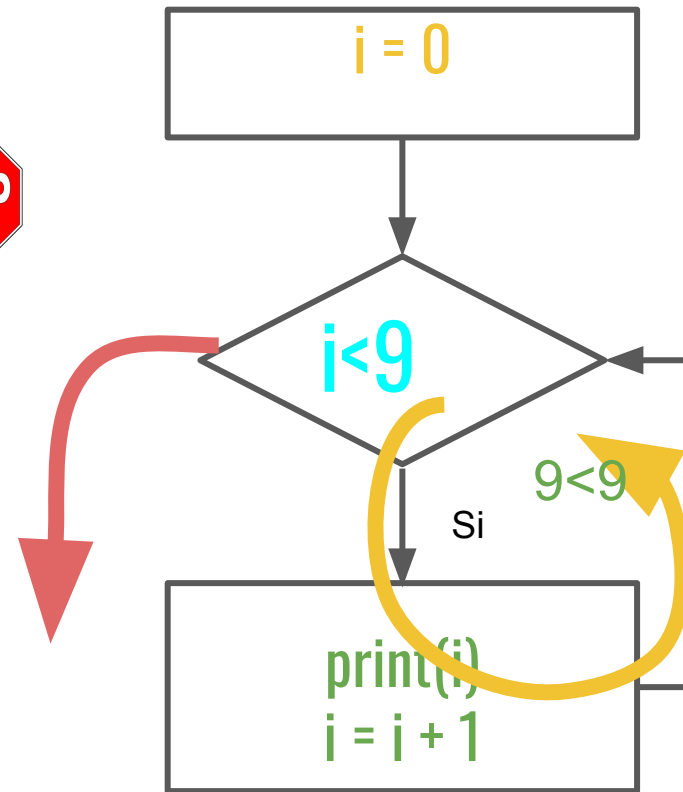
0  
1  
2  
3  
4  
5  
6  
7

# Repeticiones en Python

```
i = 0
while i < 9:
    print(i)
    i = i + 1
print("fin")
```



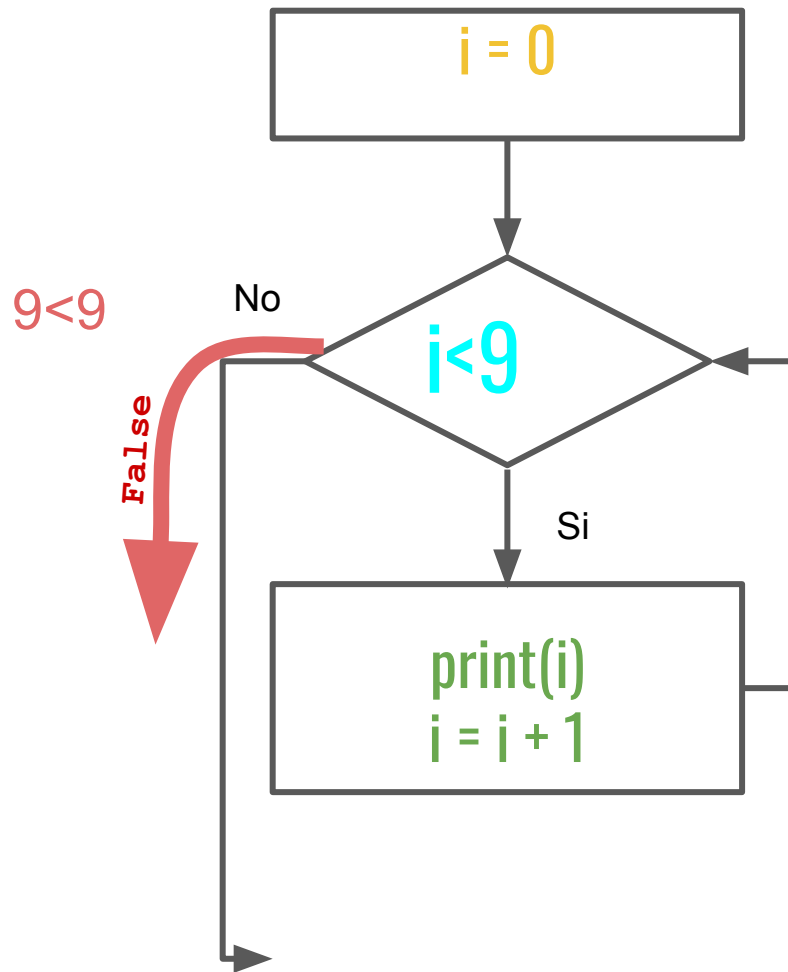
9 < 9



0  
1  
2  
3  
4  
5  
6  
7  
8

# Repeticiones en Python

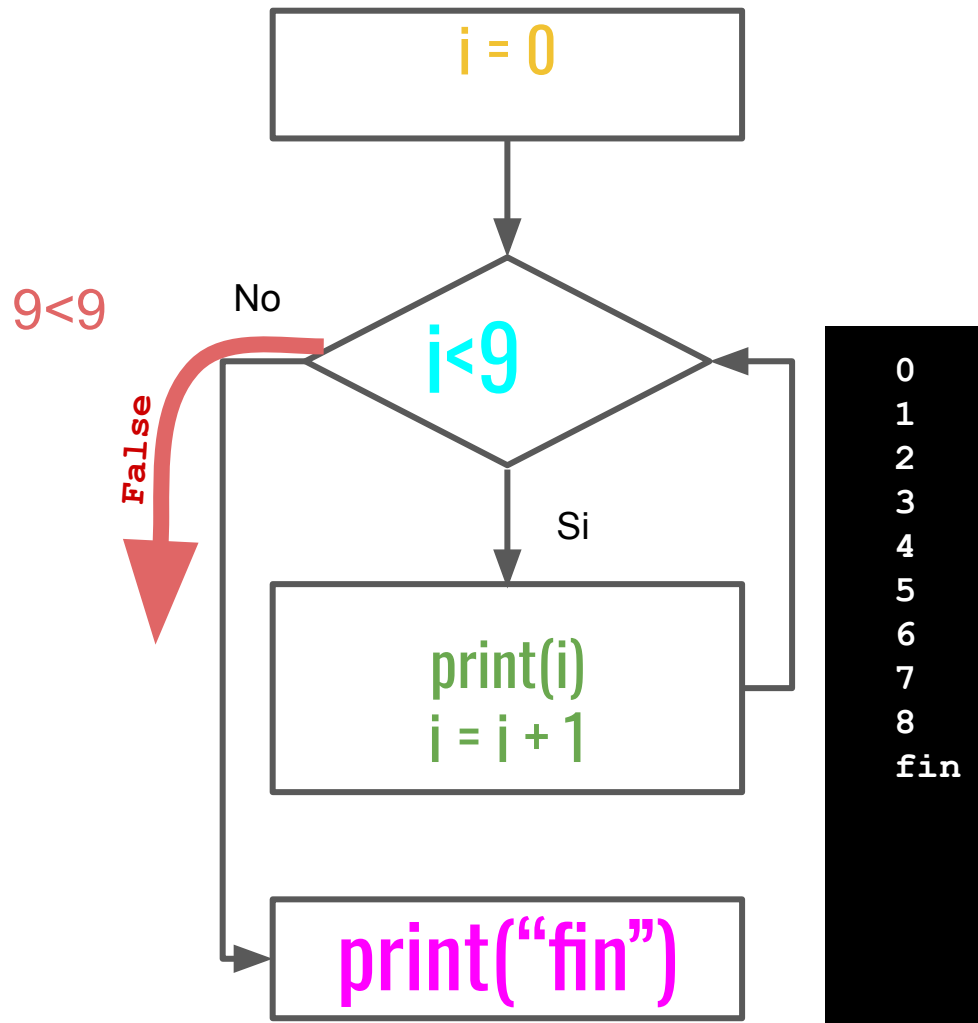
```
i = 0
while i < 9:
    print(i)
    i = i + 1
print("fin")
```



0  
1  
2  
3  
4  
5  
6  
7  
8

# Repeticiones en Python

```
i = 0
while i < 9:
    print(i)
    i = i + 1
print("fin")
```



# for **VS** while

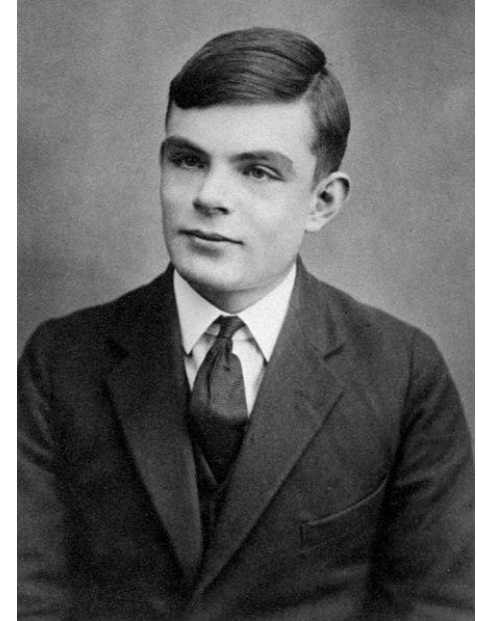
for	while
<ul style="list-style-type: none"><li>• El número de iteraciones es conocido</li><li>• Usa un contador</li><li>• Se puede reescribir con loop <code>while</code></li></ul>	<ul style="list-style-type: none"><li>• El número de iteraciones no está restringido</li><li>• Puede usar un contador, pero debe inicializarse fuera del loop e incrementarse dentro</li><li>• No necesariamente es posible reescribirlo como un loop <code>for</code></li></ul>

Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)

# Hasta el momento...

- Números, asignaciones, entrada/salida, comparaciones y repeticiones.
- Con esto ya tenemos en teoría un conocimiento de Python con el que podemos hacer todo (Turing Completo) – sin un problema se puede resolver computacionalmente, ya tenemos los mecanismos suficientes.





# Buenas prácticas de programación

- Tener más código no necesariamente es mejor
- Se mide a los programadores por la “cantidad” de funcionalidades
- Vamos a hablar de funciones, un mecanismo para descomponer y abstraer.

Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)

# Funciones

- Podemos escribir bloques de código **reusables**, que llamamos funciones
- Las funciones no son ejecutadas por un programa hasta que son **“llamadas”** o **“invocadas”** en el programa



Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)

# Funciones: Características

- Tiene un **nombre**
- Parámetros (0 o más)
- Tiene **docstring** (documentación)
- Tiene cuerpo
- Retorna algo

Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)

# Declarar una función

```
def es_par ( i ) :  
    """  
    Entrada: i, un int positivo  
    Retorna True si i es par y False de lo contrario  
    """  
    print("Dentro de es_par")  
    return i%2 == 0
```

# Declarar una función

```
def es_par ( i ) :
```

```
    """
```

```
    Entrada: i, un int positivo
```

```
    Retorna True si i es par y False de lo contrario
```

```
    """
```

```
    print("Dentro de es_par")
```

```
    return i%2 == 0
```

keyword

# Declarar una función

```
def es par ( i ) :
```

nombre

```
"""
```

```
Entrada: i, un int positivo
```

```
Retorna True si i es par y False de lo contrario
```

```
"""
```

```
print("Dentro de es_par")
```

```
return i%2 == 0
```

Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)

# Declarar una función

```
def es_par ( i ) :
```

parámetros o argumentos

```
    """
```

```
    Entrada: i, un int positivo
```

```
    Retorna True si i es par y False de lo contrario
```

```
    """
```

```
    print("Dentro de es_par")
```

```
    return i%2 == 0
```

Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)

# Declarar una función

```
def es_par ( i ):
```

especificación, docstring

```
    """
```

```
    Entrada: i, un int positivo
```

```
    Retorna True si i es par y False de lo contrario
```

```
    """
```

```
    print("Dentro de es_par")
```

```
    return i%2 == 0
```

Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)



# Declarar una función

```
def es_par ( i ) :  
    """
```

Entrada: i, un int positivo

Retorna True si i es par y False de lo contrario

```
    """
```

```
    print("Dentro de es_par")  
    return i%2 == 0
```

cuerpo de la función

# Declarar una función

```
def es_par ( i ) :  
    """  
    Entrada: i, un int positivo  
    Retorna True si i es par y False de lo contrario  
    """  
    print("Dentro de es_par")  
    return i%2 == 0
```

```
es_par(3)
```

Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)

# Declarar una función

```
def es_par ( i ) :  
    """
```

Entrada: i, un int positivo

Retorna True si i es par y False de lo contrario

```
    """
```

```
    print("Dentro de es_par")
```

```
    return i%2 == 0
```

llamada a la función con valores para los  
parámetros

```
es_par(3)
```

Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)

# En el cuerpo de la función

instrucciones a ejecutar

```
def es_par ( i ) :  
    """
```

Entrada: i, un int positivo

Retorna True si i es par y False de lo contrario

```
    """
```

```
    print("Dentro de es_par")
```

```
    return i%2 == 0
```

Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)

# En el cuerpo de la función

```
def es_par ( i ) :  
    """
```

Entrada: i, un int positivo

Retorna True si i es par y False de lo contrario

```
    """
```

```
    print("Dentro de es_par")
```

```
    return i%2 == 0
```

keyword

# En el cuerpo de la función

```
def es_par ( i ) :  
    """
```

Entrada: *i*, un int positivo

Retorna True si *i* es par y False de lo contrario

```
    """
```

```
    print("Dentro de es_par")
```

```
    return i%2 == 0
```

expresión que se evaluará para retornar

# Ámbito de las variables: Parámetros formales vs Parámetro reales

```
def es_par ( i ):  
    """  
    Entrada: i, un int positivo  
    Retorna True si i es par y False de lo contrario  
    """  
    print("Dentro de es_par")  
    return i%2 == 0  
  
i = 3  
es_par(i)
```

Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)

# Ámbito de las variables: Parámetros formales vs Parámetro reales

definición de la función

```
def es_par ( i ) :  
    """  
    Entrada: i, un int positivo  
    Retorna True si i es par y False de lo contrario  
    """  
    print("Dentro de es_par")  
    return i%2 == 0
```

`i = 3`

`es_par(i)`

Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)



# Ámbito de las variables: Parámetros formales vs Parámetro reales

parámetro formal

```
def es_par ( i ) :
```

```
    """
```

```
    Entrada: i, un int positivo
```

```
    Retorna True si i es par y False de lo contrario
```

```
    """
```

```
    print("Dentro de es_par")
```

```
    return i%2 == 0
```

```
i = 3
```

```
es_par(i)
```

Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)

# Ámbito de las variables: Parámetros formales vs Parámetro reales

parámetro real

```
def es_par ( i ) :  
    """  
    Entrada: i, un int positivo  
    Retorna True si i es par y False de lo contrario  
    """  
    print("Dentro de es_par")  
    return i%2 == 0  
  
i = 3  
es_par(i)
```

Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)

# Ámbito de las variables: Parámetros formales vs Parámetro reales

código principal: inicializa i, llama es\_par

```
def es_par ( i ) :  
    """  
    Entrada: i, un int positivo  
    Retorna True si i es par y False de lo contrario  
    """  
    print("Dentro de es_par")  
    return i%2 == 0  
  
i = 3  
es_par(i)
```

Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)

# Ámbito de las variables

```
def suma ( a, b ) :  
    a += b  
    c = a  
    return c  
  
a = 2  
b = 5  
c = suma (a, b)
```

Ámbito Global

# Ámbito de las variables

```
def suma ( a, b ) :  
    a += b  
    c = a  
    return c
```

*a* = 2

*b* = 5

*c* = *suma* (*a*, *b*)

Ámbito Global

Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)

# Ámbito de las variables

```
def suma ( a, b ) :  
    a += b  
    c = a  
    return c
```

*a* = 2

*b* = 5

*c* = *suma* (*a*, *b*)

Ámbito Global

suma

código

Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)

# Ámbito de las variables

```
def suma ( a, b ) :  
    a += b  
    c = a  
    return c  
a = 2  
b = 5  
c = suma ( a, b )
```

## Ámbito Global

suma

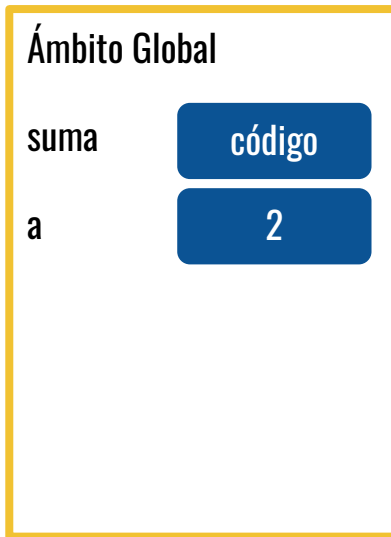
código

Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)

# Ámbito de las variables

```
def suma ( a, b ) :  
    a += b  
    c = a  
    return c  
a = 2  
b = 5  
c = suma ( a, b )
```



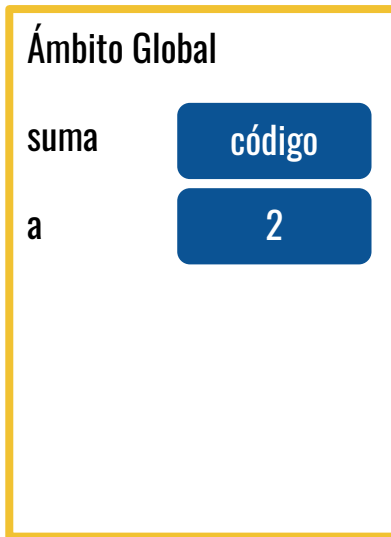
Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)



# Ámbito de las variables

```
def suma ( a, b ) :  
    a += b  
    c = a  
    return c  
  
a = 2  
b = 5  
c = suma ( a, b )
```



Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)

# Ámbito de las variables

```
def suma ( a, b ) :  
    a += b  
    c = a  
    return c  
  
a = 2  
b = 5  
c = suma ( a, b )
```

## Ámbito Global

suma

código

a

2

b

5

Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)

# Ámbito de las variables

```
def suma ( a, b ) :  
    a += b  
    c = a  
    return c  
  
a = 2  
b = 5  
c = suma ( a, b )
```

## Ámbito Global

suma

código

a

2

b

5

Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)

# Ámbito de las variables

```
def suma ( a, b ) :  
    a += b  
    c = a  
    return c  
  
a = 2  
b = 5  
c = suma ( a, b )
```

## Ámbito Global

suma

código

a

2

b

5

c

Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)

# Ámbito de las variables

```
def suma ( a, b ) :  
    a += b  
    c = a  
    return c  
  
a = 2  
b = 5  
c = suma ( a, b )
```

## Ámbito Global

suma

código

a

2

b

5

c

## Ámbito suma

a

b

c

Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)

# Ámbito de las variables

```
def suma ( a, b ) :
```

```
    a += b
```

```
    c = a
```

```
    return c
```

```
a = 2
```

```
b = 5
```

```
c = suma ( a, b )
```

## Ámbito Global

suma

código

a

2

b

5

c

## Ámbito suma

a

b

c

Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)

# Ámbito de las variables

```
def suma ( a, b ) :  
    a += b  
    c = a  
    return c  
  
a = 2  
b = 5  
c = suma ( a, b )
```

## Ámbito Global

suma

código

a

2

b

5

c

## Ámbito suma

a

2

b

c

Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)

# Ámbito de las variables

```
def suma ( a, b ) :  
    a += b  
    c = a  
    return c  
  
a = 2  
b = 5  
c = suma ( a, b )
```

## Ámbito Global

suma

código

a

2

b

5

c

## Ámbito suma

a

2

b

5

c

Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)



# Ámbito de las variables

```
def suma ( a, b ) :  
    a += b  
    c = a  
    return c  
  
a = 2  
b = 5  
c = suma ( a, b )
```

## Ámbito Global

suma

código

a

2

b

5

c

## Ámbito suma

a

2

b

5

c

Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)

# Ámbito de las variables

```
def suma ( a, b ) :  
    a += b  
    c = a  
    return c  
  
a = 2  
b = 5  
c = suma ( a, b )
```

## Ámbito Global

suma

código

a

2

b

5

c

## Ámbito suma

a

7

b

5

c

Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)

# Ámbito de las variables

```
def suma ( a, b ) :  
    a += b  
    c = a  
    return c  
  
a = 2  
b = 5  
c = suma ( a, b )
```

## Ámbito Global

suma

código

a

2

b

5

c

## Ámbito suma

a

7

b

5

c

Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)

# Ámbito de las variables

```
def suma ( a, b ) :  
    a += b  
    c = a  
    return c  
  
a = 2  
b = 5  
c = suma ( a, b )
```

## Ámbito Global

suma

código

a

2

b

5

c

## Ámbito suma

a

7

b

5

c

7

Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)

# Ámbito de las variables

```
def suma ( a, b ) :  
    a += b  
    c = a  
    return c  
  
a = 2  
b = 5  
c = suma ( a, b )
```

## Ámbito Global

suma

código

a

2

b

5

c

## Ámbito suma

a

7

b

5

c

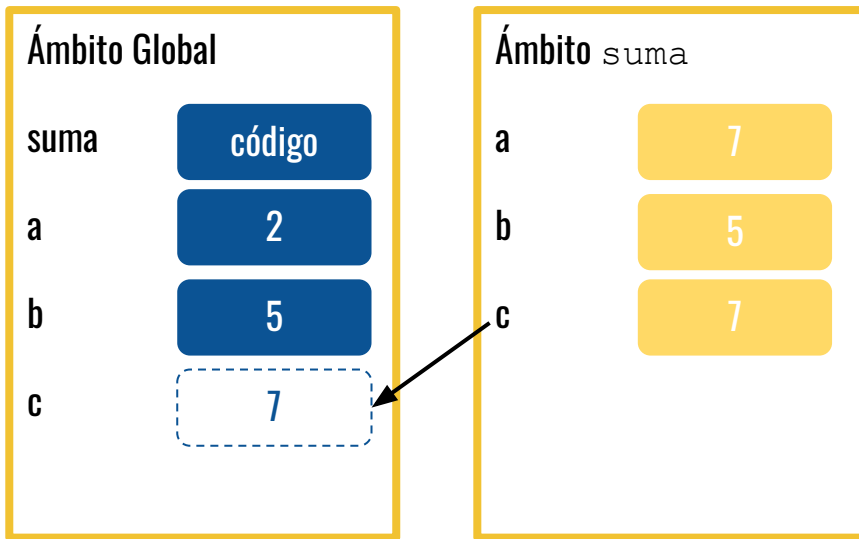
7

Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)

# Ámbito de las variables

```
def suma ( a, b ) :  
    a += b  
    c = a  
    return c  
  
a = 2  
b = 5  
c = suma ( a, b )
```



Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)

# Ámbito de las variables

```
def suma ( a, b ) :  
    a += b  
    c = a  
    return c  
  
a = 2  
b = 5  
c = suma ( a, b )
```

## Ámbito Global

suma

código

a

2

b

5

c

7

Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)

# Ámbito de las variables

```
def suma ( a, b ) :  
    a += b  
    c = a  
    return c  
  
a = 2  
b = 5  
c = suma ( a, b )
```

Ámbito Global	
suma	código
a	2
b	5
c	7

Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)



# Si la función no tuviera un statement de `return`

- Python retornará el valor `None` del tipo `NoneType` si no hay instrucción de `return`
- Esto representa la ausencia de valor

Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)

# return VS print

return	print
<ul style="list-style-type: none"><li>• <code>return</code> solo tiene significancia dentro de una función</li><li>• solo se ejecuta un <code>return</code> dentro de una función</li><li>• el código que esté dentro de una función, pero después de <code>return</code> ya no es ejecutado</li><li>• tiene un valor asociado, entregado como resultado a la llamada a la función</li></ul>	<ul style="list-style-type: none"><li>• <code>print</code> puede ser usado fuera de las funciones</li><li>• puede ejecutar varias instrucciones <code>print</code> dentro de una función</li><li>• el código dentro de una función puede ser ejecutado después de un <code>print</code></li><li>• tiene un valor asociado, impreso en la consola</li></ul>

Basado en: Ana Bell, Eric Grimson, and John Guttag. 6.0001 Introduction to Computer Science and Programming in Python. Fall 2016. Massachusetts Institute of Technology: MIT OpenCourseWare, <https://ocw.mit.edu>. License: Creative Commons BY-NC-SA.

[https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6\\_0001F16\\_Lec1.pdf](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-0001-introduction-to-computer-science-and-programming-in-python-fall-2016/lecture-slides-code/MIT6_0001F16_Lec1.pdf)

# Motivación

- ¿Cómo calculamos el logaritmo natural de un número?

# Motivación


- ¿Cómo calculamos el logaritmo natural de un número?

```
import math
x = 20
print(math.log(x))
```

# Motivación

- ¿Cómo calculamos el logaritmo natural de un número?

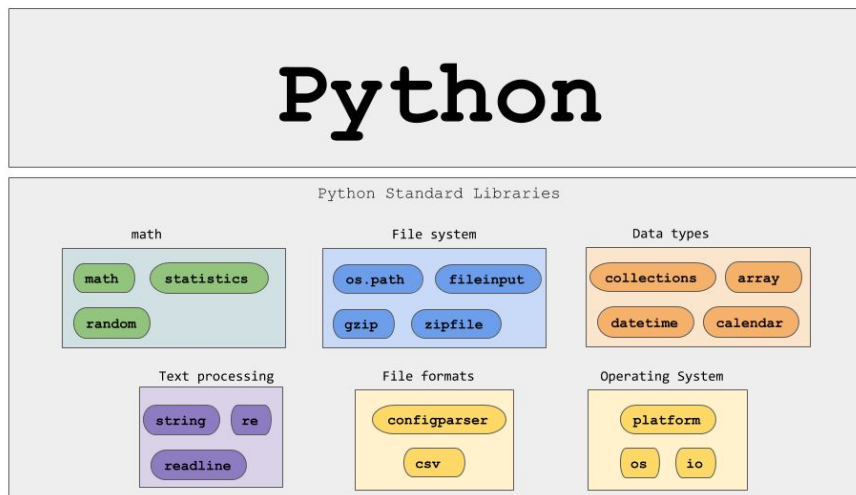
El paquete `math` es parte de la librería estándar de Python



```
import math
x = 20
print(math.log(x))
```

# Python: Librería Estándar

- Encapsula funcionalidades de: matemática, sistemas de archivos, tipos de datos, procesamiento de texto, formato de archivos y sistema operativo; extienden las funcionalidades básicas del lenguaje.



[https://ajaytech.co/  
what-are-python-libraries/](https://ajaytech.co/what-are-python-libraries/)

# Python: Librerías de terceros ('3rd party' )

- Agregan más funcionalidad, aunque su desarrollo está a cargo de terceros. Muchas de estas librerías tienen licencias open-source también

# Paquetes de Python vs. Módulos vs. Librerías

- Una librería/biblioteca es un término general que se refiere a un fragmento de código reutilizable. Por lo general, una librería/biblioteca de Python contiene una colección de módulos y paquetes relacionados. En realidad, este término a menudo se usa indistintamente con "paquete de Python" porque los paquetes también pueden contener módulos y otros paquetes (subpaquetes). Sin embargo, a menudo se supone que mientras que un paquete es una colección de módulos, una librería/biblioteca es una colección de paquetes.

<https://learnpython.com/blog/python-modules-packages-libraries-frameworks/#:~:text=Python%20Libraries&text=Actually%2C%20this%20term%20is%20often,is%20a%20collection%20of%20packages.>



# Importar un módulo de una librería

keyword



```
import math
```

```
x = 16
```

```
print(math.sqrt(x))
```

Note que para acceder a la función `sqrt` tenemos que poner la ruta completa `math.sqrt(...)`

# Importar un módulo de una librería con un alias

keyword



```
import math as m
```

```
x = 16
```

```
print(m.sqrt(x))
```

Note que a partir de la sentencia de importación del módulo, `m` es el alias.

# Importar elementos específicos de un módulo de una librería

keyword



```
from math import sqrt, cos
```

```
x = 16
```

```
y = 3.14
```

```
print(sqrt(x))
```

```
print(cos(y))
```

Note que ya no es necesario escribir `math.sqrt(...)` o `math.cos(...)`