

FACULTAD DE
INGENIERÍA Y CIENCIAS



UAI
UNIVERSIDAD ADOLFO IBÁÑEZ

INTRODUCCIÓN A PYTHON

INTRODUCCIÓN

Miguel Carrasco
miguel.carrasco@uai.cl



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.



Guido Van Rossum

(creador de Python)

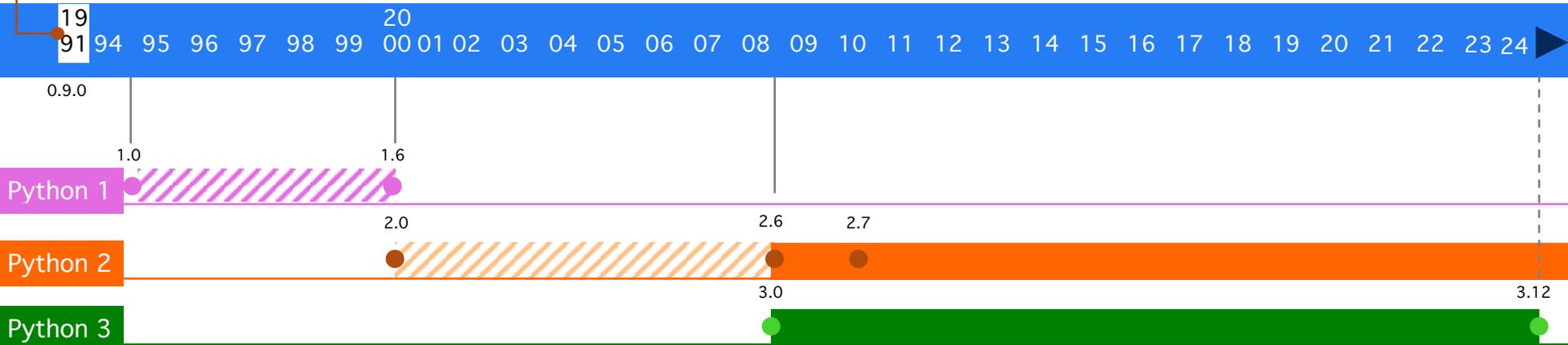
“es un científico de la computación conocido por ser el autor del lenguaje de programación Python en 1991”
(fuente: Wikipedia)



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:

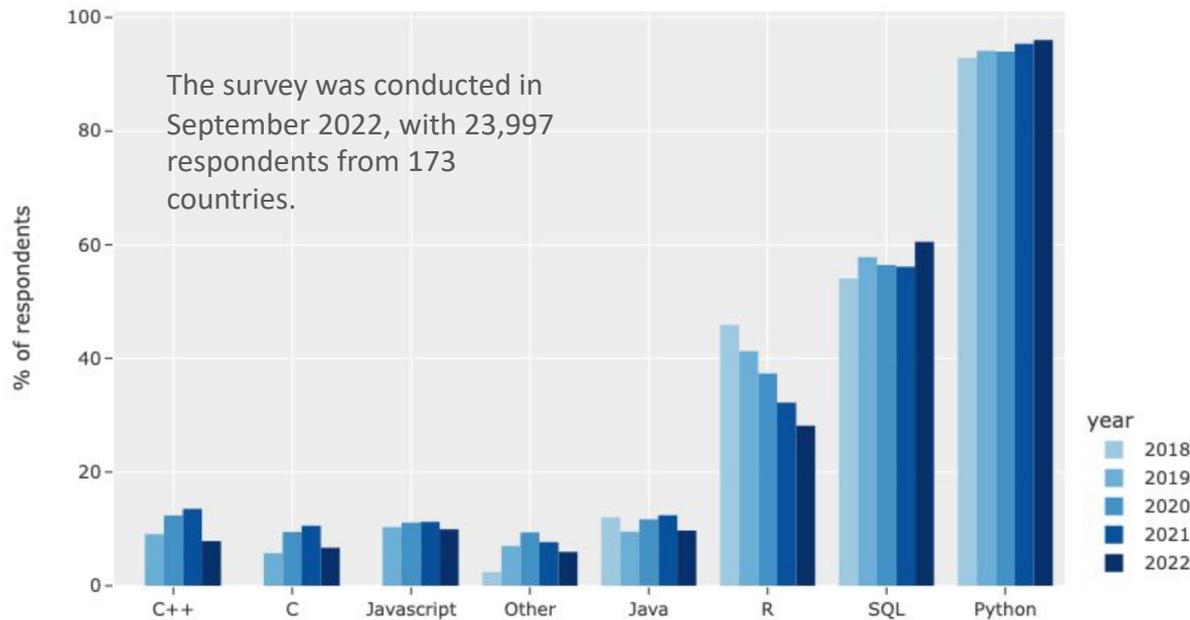
- Facilidad de lectura (código interpretado)
- Sistema de tipo dinámico
- Manejo automático de administración de memoria



2022

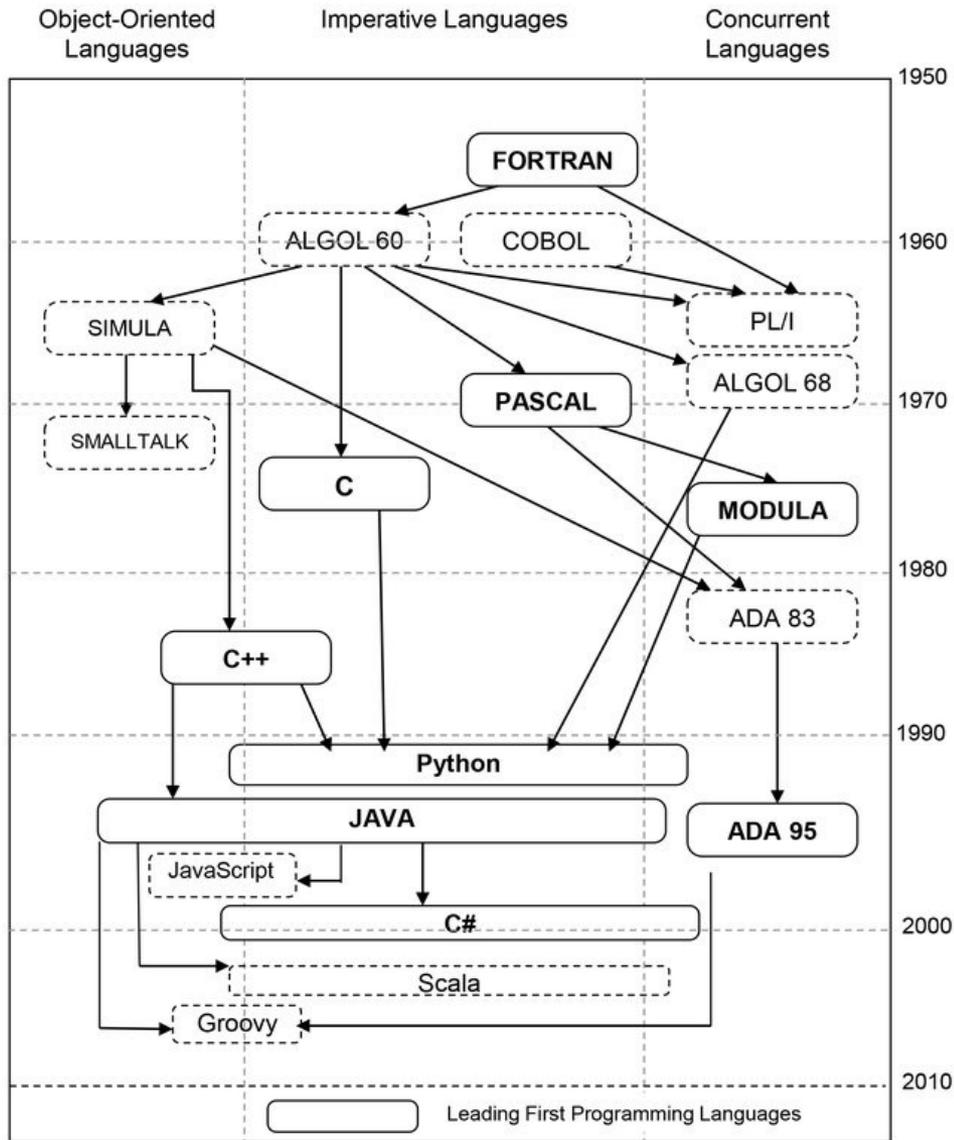
Kaggle DS & ML Survey 2022

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



fuentes: <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

Te damos la bienvenida a Colaboratory

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda

Compartir Configuración Perfil

Índice

- Introducción
- Ciencia de datos
- Aprendizaje automático
- Más recursos
- Ejemplos de aprendizaje automático
- Sección

+ Código + Texto Copiar en Drive

Conectar Editen

¿Qué es Colaboratory?

Colaboratory, o Colab, te permite escribir y ejecutar código de Python en un navegador, con

- Sin configuración requerida
- Acceso gratuito a GPU
- Facilidad para compartir

No importa si eres **estudiante**, **científico de datos** o **investigador de IA**, ya que Colab facilita tu trabajo. Mira [este video introductorio sobre Colab](#) para obtener más información al respecto, o bien comienza a usarlo más abajo.

Introducción

El documento que estás leyendo no es una página web estática, sino un entorno interactivo denominado **notebook de Colab**, que permite escribir y ejecutar código.

Por ejemplo, esta es una **celda de código** con una secuencia de comandos Python corta que calcula un valor, lo almacena en una variable y devuelve el resultado:

```
[ ] 1 seconds_in_a_day = 24 * 60 * 60
     2 seconds_in_a_day
```

86400

A fin de ejecutar el código en la celda anterior, haz clic en él para seleccionarlo y, luego, presiona el botón de reproducción ubicado a la izquierda del código o usa la combinación de teclas "Command/Ctrl + Intro". Para editar el código, solo haz clic en la celda y comienza a editar.

Las variables que defines en una celda pueden usarse en otras:

► <https://colab.research.google.com/>

offline

Python

PSF

Docs

PyPI

Jobs

Community



GO

Socialize

About

Downloads

Documentation

Community

Success Stories

News

Events

Download the latest version for Mac OS X

Download Python 3.6.4

Download Python 2.7.14

Wondering which version to use? [Here's more about the difference between Python 2 and 3.](#)

Looking for Python with a different OS? Python for [Windows](#), [Linux/UNIX](#), [Mac OS X](#), [Other](#)

Want to help test development versions of Python? [Pre-releases](#)



offline



Products

Why Anaconda?

Solutions

Resources

Company

Download



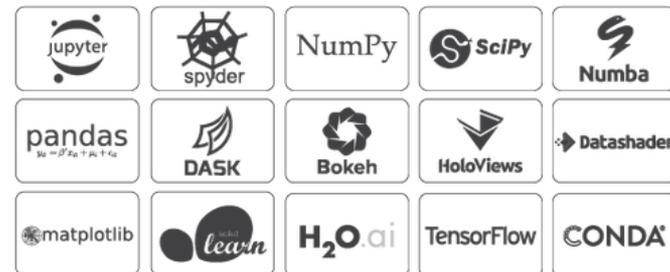
Anaconda Distribution

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**



The screenshot displays the Anaconda Navigator application window. At the top, the title bar reads "Anaconda Navigator". Below the title bar, the main header features the "ANACONDA NAVIGATOR" logo on the left and a "Sign in to Anaconda Cloud" button on the right. A left-hand sidebar contains navigation options: "Home" (selected), "Environments", "Learning", and "Community". At the bottom of the sidebar are links for "Documentation" and "Developer Blog", along with social media icons for Twitter, YouTube, and GitHub.

The main content area is titled "Applications on" and shows a dropdown menu set to "base (root)" and a "Channels" button. A "Refresh" button is located in the top right of this section. The applications are displayed in a grid:

- Spyder 3.3.2**: Scientific PYTHON Development EnviRnment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features. A "Launch" button is visible.
- Glueviz 0.13.3**: Multidimensional data visualization across files. Explore relationships within and among related datasets. An "Install" button is visible.
- 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. An "Install" button is visible.
- RStudio 1.1.456**: A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks. An "Install" button is visible.
- VS Code 1.33.1**: Streamlined code editor with support for development operations like debugging, task running and version control. An "Install" button is visible.

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

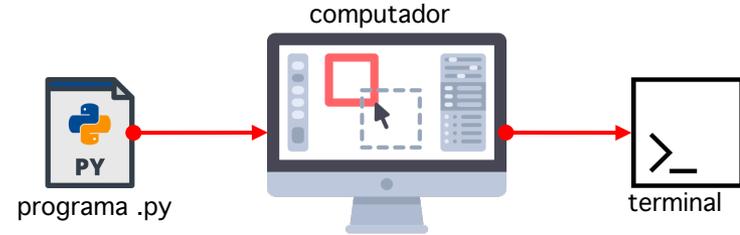


```
self.FidValue = OrderedDict(sorted(self.items(), key=lambda item: item[0]))
#Read item in dictionary
for key, value in item.FidValue.items():
    typeOfFID = mapFidType.get(key)
    if (typeOfFID == "DATE"):
        d = datetime.datetime.strptime(str(value), "%Y-%m-%d")
        dataCal = datetime.date.strptime(str(value), "%Y-%m-%d")
        FidAndValue = FidAndValue + value
    else: FidAndValue = FidAndValue + value
```

```
try:
    start = date(int(self.start_year.get(self.months.index(self.start_month)),
                int(self.start_day.get(self.months.index(self.start_month))),
                int(self.start_year.get(self.months.index(self.start_month))))
    end = date(int(self.end_year.get(self.months.index(self.end_month)),
                int(self.end_day.get(self.months.index(self.end_month))),
                int(self.end_year.get(self.months.index(self.end_month))))
```



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



spyder

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 Created on Tue Jul 2 14:04:41 2019
5
6 @author: mlacarrasco
7
8
9 |
```

Usage

Here you can get help of any object by pressing **Cmd+I** in front of it, either on the Editor or the Console.

Help can also be accessed by pressing **Cmd+H** in front of a variable name. You can also type **> Help.**

Variable explorer | File explorer | Help

IPython console

Console 1/A

Python 3.7.0 (default, Jun 28 2018, 07:39:16)
Type "copyright", "credits" or "license" for more information.

IPython 7.2.0 -- An enhanced Interactive Python.

In [1]:

▼ 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.

2

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3 """
4 Created on Tue Jul  2 14:04:41 2019
5
6 @author: mlacarrasco
7 """
8
9 print('hola')
```

1

Secuencia de pasos:

1 → 2 → 3

3

Python console

```
Python 3.7.0 (default, Jun 28 2018, 07:39:16)
Type "copyright", "credits" or "license" for more information.

IPython 7.2.0 -- An enhanced Interactive Python.

In [1]: runfile('/Users/mlacarrasco/Dropbox/_BACKUP/Asignaturas/
2019/sem_01/MagisterDataScience/clases/modulo_1_control/code/
test.py', wdir='/Users/mlacarrasco/Dropbox/_BACKUP/Asignaturas/2019/
sem_01/MagisterDataScience/clases/modulo_1_control/code')
hola

In [2]:
```

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.

```
1#!/usr/bin/env python3
2# -*- coding: utf-8 -*-
3
4Created on Tue Jul  2 14:04:41 2019
5
6@author: mlacarrasco
7
8
9 print('este es mi primer texto)
```

File `"/anaconda3/lib/python3.7/site-packages/spyder_kernels/customize/spydercustomize.py"`, line 704, in `runfile`
`execfile(filename, namespace)`

File `"/anaconda3/lib/python3.7/site-packages/spyder_kernels/customize/spydercustomize.py"`, line 108, in `execfile`
`exec(compile(f.read(), filename, 'exec'), namespace)`

File `"/Users/mlacarrasco/Dropbox/_BACKUP/Asignaturas/2019/sem_01/MagisterDataScience/clases/modulo_1_control/code/test.py"`, line 9
`print('este es mi primer texto)`
^
SyntaxError: EOL while scanning string literal

In [3]:
In [3]:

Run file Permissions: RW End-of-lines: LF Encoding: UTF-8 Line: 9 Column: 31 Memory: 63 %

✗ print('este es mi primer texto)

👍 print('este es mi primer 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

The image shows a screenshot of the Spyder Python IDE. The editor window displays a Python script with the following code:

```
1#!/usr/bin/env python3
2# -*- coding: utf-8 -*-
3"""
4Created on Tue Jul  2 14:04:41 2019
5
6@author: mlacarrasco
7"""
8
9print('hola buenos'+ 'amigos')
```

A blue callout box points to the concatenation operation in line 9, containing the text "Cadena de caracteres".

Below the code editor, there is a visual representation of the input and output of the program:

INPUT: h o l a b u e n o s + a m i g o s

OUTPUT: h o l a b u e n o s a m i g o s

▼ 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

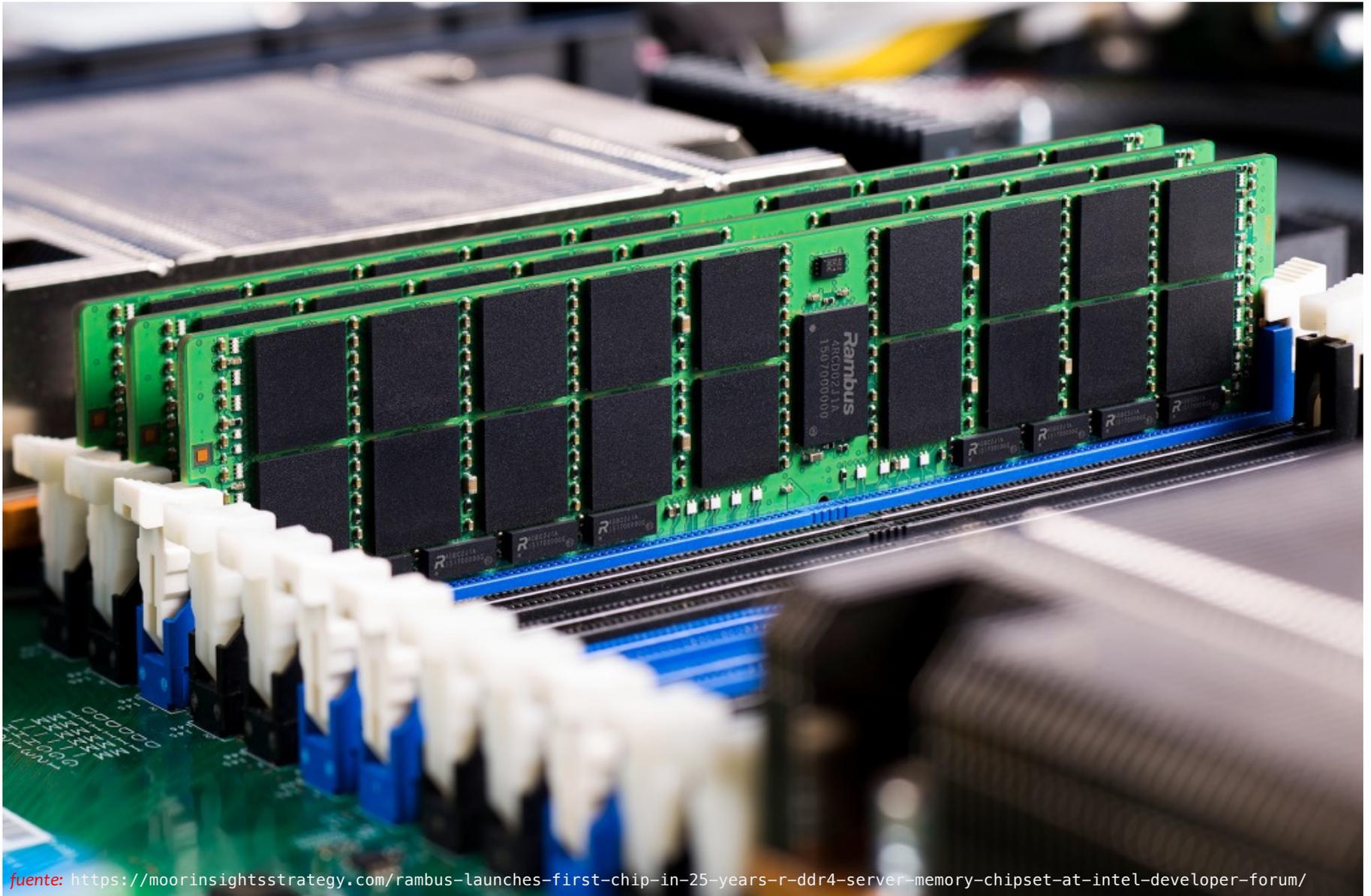
The screenshot shows the Spyder Python IDE interface. The editor window displays a Python script with the following code:

```
1#!/usr/bin/env python3
2# -*- coding: utf-8 -*-
3''''
4Created on Tue Jul  2 14:04:41 2019
5
6@author: mlacarrasco
7''''
8
9print('hola buenos', 'amigos')
```

A blue callout box points to the print statement with the text "Cadena de caracteres". Below the code, the input and output are shown:

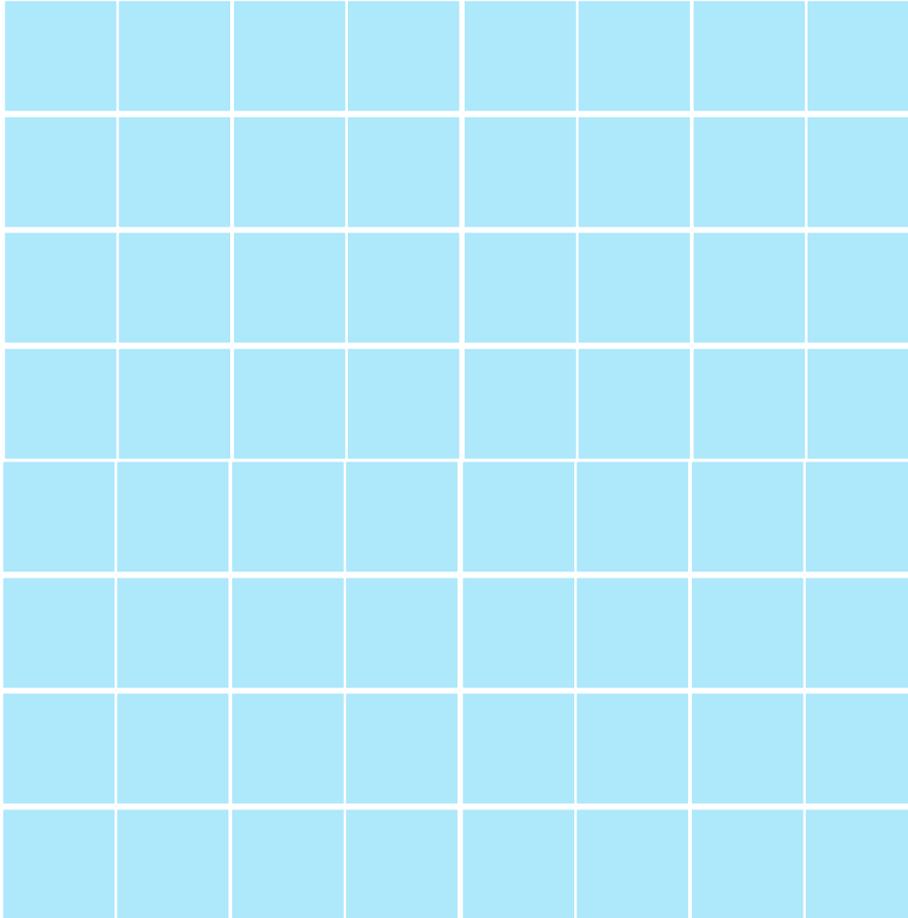
INPUT h o l a b u e n o s , a m i g o s

OUTPUT h o l a b u e n o s a m i g o s



fuerite: <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

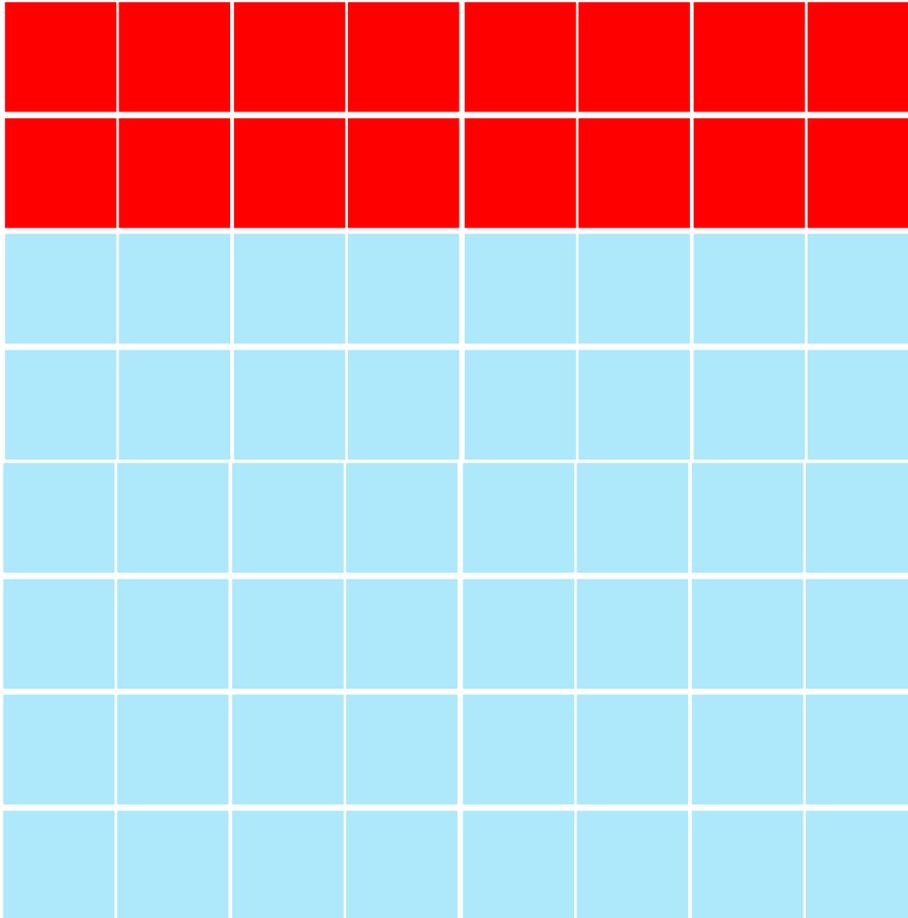
Supongamos que la memoria que tenemos dispone de 64 bloques.

Cada bloque puede guardar un número de 8 bits.

¿Cuántos bytes podemos almacenar?



► Creación de variables



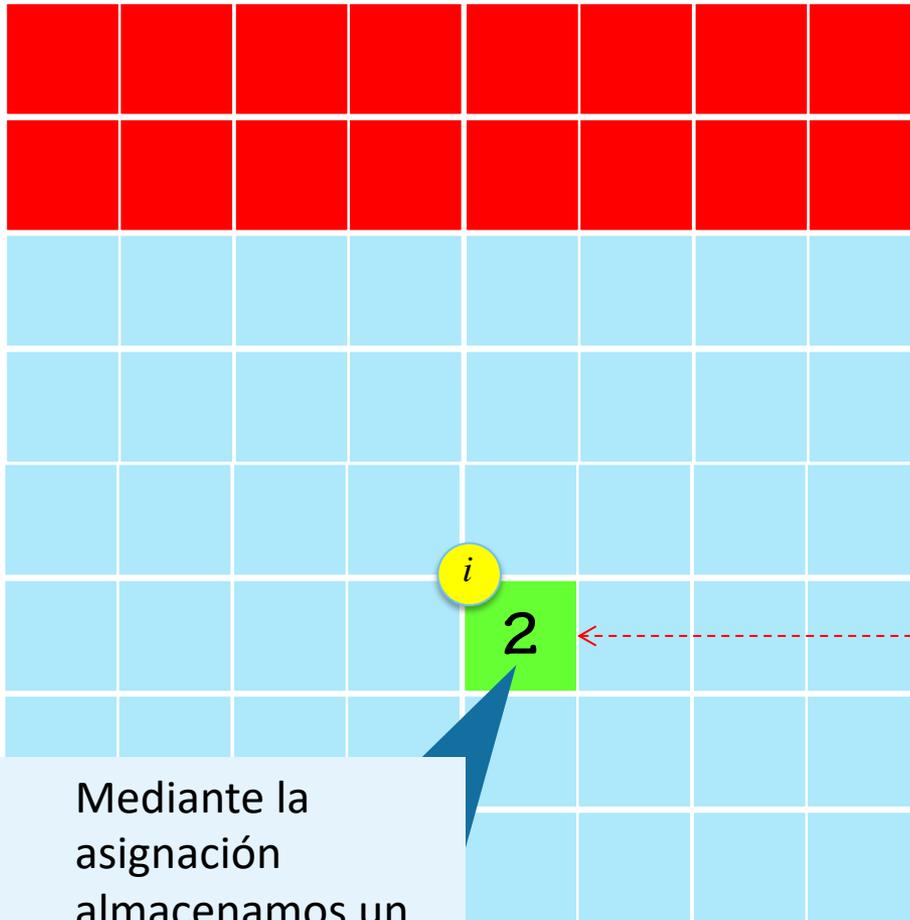
vista de bloques en la memoria del computador

Cuando encendemos 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

La asignación es siempre del lazo derecho al izquierdo



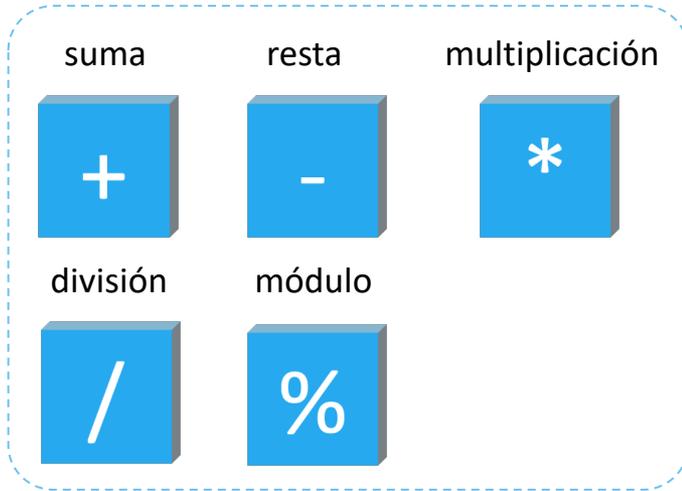
Mediante la asignación almacenamos un valor en la memoria

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



```
self.FidValue = OrderedDict(sorted(self.items(), key=lambda item: item[0]))  
#Read item in dictionary  
for key, value in item.FidValue.items():  
    typeOfFID = mapFidType.get(key)  
    if (typeOfFID == "DATE"):  
        d = datetime.datetime.strptime(str(value), "%Y-%m-%d")  
        dataCal = datetime.date.strptime(str(value), "%Y-%m-%d")  
        FidAndValue = FidAndValue + value  
    else: FidAndValue = FidAndValue + value
```

```
try:  
    start = date(int(self.start_year.get(self.start_year.index(self.start_year)),  
                int(self.start_day.get(self.start_day.index(self.start_year))),  
                int(self.start_month.get(self.start_month.index(self.start_year))))  
  
    end = date(int(self.end_year.get(self.end_year.index(self.start_year)),  
              int(self.end_day.get(self.end_day.index(self.start_year))),  
              int(self.end_month.get(self.end_month.index(self.start_year))))
```



Operaciones aritméticas

Python permite efectuar las siguientes operaciones aritméticas

1

```
w = 100
i = 2 + w
```

2

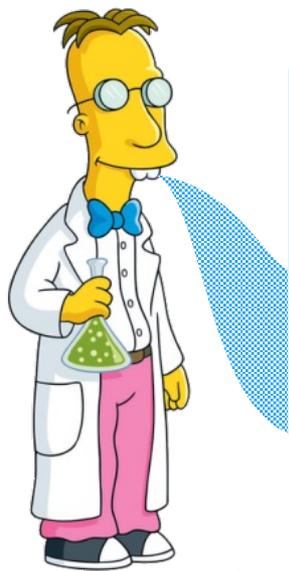
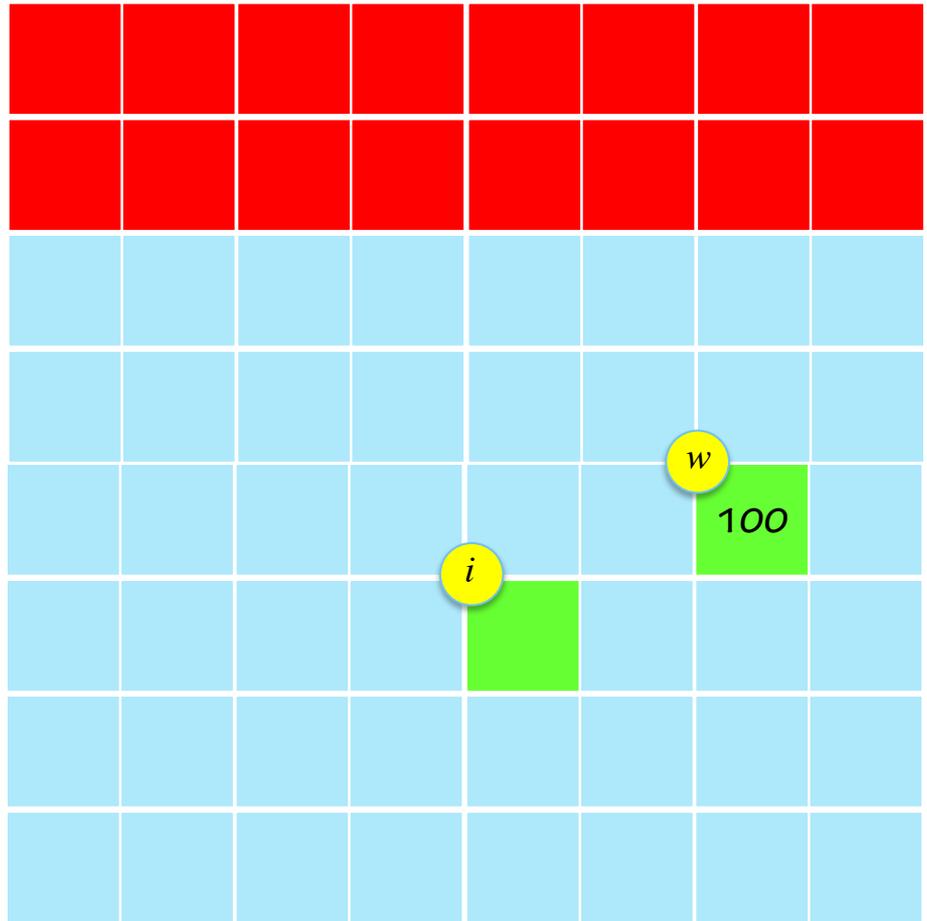
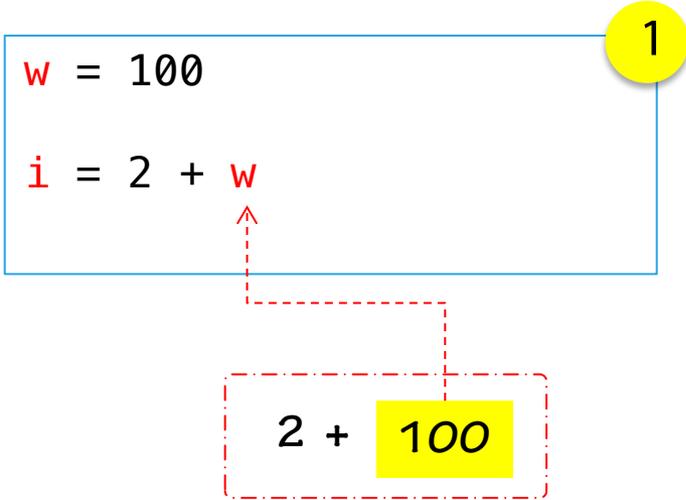
```
w = 100
i = (2 + w)
i = i - 3
```

3

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

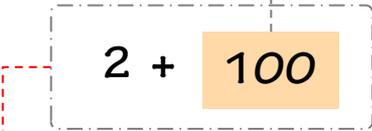
4

```
w = 100
i = 2
w = w / 2
i = w - i
```

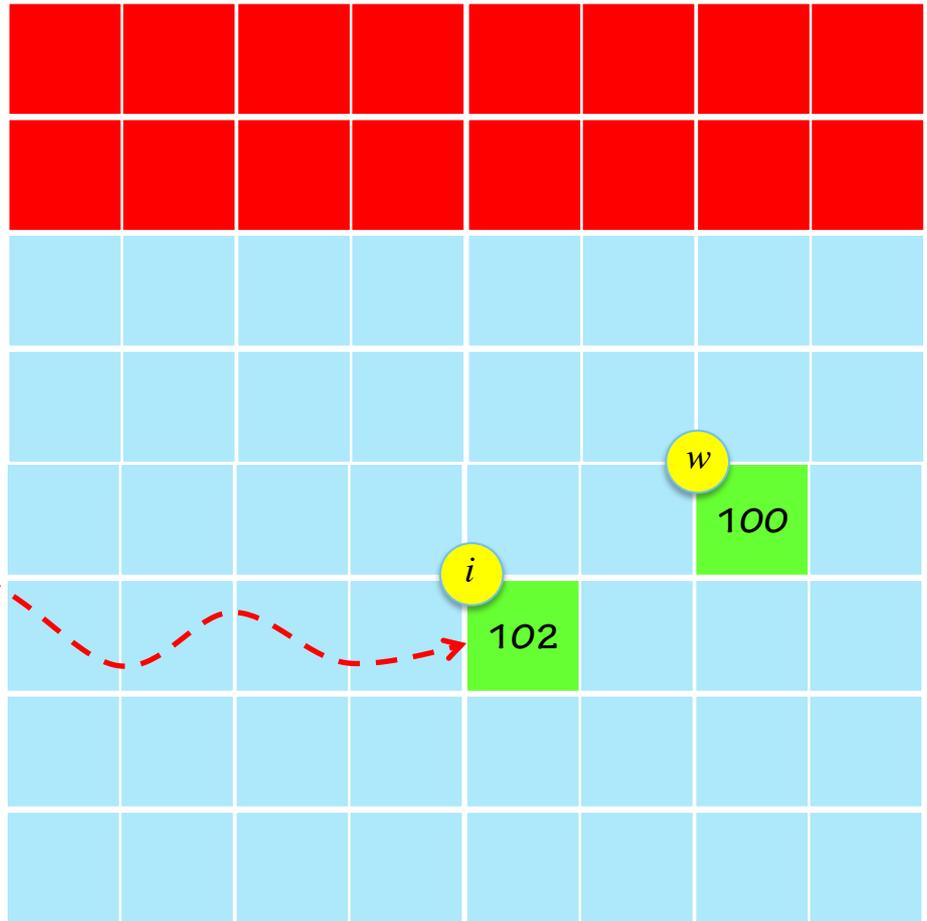



Siempre debe calcularse (evaluarse) la parte derecha de la asignación y luego se almacena el resultado al lado izquierdo

```
w = 100  
i = 2 + w
```

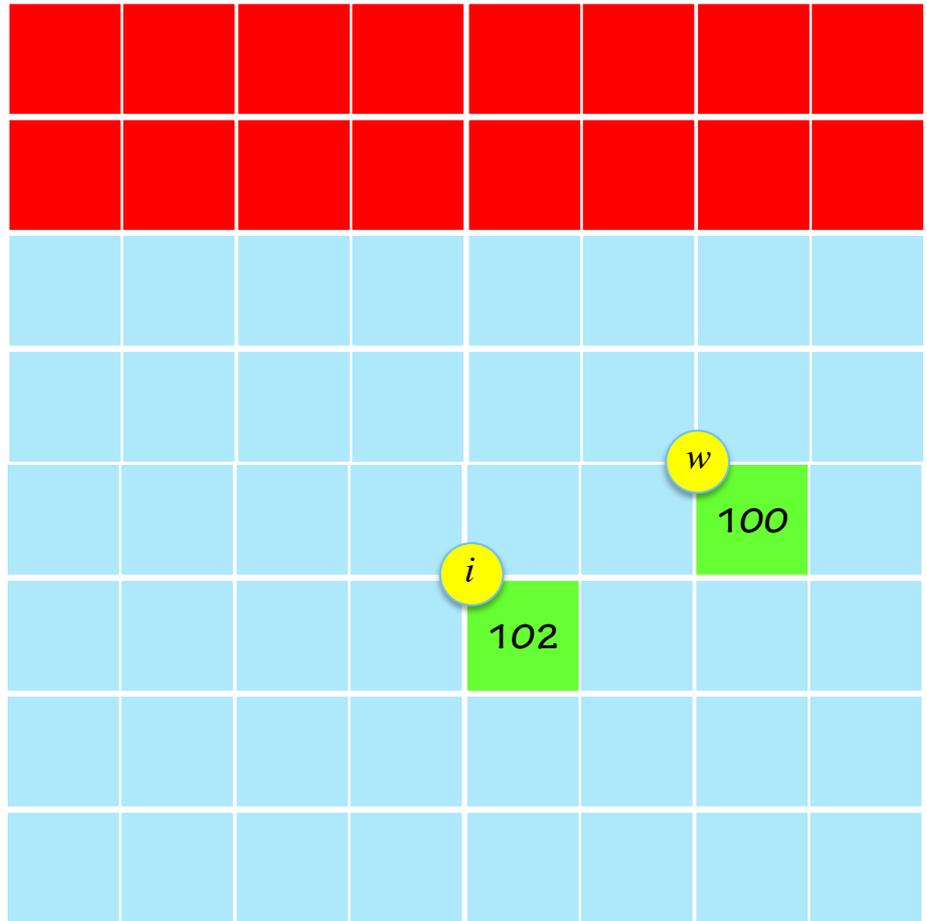
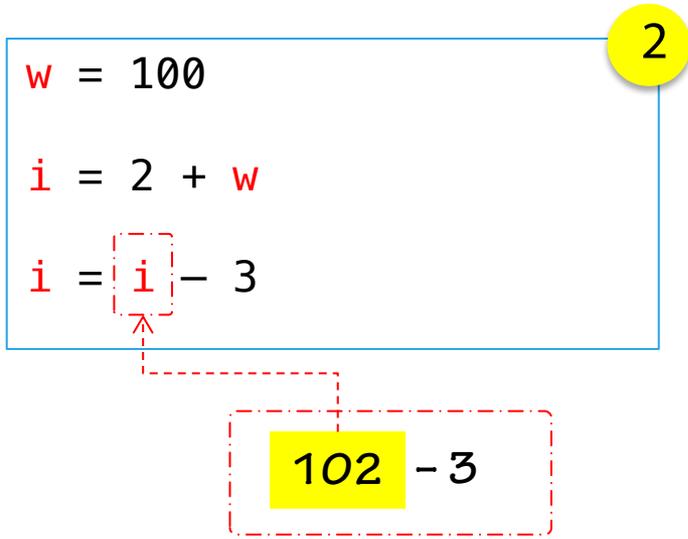


102



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

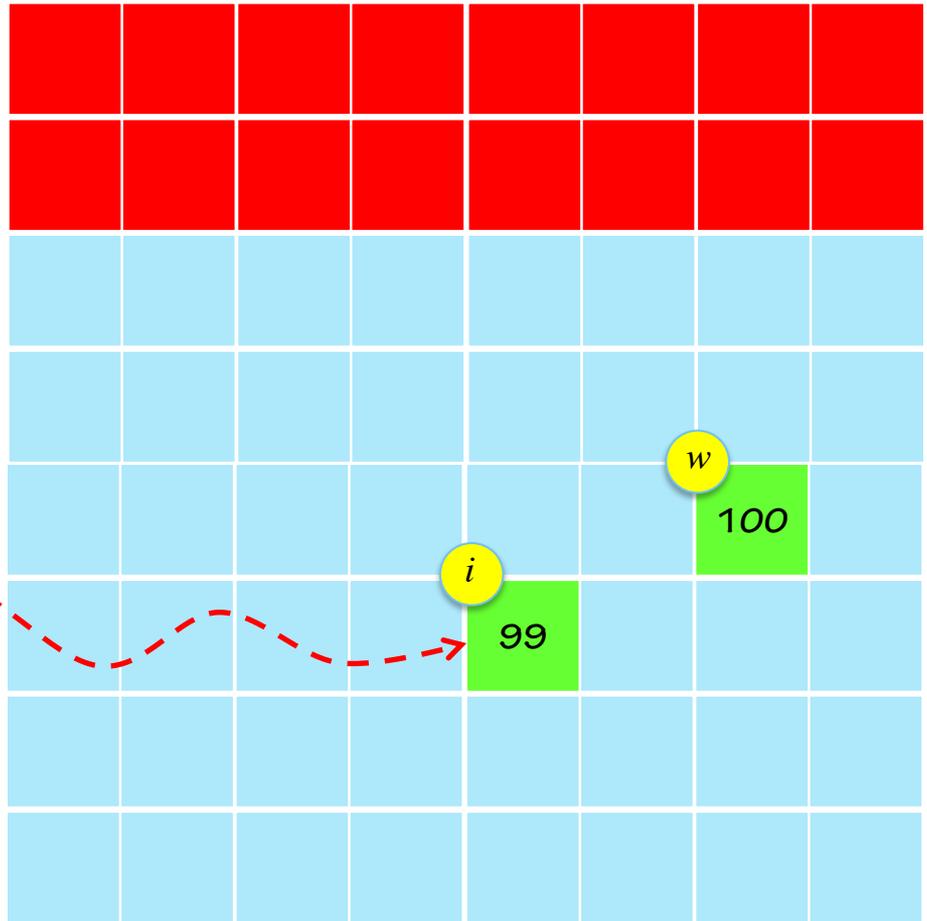
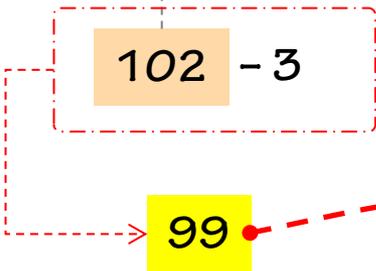




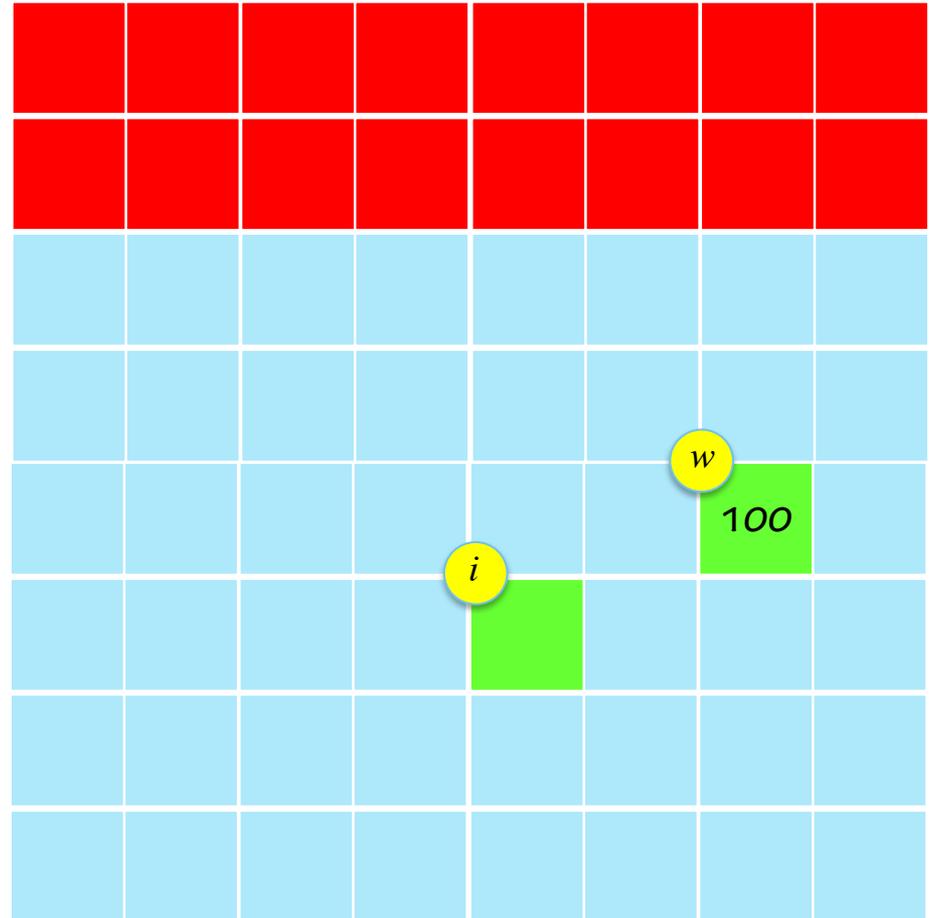
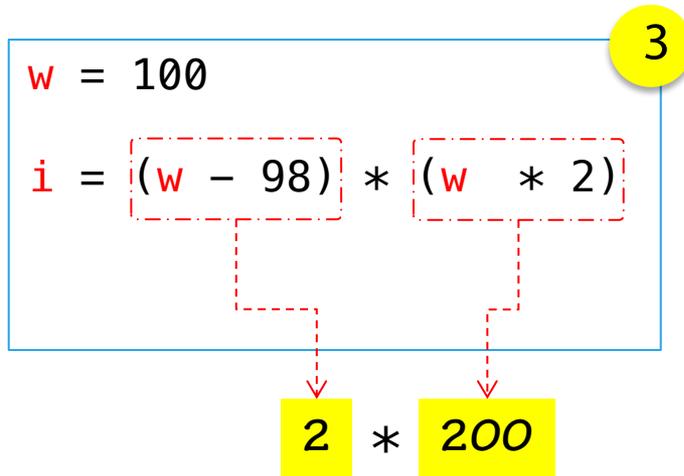
Lección 3

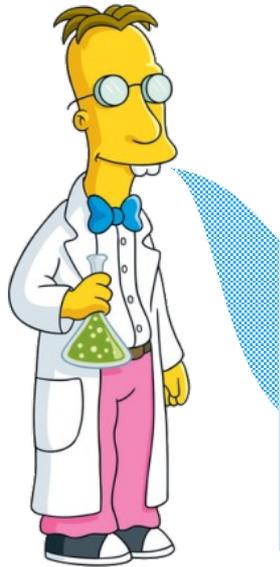
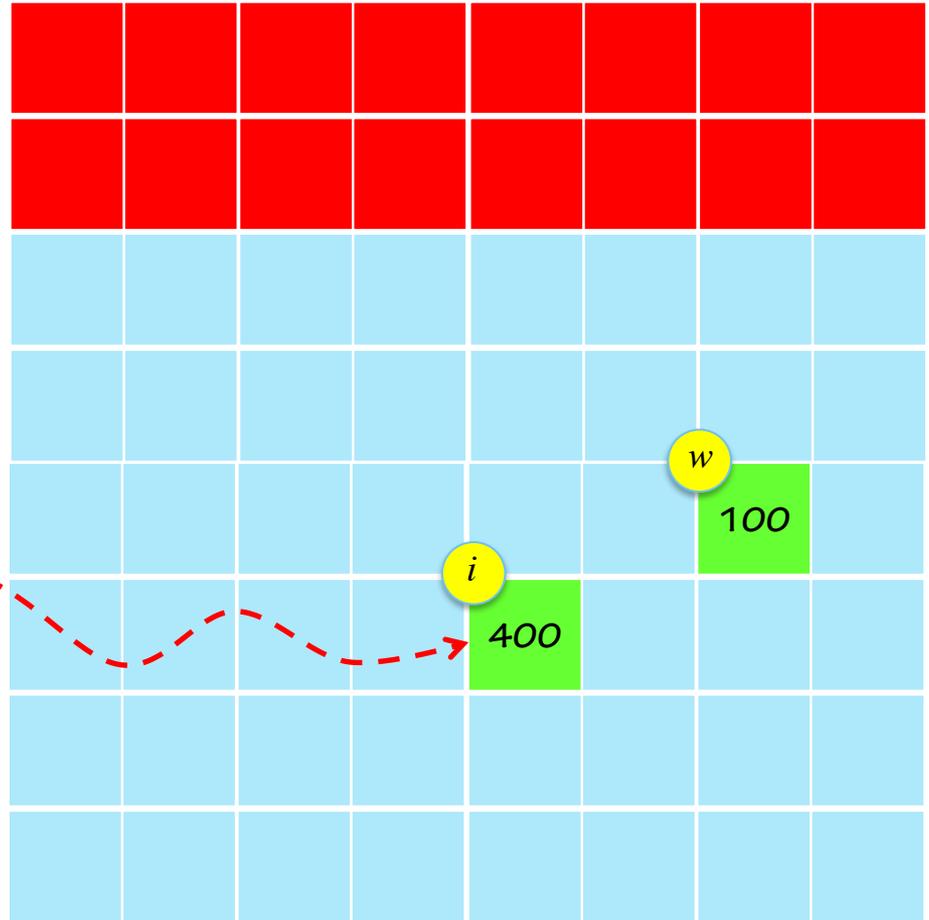
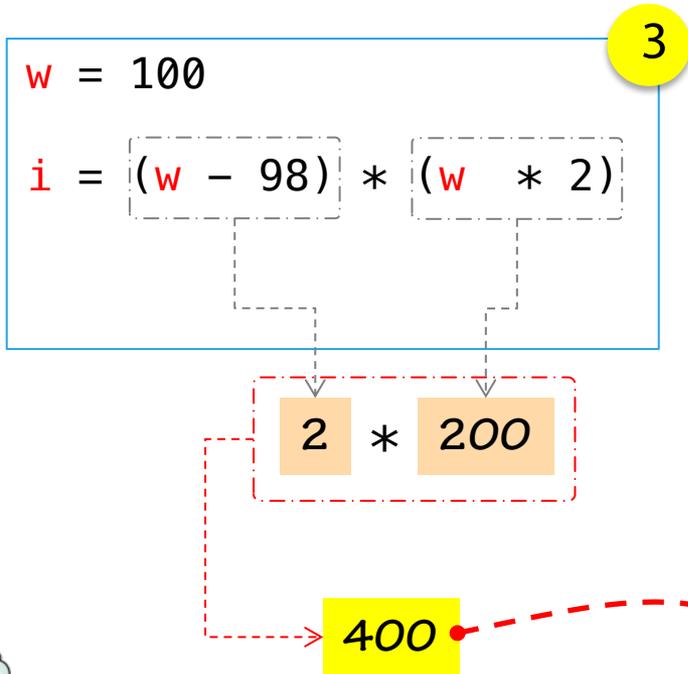
operaciones aritméticas

```
w = 100  
i = 2 + w  
i = i - 3
```



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

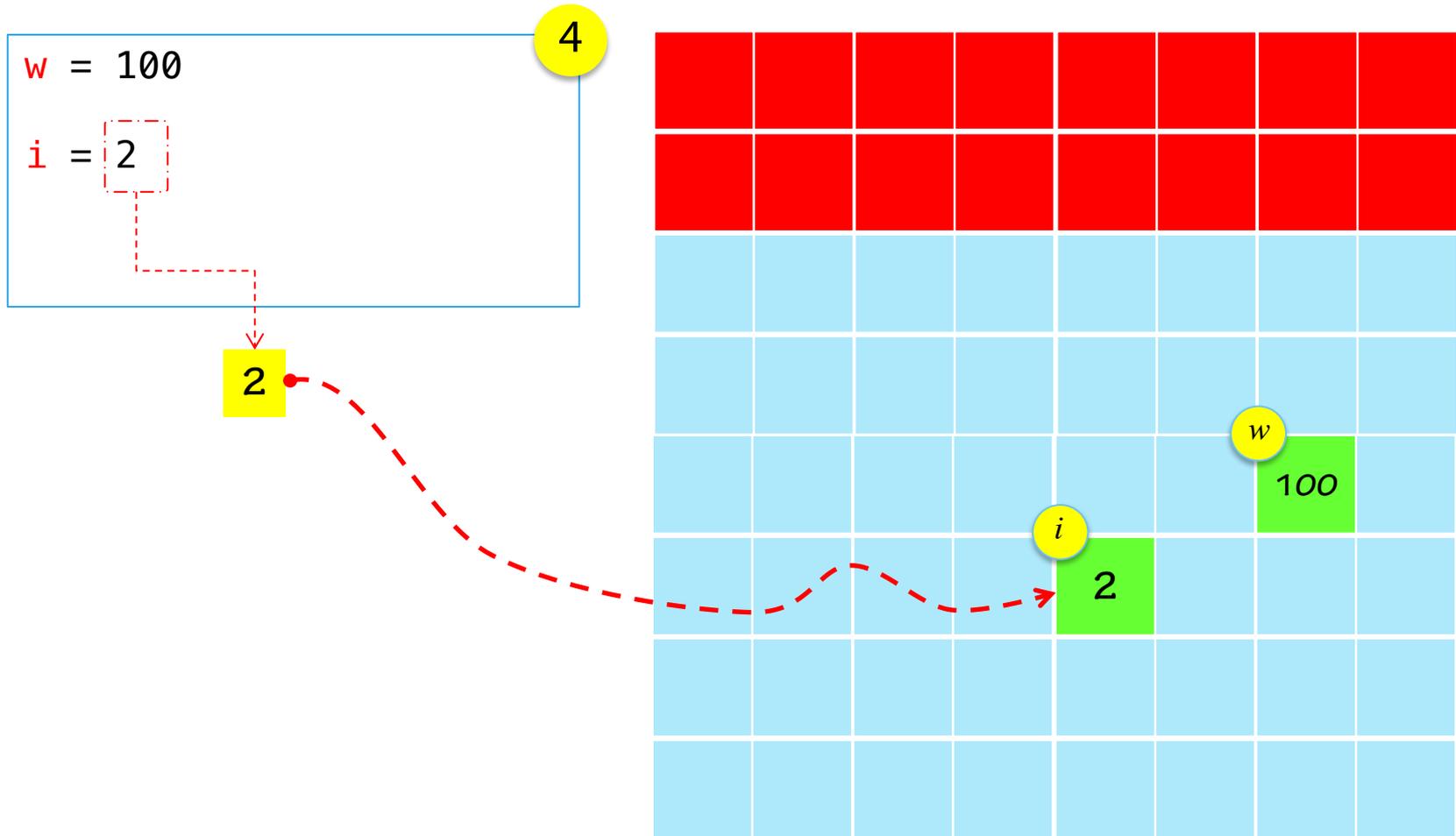


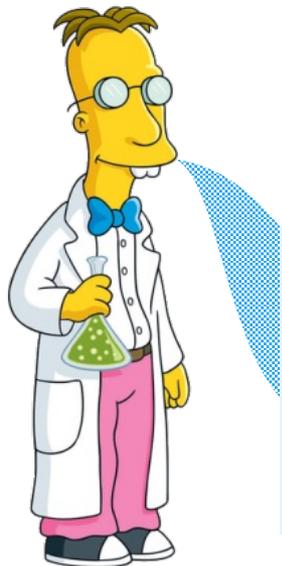
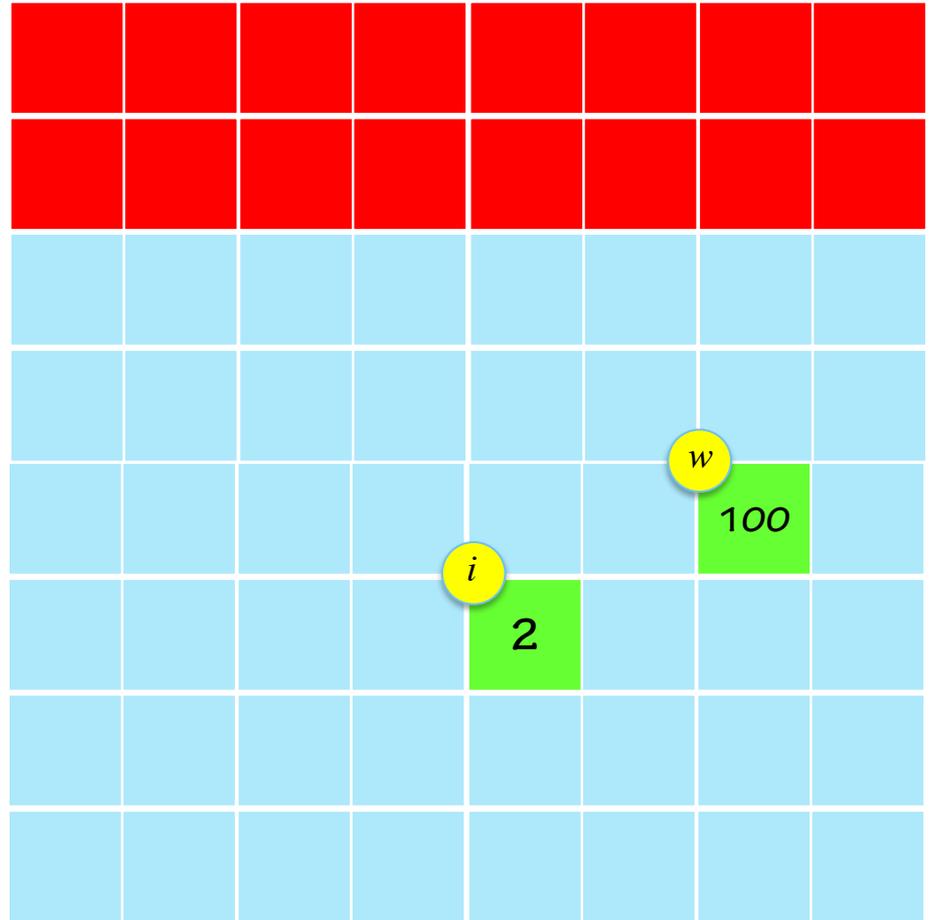
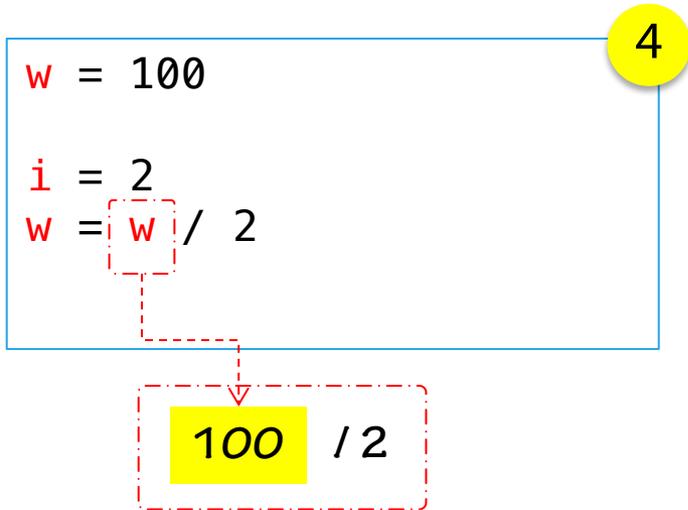


Recuerde, siempre debe terminar de calcular las operaciones y luego se asigna a la variable definida.

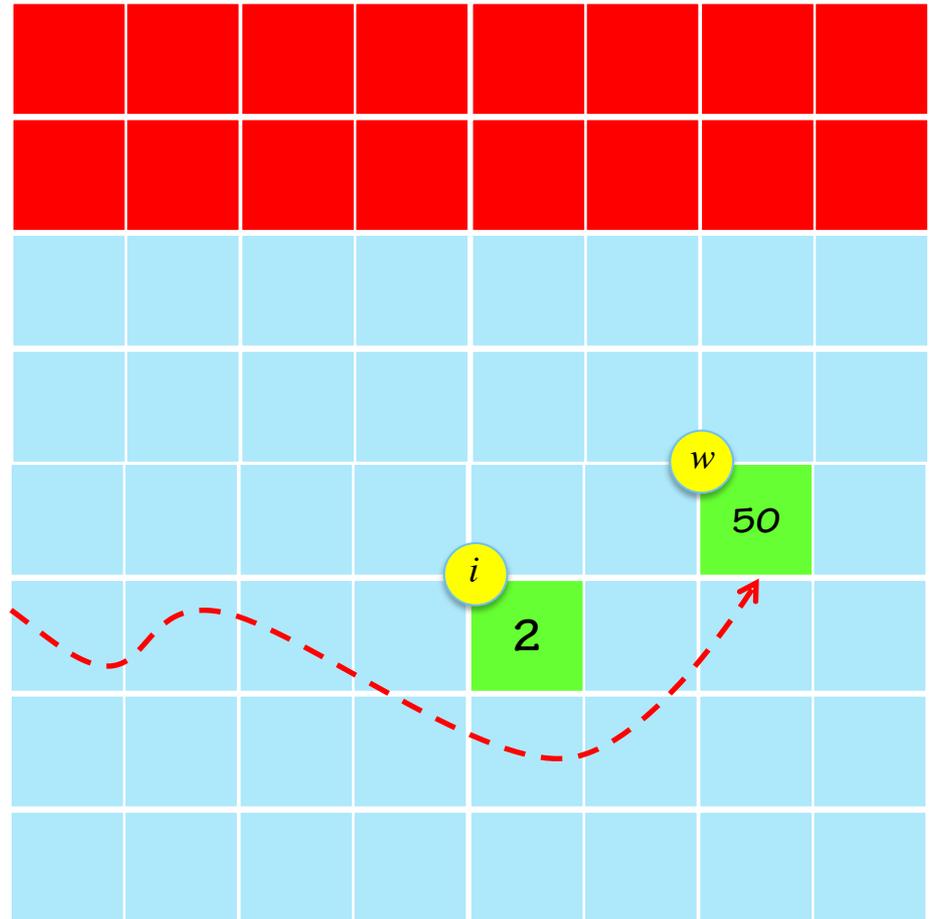
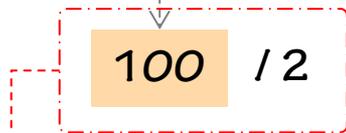
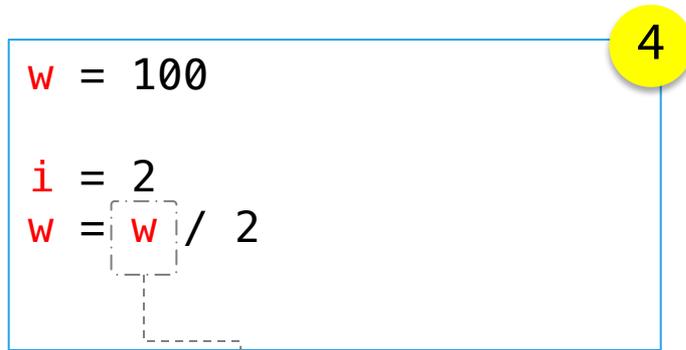
Lección 3

operaciones aritméticas



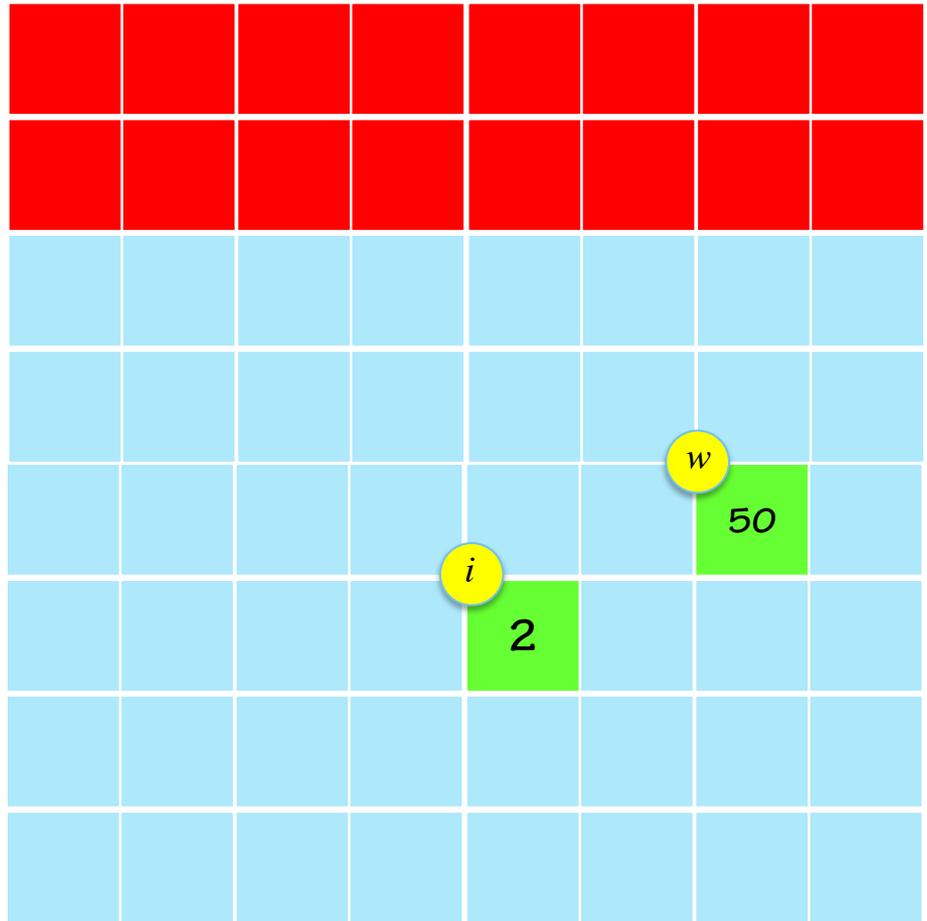
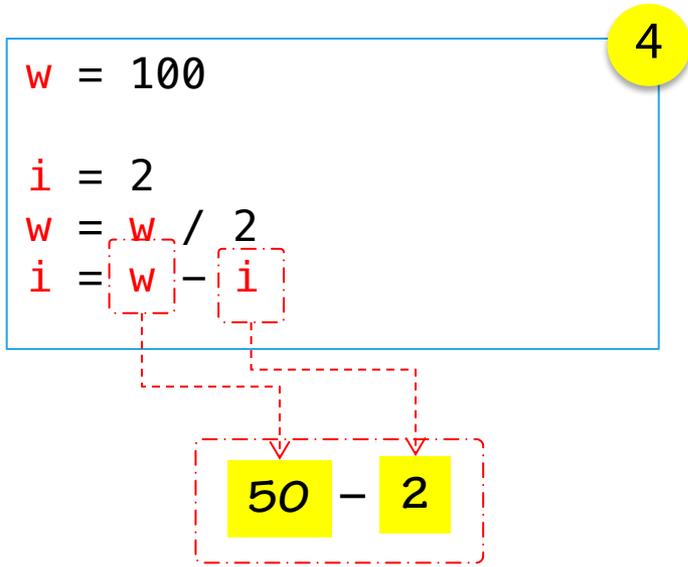


Recuerde, siempre debe terminar de calcular las operaciones y luego se asigna a la variable definida.



Lección 3

operaciones aritméticas

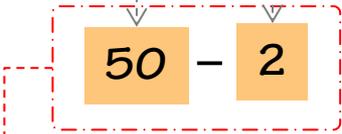


Lección 3

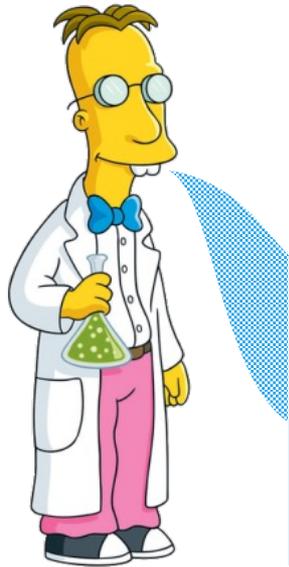
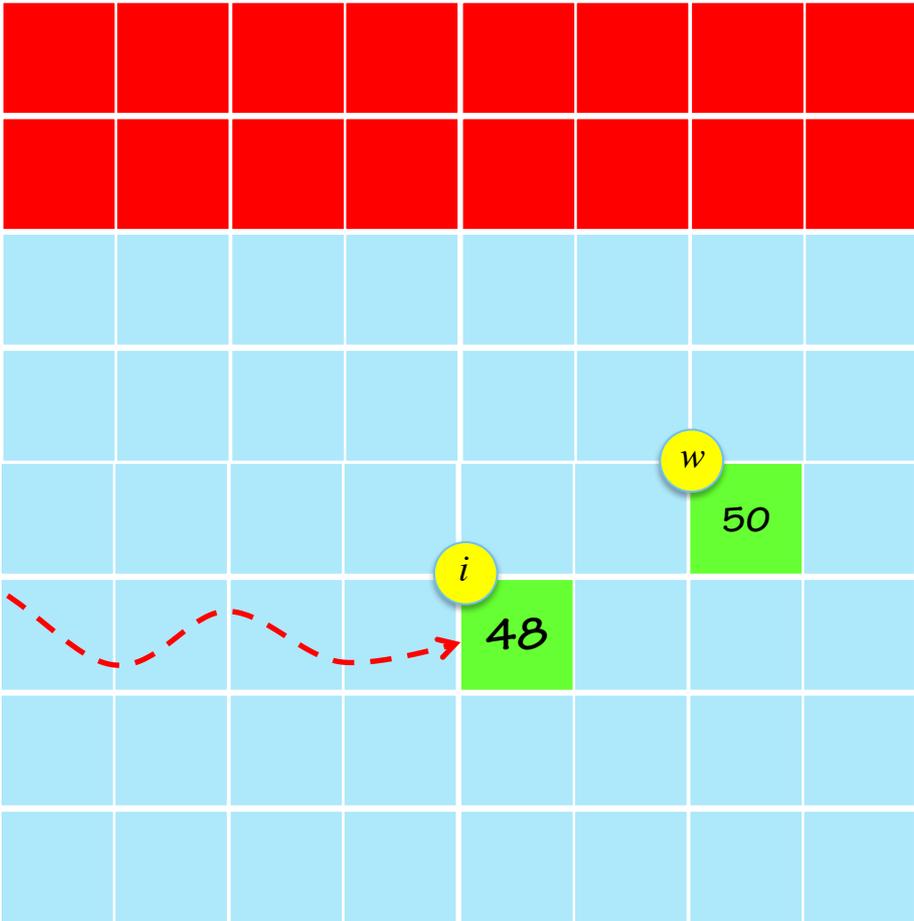
operaciones aritméticas

```
w = 100  
i = 2  
w = w / 2  
i = w - i
```

4

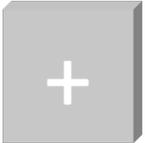


48



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

suma



resta



división



módulo



multiplicación



$a \% b = m$ → $a = b \times k + m$

$11 = 2 \times 5 + 1$
 $18 = 2 \times 9 + 0$

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

1^{ro}

paréntesis



2^{do}

exponente



3^{ro}

multipli
cación



izquierda a derecha

división



módulo



4^{to}

izquierda a derecha

suma



resta





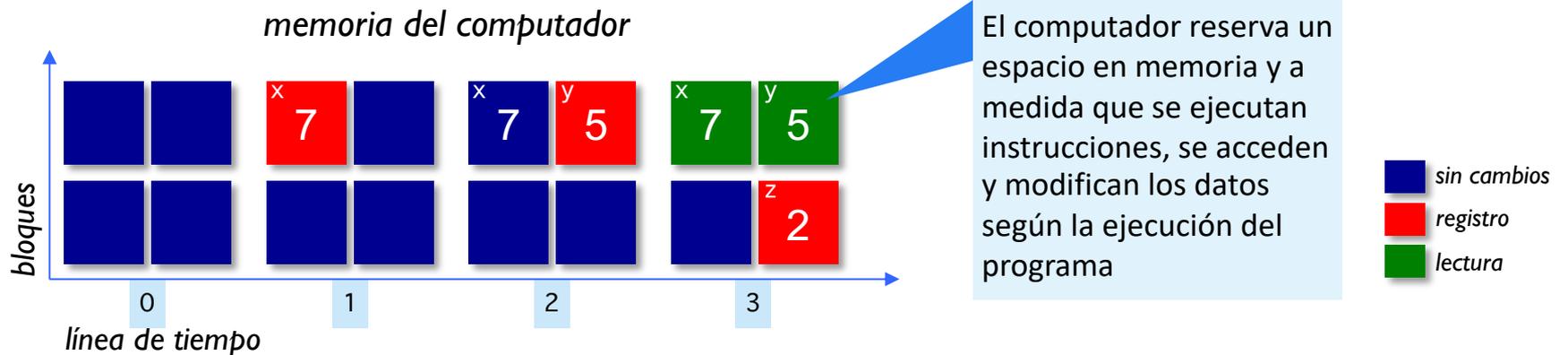
Recordemos que una variable representa **un espacio de memoria en el computador** que puede ser modificado en el tiempo y permite el registro y acceso a los datos.

Matemáticas

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

Python

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





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.

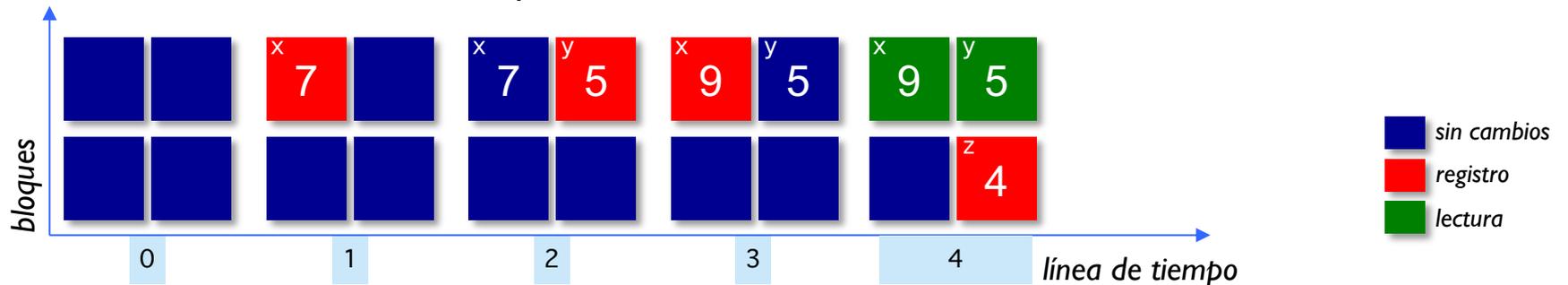
Matemáticas

```
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)
```

memoria del computador



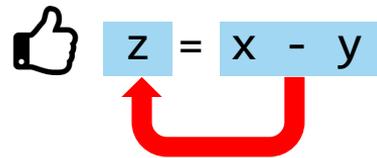
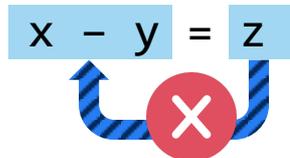


Una variable representa **un espacio de memoria en el computador** que puede ser modificado en el tiempo y permite el registro y acceso a los datos.

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)

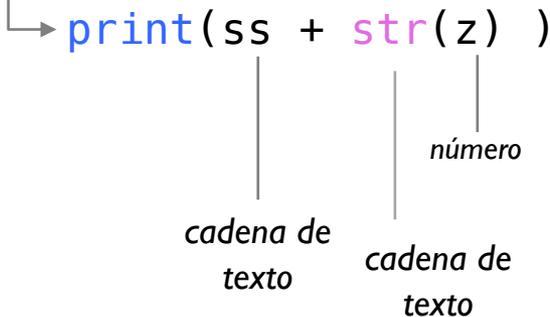
The screenshot shows a Repl.it environment with a Python script named `main.py` and its output. The script contains the following code:

```
1 x = 9
2 y = 7
3 z = x - y
4 ss = "el resultado es: "
5 print(ss + str(z))
```

The output of the script is:

```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
el resultado es: 2
```

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





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?

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



```
self.FidValue = OrderedDict(sorted(self.items(), key=lambda item: item[0]))  
#Read item in dictionary  
for key, value in item.FidValue.items():  
    typeOfFID = mapFidType.get(key)  
    if (typeOfFID == "DATE"):  
        d = datetime.datetime.strptime(str(value), "%Y-%m-%d")  
        dataCal = datetime.date.strptime(str(value), "%Y-%m-%d")  
        FidAndValue = FidAndValue + value  
    else: FidAndValue = FidAndValue + value
```

```
try:  
    start = date(int(self.start_year.get(self.start_year.index(self.start_year)),  
                int(self.start_day.get(self.start_year.index(self.start_year))),  
                int(self.start_month.get(self.start_year.index(self.start_year))))  
  
    end = date(int(self.end_year.get(self.end_year.index(self.end_year)),  
              int(self.end_day.get(self.end_year.index(self.end_year))),  
              int(self.end_month.get(self.end_year.index(self.end_year))))
```

▼ input()

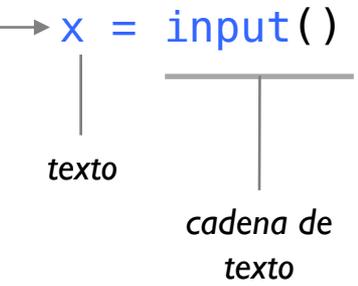
El comando `input()` permite ingresar un número por la terminal. El programa espera que el usuario escriba un número y presione la tecla "enter"

The screenshot shows a Python REPL interface with a code editor on the left and a terminal on the right. The code editor contains the following code:

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

The terminal output shows the execution of the code:

```
Python 3.6.1 (default, Dec 2015, 13:05:11)  
[GCC 4.8.2] on linux  
234  
el numero ingresado es: 234
```



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"

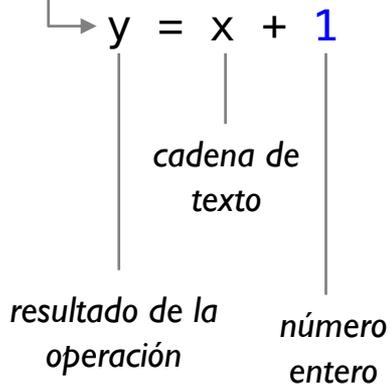
```
Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux

54
Traceback (most recent call last):
  File "python", line 4, in <module>
TypeError: must be str, not int
```

The screenshot shows a Python REPL window with a file named `main.py`. The code in the file is:

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

The error message indicates that the operation `x+1` failed because `x` is a string, not an integer.



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.

The screenshot shows a Python REPL interface with a file named `main.py`. The code in the editor is:

```

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

```

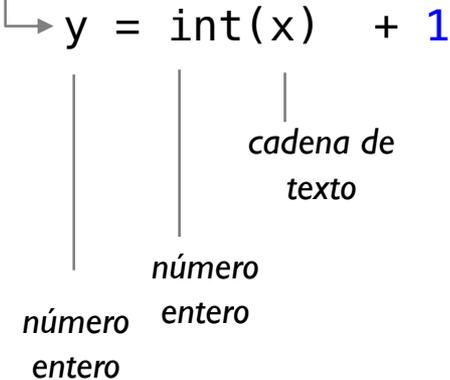
The output in the terminal is:

```

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.



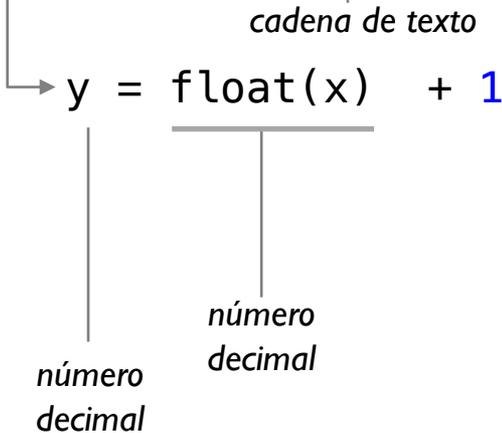
▼ float()

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

```

Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
>
1.2
el resultado es: 2.2
>
    
```

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.



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



```
self.FidValue = OrderedDict(sorted(self.items(), key=lambda item: item[0]))
#Read item in dictionary
for key, value in item.FidValue.items():
    typeOfFID = mapFidType.get(key)
    if (typeOfFID == "DATE"):
        d = datetime.datetime.strptime(str(value), "%Y-%m-%d")
        dataCal = datetime.date.strptime(str(d), "%Y-%m-%d")
        FidAndValue = FidAndValue + value
    else: FidAndValue = FidAndValue + value
```

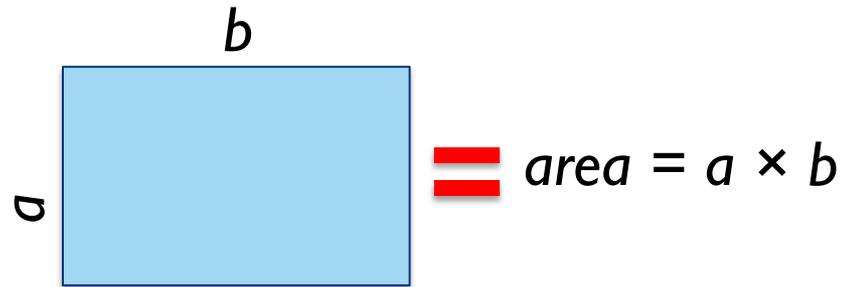
```
try:
    start = date(int(self.start_year.get(
        self.months.index(self.start_month)),
        int(self.start_day.get(
            self.months.index(self.start_month)),
            int(self.start_year.get(
                self.months.index(self.start_month)),
                int(self.start_day.get(
                    self.months.index(self.start_month))

    end = date(int(self.end_year.get(
        self.months.index(self.end_month)),
        int(self.end_day.get(
            self.months.index(self.end_month)),
            int(self.end_year.get(
                self.months.index(self.end_month)),
                int(self.end_day.get(
                    self.months.index(self.end_month))
```



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

Realice un programa que calcule el área de un rectángulo.

- ¿Cómo se calcula el área?→ $area = a \times b$
- ¿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()`

```
repl.it
main.py
1 msg1 = "Ingrese numero: "
2 msg2= "El area es: "
3
4 print(msg1)
5 a= input()
6
7 print(msg1)
8 b= input()
9
10 area= float(a)*float(b)
11 print(msg2 + str(area))

Python 3.6.1 (default, Dec 2015, 13:05:11)
[GCC 4.8.2] on linux
Ingrese numero:
12
Ingrese numero:
5
El area es: 60.0
```


comparación



distinto de



mayor



menor



mayor igual



menor igual



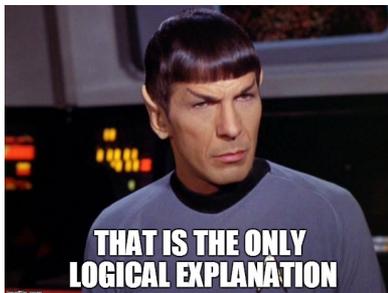
Resultados posibles

verdadero
(TRUE)

1

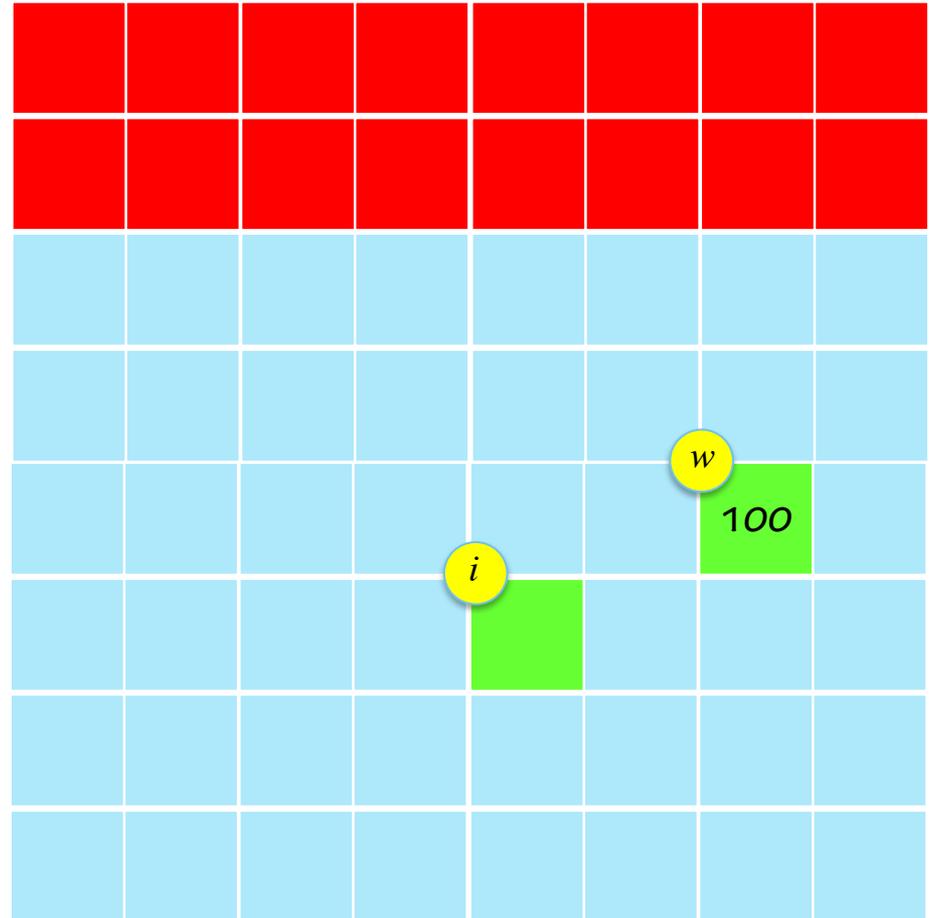
falso
(FALSE)

0



```
w = 100  
i = 99 == w
```

```
99 == 100
```



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

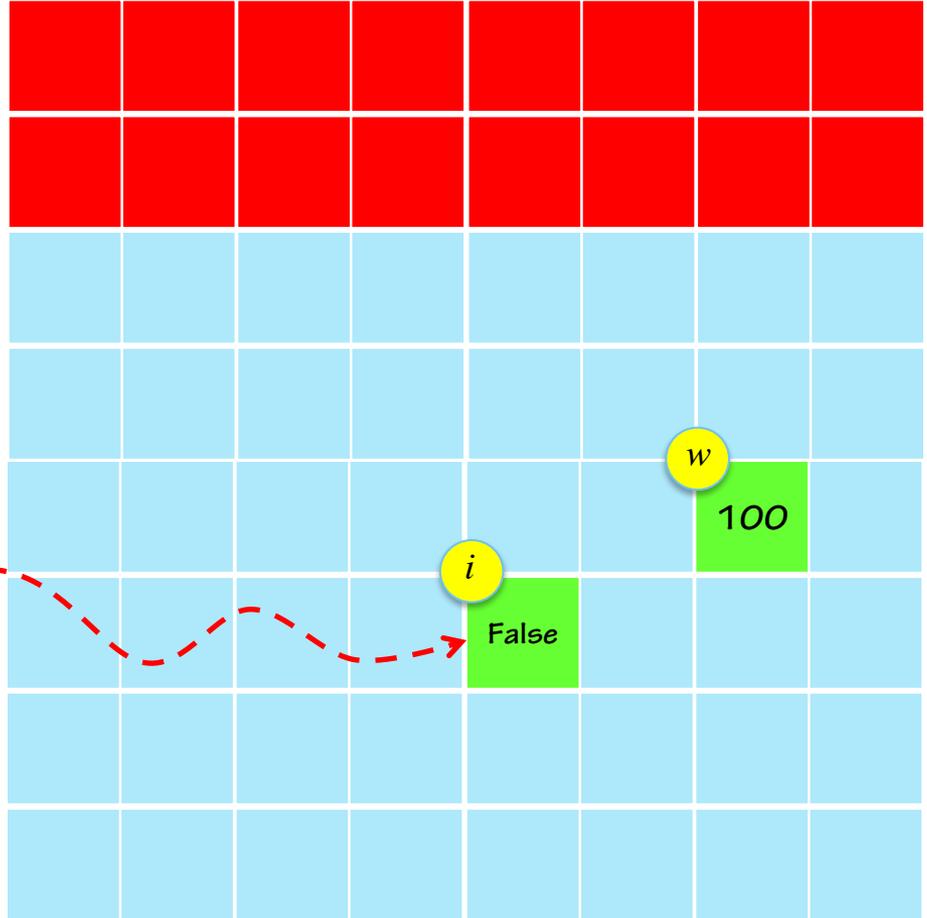
Condicionales

operadores relacionales

```
w = 100  
i = 99 == w
```

99 == 100

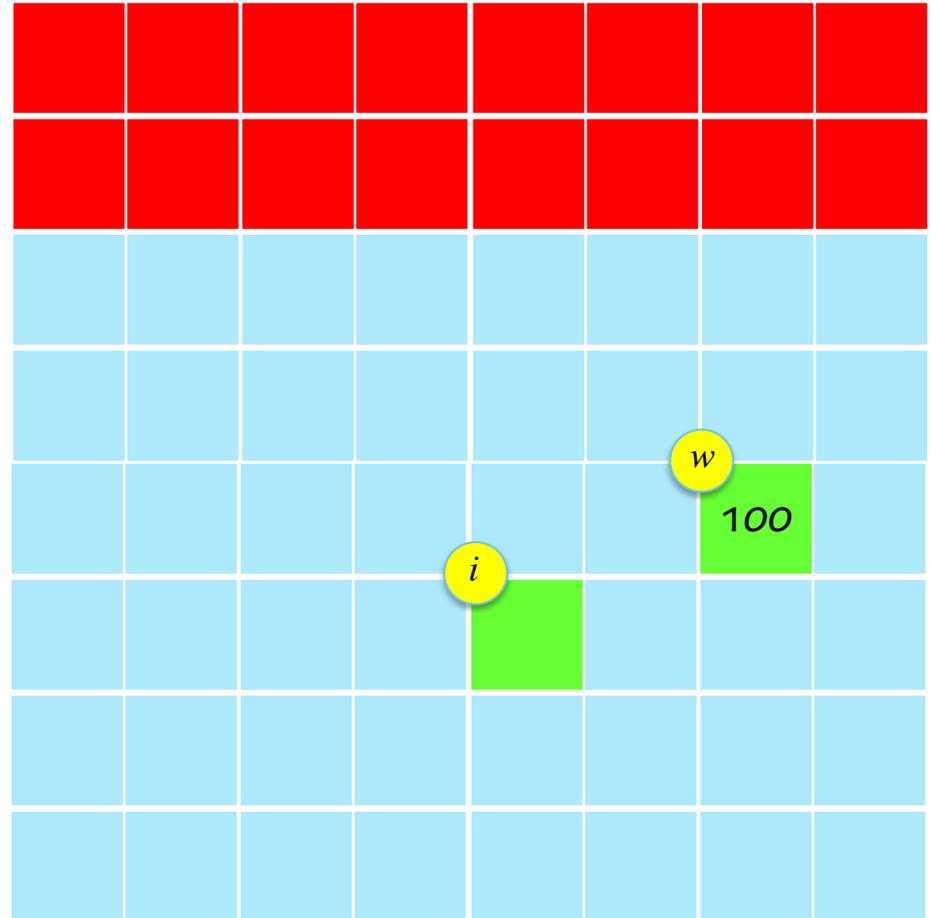
False



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.

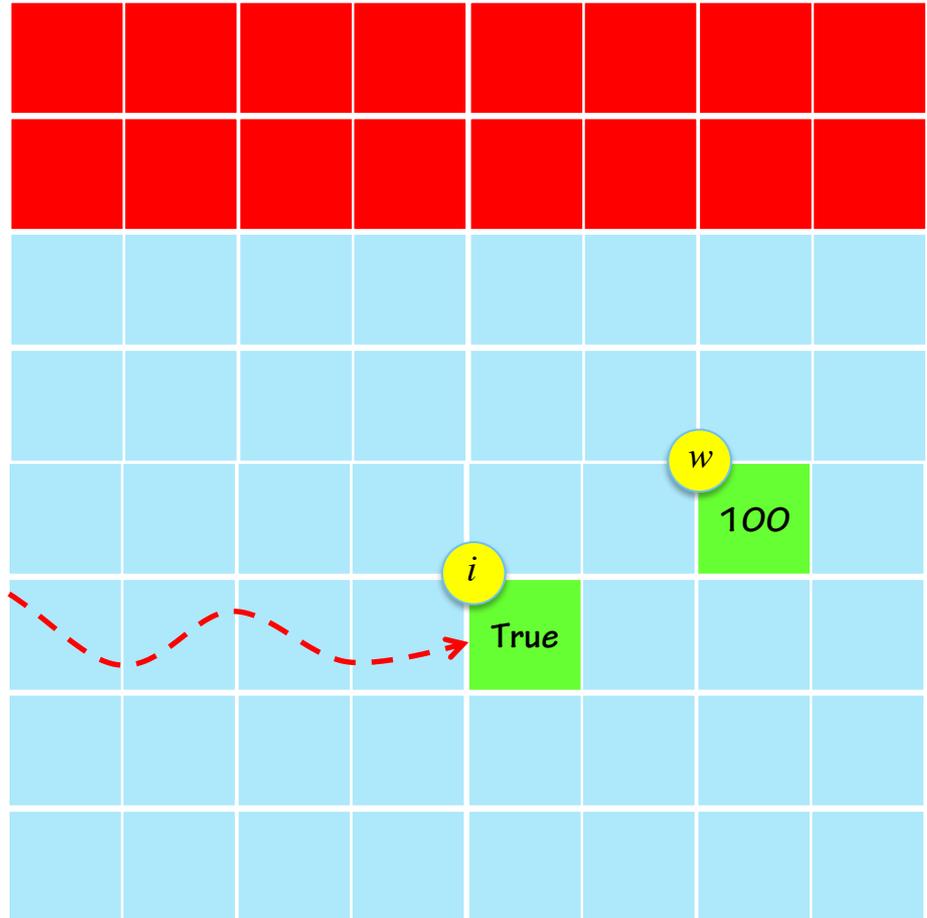
Condicionales

operadores relacionales

```
w = 100  
i = 100 >= w
```

```
100 >= 100
```

True



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				
1 ^{ro}	2 ^{do}	3 ^{ro}	4 ^{to}	
==	!=	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

1 ^{ro}	2 ^{do}	3 ^{ro}	4 ^{to}
==	!=	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		

precedencia

1 ^{ro}	2 ^{do}	3 ^{ro}	4 ^{to}
==	!=	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)	True and True	True
not(25>1) and (10>2)		
(-8<-3) or 100<99		
not(25>1) and (-8<-3) or 100<99		

precedencia

1 ^{ro}	2 ^{do}	3 ^{ro}	4 ^{to}
==	!=	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)	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		

precedencia

1 ^{ro}	2 ^{do}	3 ^{ro}	4 ^{to}
==	!=	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)	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		

precedencia

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

operando

operando

Tabla de verdad

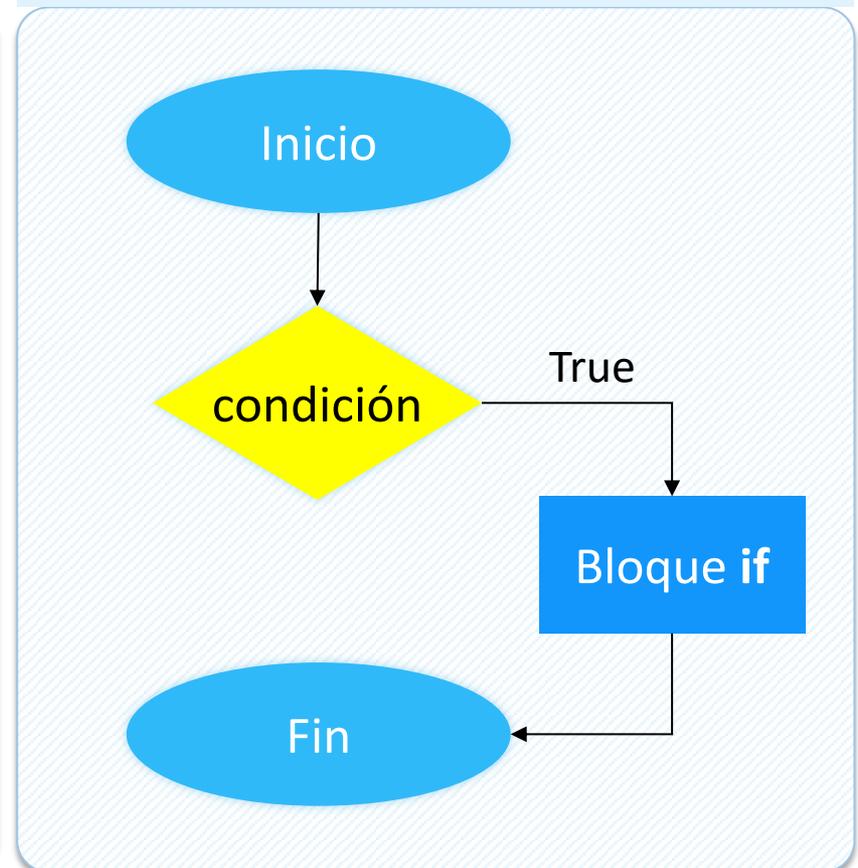
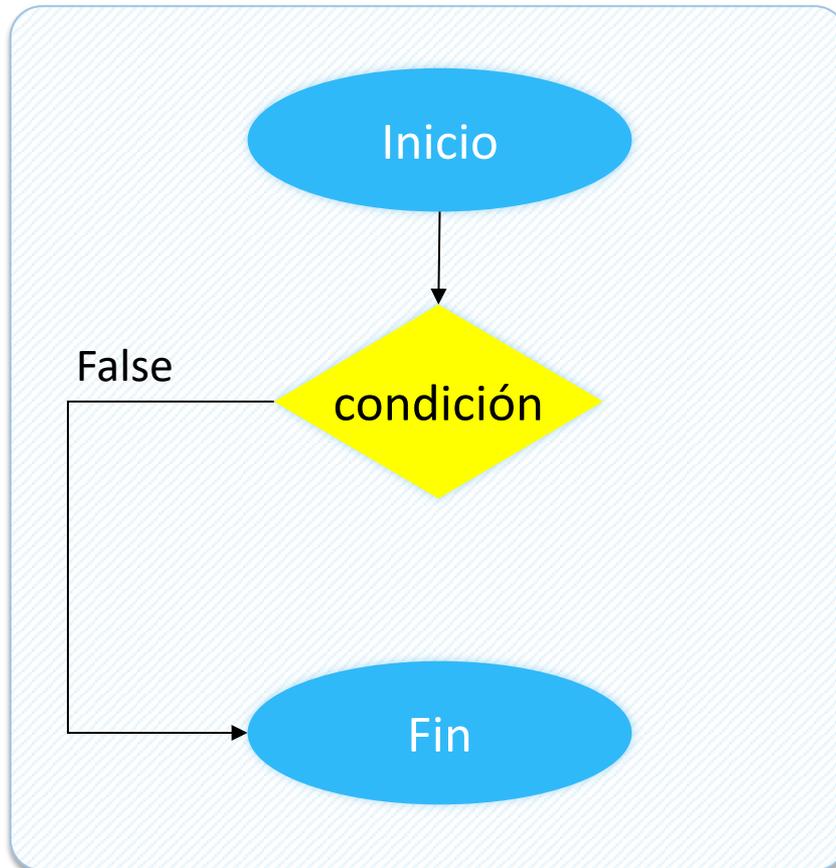
a	b	not(a)	a and b	a or b
True	True	False	True	True
True	False	False	False	True
False	True	True	False	True
False	False	True	False	False

Ejemplo

Expresión	Evaluación	Resultado
(25>1) and (10>2)	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

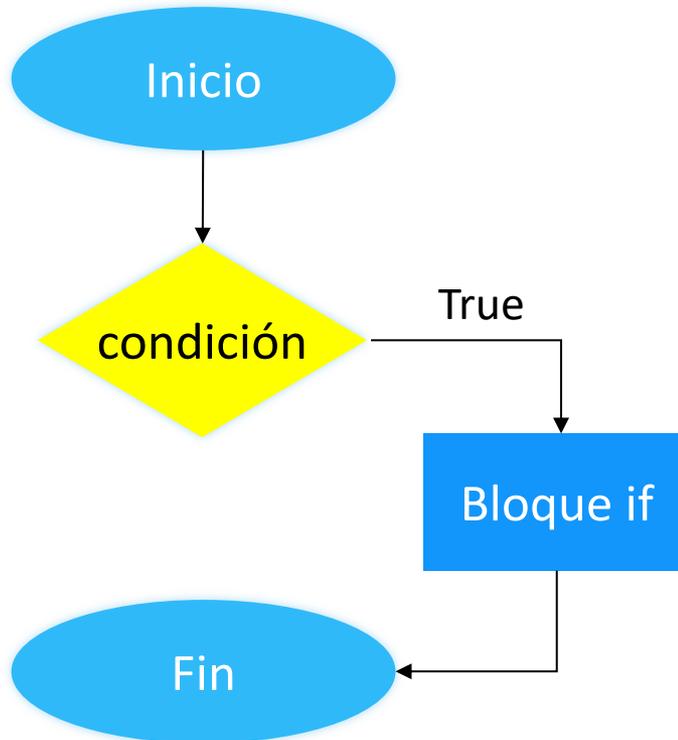
`if <condición lógica> :`

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:
```

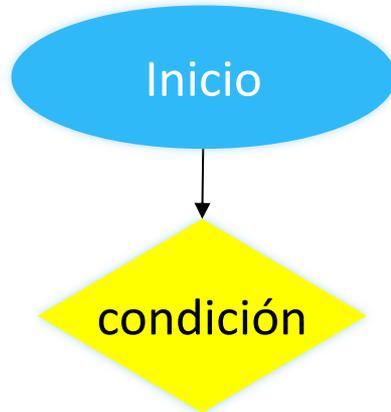
bloque IF

```
area = 9
```

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

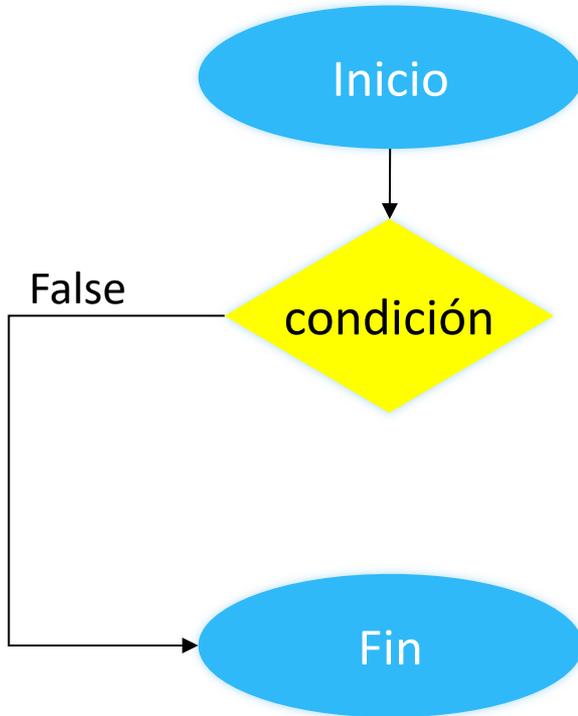


```
area = 10
w = 8

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

area = 9
```

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



```
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.



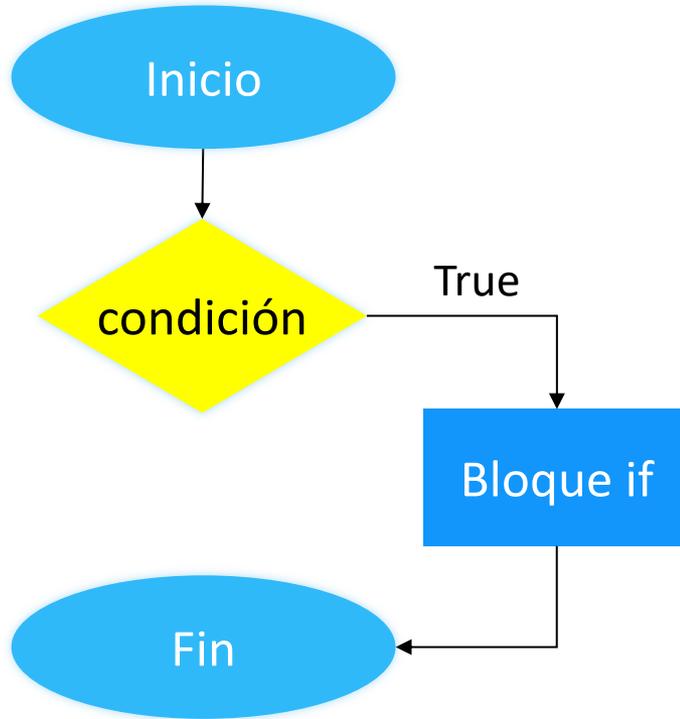
```
area = 10
w = 8

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?



```
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.

▼ if(condición):

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

The screenshot shows a Python REPL interface. On the left, a code editor displays the following code in `main.py`:

```
1 area = 10
2 w = 8
3
4 if ( area > 10 ):
5     print("El area es mayor a 10")
6     w = 11
7
8 area = 9
9 print(str(w))
```

On the right, the terminal output shows the Python version and GCC version, followed by a prompt. A blue callout box points to the terminal with the text: "¿Qué resultado se muestra por la terminal?"

Tabla de comparaciones lógicas

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

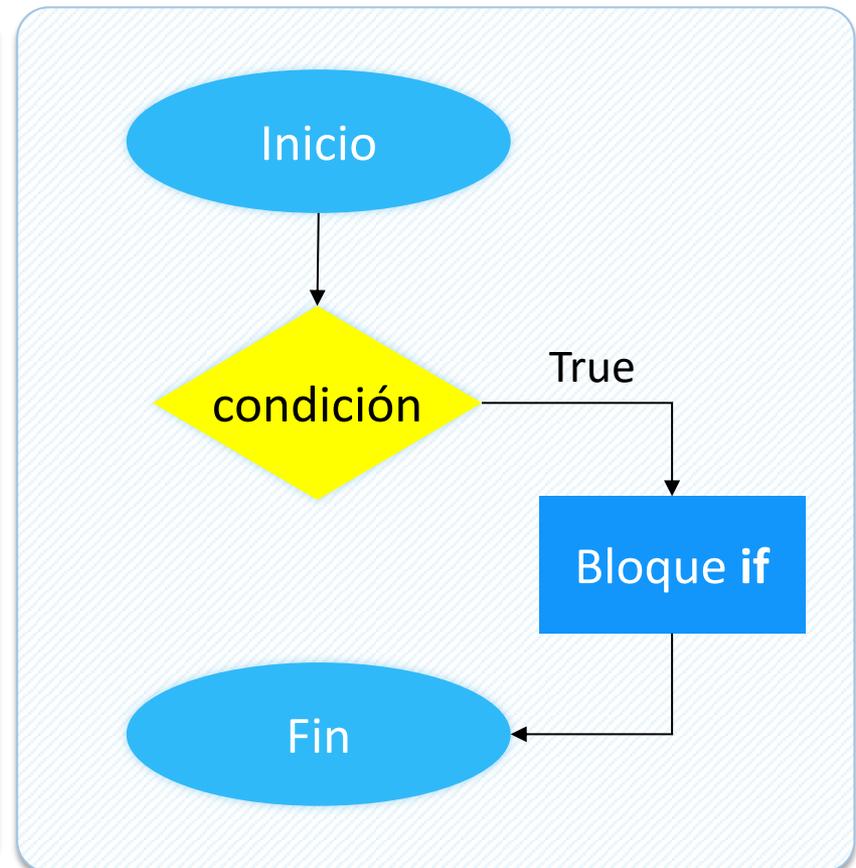
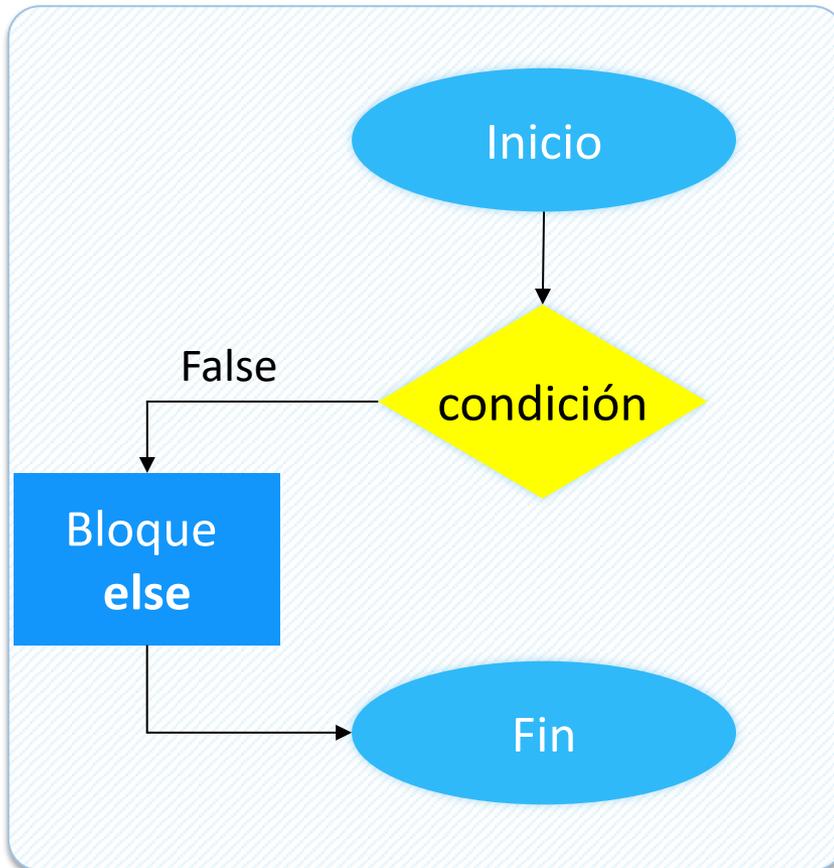
Resultados posibles

1	verdadero (TRUE)
0	falso (FALSE)

```
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**.

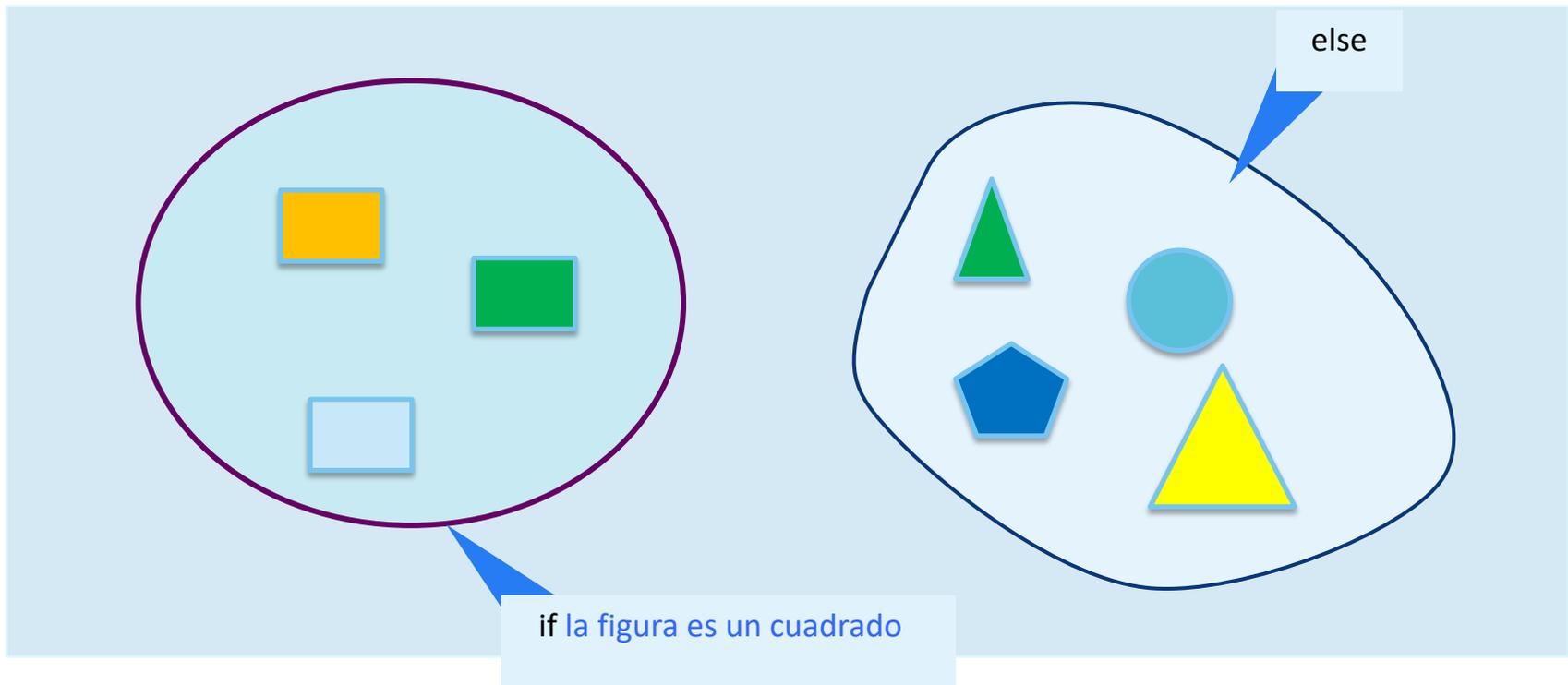


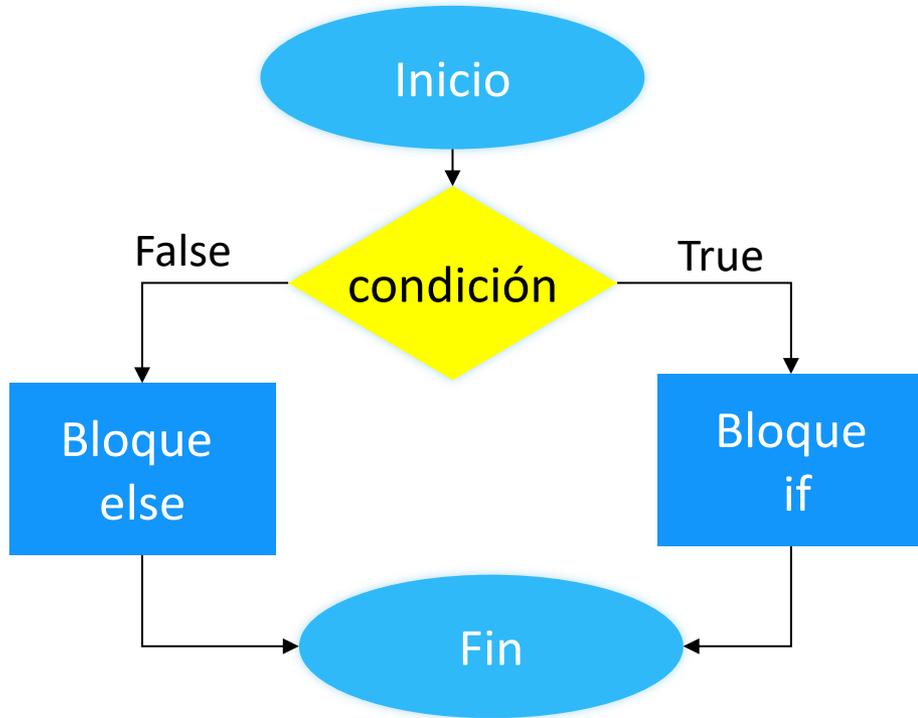
```
if <condición lógica> :
```

```
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





```
area = 10  
w = 8
```

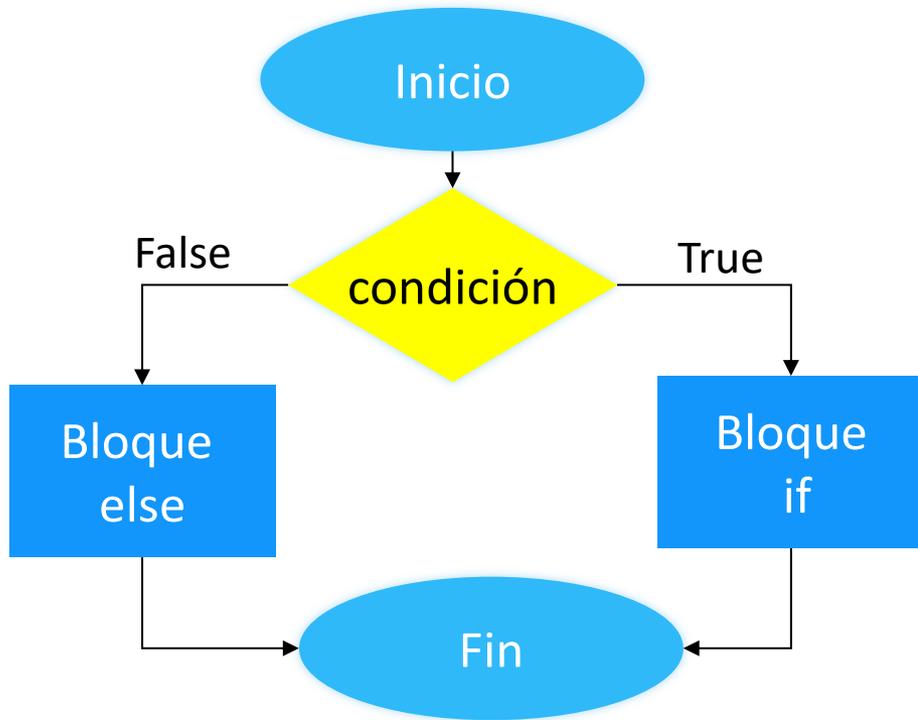
```
if area > 10:
```

bloque IF

```
else:
```

bloque ELSE

```
print(w)
```



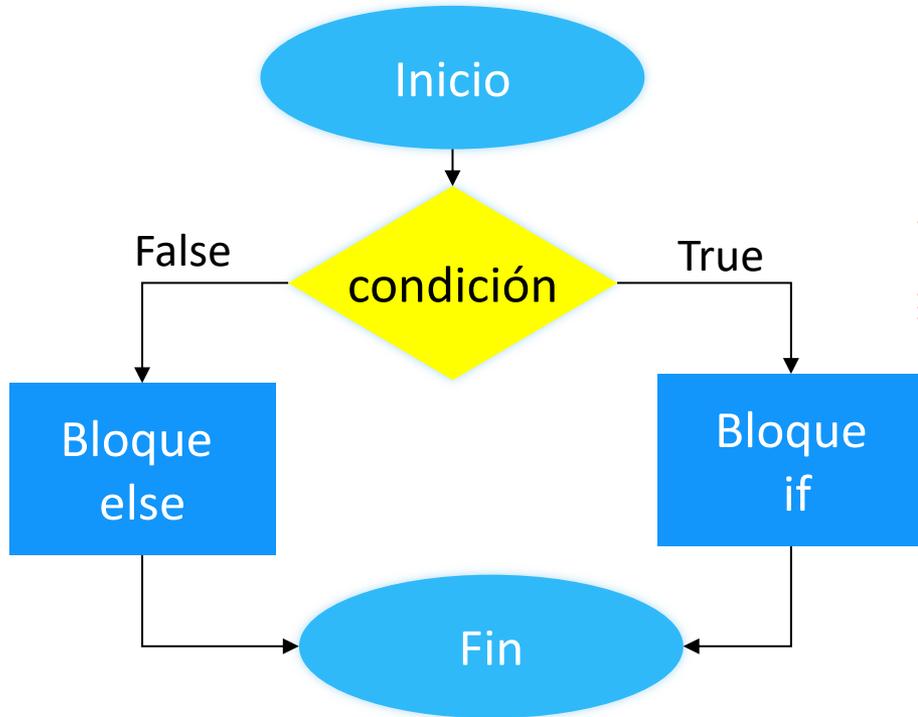
```
area = 10
w = 8

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?



```
area = 10
w = 8
if area > 10:
    print('El area es mayor a 10')
    w = 11
else:
    area = 9
    w = 7
print(w)
```

A red dashed arrow labeled 'al bloque else' points from the 'else:' line to the 'else:' block.

¿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

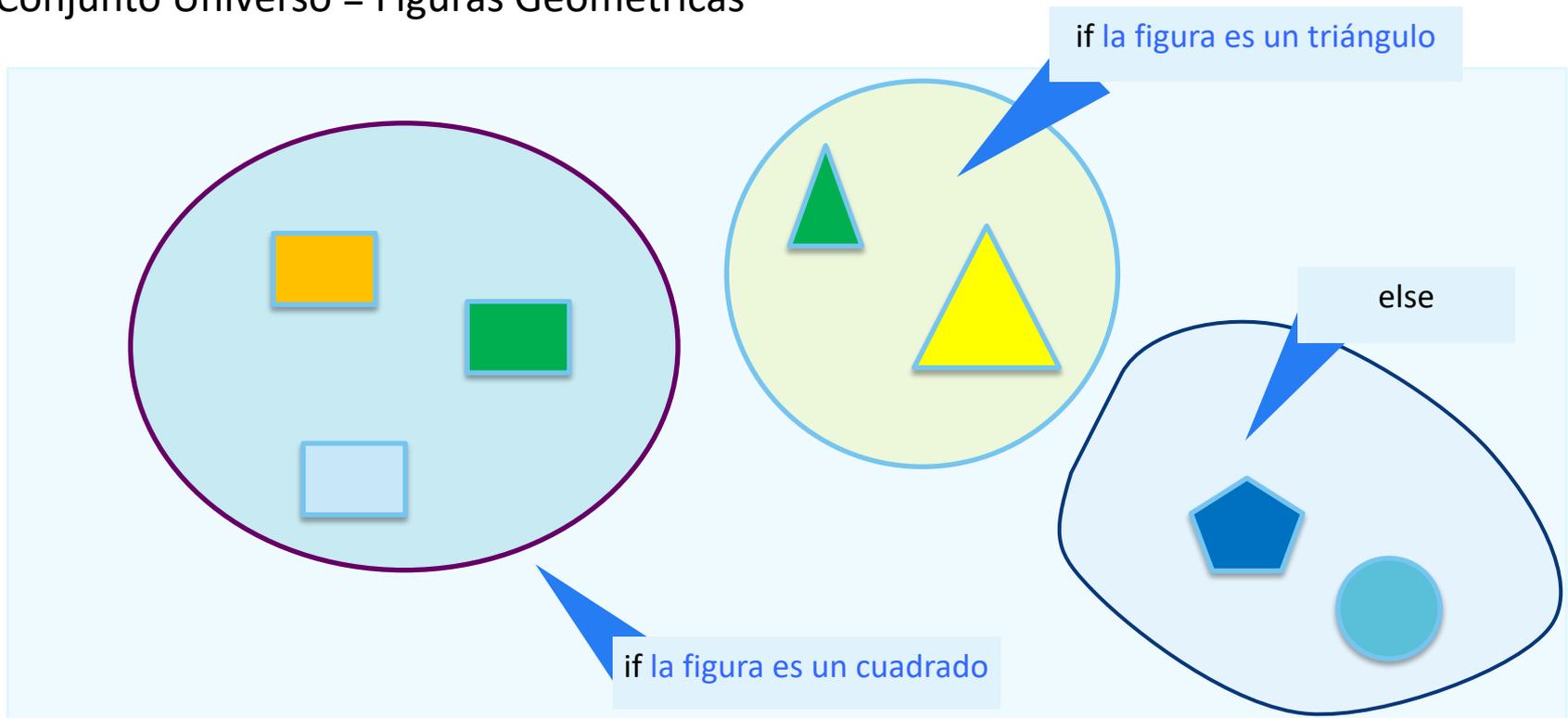


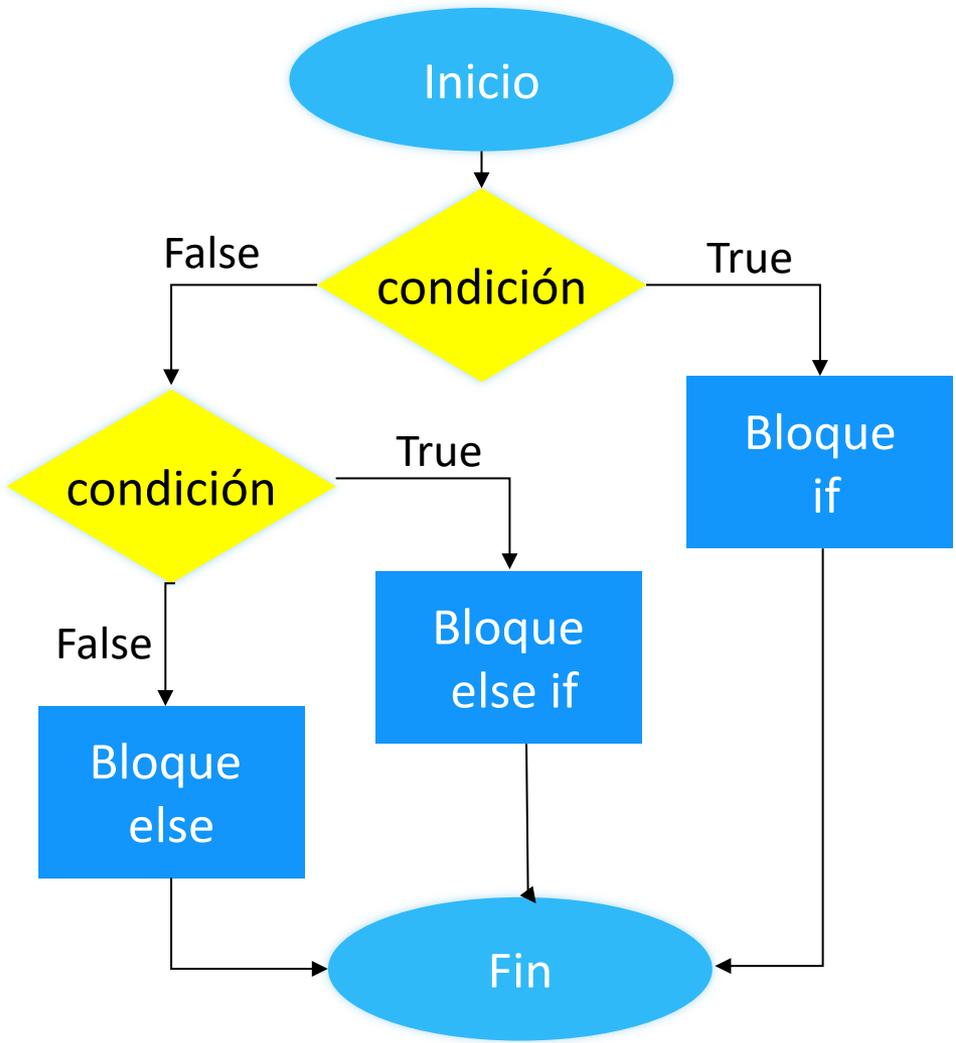
EMPUJE LA PUERTA
SI SE ABRE,
ESTA ABIERTO
CASO CONTRARIO, NO



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

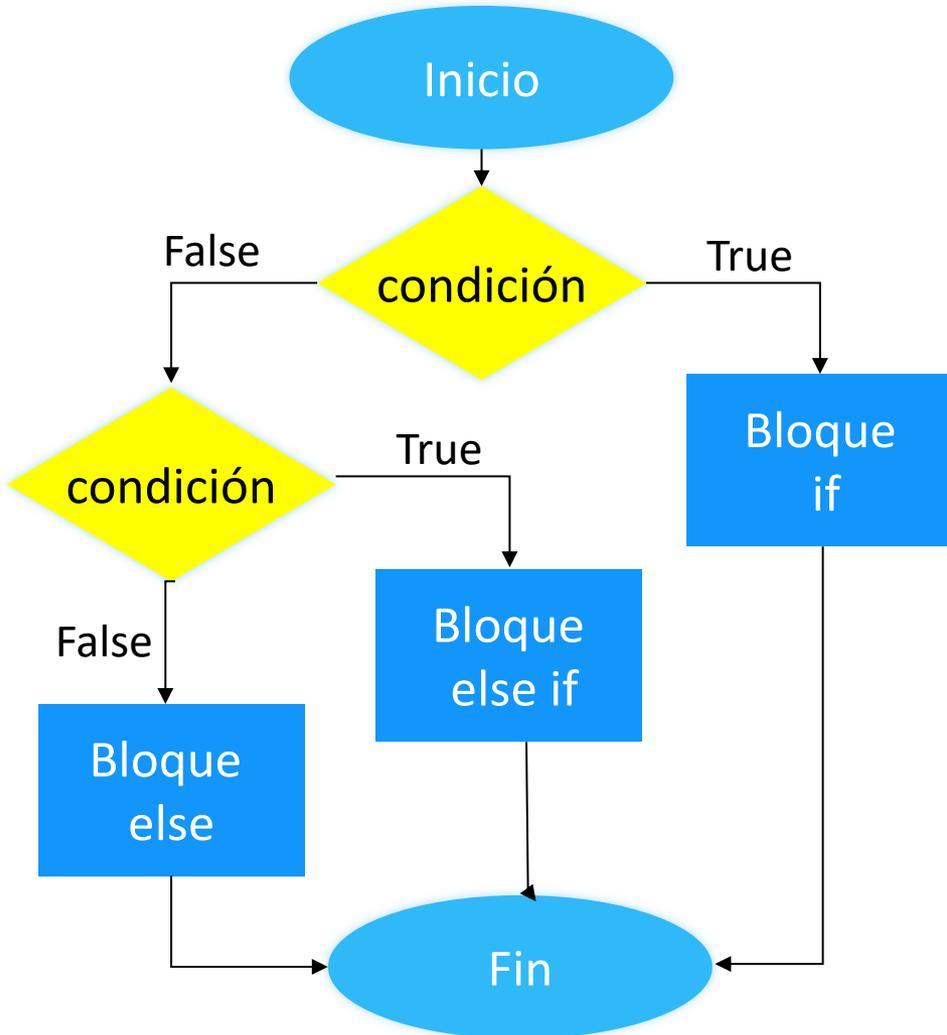




```
area = 5
w = 8

if area > 10:
    bloque if
else:
    if area > 5:
        bloque elif
    else:
        bloque else

print(w)
```



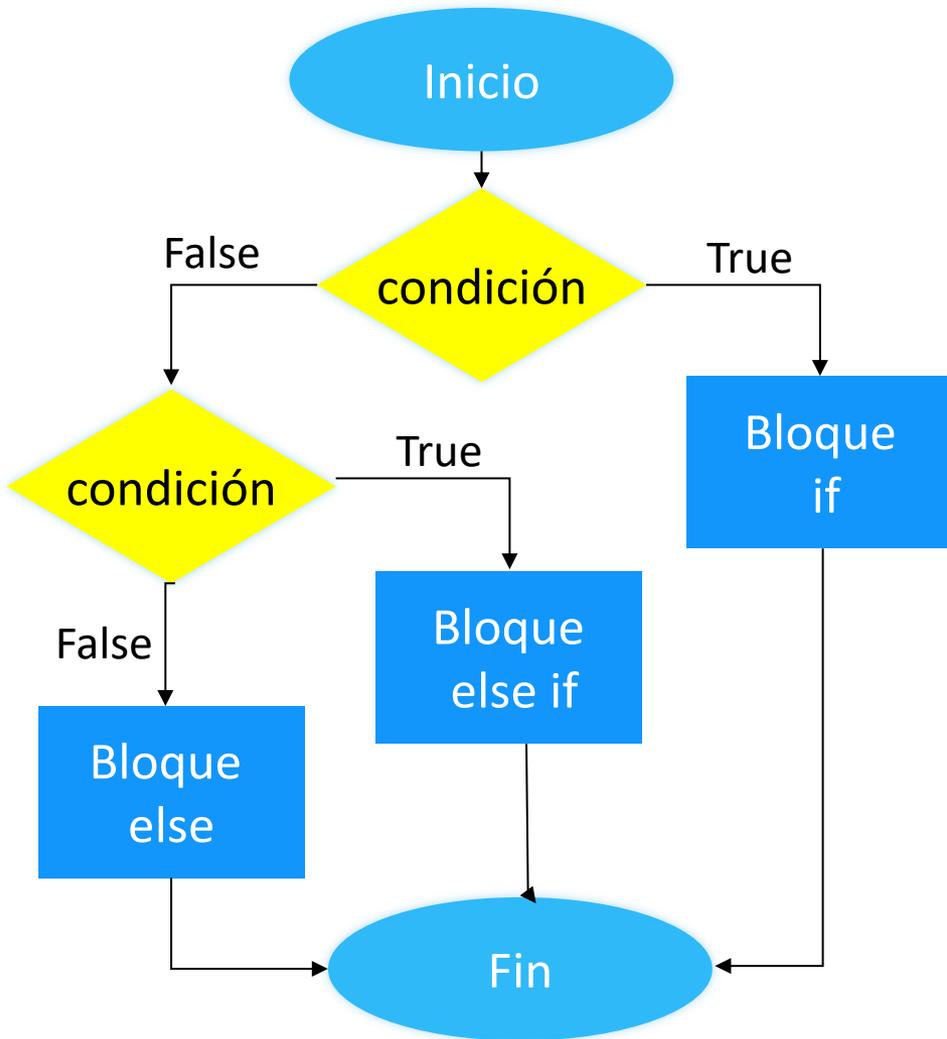
```
area = 5
w = 8

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?



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

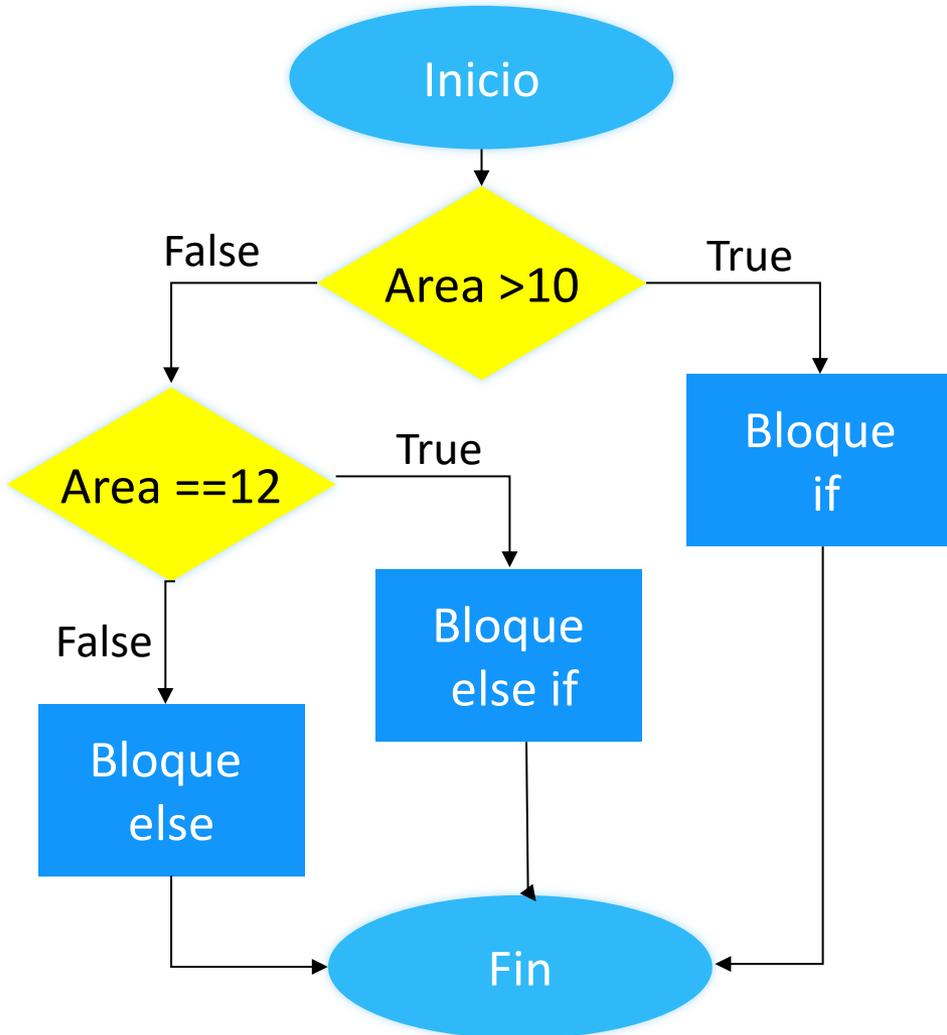
```
area = 5
w = 8

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)
```



```
area = 5
w = 8

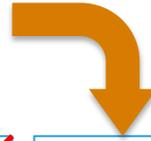
if area > 10:
    print('El área es mayor a 10')
    w = 11

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)



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?*



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")
```



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?



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')
```



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’



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

