FACULTAD DE
INGENIERÍA Y CIENCIAS

UNIVERSIDAD ADOLFO IBÁÑEZ

Introducción a Python

Introducción

Miguel Carrasco
miguel.carrasco@uai.cl

Profesor...

Miguel Carrasco

- International Engineering Educator ING.PAED.IGIP (IGIP, Austria), 2023
- Docteur en Informatique de la Université Pierre et Marie Curie, (Paris VI, Sorbonne University, Francia) (2010), *distinción máxima*
- Doctor en Ciencias de la Ingeniería (2010) y Magíster en Ingeniería (2008), Pontificia Universidad Católica de Chile.
- Ingeniero Civil en Informática y Magíster en Informática Universidad de Santiago de Chile (2004), *distinción máxima*

Áreas de trabajo

- Profesor Asociado de la Facultad de Ingeniería y Ciencias de la Universidad Adolfo Ibáñez, Chile.
- Sus áreas de interés son la inspección automática, visión por computador, procesamiento de imágenes, interfaz humano computador, Internet of Things.

Temario

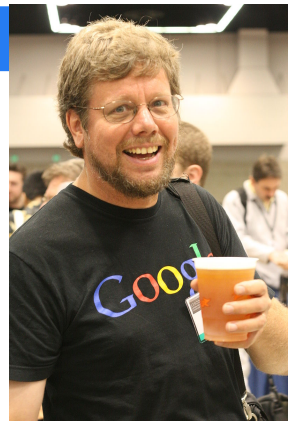
- Introducción a Python
  - Historia
  - Probando Python

Cronología

Guido Van Rossum

(creador de Python)

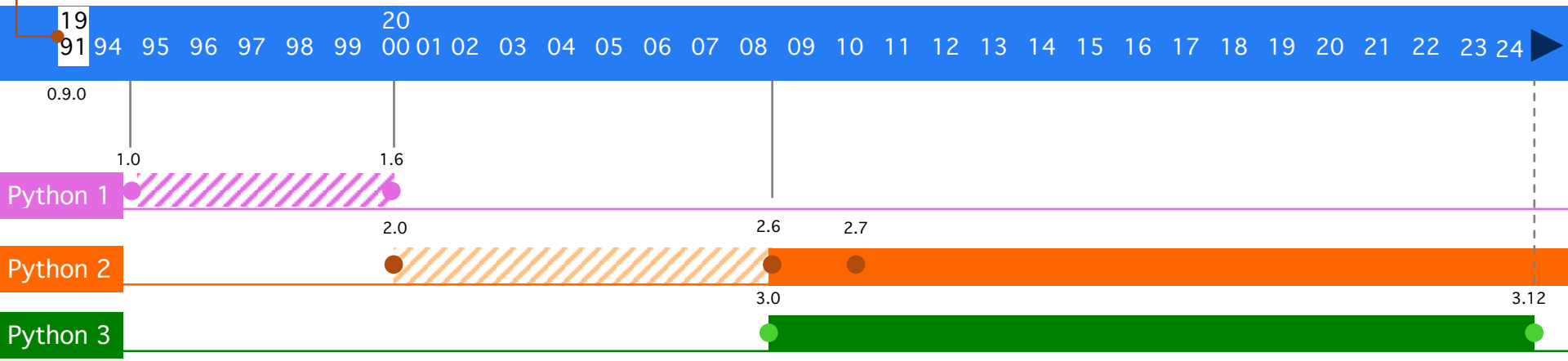
"es un científico de la computación conocido por ser el autor del lenguaje de programación Python en 1991" (fuente: Wikipedia)



Python es un lenguaje de propósito general y de tipo script. Es empleado para aplicaciones de interfaces, bases de datos, computación científica, aplicaciones web (client-server web programming) entre otras.

Principales características:

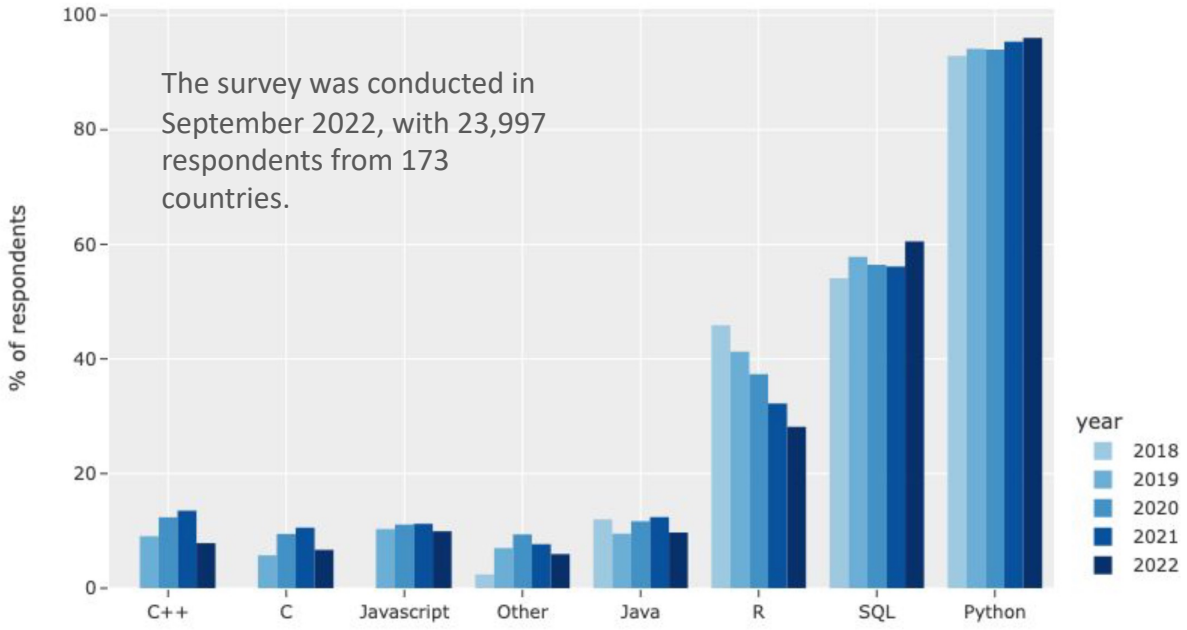
Facilidad de lectura (código interpretado)
Sistema de tipo dinámico
Manejo automático de administración de memoria



2022

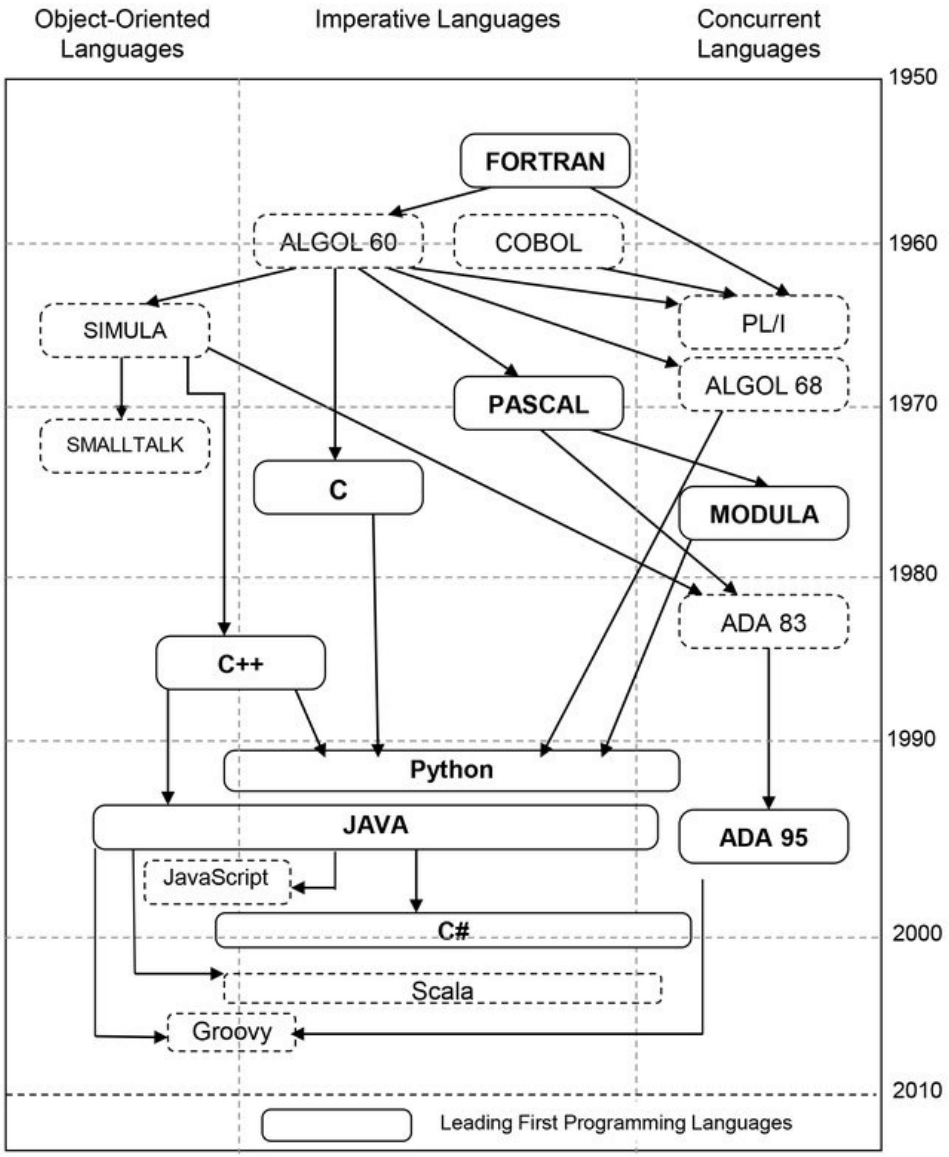
Kaggle DS & ML Survey 2022

Python and SQL remain the two most common programming skills for data scientists



fuente 1: https://storage.googleapis.com/kaggle-media/surveys/Kaggle%20State%20of%20Machine%20Learning%20and%20Data%20Science%20Report%202022.pdf

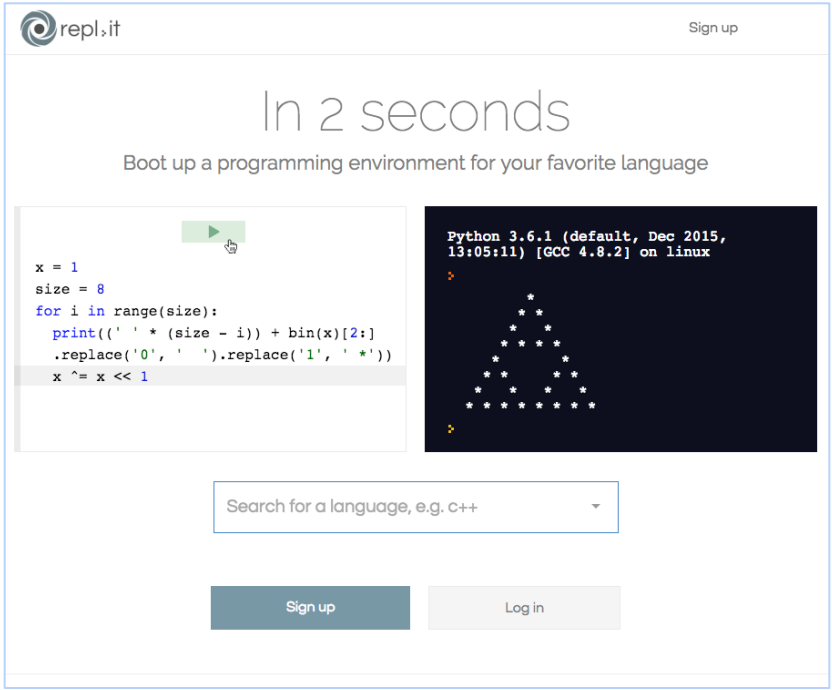
¿ Lenguajes ?



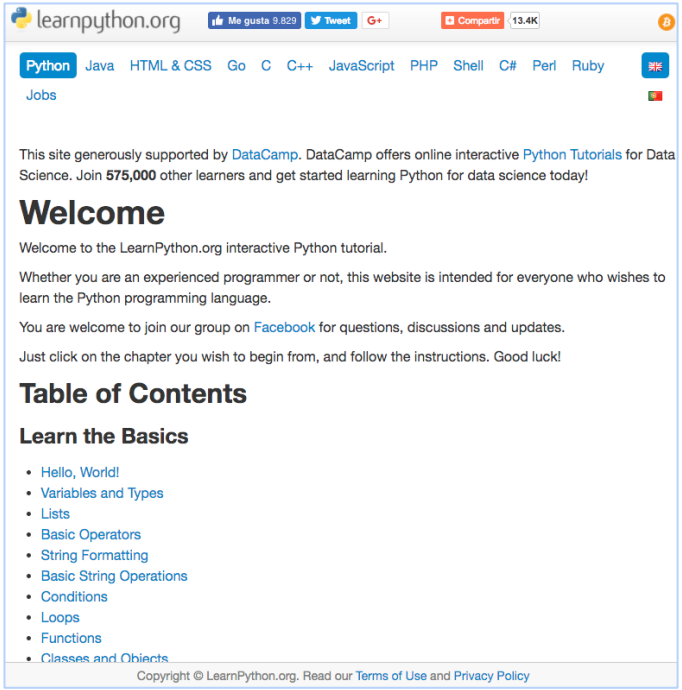
fuente: Farooq, Shoaib & Khan, Sher afzal & Ahmad, Farooq & Islam, Saeed & Abid, Adnan. (2014). An Evaluation Framework and Comparative Analysis of the Widely Used First Programming Languages. PloS one. 9. e88941. 10.1371/journal.pone.0088941.

online

- Ambientes de trabajo



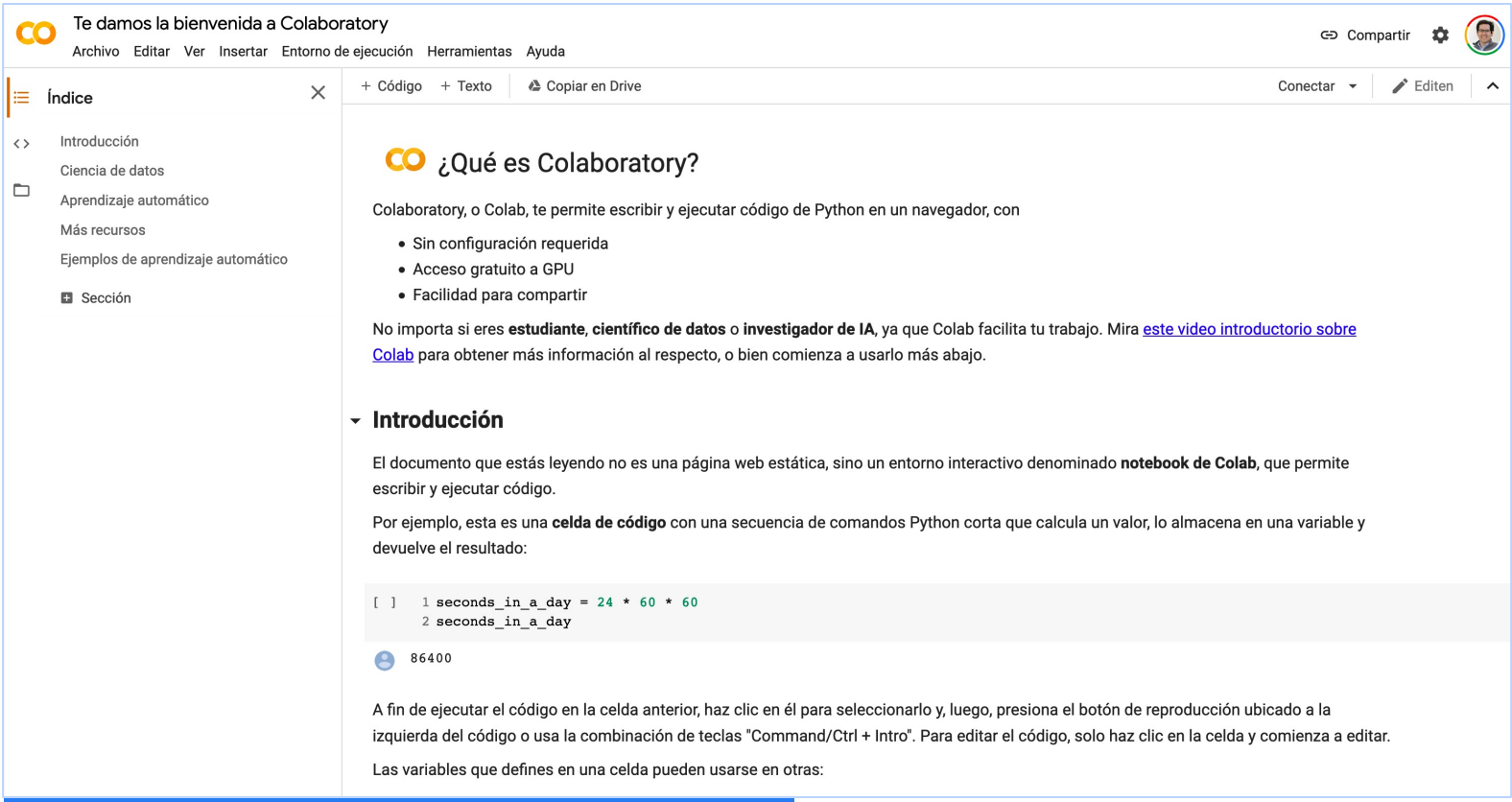
repl.it



www.learnpython.org

online

- Ambientes de trabajo



https://colab.research.google.com/

Instalación

offline

Instalación

FACULTAD DE INGENIERÍA Y CIENCIAS — UNIVERSIDAD ADOLFO IBÁÑEZ

offline

ANACONDA.

Products | Why Anaconda? | Solutions | Resources | Company | **Download**

Anaconda Distribution

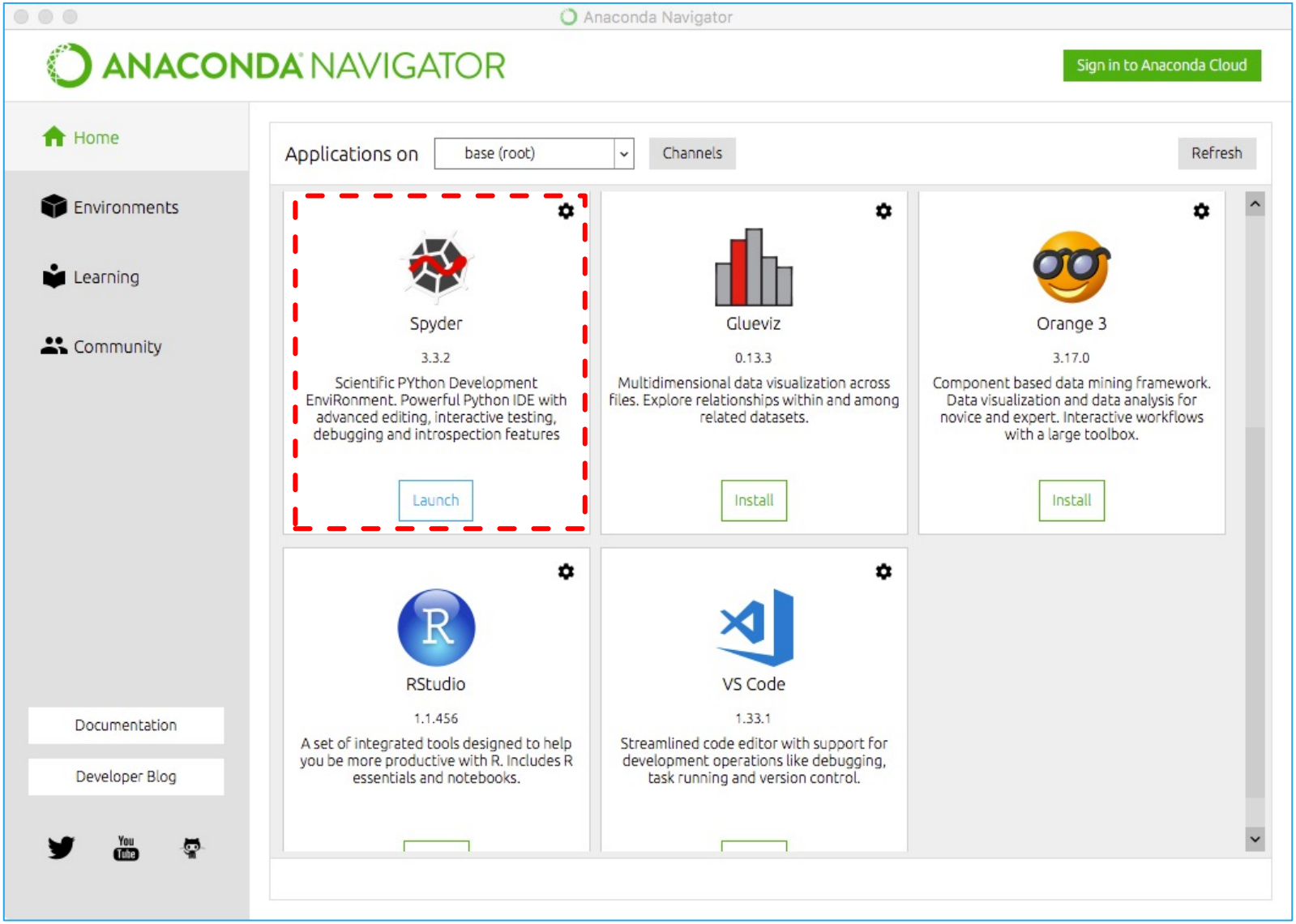
The World's Most Popular Python/R Data Science Platform

Download

The open-source Anaconda Distribution is the easiest way to perform Python/R data science and machine learning on Linux, Windows, and Mac OS X. With over 11 million users worldwide, it is the industry standard for developing, testing, and training on a single machine, enabling *individual data scientists* to:

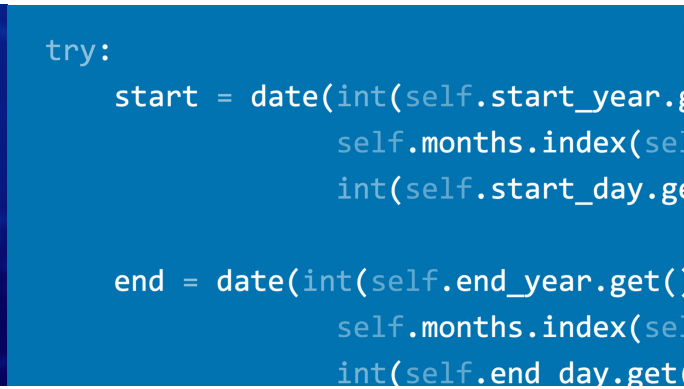
- Quickly download 1,500+ Python/R data science packages
- Manage libraries, dependencies, and environments with Conda
- Develop and train machine learning and deep learning models with scikit-learn, TensorFlow, and Theano
- Analyze data with scalability and performance with Dask, NumPy, pandas, and Numba
- Visualize results with Matplotlib, Bokeh, Datashader, and Holoviews

Spyder

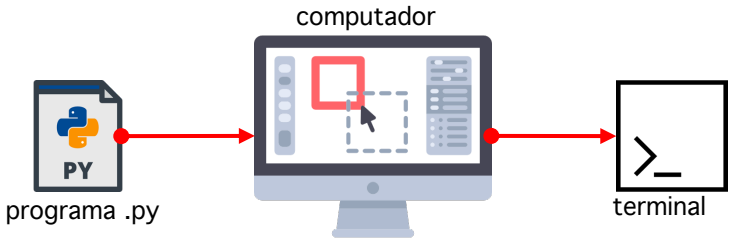

Anaconda Navigator
ANACONDA NAVIGATOR
Sign in to Anaconda Cloud
Home
Environments
Learning
Community
Documentation
Developer Blog
Applications on
base (root)
Channels
Refresh
Spyder
3.3.2
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features
Launch
Glueviz
0.13.3
Multidimensional data visualization across files. Explore relationships within and among related datasets.
Install
Orange 3
3.17.0
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.
Install
RStudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.
VS Code
1.33.1
Streamlined code editor with support for development operations like debugging, task running and version control.

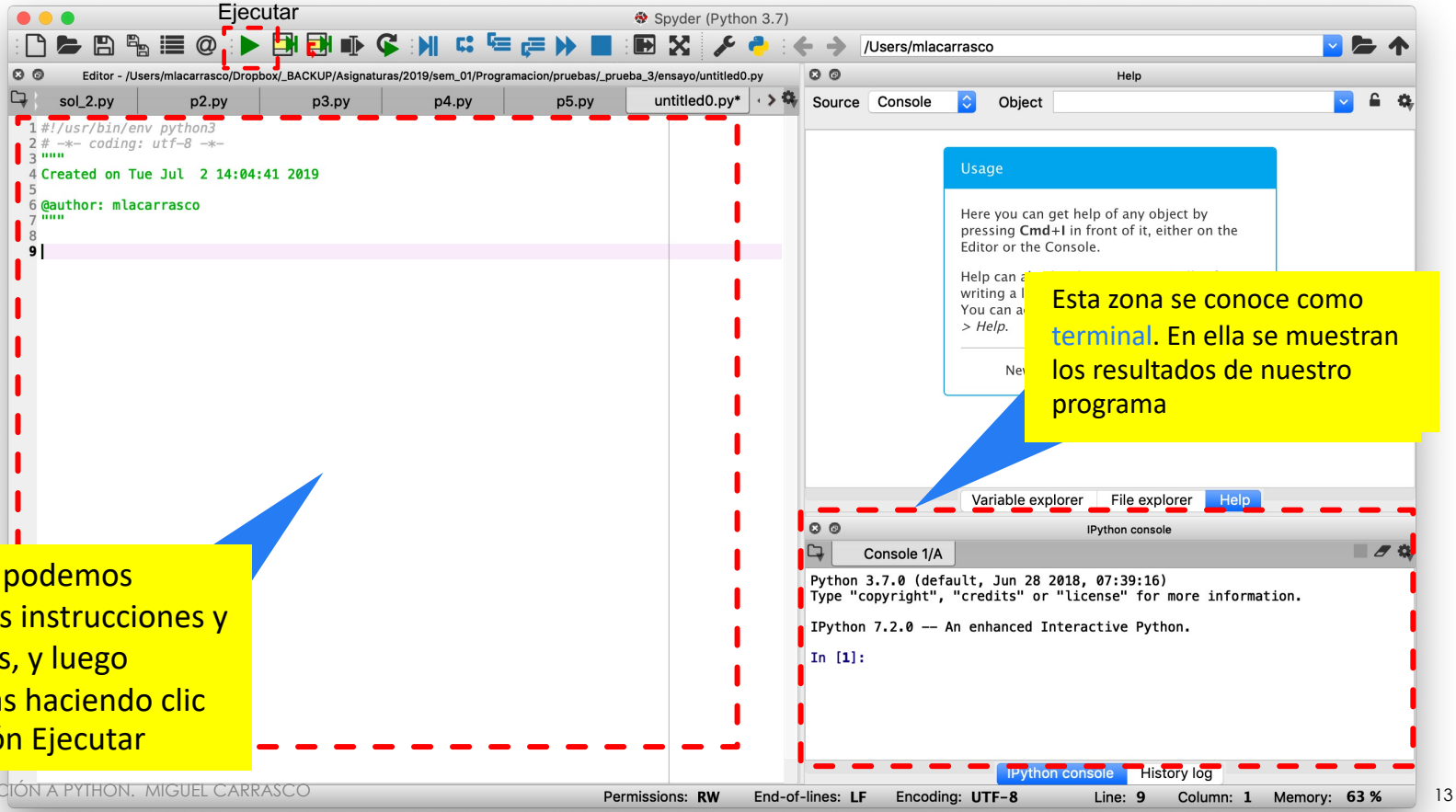
Temario

- Introducción a Python
- Primeros pasos
  - Lección 01: Despliegue de datos



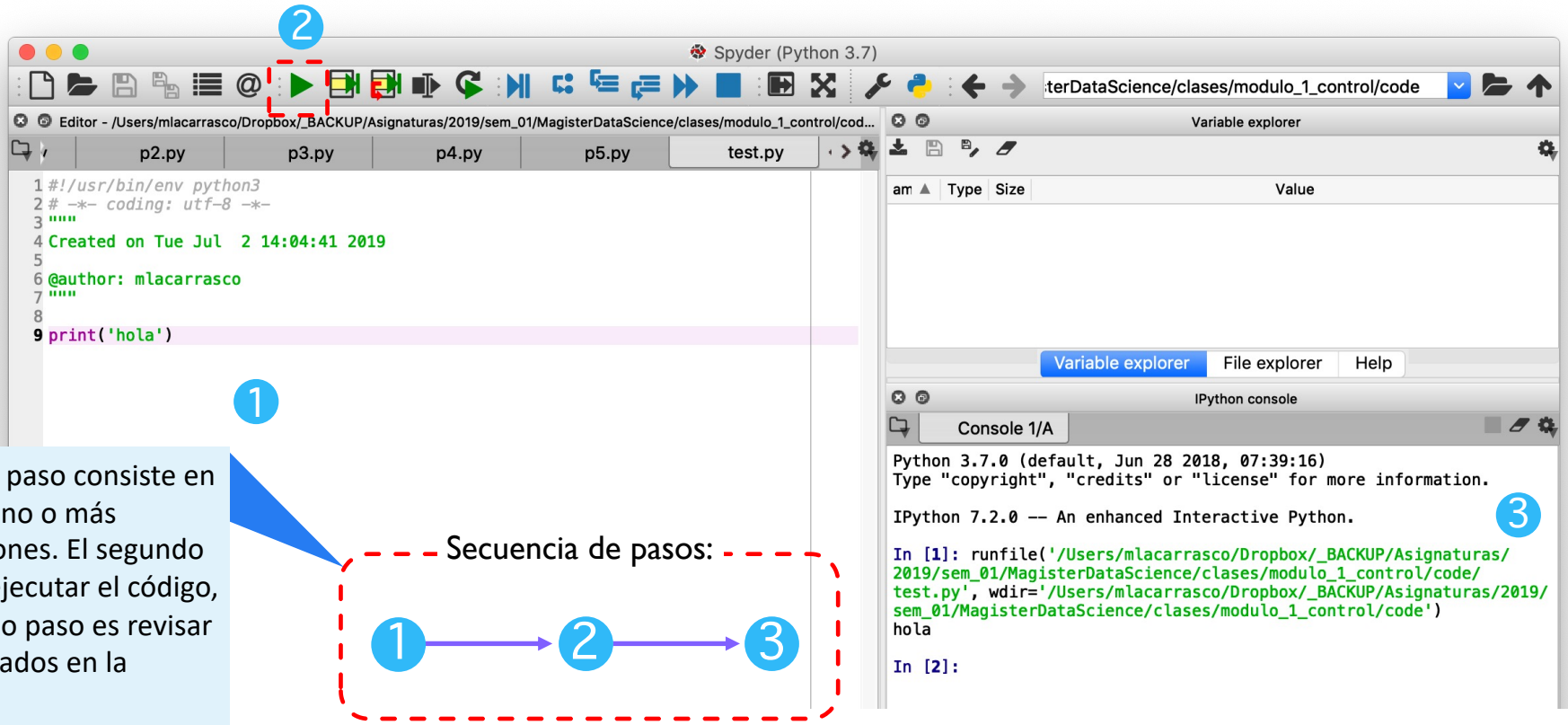
Un programa en Python está conformado por líneas de texto que contienen una o más instrucciones y sentencias del lenguaje





print()

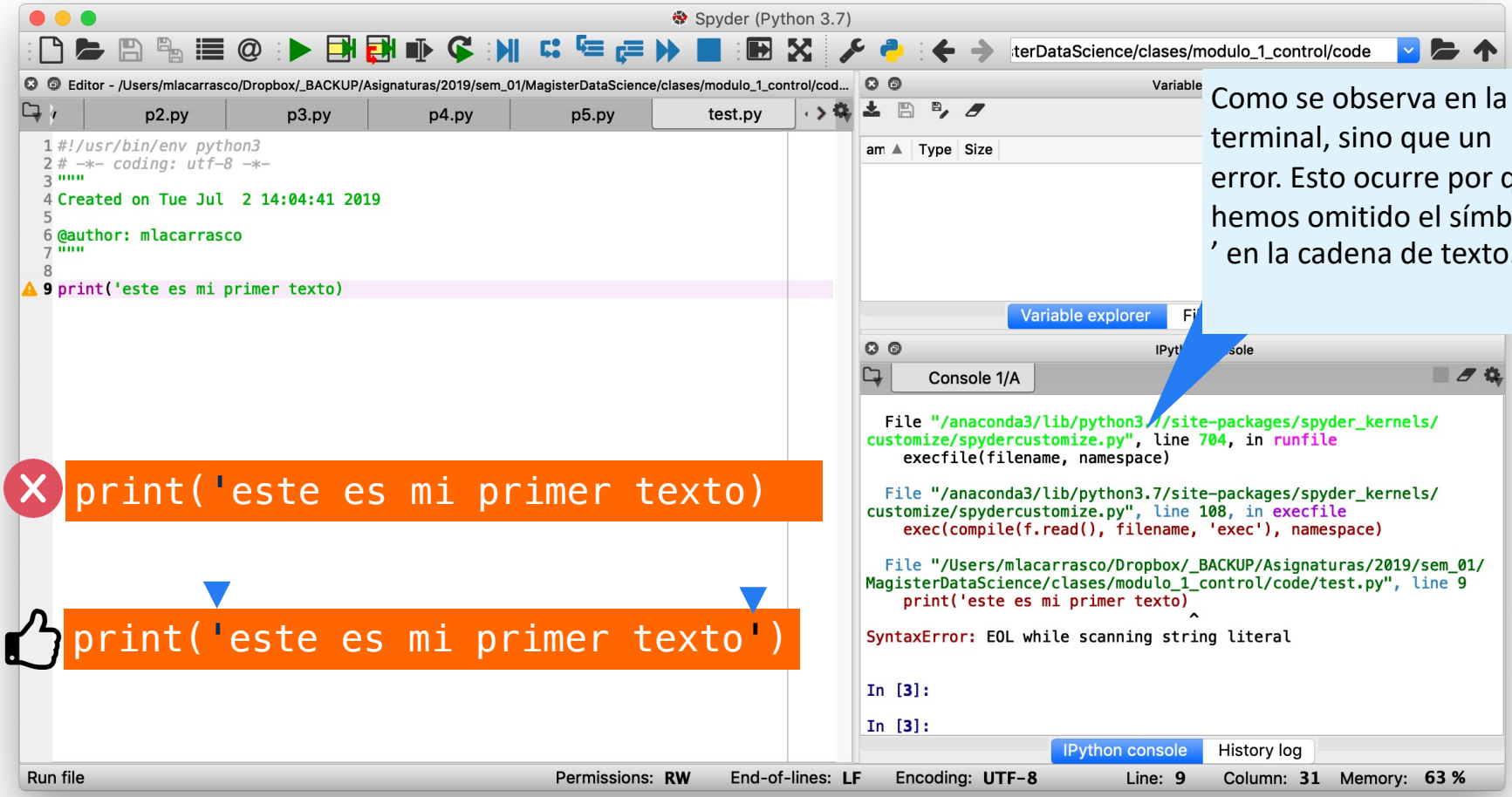
El comando print() permite desplegar el resultado de un variable o un texto por la terminal. Si escribimos texto, éste debe ir escrito entre comillas. Si escribimos una variable, éste despliega el valor almacenado en dicha variable.



El primer paso consiste en escribir uno o más instrucciones. El segundo paso es ejecutar el código, y el último paso es revisar los resultados en la terminal

print()

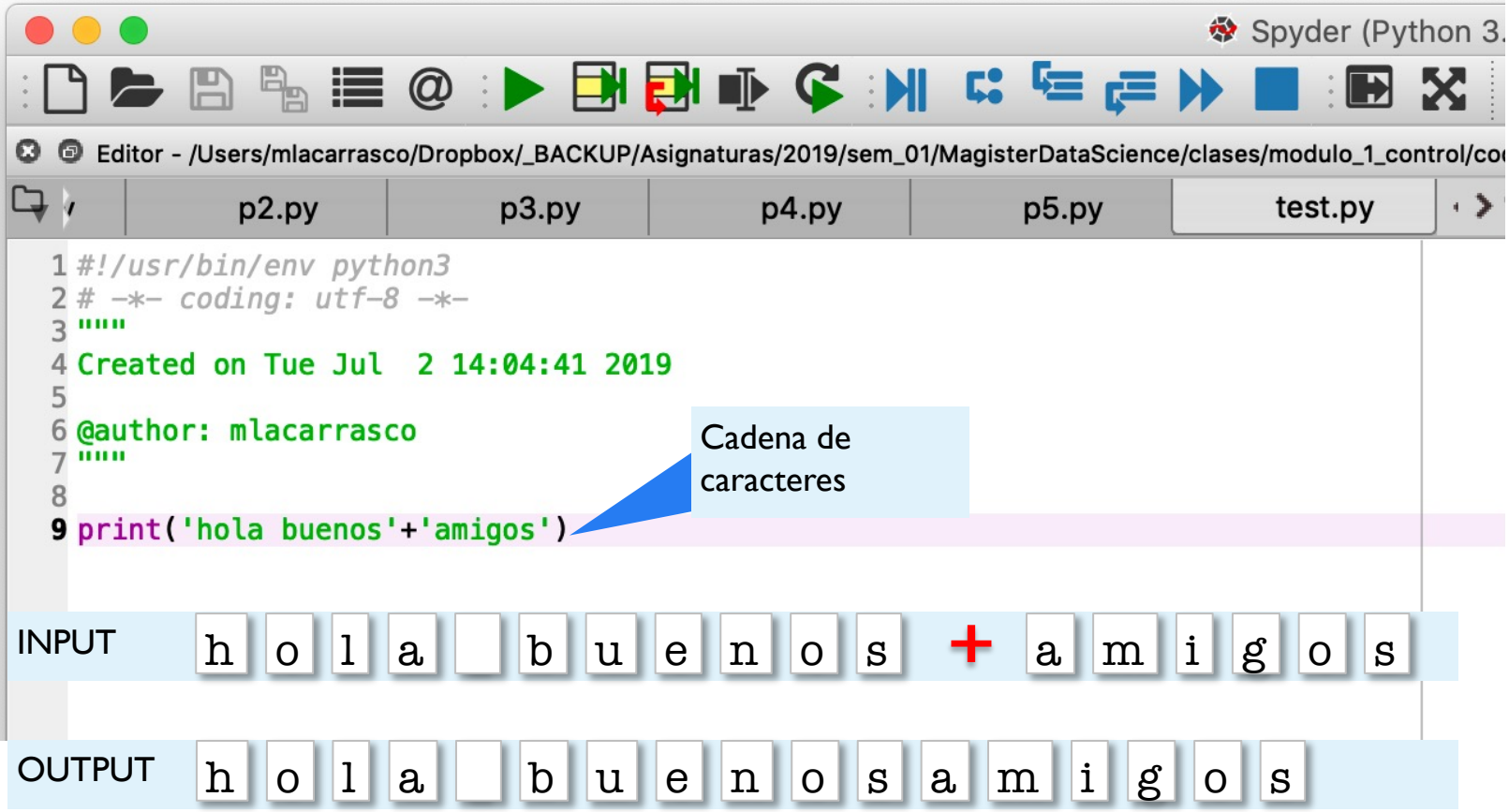
Si escribimos texto, éste debe ir escrito entre comillas. En caso contrario, el computador arrojará un error ya que no sabrá donde termina la cadena de texto.



Como se observa en la terminal, sino que un error. Esto ocurre por que hemos omitido el símbolo ’ en la cadena de texto.

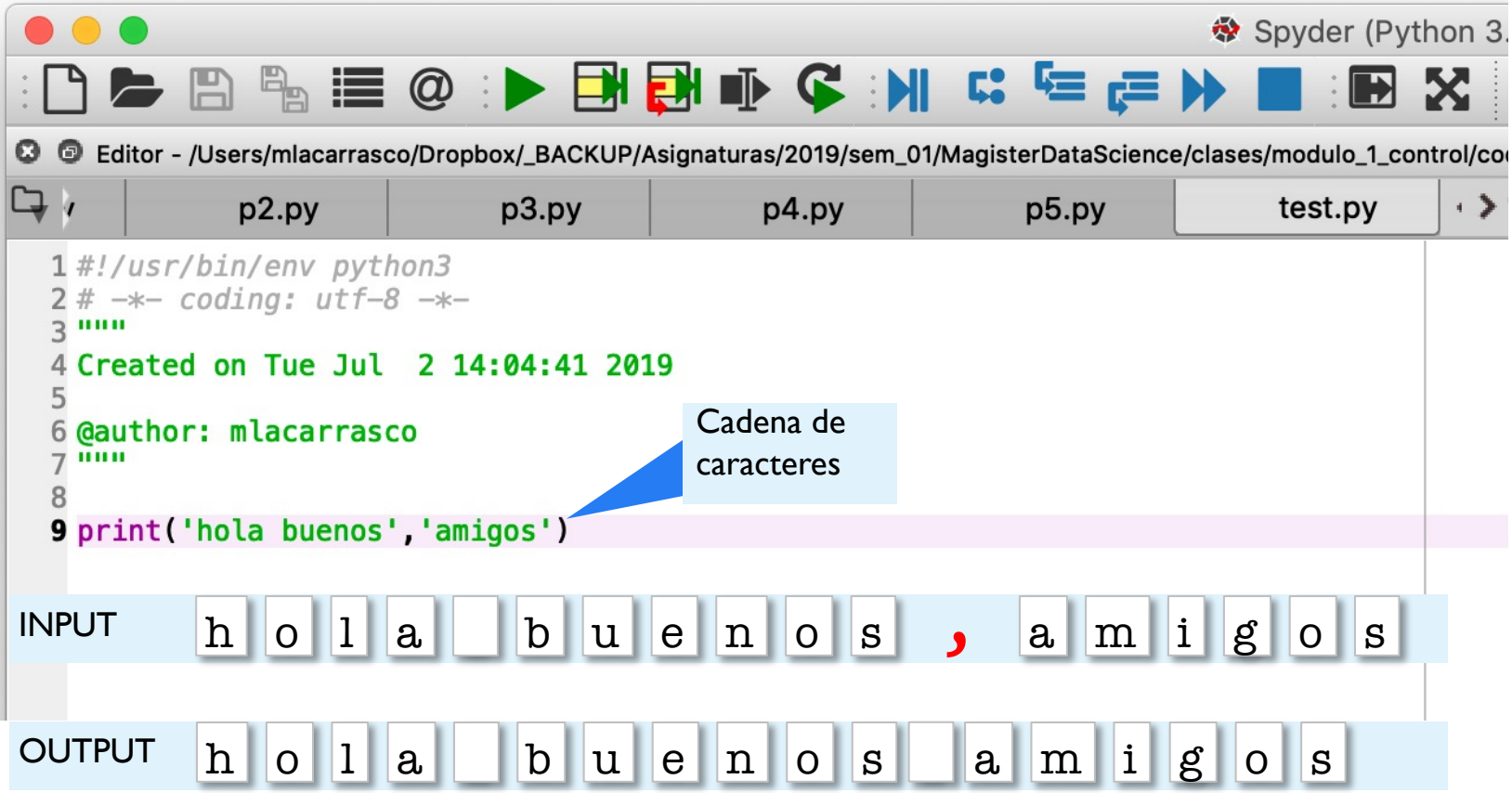
print()

Para unir dos cadenas de texto, podemos emplear el símbolo +
Es importante notar que une directamente las cadenas sin dejar un espacio en blanco entre éstas



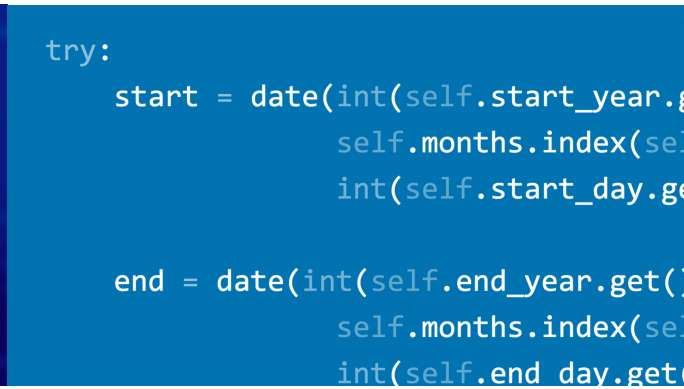
print()

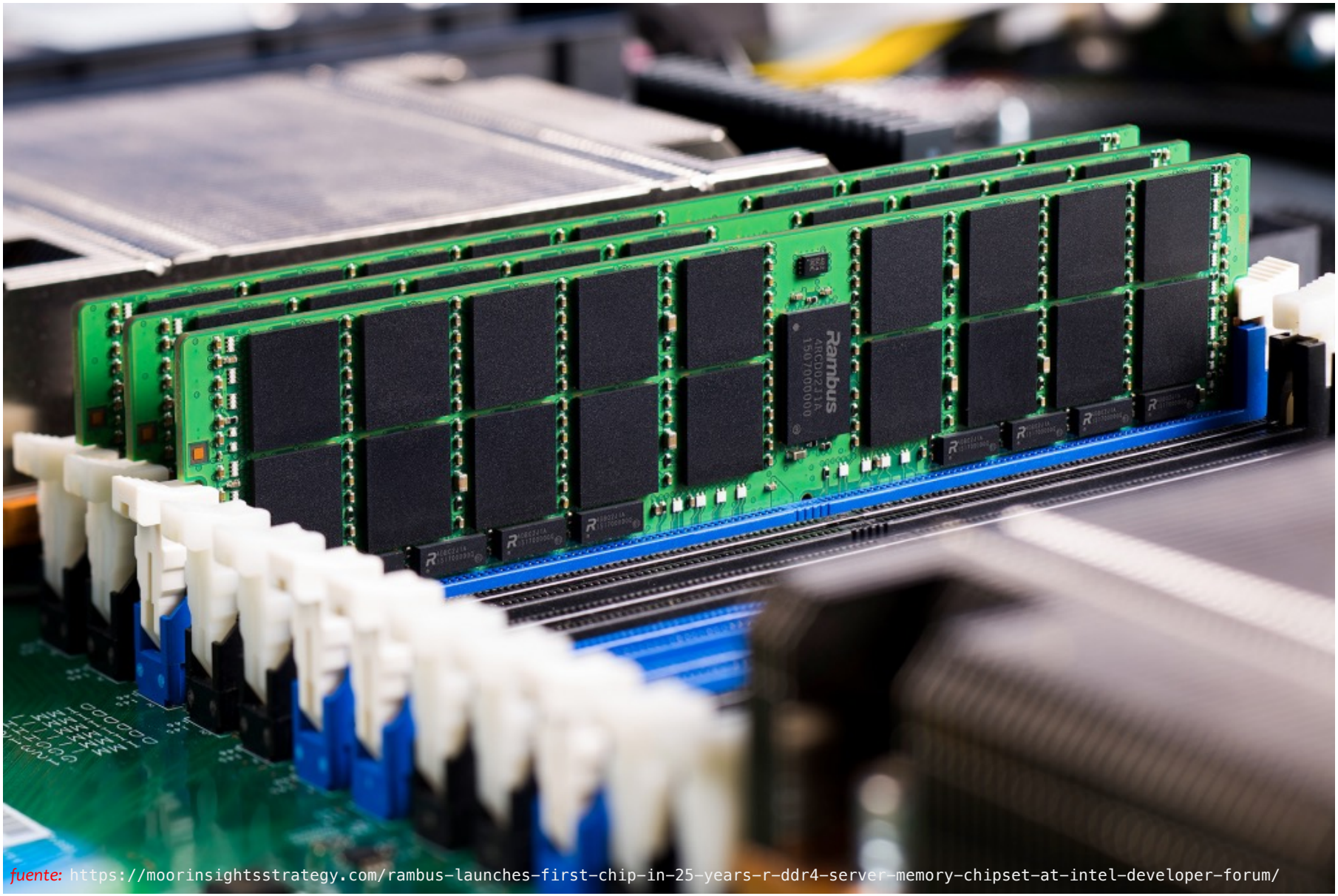
Para imprimir más de una cadena de caracteres, o mezclar cadenas de caracteres con números u otro tipo de datos, podemos hacerlo usando print separando los objetos por coma



Temario

- Introducción a Python
- Primeros pasos
  - Lección 01: Despliegue de resultados
  - Lección 02: Variables





fuente: https://moorinsightsstrategy.com/rambus-launches-first-chip-in-25-years-r-ddr4-server-memory-chipset-at-intel-developer-forum/

- Creación de variables

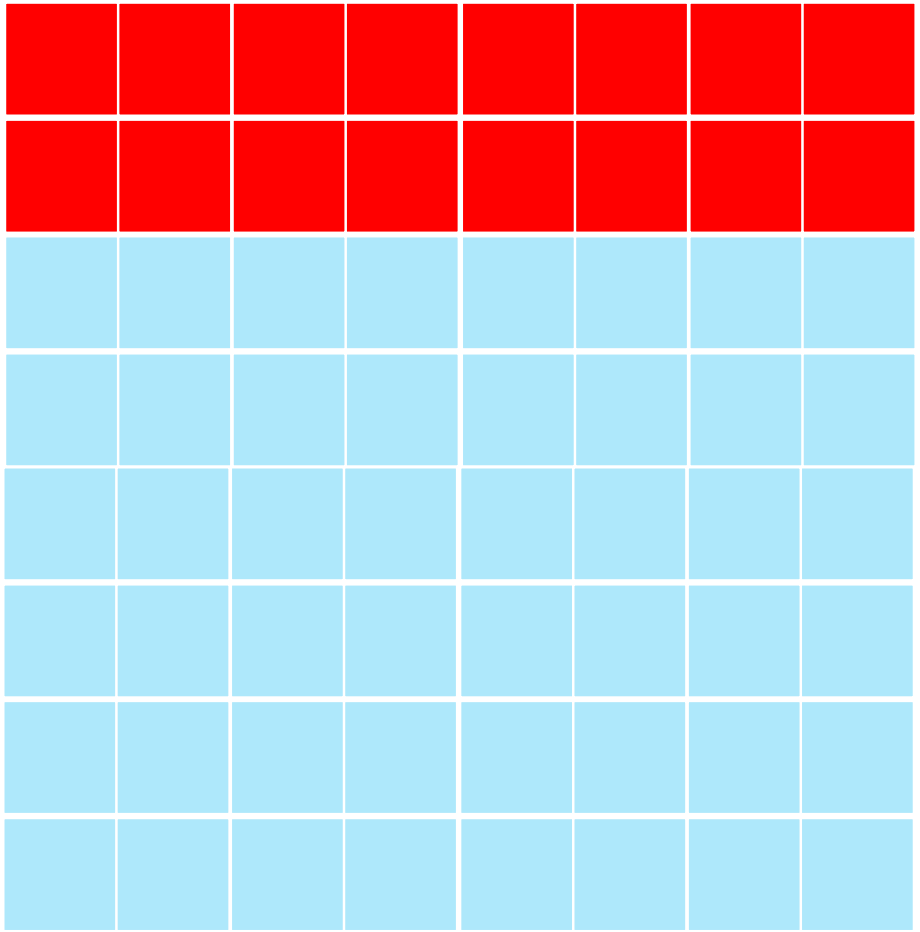
vista de bloques en la memoria del computador

Supongamos que la memoria que tenemos dispone de 64 bloques.

Cada bloque puede guardar un número de 8 bits.

¿Cuantos bytes podemos almacenar?

- Creación de variables

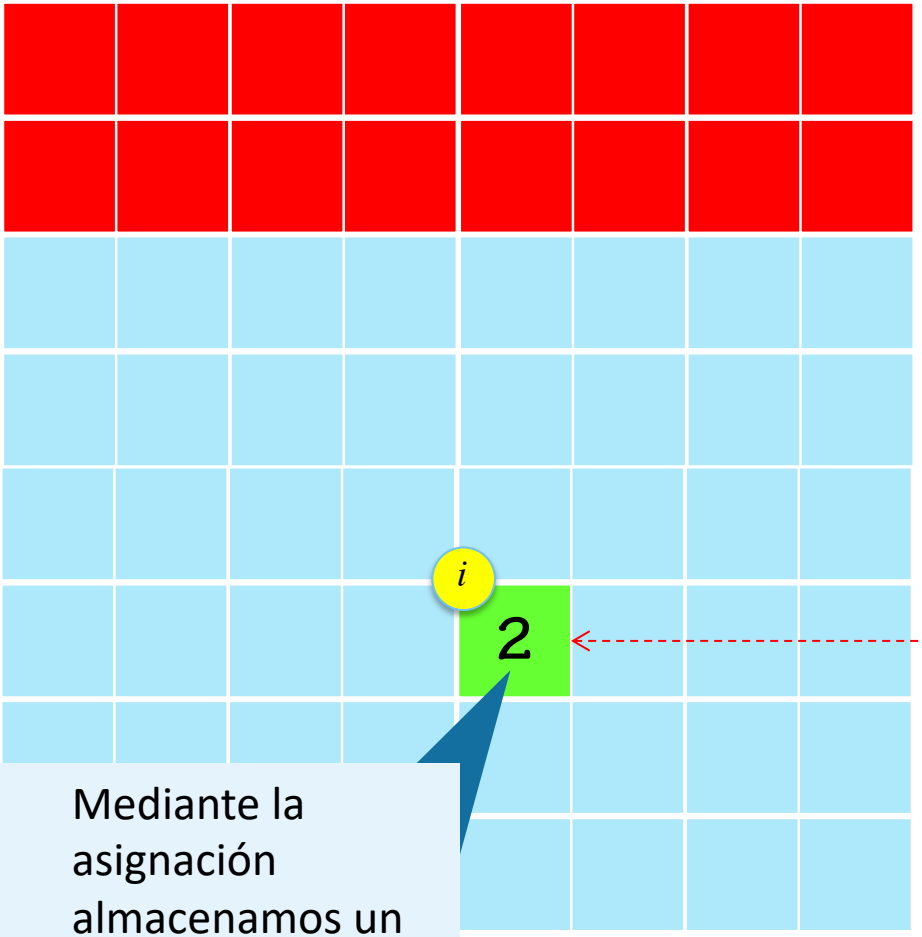


vista de bloques en la memoria del computador

Cuando encedemos el computador, el sistema operativo utiliza parte de la memoria RAM

Lo mismo sucede con los programas que empleamos

- Creación de variables




```
i = 2
```

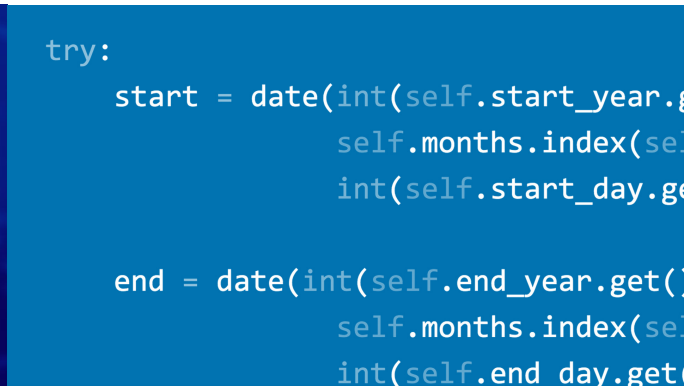
operador asignación

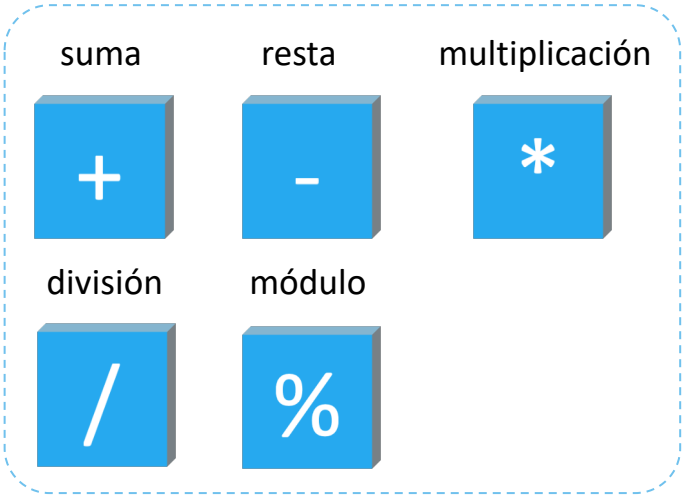
Mediante la asignación almacenamos un valor en la memoria

La asignación es siempre del lazo derecho al izquierdo

Temario

- Introducción a Python
- Primeros pasos
  - Lección 01: Despliegue de resultados
  - Lección 02: Variables
  - Lección 03: Operaciones aritméticas





Operaciones aritméticas

Python permite efectuar las siguientes operaciones aritméticas

1
```
w = 100

i = 2 + w
```

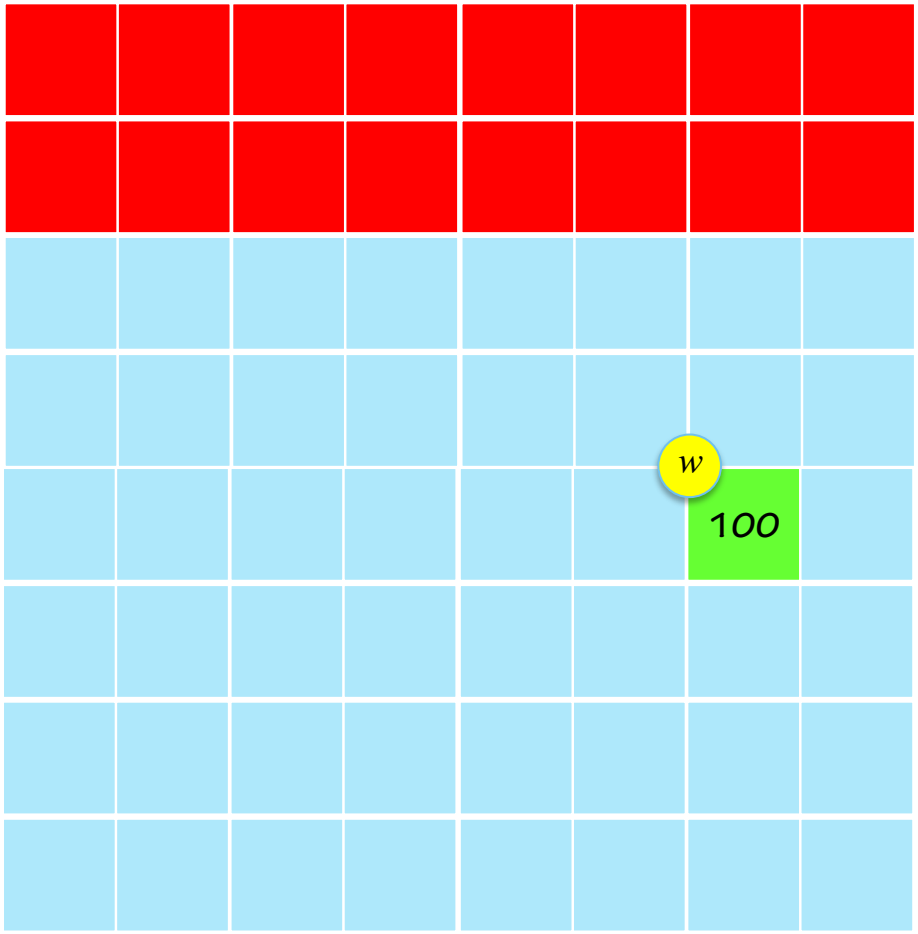
2
```
w = 100

i = (2 + w)
i = i - 3
```

3
```
w = 100

i = (w - 98) * (w  * 2)
```

4
```
w = 100

i = 2
w = w / 2
i = w - i
```

```
w = 100
```

1

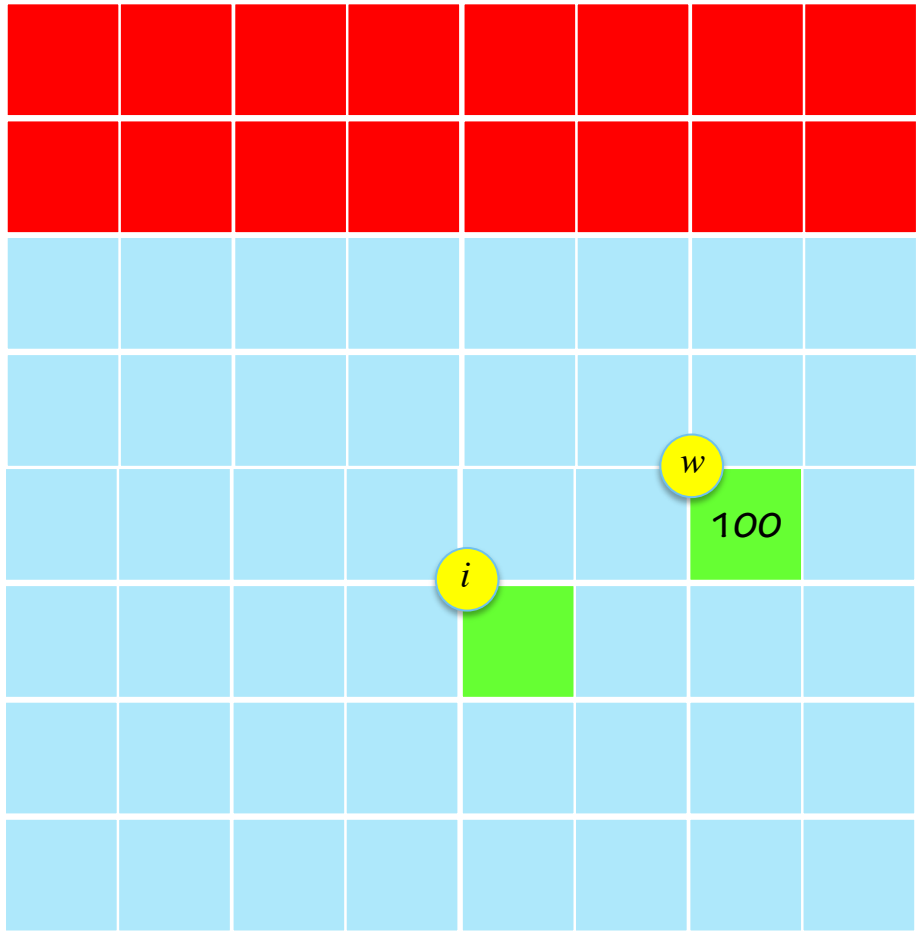


Toda variable tiene un tipo de datos definido. Si empleamos varias variables del mismo tipo solo debemos separarlas por comas.

```
w = 100

i = 2 + w
```

1

2 + 100



Siempre debe calcularse (evaluarse) la parte derecha de la asignación y luego se almacena el resultado al lado izquierdo

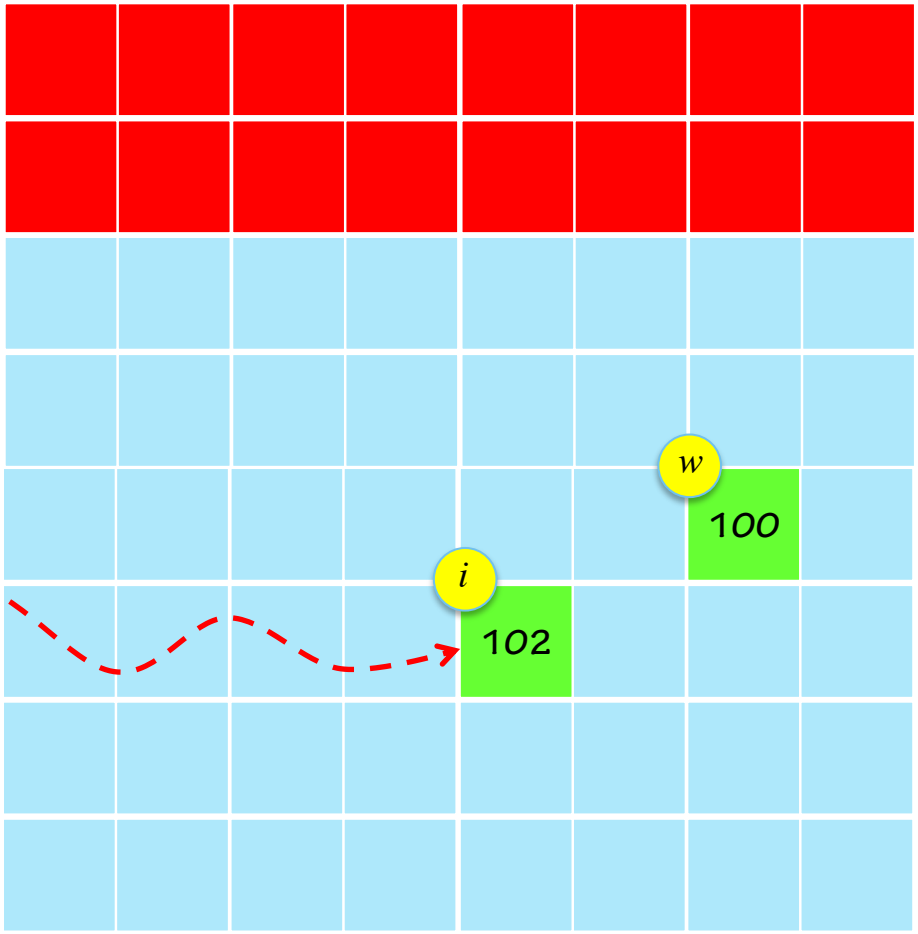
```
w = 100

i = 2 + w
```

1

2 + 100

102

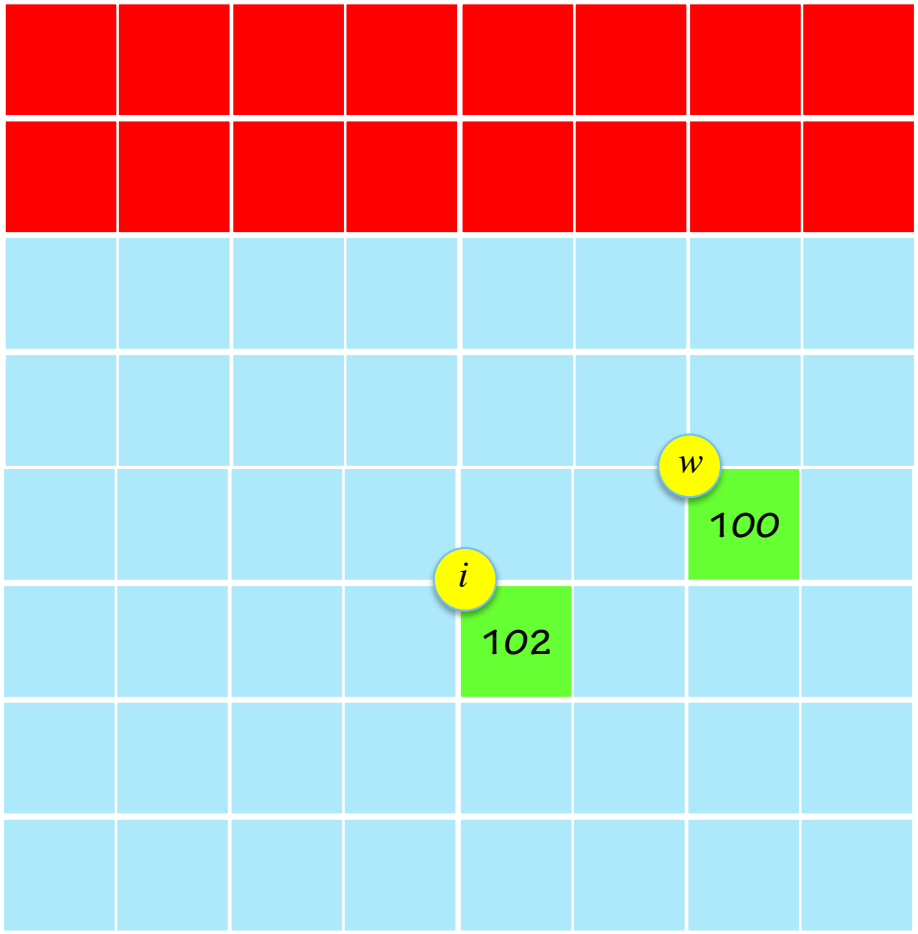
Siempre debe calcularse (evaluarse) la parte derecha de la asignación y luego se almacena el resultado al lado izquierdo



```
w = 100

i = 2 + w

i = i - 3
```

2

102 - 3

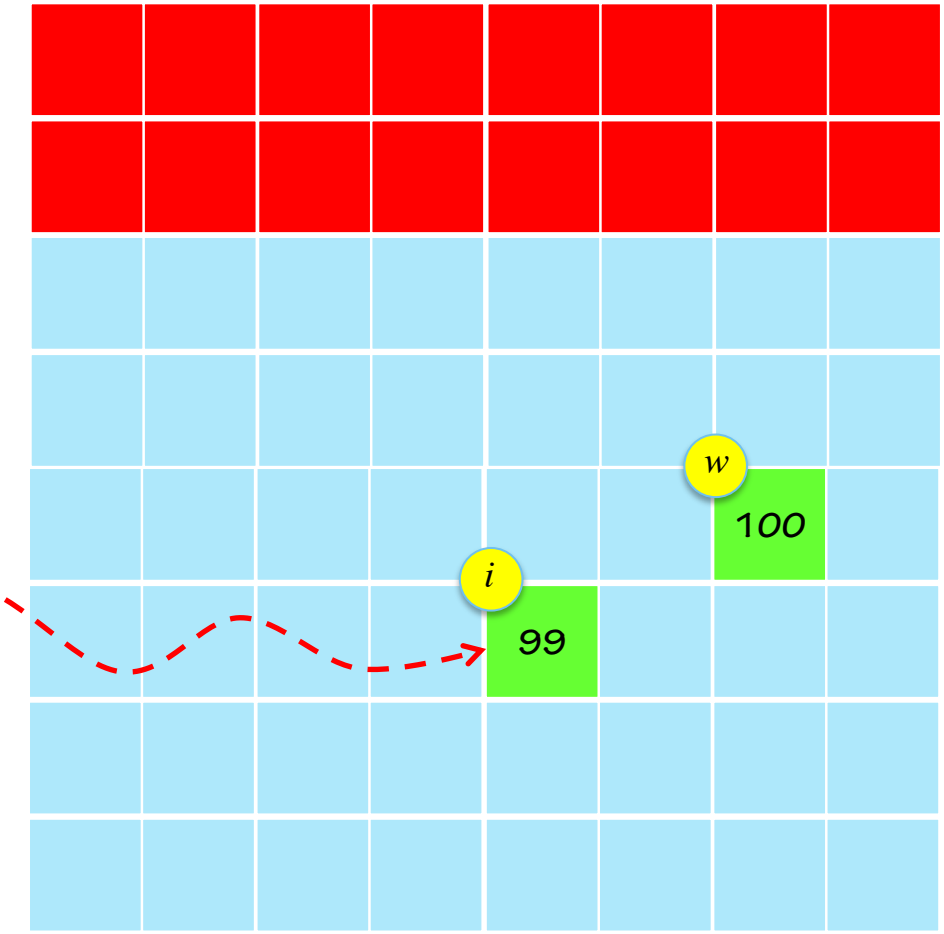


```
w = 100

i = 2 + w

i = i - 3
```

2

102 - 3

99

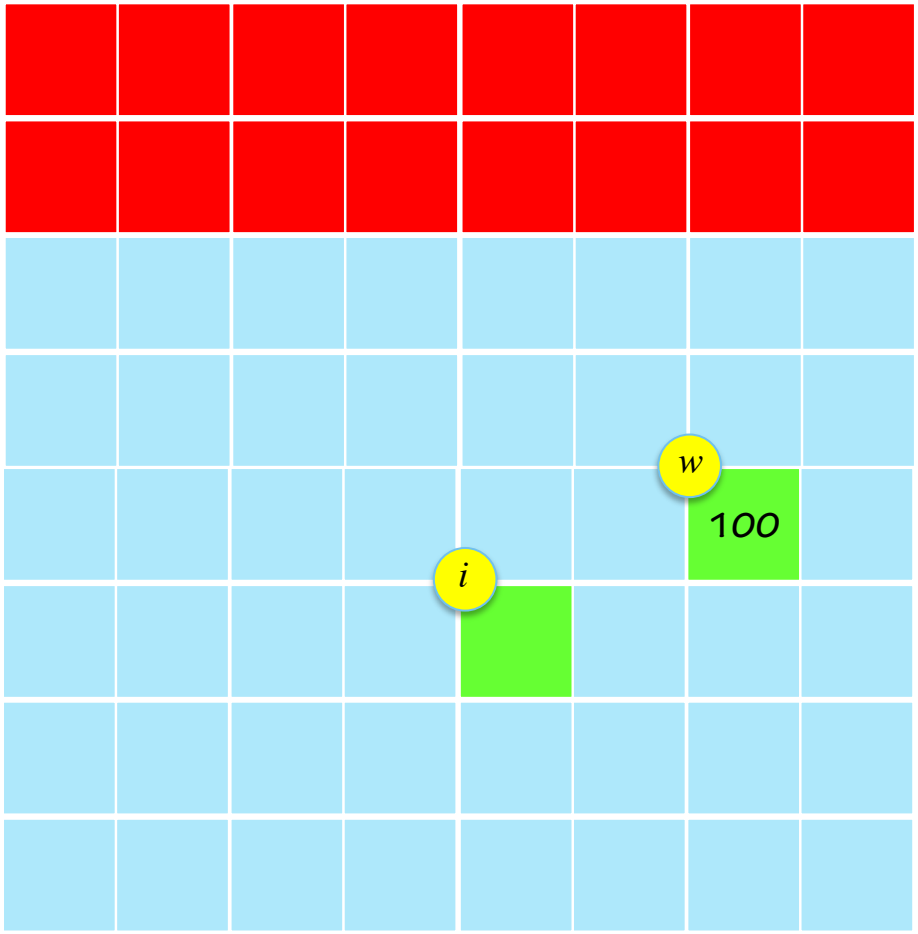


Si observamos la variable i, ésta no ha modificado su valor hasta que se haya asignado algo en ella

```
w = 100

i = (w - 98) * (w  * 2)
```

3

2 * 200



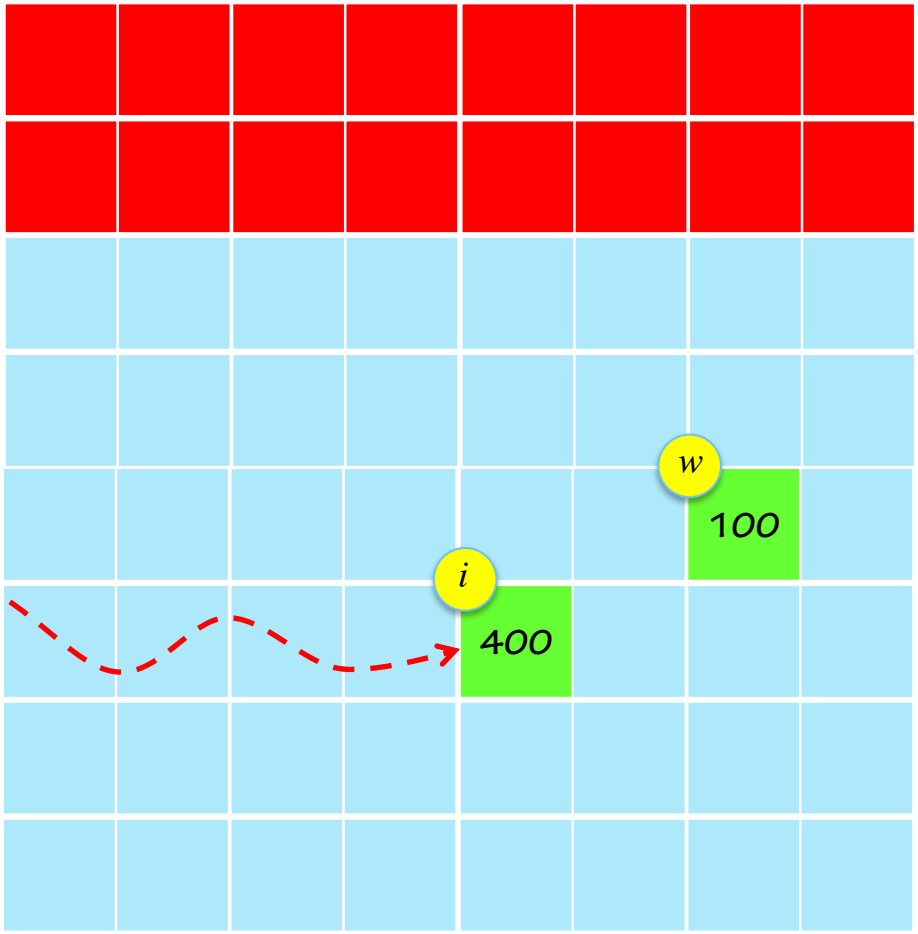
```
w = 100
i = (w - 98) * (w  * 2)
```

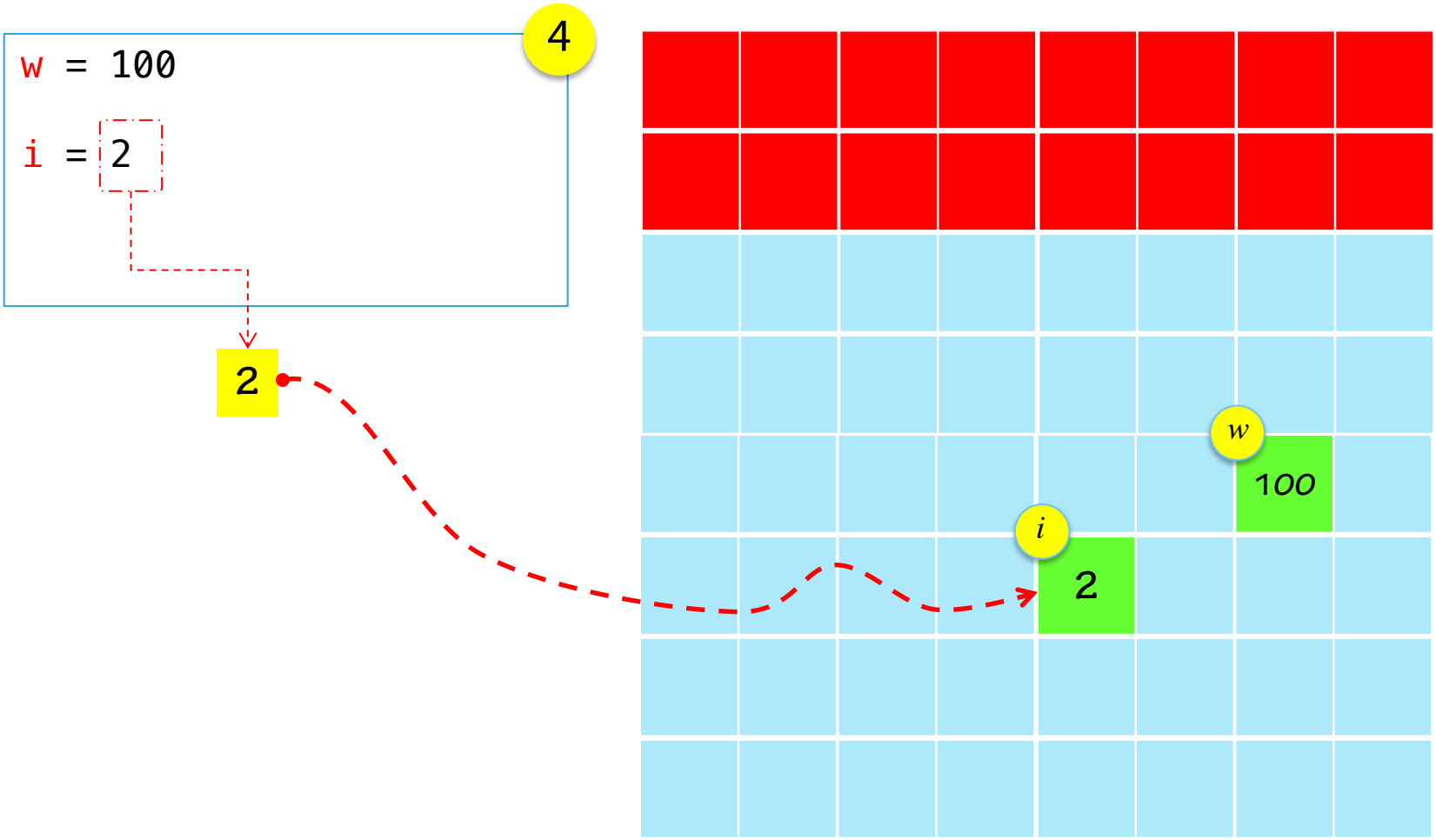
3

2 * 200

400

Recuerde, siempre debe terminar de calcular las operaciones y luego se asigna a la variable definida.

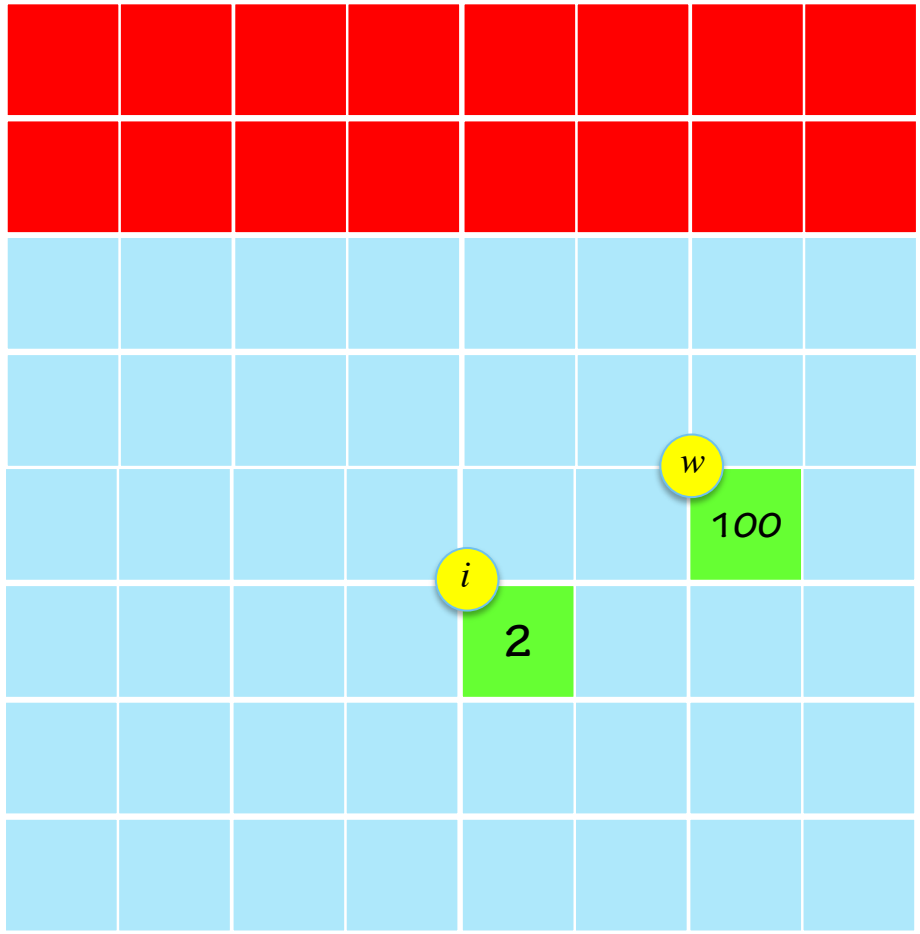



4
w = 100
i = 2
2
w
100
i
2

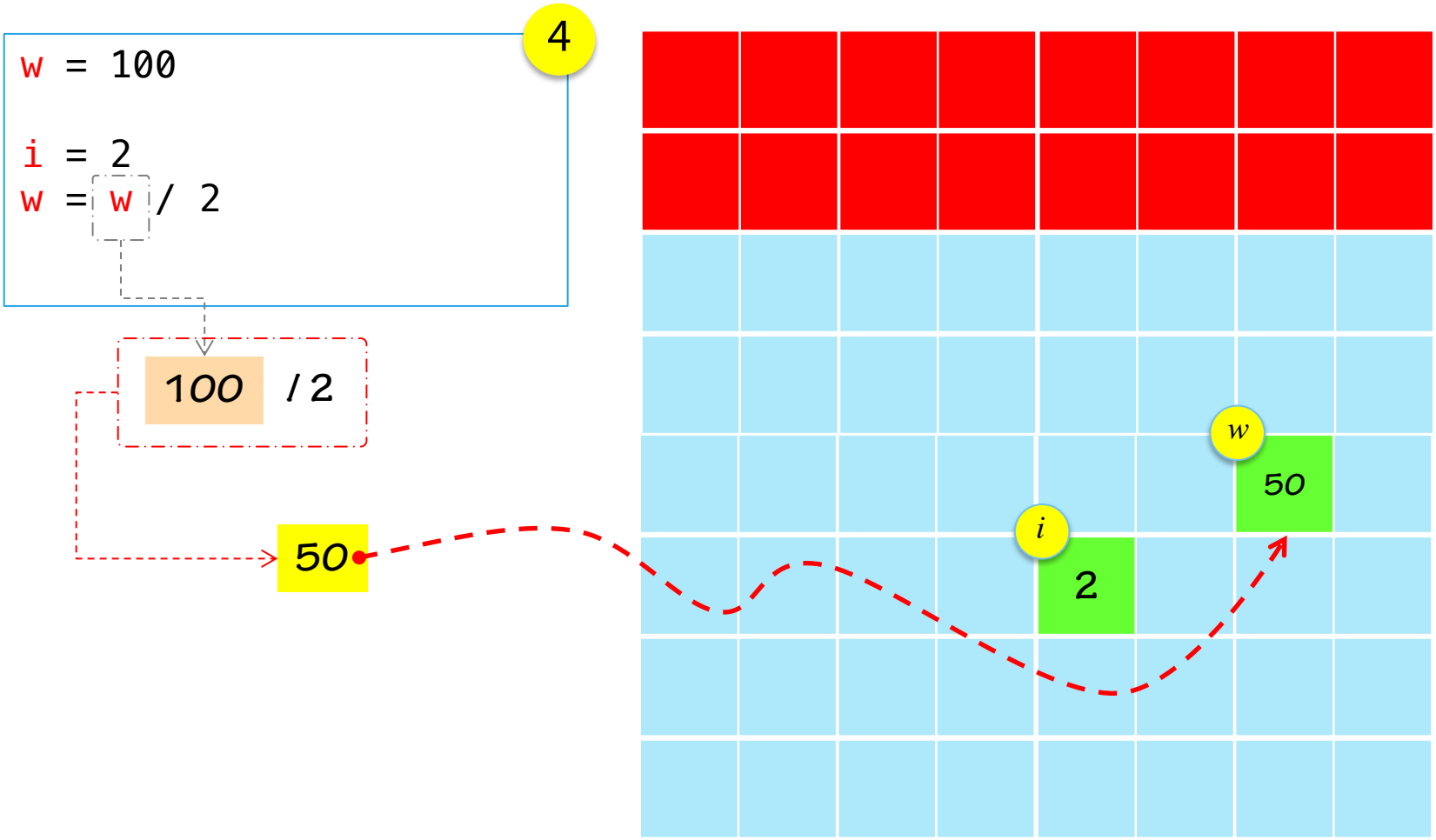
```
w = 100

i = 2
w = w / 2
```

4

100 / 2



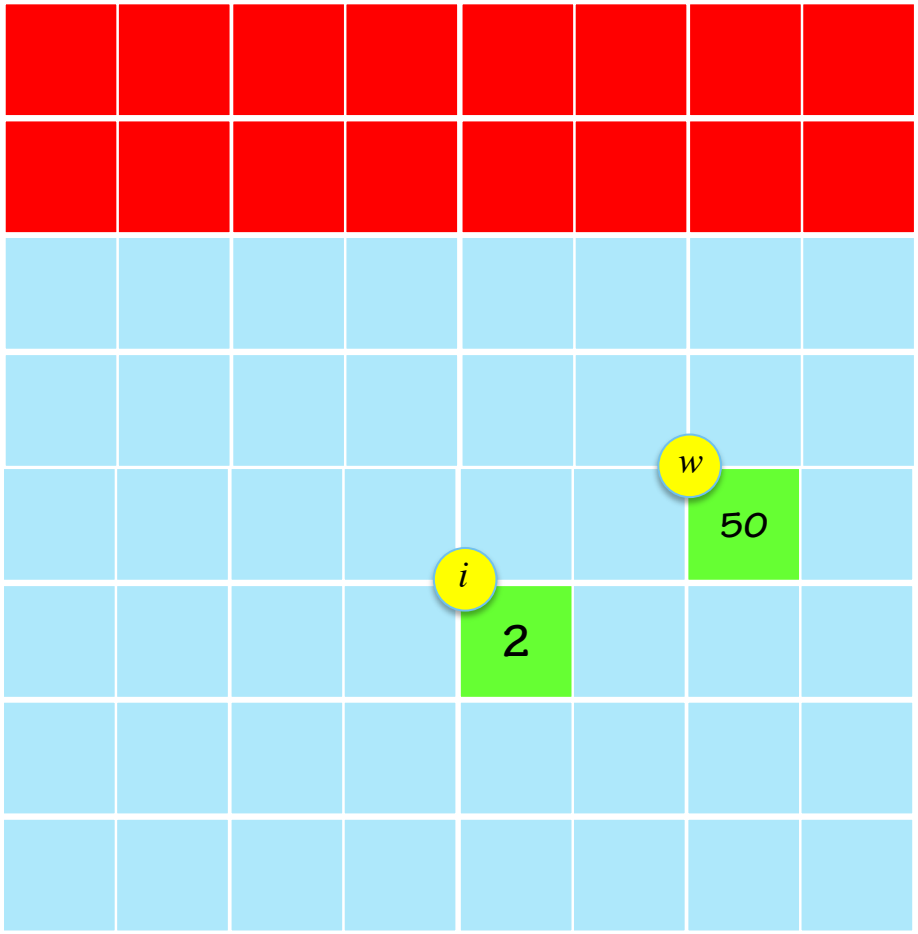
Recuerde, siempre debe terminar de calcular las operaciones y luego se asigna a la variable definida.


4
w = 100
i = 2
w = w / 2
100 / 2
50
w
50
i
2

```
w = 100

i = 2
w = w / 2
i = w - i
```

4

50 - 2

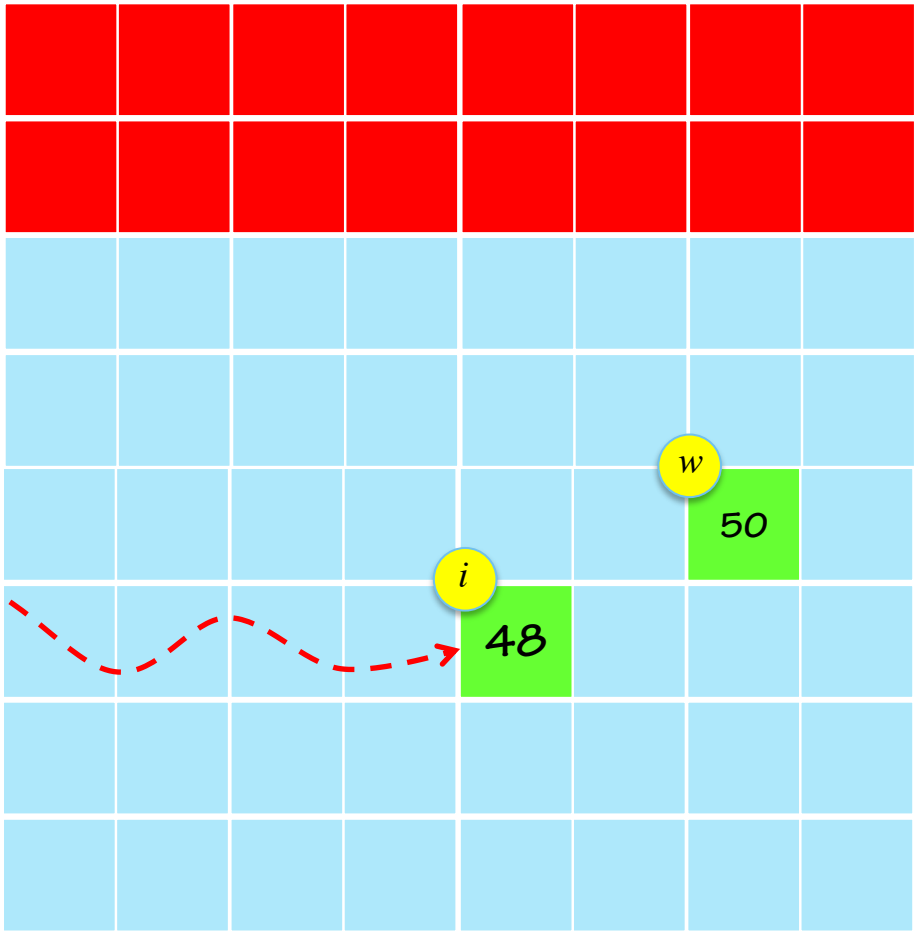


```
w = 100

i = 2
w = w / 2
i = w - i
```

4

50 – 2

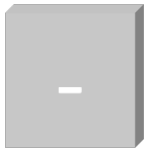
48



Recuerde, siempre debe terminar de calcular las operaciones y luego se asigna a la variable definida.

suma

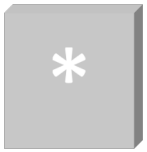
resta



división

módulo

multiplicación



$$a\%b = m \rightarrow a = b \times k + m$$

$$11 = 2 \times 5 + 1$$

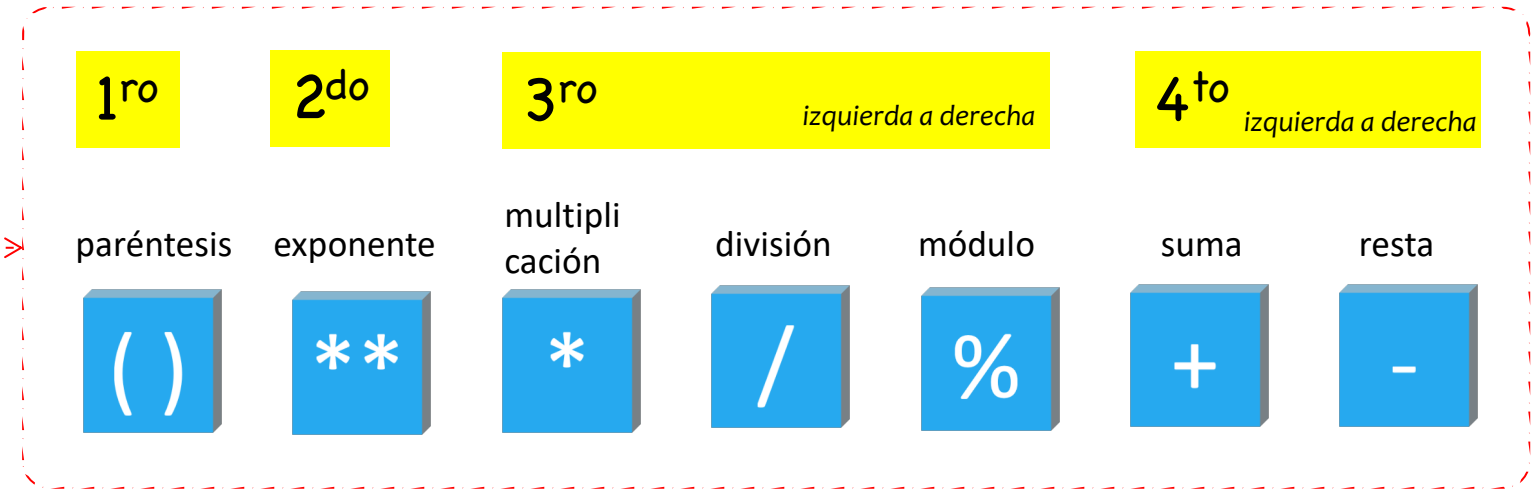
$$18 = 2 \times 9 + 0$$

módulo o resto división

módulo o resto división

Orden de Precedencia

Recuerda. El orden de precedencia es importante. Este es el orden que utiliza Python



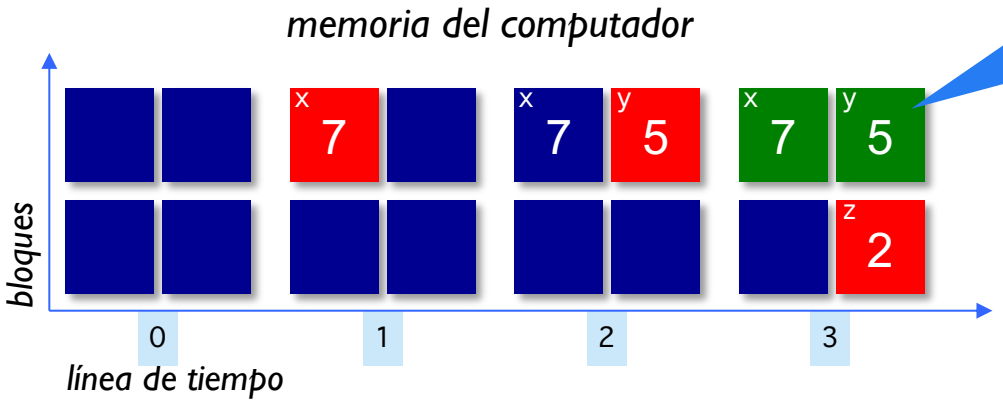
Recordemos que una variable representa un espacio de memoria en el computador que puede ser modificado en el tiempo y permite el registro y acceso a los datos.

Matemáticas

```
x = 7
y = 5
z = x - y
```

Python

```
x = 7
y = 5
z = x - y
print(z)
```



El computador reserva un espacio en memoria y a medida que se ejecutan instrucciones, se acceden y modifican los datos según la ejecución del programa

sin cambios
registro
lectura

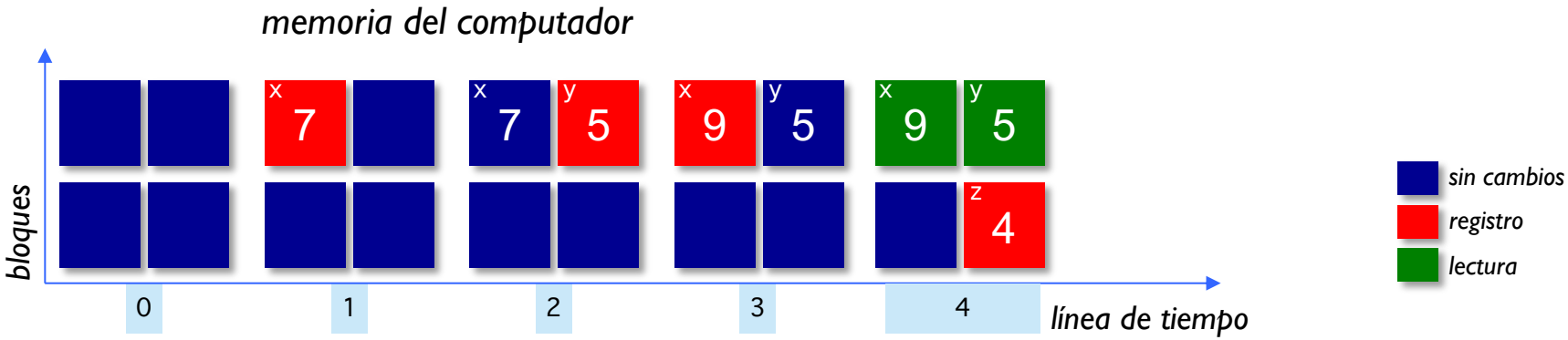
Una variable representa un espacio de memoria en el computador que puede ser modificado en el tiempo y permite el registro y acceso a los datos.

Matemáticas

```
x = 7
y = 5
x = 9
z = x - y
```

Python

```
1 x = 7
2 y = 5
3 x = 9
4 z = x - y
5 print(z)
```

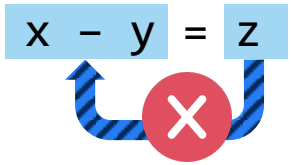


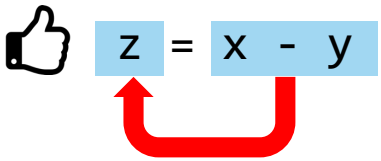
Una variable representa un espacio de memoria en el computador que puede ser modificado en el tiempo y permite el registro y acceso a los datos.

Python

```
x = 7
y = 5
x = 9
z = x - y
print(z)
```

Importante

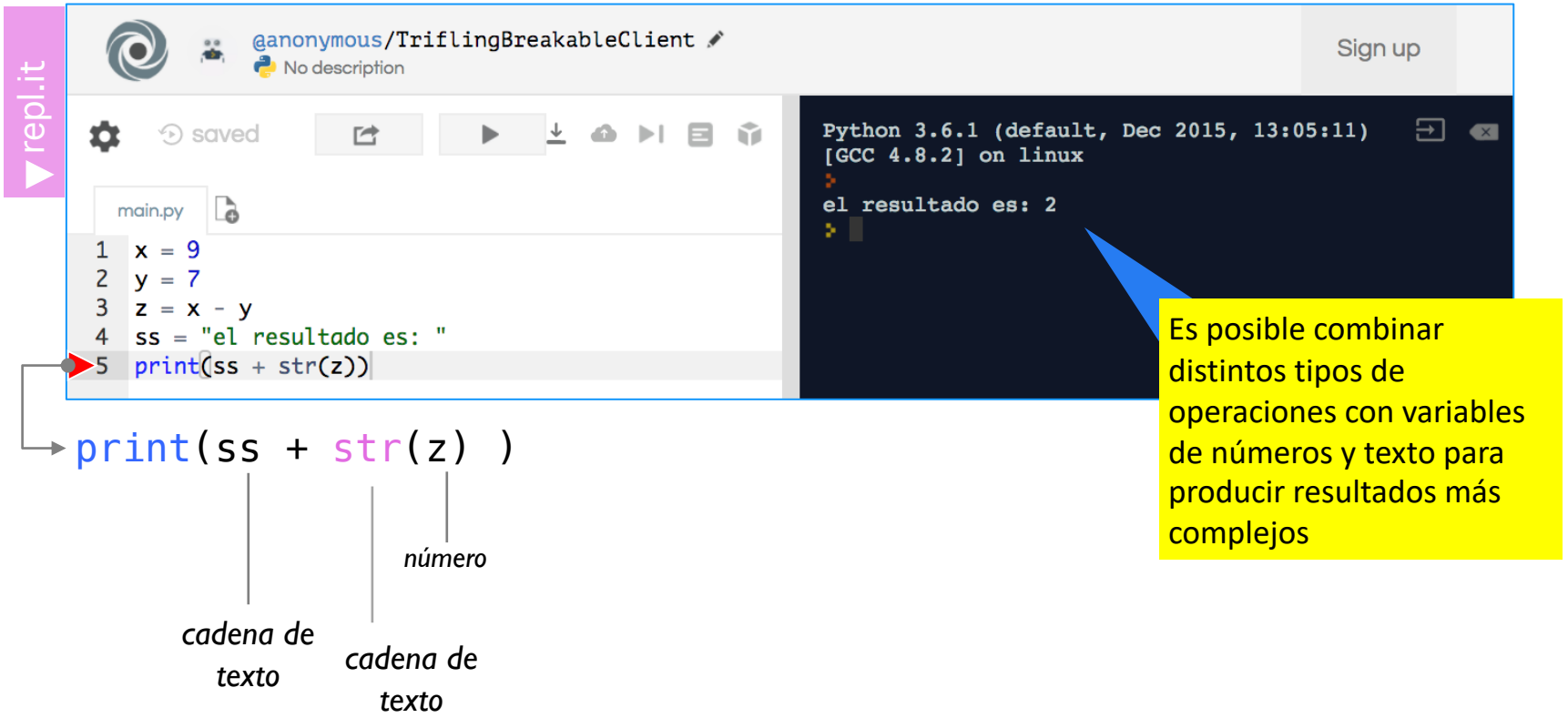




La asignación es siempre de derecha a izquierda

str()

Vimos anteriormente que print me permite imprimir varios valores separados por coma. Una alternativa es el comando str() que permite transformar un número en una cadena de texto. Con ello podemos concatenar distintos tipos de datos (números o texto)



Es posible combinar distintos tipos de operaciones con variables de números y texto para producir resultados más complejos

¡Prueba tú!

Tiempo : 5 minutos

Vamos a comenzar usando Python como una calculadora. Para ello muestra en pantalla una operación matemática y su resultado. Por ejemplo:

```
El resultado de 2 * 3 + 4 – 12 ** 5 es  -248822
```

¿Qué operaciones matemáticas ocupaste?

¿Tuviste algún problema al imprimir en pantalla?

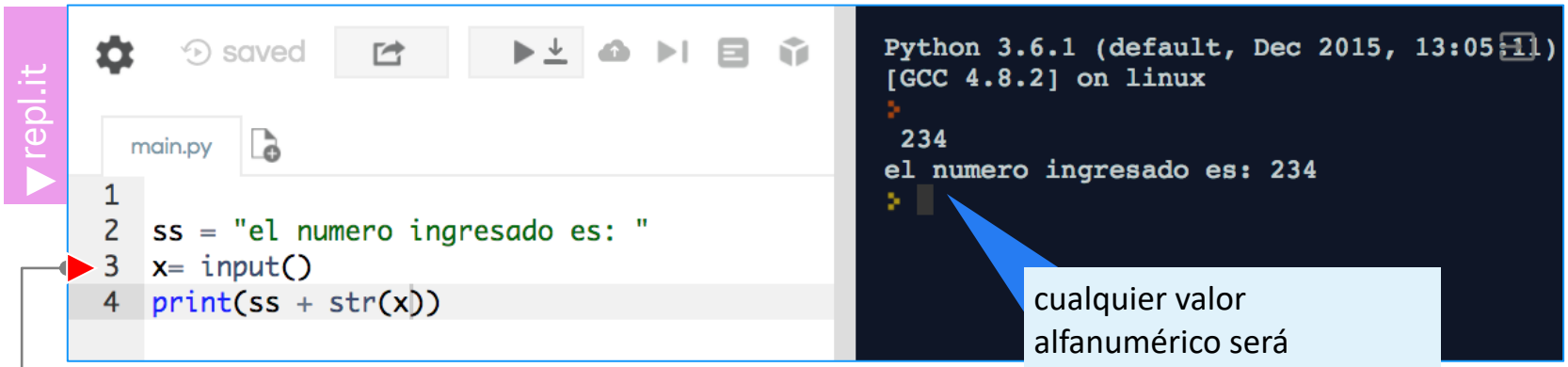
Temario

- Introducción a Python
- Primeros pasos
  - Lección 01: Despliegue de resultados
  - Lección 02: Variables
  - Lección 03: Operaciones aritméticas
  - Lección 04: Ingreso de datos

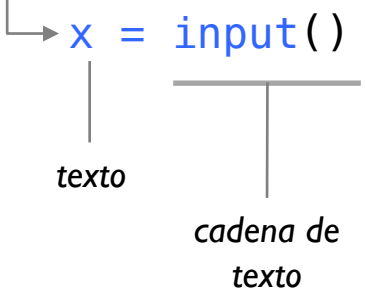
```
try:
    start = date(int(self.start_year.
                     self.months.index(se
                     int(self.start_day.ge

    end = date(int(self.end_year.get(
                     self.months.index(se
                     int(self.end_day.get
```

input()

El comando input() permite ingresar un numero por la terminal. El programa espera que el usuario escriba un número y presione la tecla “enter”

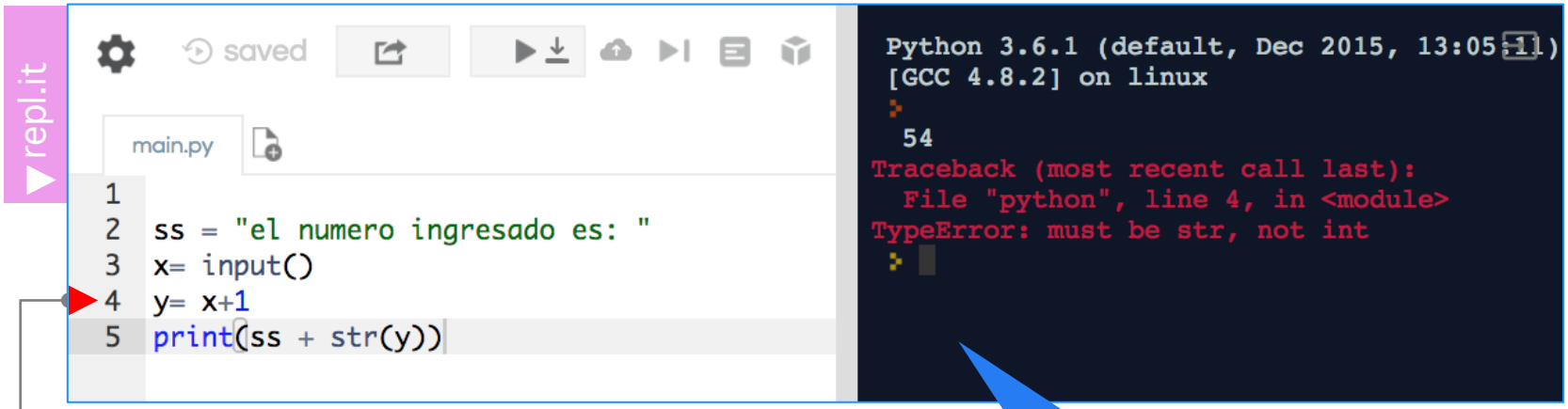


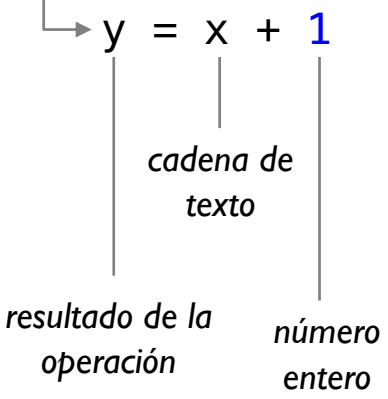
cualquier valor alfanumérico será procesado como una cadena de caracteres.
Es tarea del programador transformar los caracteres numéricos a números



input()

El comando input() permite ingresar un número por la terminal. El programa espera que el usuario escriba un número y presione la tecla “enter”





El error se genera por que el intérprete comprende que el valor almacenado en la variable ‘x’ es una cadena de caracteres y no un valor entero (número)

int()

El comando int() transforma una cadena alfanumérica en un número entero. **Importante**: Usted debe ingresar un número entero sin decimales, sin caracteres.

repl.it

saved

main.py

```
1
2  ss = "el numero ingresado es: "
3  x= input()
4  y= int(x) + 1
5  print(ss + str(y))
```

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
 54
el numero ingresado es: 55
```

El proceso anterior se denomina conversión de tipos. Consiste en cambiar el tipo de datos de un origen a uno de destino según el formato seleccionado.

```
y = int(x)  + 1
```

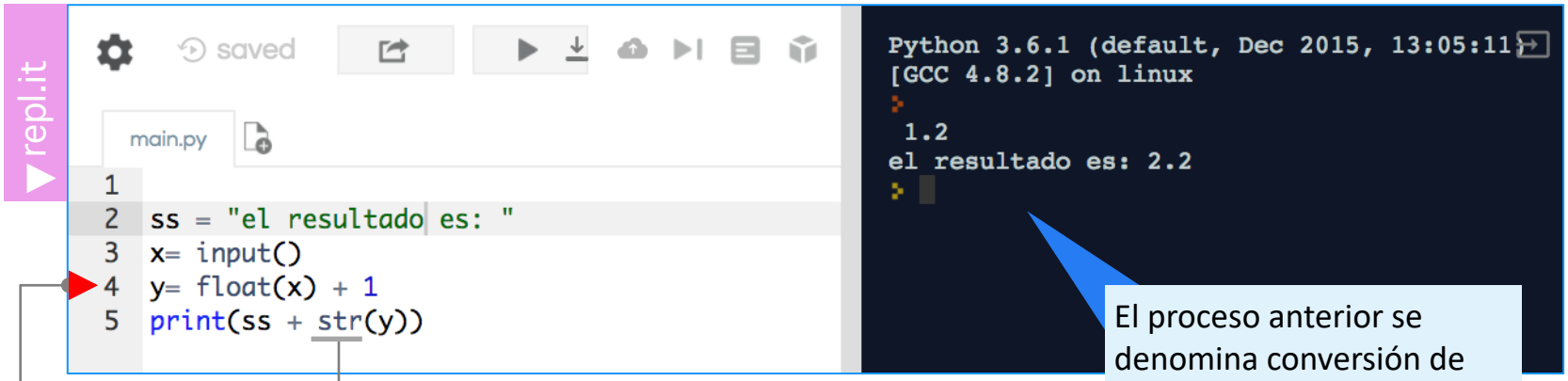
cadena de texto

número entero

número entero

float()

El comando float() transforma una cadena alfanumérica en un número flotante. Importante: Usted puede ingresar un número con decimales (o punto flotante) pero sin caracteres



cadena de texto

```
y = float(x)  + 1
```

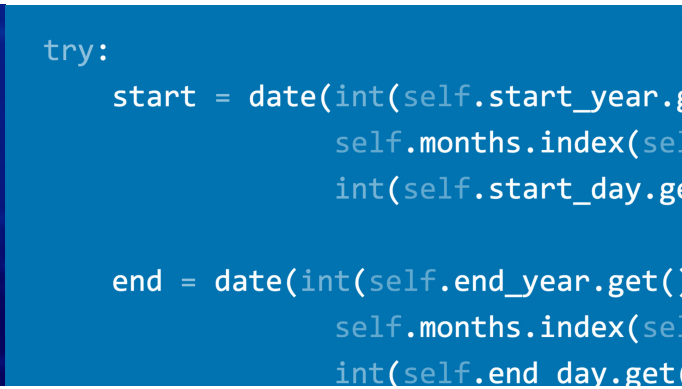
número decimal

número decimal

El proceso anterior se denomina conversión de tipos. Consiste en cambiar el tipo de datos de un origen a uno de destino según el formato seleccionado.

Temario

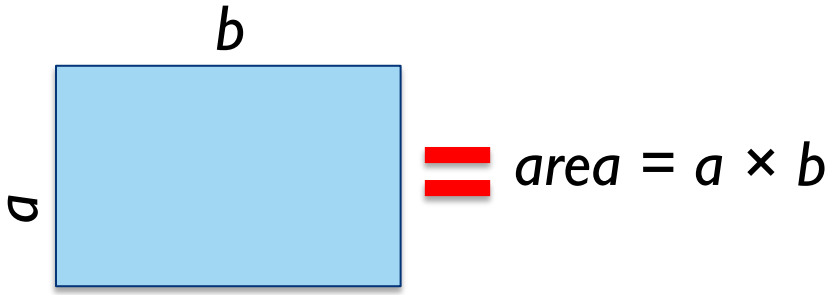
- Introducción a Python
- Primeros pasos
  - Lección 01: Despliegue de resultados
  - Lección 02: Variables
  - Lección 03: Operaciones aritméticas
  - Lección 04: Ingreso de datos
  - Ejercicio 1



Ejercicio 1

Tiempo : 5 minutos

Realice un programa que calcule el área de un rectángulo.



▼ Preguntas

¿Cómo se calcula el área?
¿Cómo solicito los datos al usuario?
¿Cómo realizo la multiplicación?
¿Los valores son números enteros o decimales?
¿El resultado es un entero o decimal?
¿Cómo se muestra el resultado?

Proceso de pensamiento guiado con preguntas, también conocido como pensamiento crítico

Ejercicio 1

Realice un programa que calcule el área de un rectángulo.

¿Cómo se calcula el área? → `area = a × b`

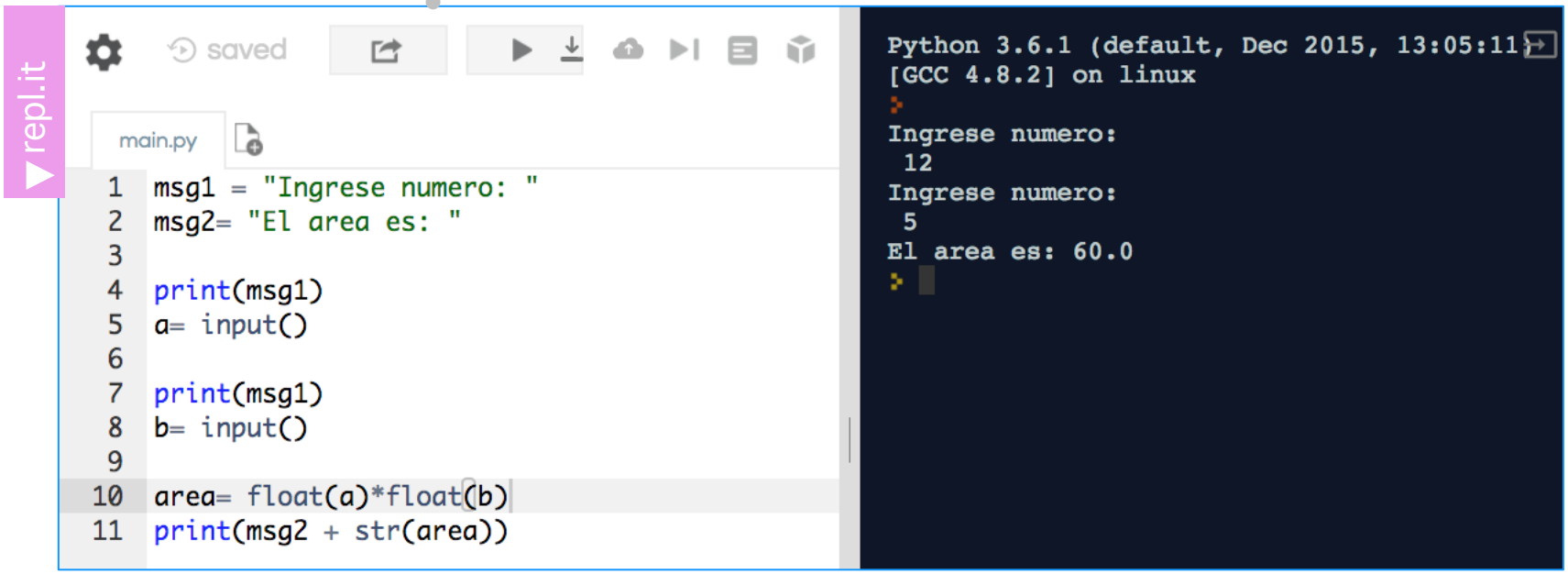
¿Cómo solicito los datos al usuario? → `input()`

¿Cómo realizo la multiplicación? → `*`

¿Los valores son números enteros o decimales? → `float()`

¿Donde almaceno el resultado? → `area`

¿Cómo se muestra el resultado? → `print() y str()`

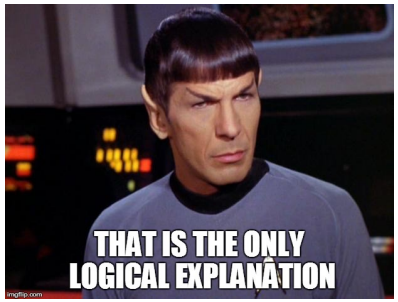


Temario

- Introducción a Python
- Primeros pasos
- Estructuras de control
  - Operadores lógicos
  - Condicionales IF-ELSE


```
try:
    start = date(int(self.start_year.
                     self.months.index(se
                     int(self.start_day.ge

    end = date(int(self.end_year.get(
                     self.months.index(se
                     int(self.end_day.get
```

comparación	==	distinto de	!=
mayor	>	menor	<
mayor igual	>=	menor igual	<=



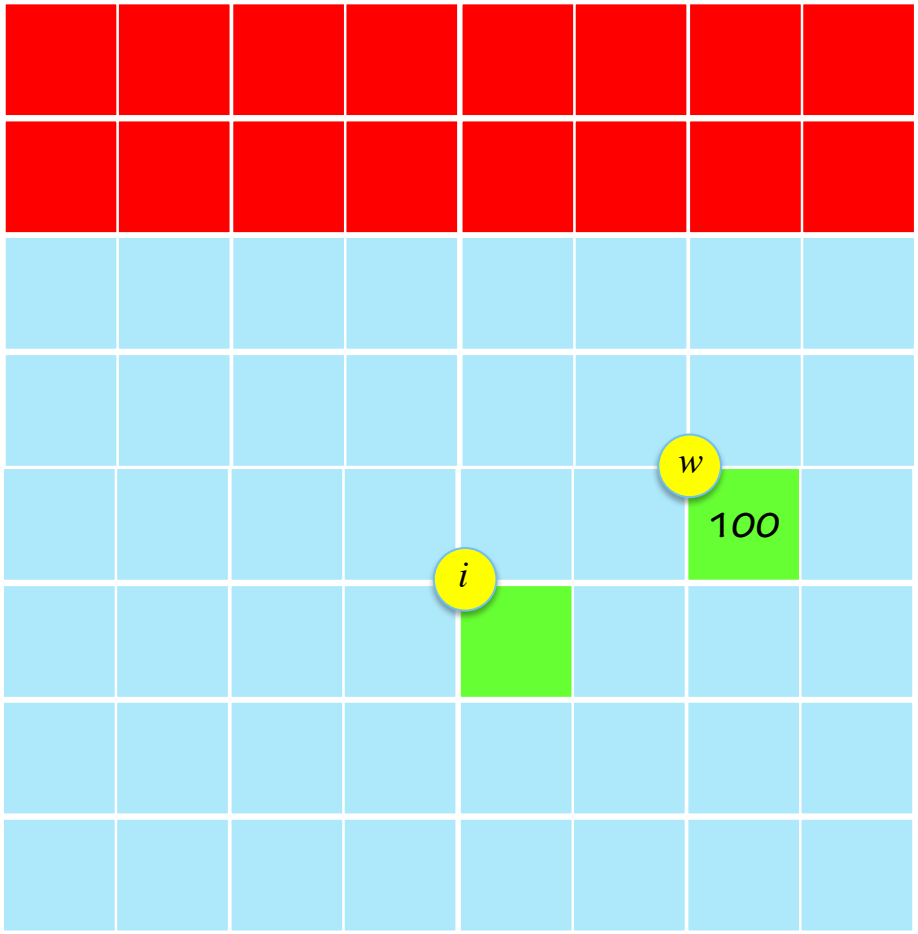
Resultados posibles

verdadero (TRUE) **1**

falso (FALSE) **0**

```
w = 100

i = 99 == w
```

```
99 == 100
```

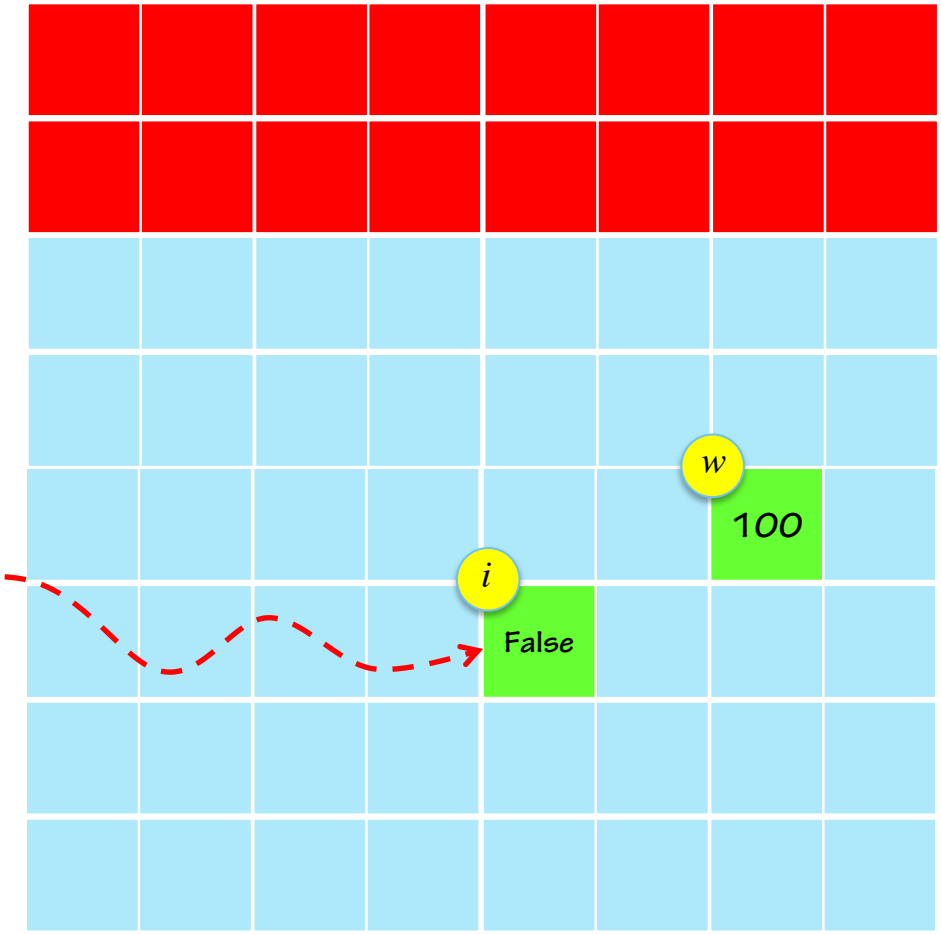


En el caso de las operaciones relacionales, el único resultado posible es 0 ó 1.

```
w = 100

i = 99 == w
```

99 == 100

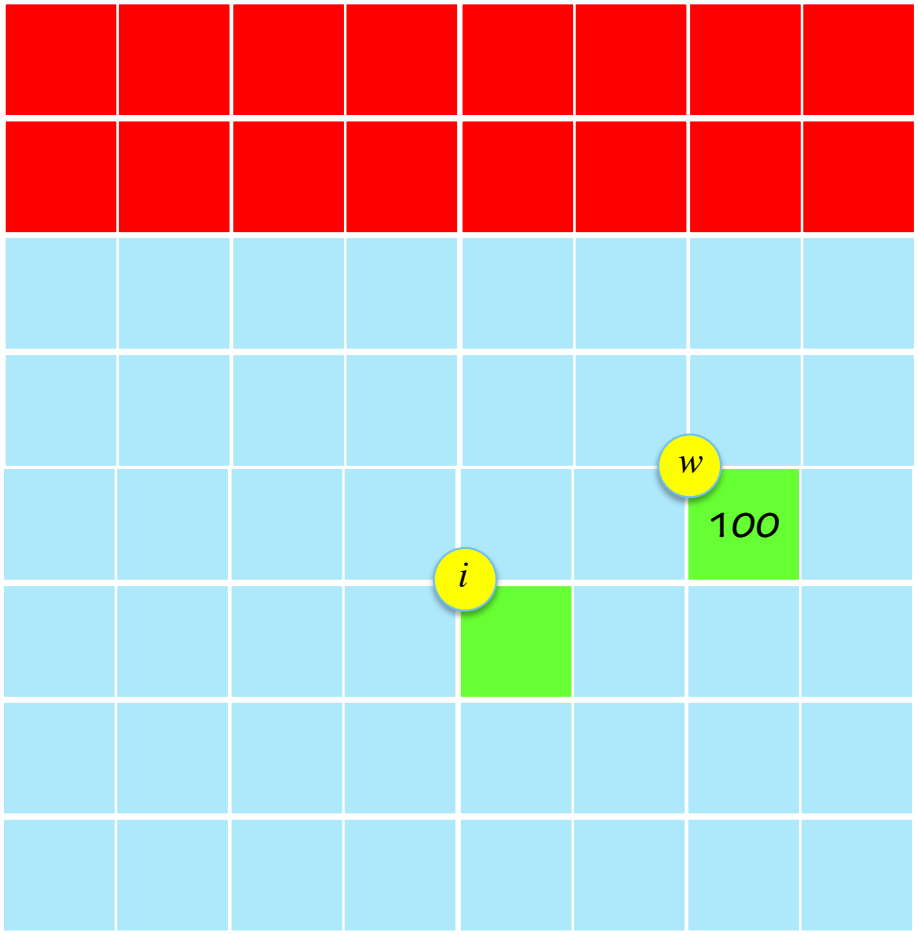
False



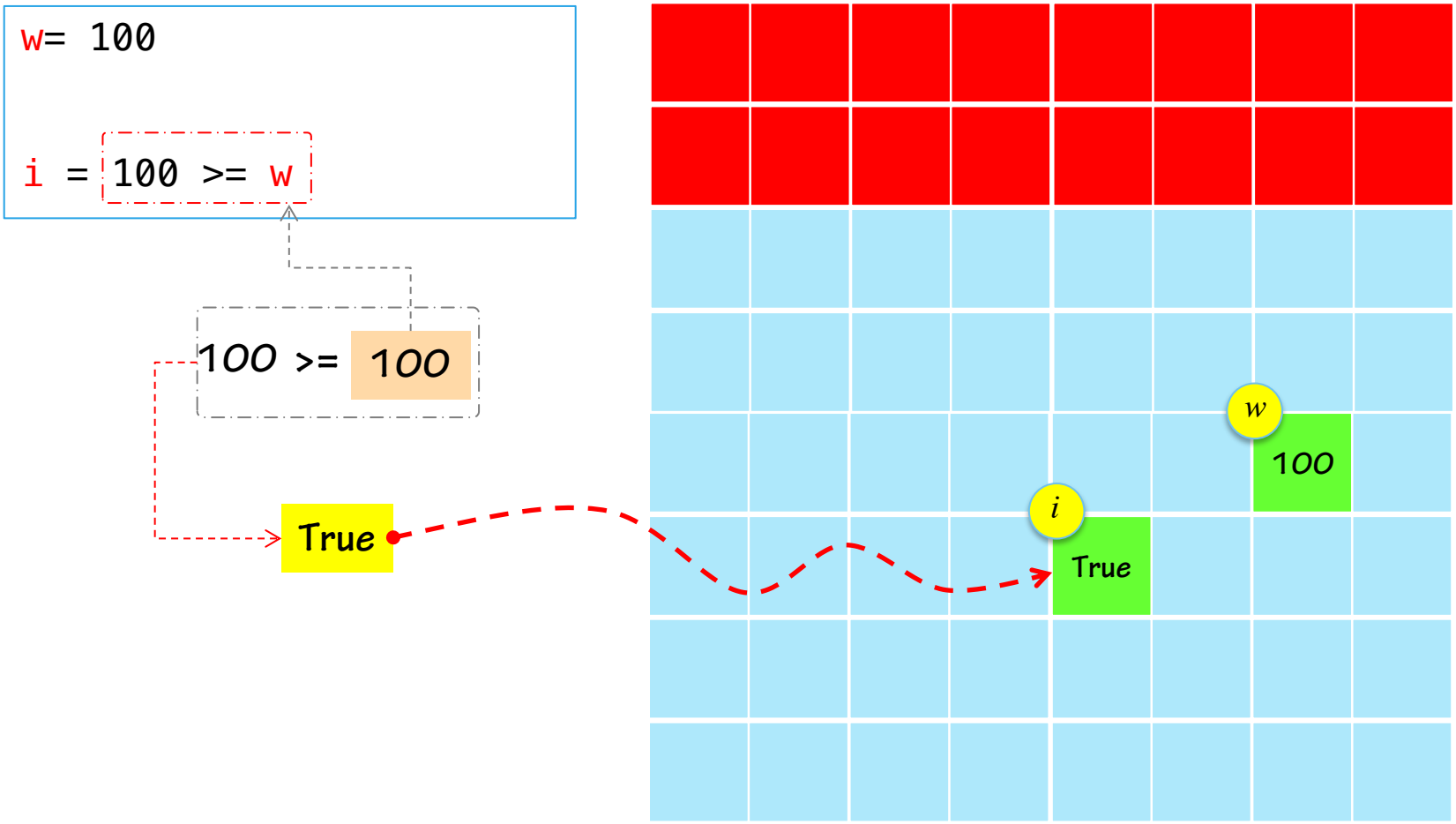
En el caso de las operaciones relacionales, el único resultado posible es False o True

```
w = 100

i = 100 >= w
```

```
100 >= 100
```



En el caso de las operaciones relacionales, el único resultado posible es True o False.


w= 100
i = 100 >= w
100 >= 100
True
w
100
i
True

operando operando

Tabla de verdad

a	b	not(a)	a and b	a or b
True	True	False	True	True
True	False	False	False	True
False	True	True	False	True
False	False	True	False	False

precedencia

1ro		2do	3ro	4to
==	!=	not	and	or

Operadores lógicos

and — Si ambos operandos son verdaderos, la condición se hace verdadera

or — Si cualquiera de los operandos es verdadero, la condición se hace verdadera

not — Revierte el resultado lógico de su operando. Si la condición es verdadera, se revierte a falso

precedencia

1ro	2do	3ro	4to
==	!=	not	and	or

operando operando

Tabla de verdad

a	b	not(a)	a and b	a or b
True	True	False	True	True
True	False	False	False	True
False	True	True	False	True
False	False	True	False	False

Ejemplo

Expresión	Evaluación	Resultado
(25>1) and (10>2)		
not(25>1) and (10>2)		
(-8<-3) or 100<99		
not(25>1) and (-8<-3) or 100<99		

Condicionales
operadores lógicos

precedencia

1ro	2do	3ro	4to
==	!=	not	and	or

Tabla de verdad

operando a	operando b	not(a)	a and b	a or b
True	True	False	True	True
True	False	False	False	True
False	True	True	False	True
False	False	True	False	False

Ejemplo

Expresión	Evaluación	Resultado
(25>1) and (10>2)	True and True	True
not(25>1) and (10>2)		
(-8<-3) or 100<99		
not(25>1) and (-8<-3) or 100<99		

Condicionales
operadores lógicos

precedencia

1ro	2do	3ro	4to
==	!=	not	and	or

Tabla de verdad

operando a	operando b	not(a)	a and b	a or b
True	True	False	True	True
True	False	False	False	True
False	True	True	False	True
False	False	True	False	False

Ejemplo

Expresión	Evaluación	Resultado
(25>1) and (10>2)	True and True	True
not(25>1) and (10>2)	False and True	False
(-8<-3) or 100<99		
not(25>1) and (-8<-3) or 100<99		

Condicionales
operadores lógicos

precedencia

1ro	2do	3ro	4to
==	!=	not	and	or

Tabla de verdad

operando a	operando b	not(a)	a and b	a or b
True	True	False	True	True
True	False	False	False	True
False	True	True	False	True
False	False	True	False	False

Ejemplo

Expresión	Evaluación	Resultado
(25>1) and (10>2)	True and True	True
not(25>1) and (10>2)	False and True	False
(-8<-3) or 100<99	True or False	True
not(25>1) and (-8<-3) or 100<99		

Condicionales
operadores lógicos

precedencia

1ro	2do	3ro	4to
==	!=	not	and	or

Tabla de verdad

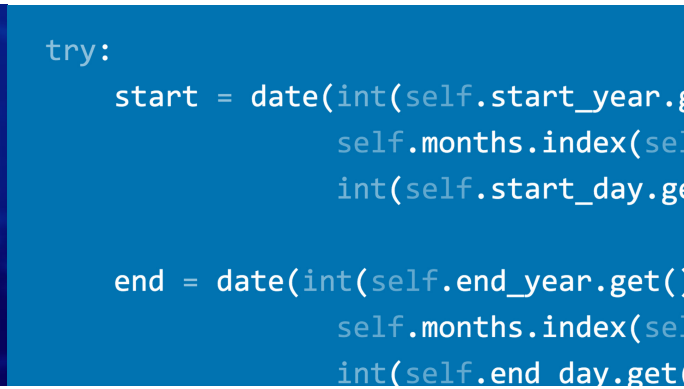
operando a	operando b	not(a)	a and b	a or b
True	True	False	True	True
True	False	False	False	True
False	True	True	False	True
False	False	True	False	False

Ejemplo

Expresión	Evaluación	Resultado
(25>1) and (10>2)	True and True	True
not(25>1) and (10>2)	False and True	False
(-8<-3) or 100<99	True or False	True
not(25>1) and (-8<-3) or 100<99	(False and True) or False = False or False	False

Temario

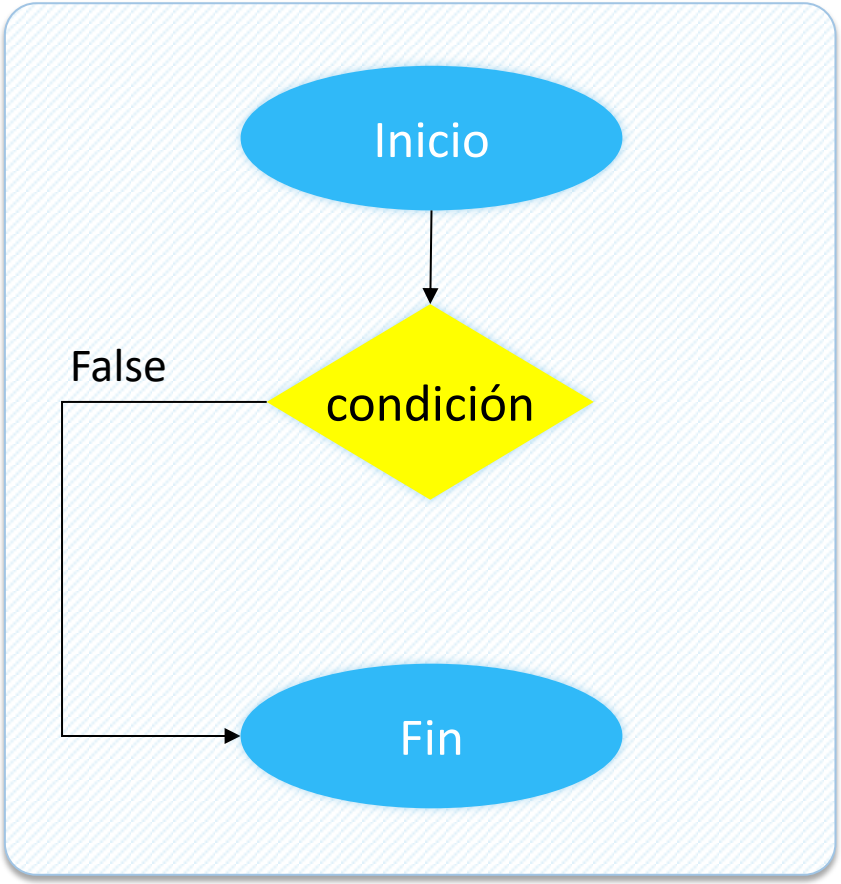
- Introducción a Python
- Primeros pasos
- Estructuras de control
  - Operadores lógicos
  - Condicionales IF-ELSE

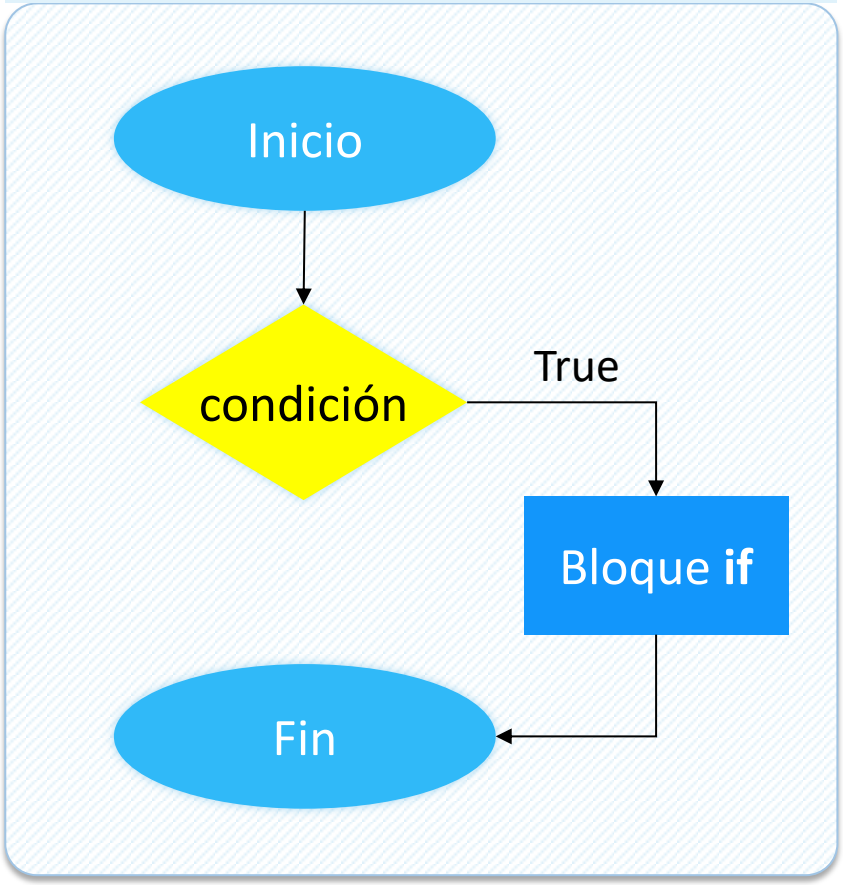


Condicionales

```
if <condición lógica> :
```

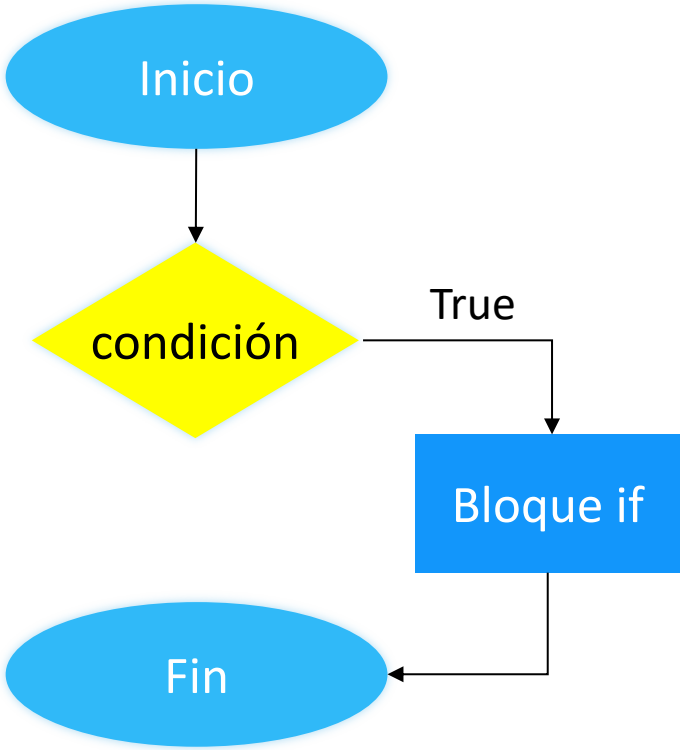
Si la condición es verdadera (True o un entero positivo), el control del programa se dirige al bloque de instrucciones del **if**. Luego continúa su ejecución fuera del bloque





```
if <condición lógica> :
```

Si la condición es verdadera, el control del programa se dirige al bloque de instrucciones del **if**. Luego continúa su ejecución fuera del bloque




```
area = 10
w = 8

if area > 10:
    area = 9
```

condición

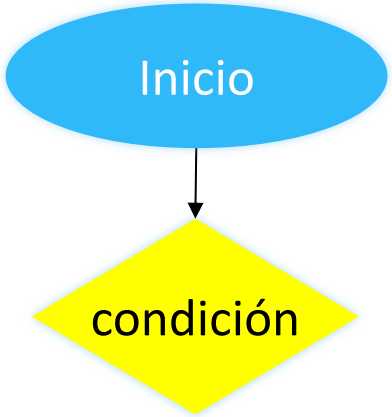
bloque IF

bloque de instrucciones

Importante

Las instrucciones dentro del bloque deben estar **indentadas** con al menos un espacio. Si no lo hace, el intérprete no ejecutará el programa

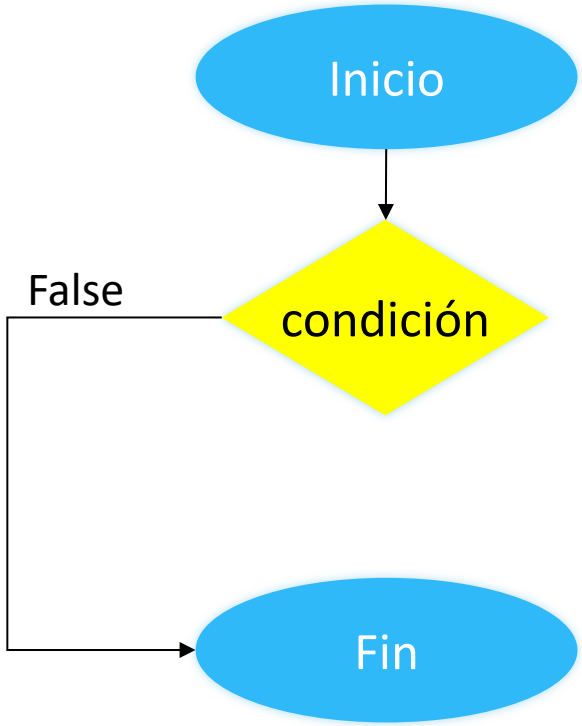
Condicionales




```
area = 10
w = 8

if area > 10:
    print('El area es mayor a 10')
    w = 11

area = 9
```

¿Qué valor tiene la variable **w** al término del programa?

Condicionales

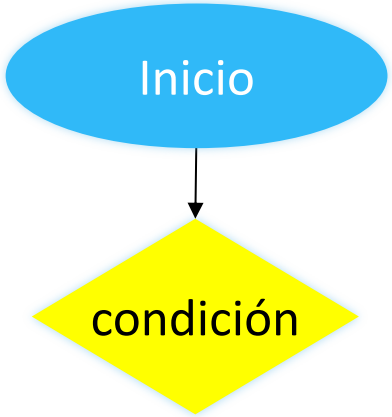



```
area = 10
w = 8

if area > 10:
    print('El area es mayor a 10')
    w = 11

area = 9
```

¿Qué valor tiene la variable w al término del programa?

la variable w tiene el valor 8 ya que NO ingresa al bloque IF, y NO se modifica el valor asignado al comienzo del programa.

Condicionales

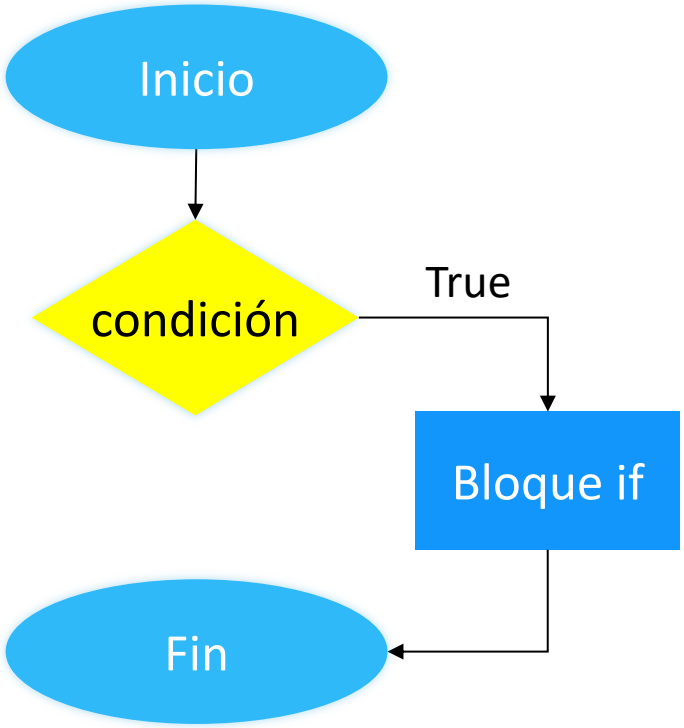



```
area = 10
w = 8

if area >= 10:
    print('El area es mayor a 10')
    w = 11

area = 9
```

condición

¿Qué valor tiene la variable w al término del programa?

Condicionales




```
area = 10
w = 8

if area >= 10:
    print('El area es mayor a 10')
    w = 11

area = 9
```

¿Qué valor tiene la variable w al término del programa?

la variable w el valor 11 ya que ingresa al bloque IF, y en ella se modifica el valor almacenado en la variable w.

Condicionales

▼ if(condición):

La instrucción **if():** nos permite evaluar una condición lógica que tiene dos resultados posibles: verdadero o falso.

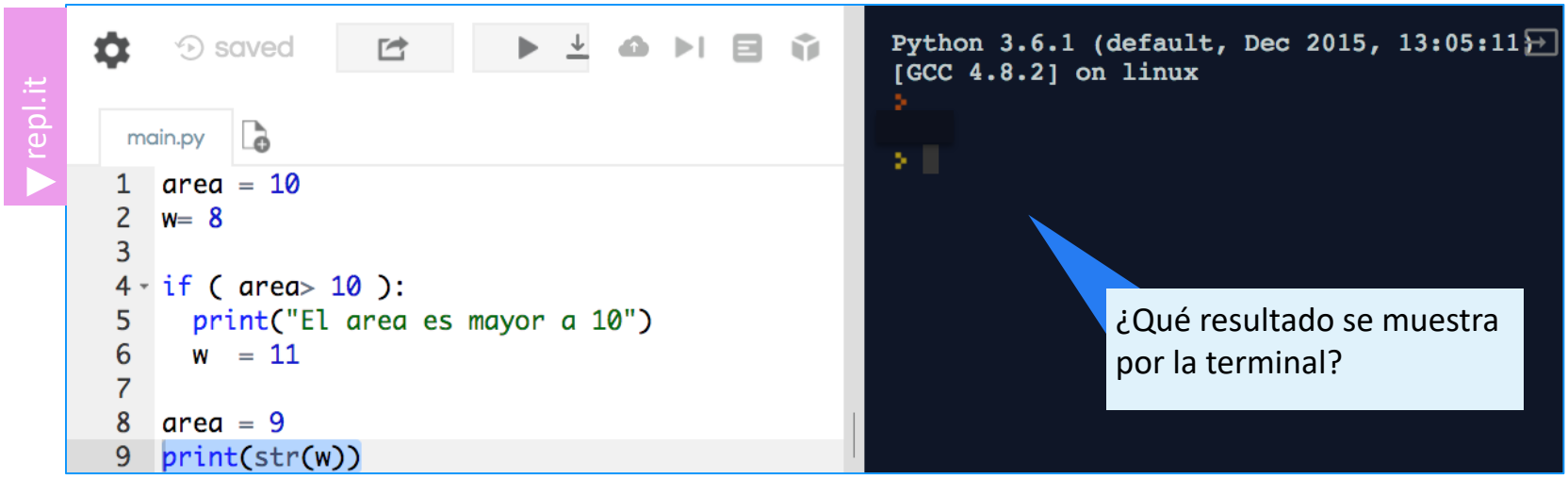


Tabla de comparaciones lógicas

comparación	==	distinto de	!=
mayor	>	menor	<
mayor igual	>=	menor igual	<=

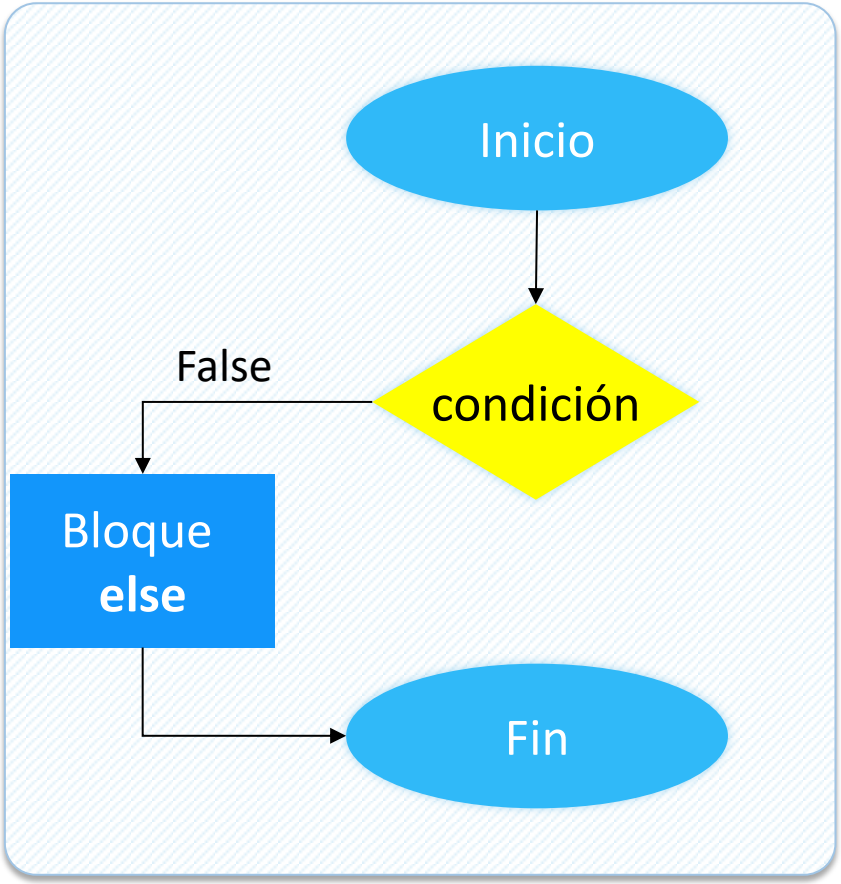
Resultados posibles

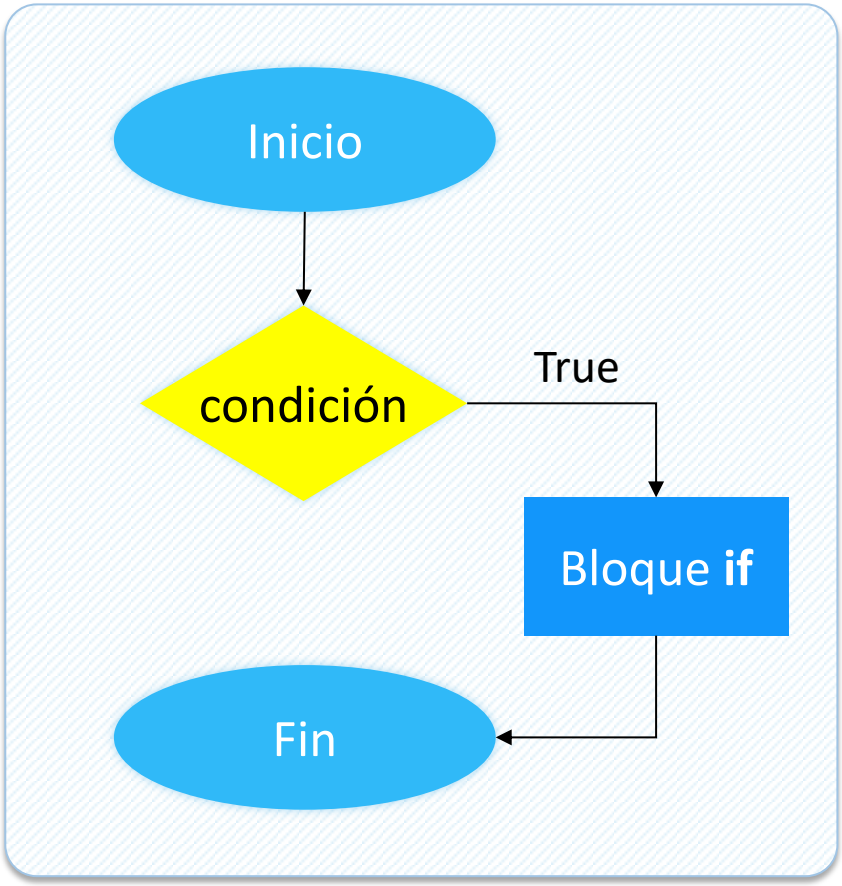
1	verdadero (TRUE)
0	falso (FALSE)

Condicionales

```
if  <condición lógica> :

else:
```

Si la condición es **verdadera**, el control del programa se dirige al bloque de instrucciones del **if**. Si es falsa realiza el bloque **else**.



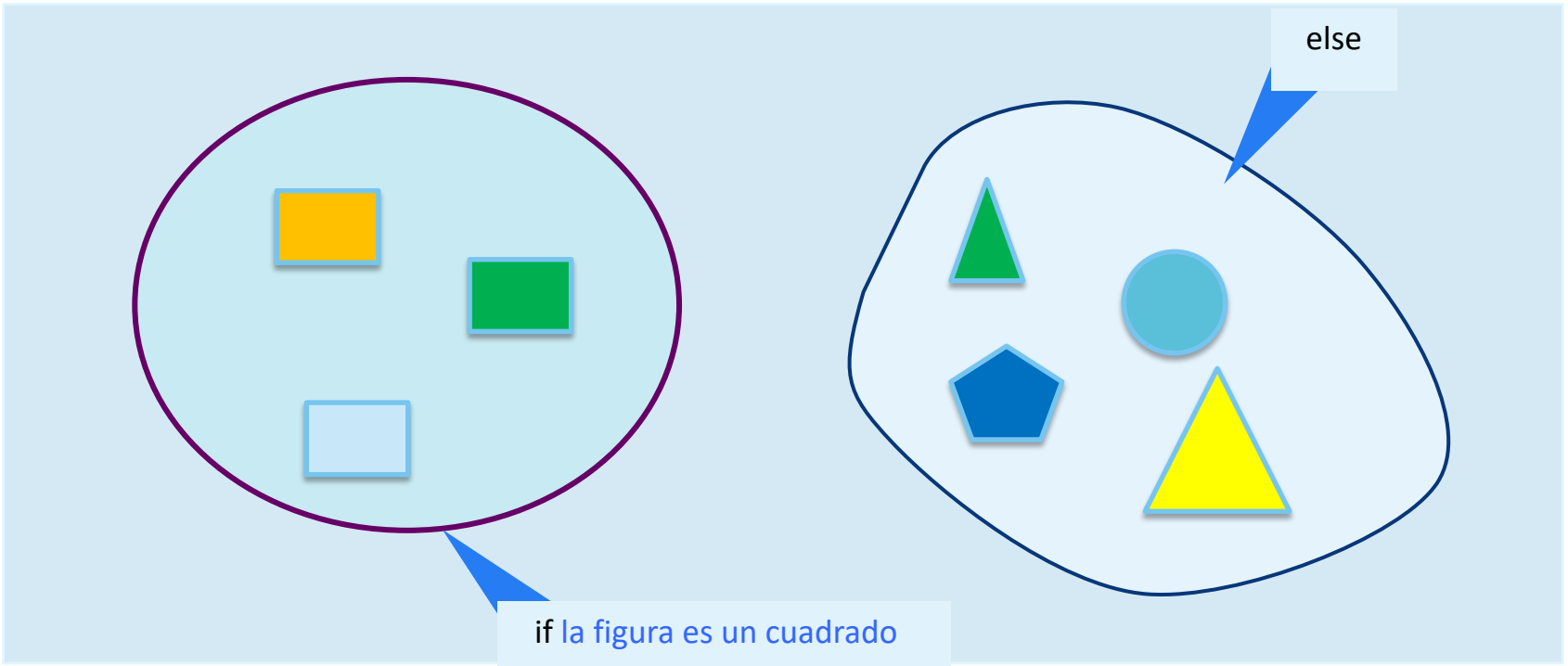


Condicionales

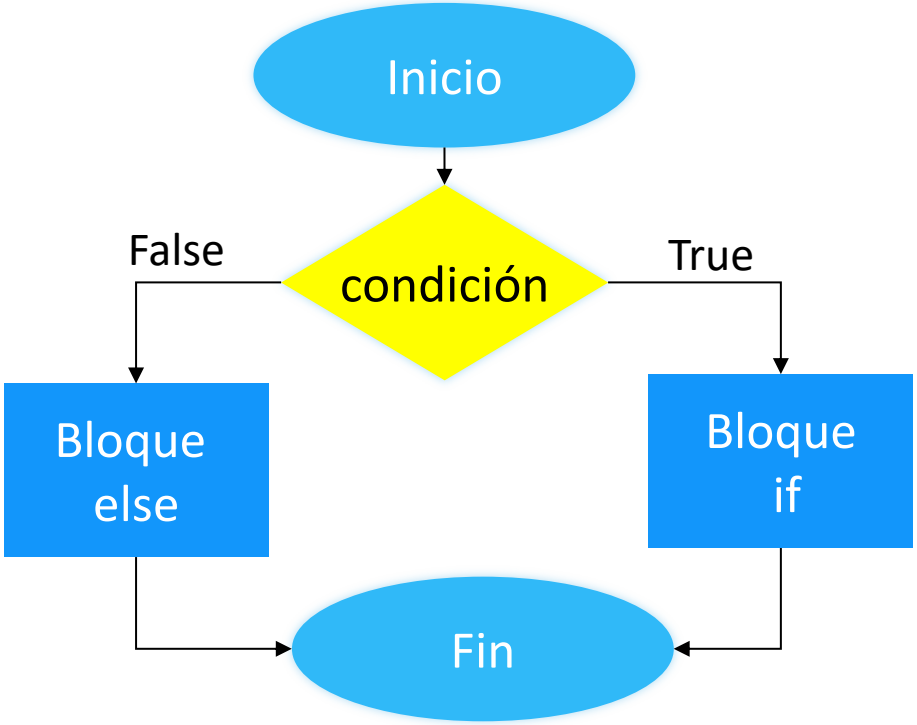
```
if  <condición lógica> :

else:
```

Si la condición es **verdadera**, el control del programa se dirige al bloque de instrucciones del **if**. Si es falsa realiza el bloque **else**.

Conjunto Universo = Figuras Geométricas

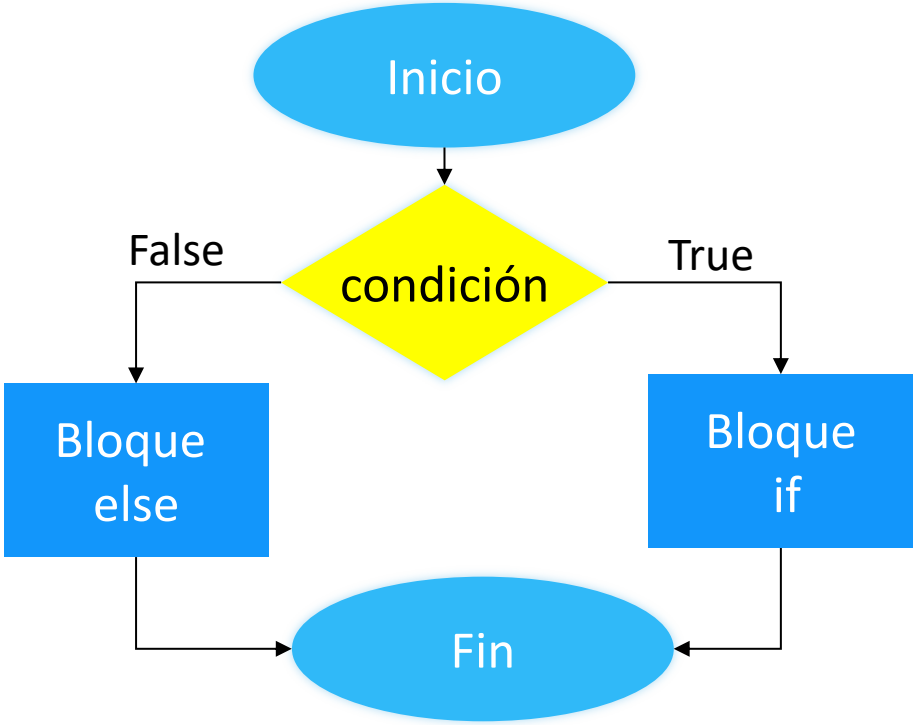


Condicionales




```
area = 10
w = 8

if area > 10:
    bloque IF
else:
    bloque ELSE
print(w)
```

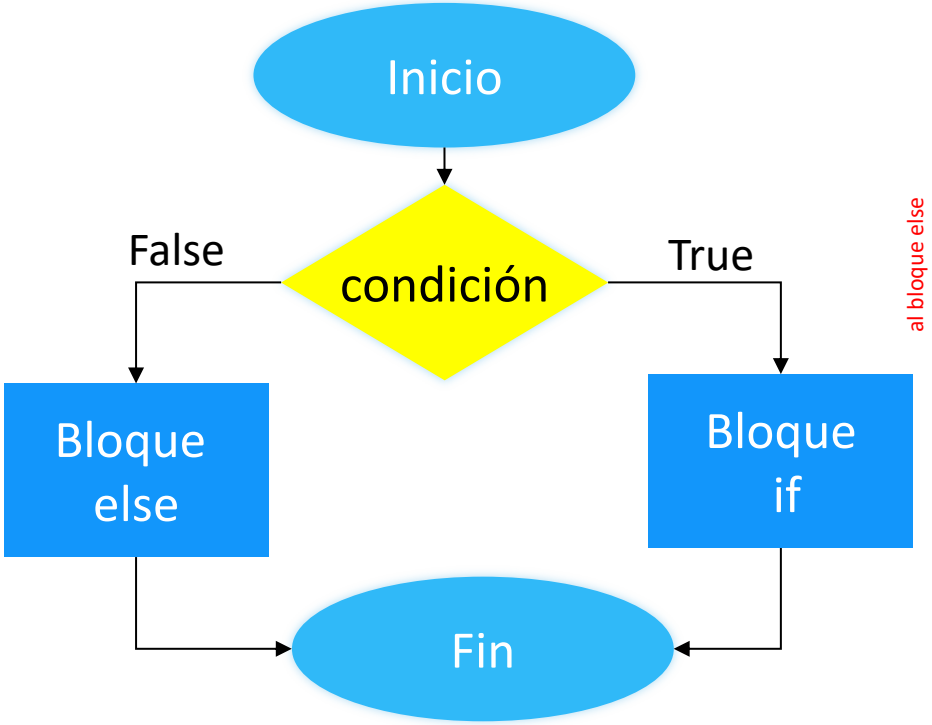
Condicionales




```
area = 10
w = 8

if area > 10:
    print('El area es mayor a 10')
    w = 11

else:
    area = 9
    w = 7

print(w)
```

¿Qué valor tiene la variable w al término del programa?

Condicionales




```
area = 10
w = 8

if area > 10:
    print('El area es mayor a 10')
    w = 11

else:
    area = 9
    w = 7

print(w)
```

al bloque else

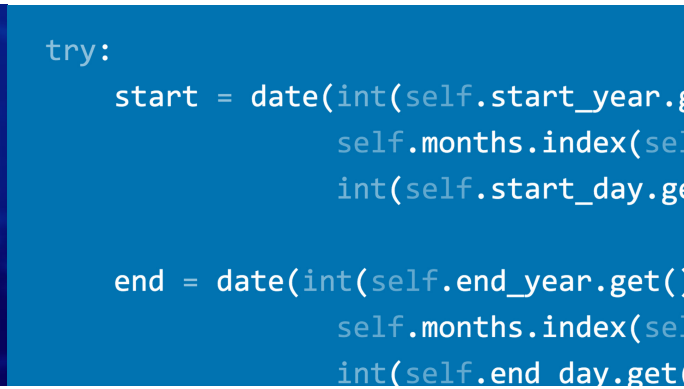
¿Qué valor tiene la variable **w** al término del programa?

la variable w tiene el valor 7 ya que NO ingresa al bloque if, pero SI ingresa al bloque else, modificando su valor


EMPUJE LA PUERTA
SI SE ABRE,
ESTA ABIERTO
CASO CONTRARIO, NO

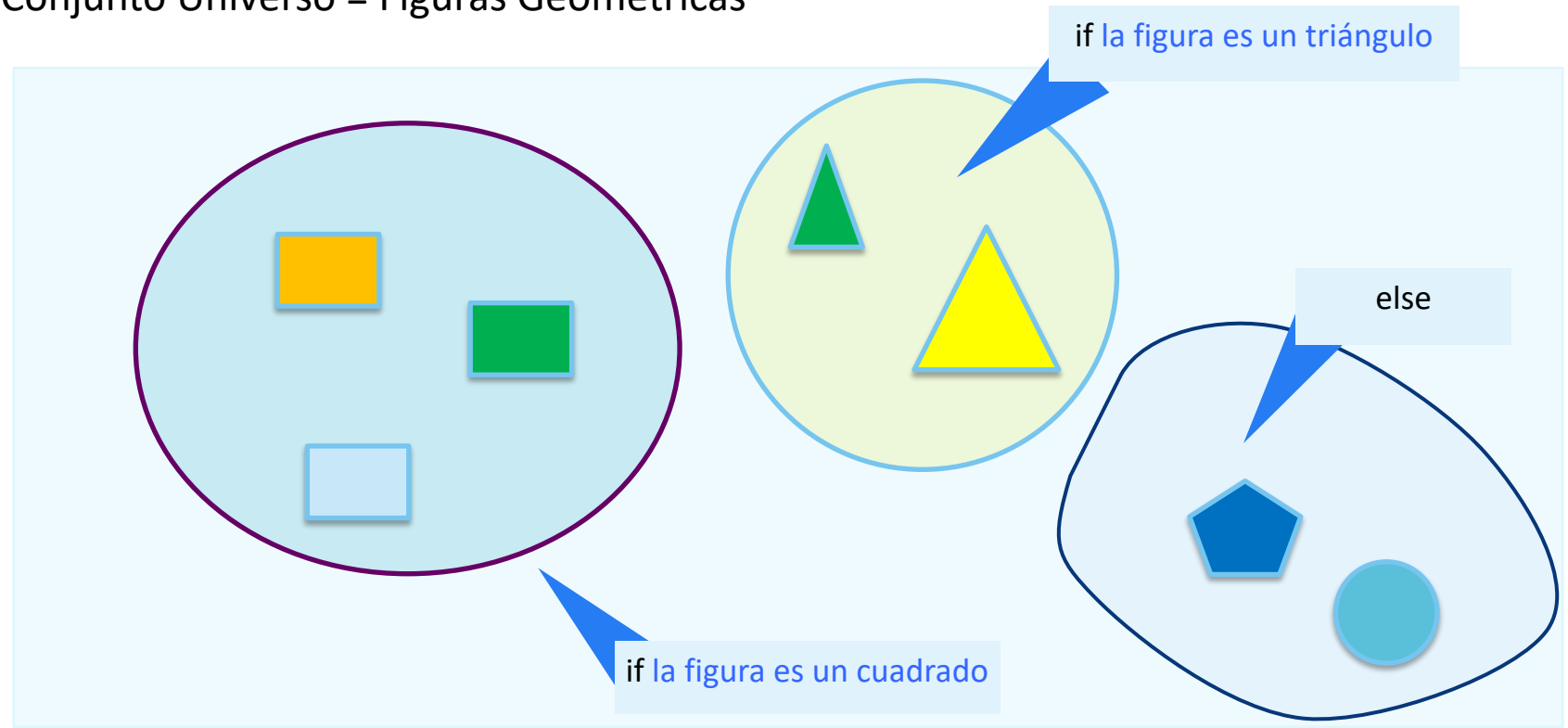
Temario

- Introducción a Python
- Primeros pasos
- Estructuras de control
  - Condicionales IF-ELSE
  - Condicionales IF-ELIF-ELSE

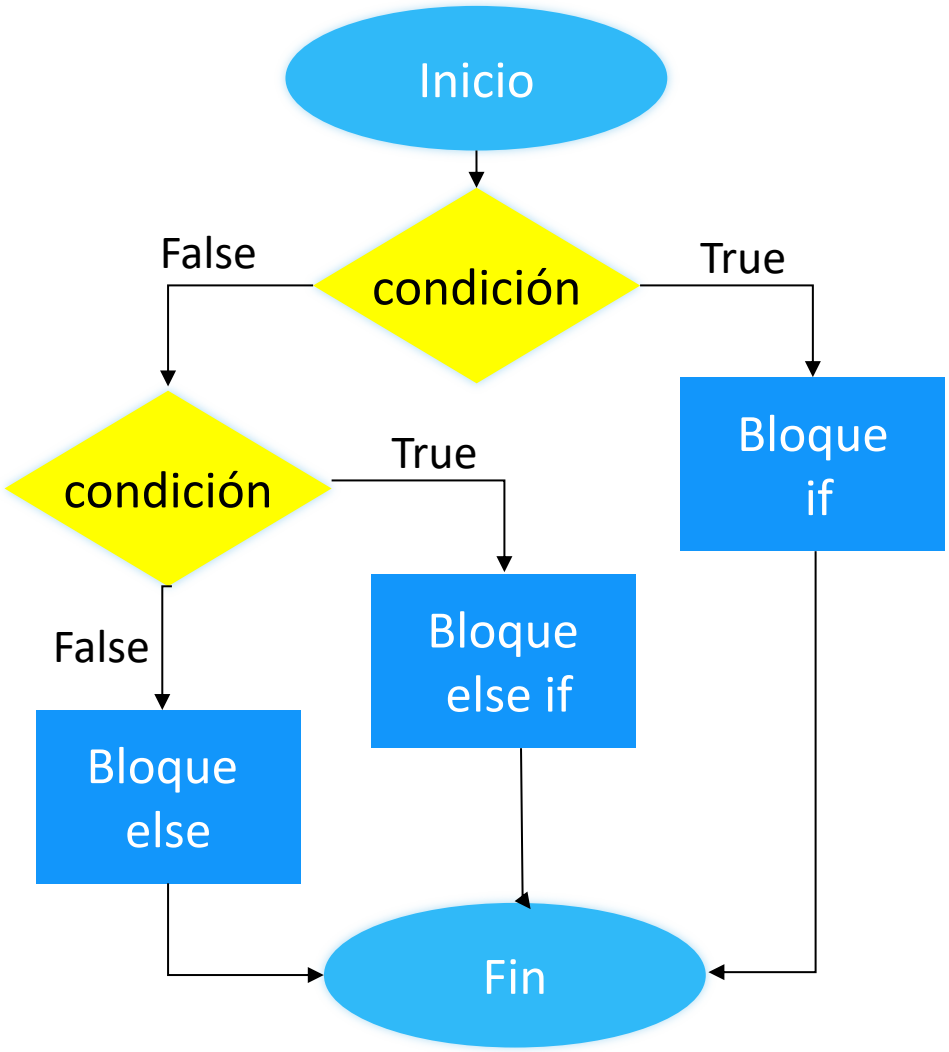


Hay veces que tenemos que separar nuestro conjunto universo en más de dos grupos: cuadrados, triángulos y el resto
¿Cómo se escribirán este tipo de condiciones en Python?

Conjunto Universo = Figuras Geométricas

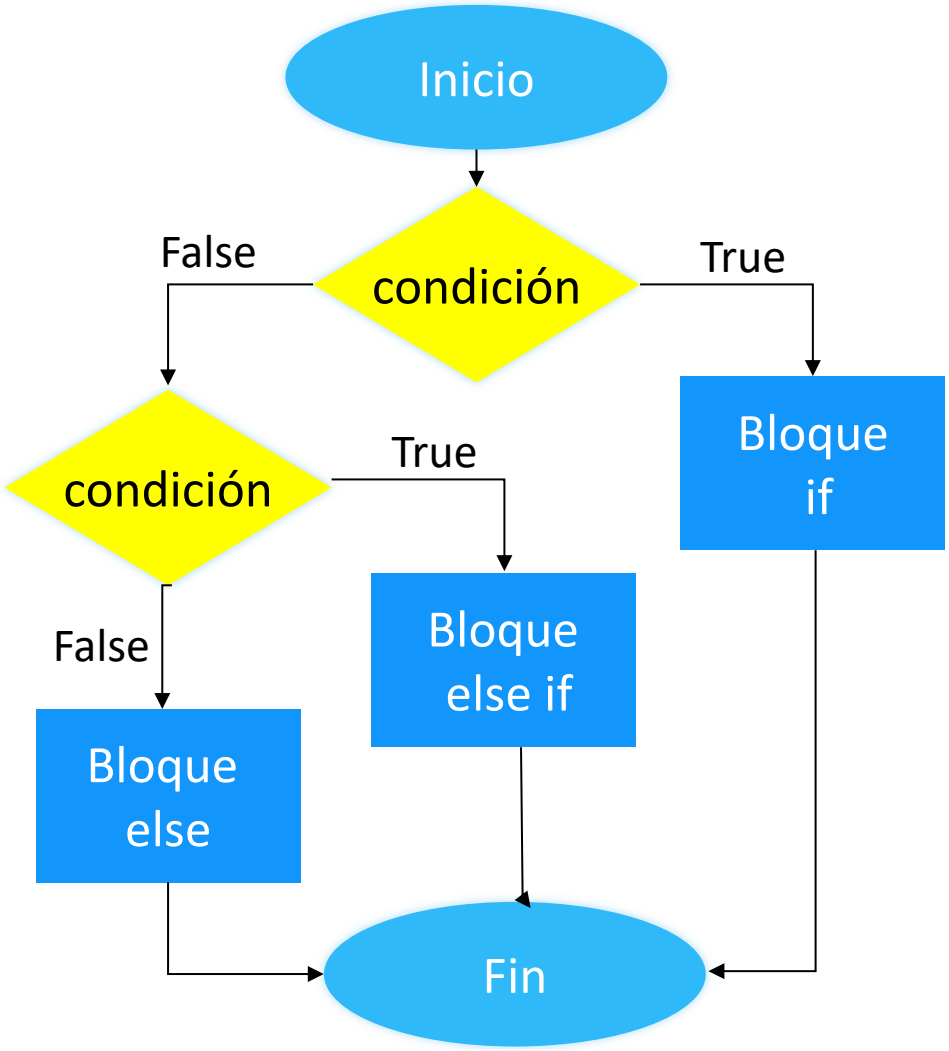


Condicionales




```
area = 5
w = 8

if area > 10:
    bloque if
else:
    if area > 5:
        bloque elif
    else:
        bloque else
print(w)
```

Condicionales

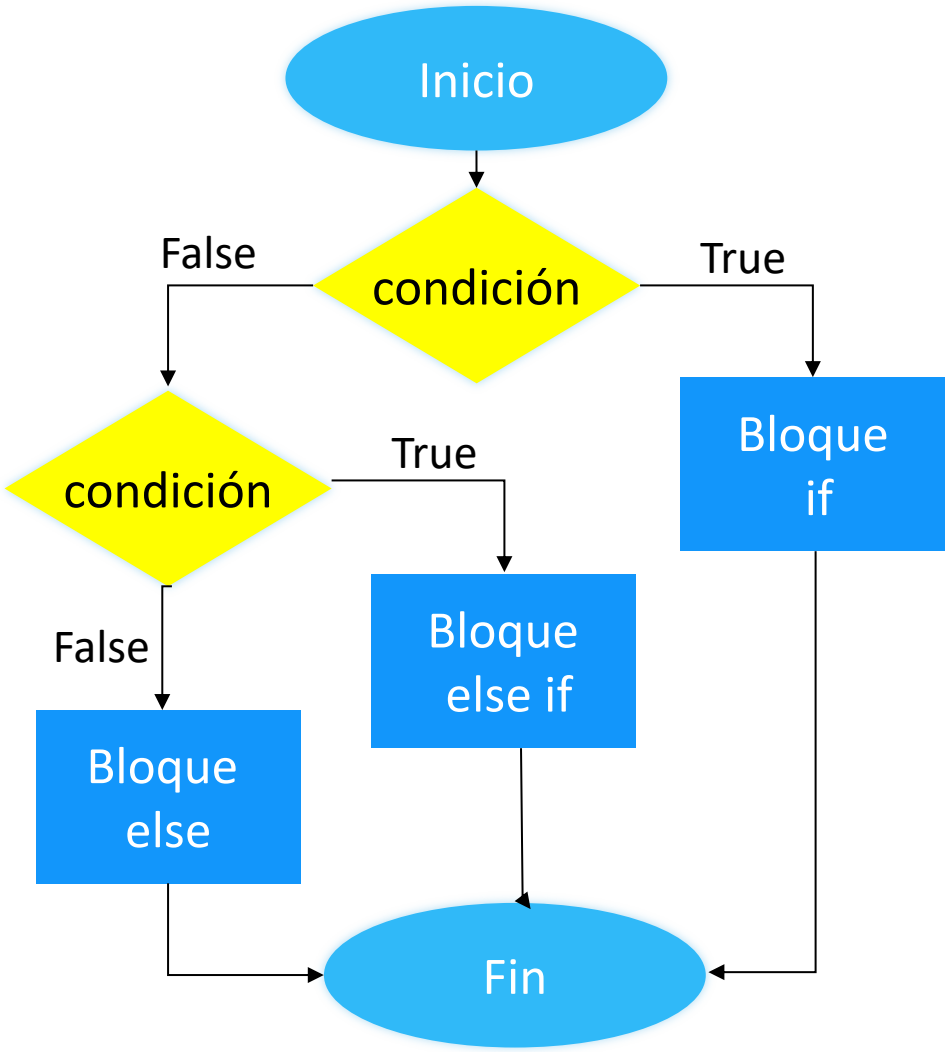



```
area = 5
w = 8

if area > 10:
    print('El area es mayor a 10')
    w = 11

else:
    if area > 5:
        print('El area es mayor a 5')
        w = 6

    else:
        print('El area es menor a 6')
        w = 1

print(w)
```

¿se fijaron que hay que indentar más?

¿Qué valor tiene la variable **w** al término del programa?

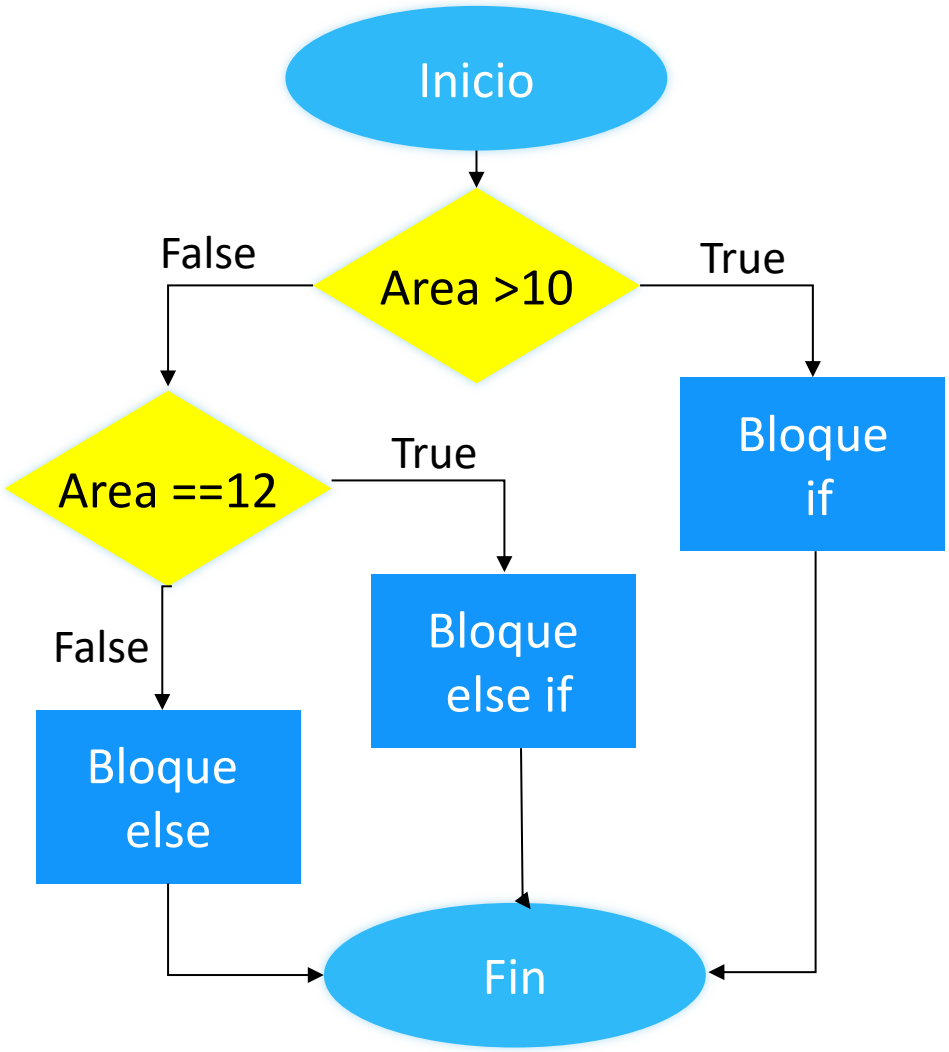
Condicionales



En Python podemos compactar las condiciones anidadas haciendo uso de `elif` SIEMPRE y CUANDO sean excluyentes

```
area = 5
w = 8

if area > 10:
    print('El área es mayor a 10')
    w = 11

elif area > 5:
    print('El área es mayor a 5')
    w = 6

else:
    print('El área es menor a 6')
    w = 1

print(w)
```

Condicionales




```
area = 5
w = 8

if area > 10:
    print('El área es mayor a 10')
    w = 11

elif area == 12:
    print('El área es 12')
    w = 6

else:
    print('área es menor o igual a 10')
    w = 1

print(w)
```

Esto no funciona..debe usarse:

```
area = 5
w    = 8

if area > 10:
    print("El area es mayor o a 10")
    w = 11

elif area == 12:
    print("El area es 12")
    w = 6

else:
    print("area es menor o igual a 10")
    w = 1

print(w)
```

```
area = 5
w    = 8

if area > 10:
   if area == 12:
       print("El area es 12")
       w = 6
   else:
       print("El area es mayor a 10")
       w = 11

else:
       print("area es menor o igual a 10")
       w = 1

print(w)
```

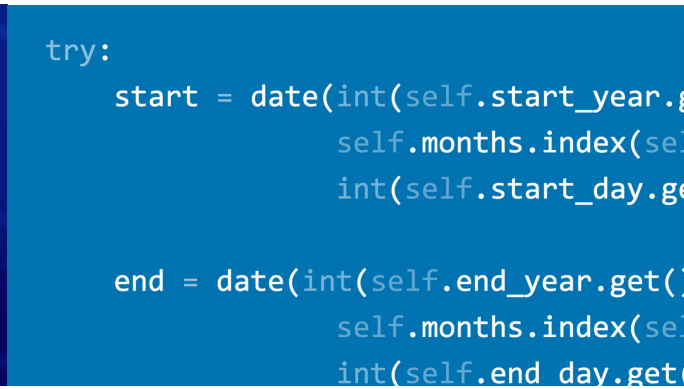
```
bool result;

if (value_1 == true && value_2 == true)
    result = true;

else if (value_1 == true && value_2 == false)
    result = false;

else if (value_1 == false && value_2 == true)
    result = false;

else if (value_1 == false && value_2 == false)
    result = true;

if (result == true)
    return true;

else if (result == false)
    return false;
```

```
return value_1 == value_2;
```

(advertencia: esto es un XOR)

Temario

- Introducción a Python
- Primeros pasos
- Estructuras de control
  - Condicionales IF-ELSE
  - Condicionales IF-ELIF-ELSE
  - Ejercicios



Ejercicio 1

Tiempo : 15 minutos

Realice un programa que le pregunte al usuario si le gustan los dulces y muestre el mensaje “A mi también” si escribe algunas de las siguientes respuestas: { Sí, Si, si, sí }

▼ Preguntas

¿Cómo solicito los datos al usuario?

¿Cómo creo una condición compuesta?

¿Cómo se muestra el resultado?

Ejercicio 1

Tiempo : 15 minutos

Realice un programa que le pregunte al usuario si le gustan los dulces y muestre el mensaje “A mi también” si escribe algunas de las siguientes respuestas: { Sí, Si, si, sí }

Solución propuesta #1

```
print('Le gustan los dulces')
rsp= input()

if rsp == 'Si' or rsp =='Sí' or rsp == 'si' or rsp == 'sí':
   print('A mi tambien')
```

Solución propuesta #2

```
if input('Te gustan los dulces') in ('Sí','Si','si', 'sí'):
   print("A mi tambien")
```

Ejercicio 2

Tiempo : 5 minutos

Realice un programa que pregunte un número al usuario y luego imprima en pantalla si el número es par o impar

▼ Preguntas

¿Cómo solicito los datos al usuario?

¿Cómo sé que un número es par?

¿Cómo se muestra el resultado?

Ejercicio 2

Tiempo : 5 minutos

Realice un programa que pregunte un número al usuario y luego imprima en pantalla si el número es par o impar

Solución propuesta

```
print('Ingresa un número')
num = int(input())

if num%2==0:
   print('El numero es par')
else:
   print('El numero es impar')
```

Ejercicio 3

Tiempo : 15 minutos

Twitter tiene un límite de 28 caracteres. Realice un programa que permita al usuario ingresar un texto e imprimir por pantalla “Escribe algo” si el usuario no escribió un texto, “OK” si el texto tiene 28 caracteres o menos, o “Texto muy largo” en caso contrario

▼ Hint

Usa la función len() para obtener el largo de un texto

```
saludo = 'hola'
largo = len(saludo)
print('El largo de', saludo, 'es', largo)
```

Imprimirá en pantalla ‘El largo de hola es 4’

Ejercicio 3

Tiempo : 15 minutos

Twitter tiene un límite de 28 caracteres. Realice un programa que permita al usuario ingresar un texto e imprimir por pantalla “Escribe algo” si el usuario no escribió un texto, “OK” si el texto tiene 28 caracteres o menos, o “Texto muy largo” en caso contrario

Solución propuesta

```
print('Escribe un mensaje')
texto = input()
largo = len(texto)

if largo == 0:
   print('Escribe algo')
elif largo>0 and largo<28:
   print('OK')
else:
   print('Texto muy largo')
```


WHEN YOU START PROGRAMMING
AFTER A WHILE
Hello World!
PYCODERS
@pycoders
Data Structures