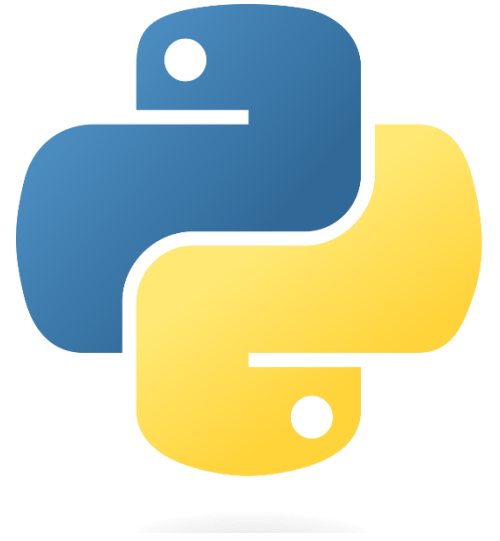


script.py

```
1 facebook = "Facebook's rating is"  
2 fb_rating = 3.5  
3  
4 fb_rating_str = str(3.5)  
5 fb = facebook + ' ' + fb_rating_str  
6  
7 print(fb)  
8
```



Python

Programación

Introducción

- Creado en 1991 por Guido van Rossum



Conceptos

- Lenguaje de programación de alto nivel
- Sintaxis intuitiva
- Lenguaje interpretado
 - NO tiene fase de compilación intermedia como C o Java
 - Escritura dinámica
 - No necesitas indicar el tipo de dato al declarar una variable
 - La variable NO está vinculada a un tipo de dato concreto

Paradigmas de programación

- Python admite varios paradigmas de programación
 - Programación orientada a objetos
 - Programación procedimental
 - Programación funcional

Tradicional

```
bool estaCosme = false;
foreach (var alumno in alumnos)
{
    if (alumno.Name == "Cosme")
    {
        estaCosme = true;
        break;
    }
}
```

Funcional

```
alumnos.Any(
    a => a.Name == "Cosme");
```

Paradigmas de programación



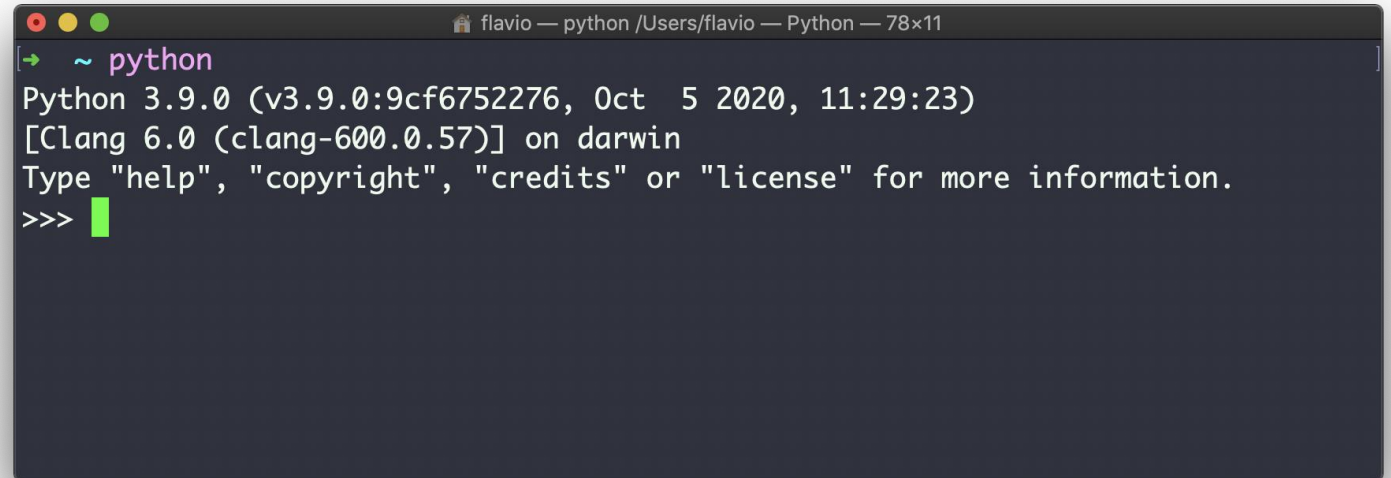
Comenzamos

- Instalar paquete oficial de Python.org
 - Downloads

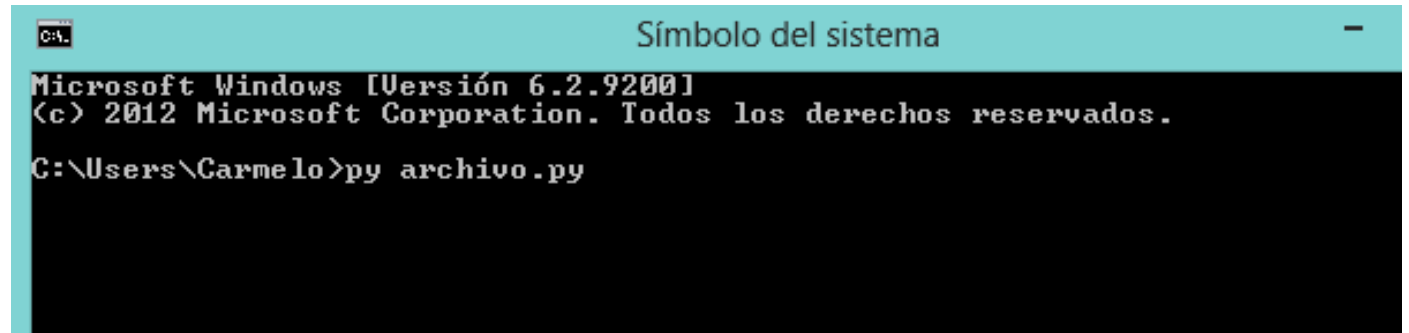


Ejecutar Python

- Desde el intérprete
- Desde archivo
 - `py archivo.py`



```
flavio — python /Users/flavio — Python — 78x11
→ ~ python
Python 3.9.0 (v3.9.0:9cf6752276, Oct 5 2020, 11:29:23)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

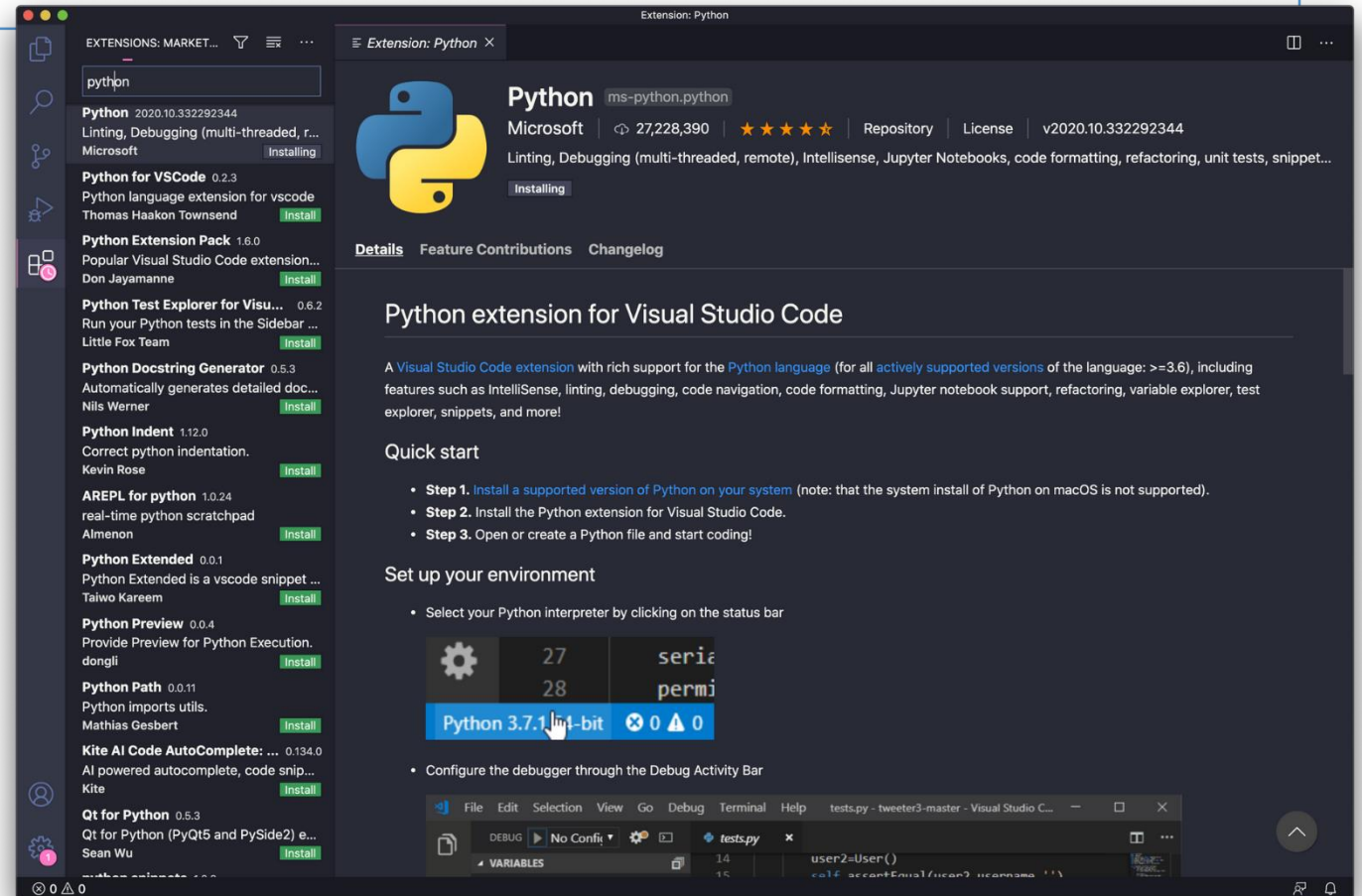


```

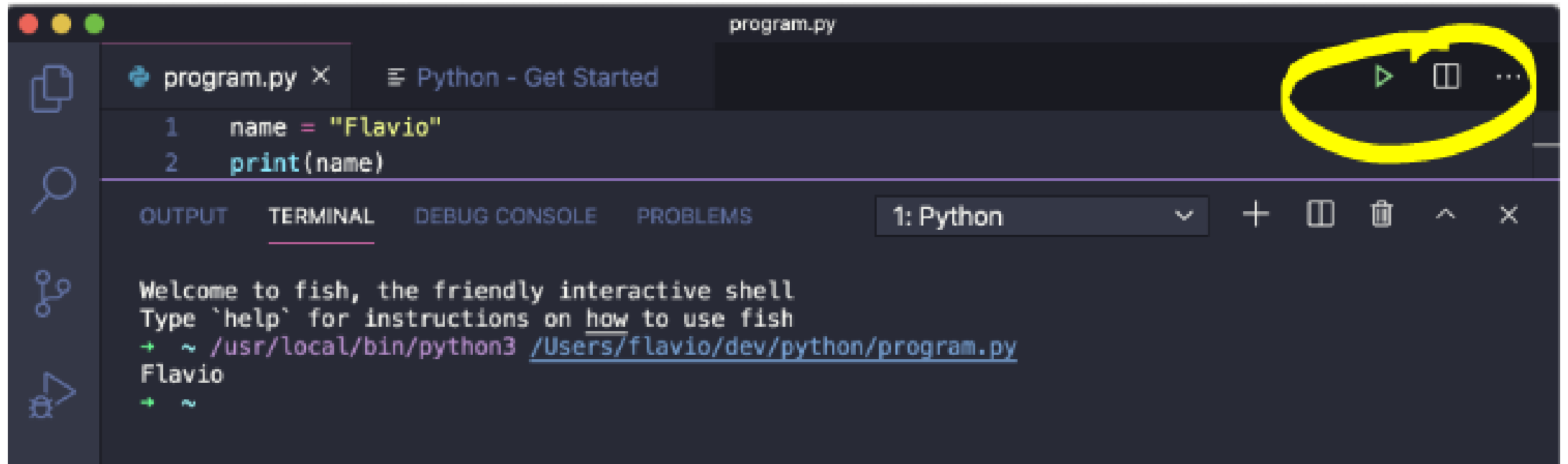
Símbolo del sistema
Microsoft Windows [Versión 6.2.9200]
(c) 2012 Microsoft Corporation. Todos los derechos reservados.
C:\Users\Carmelo>py archivo.py
```

Visual Studio Code

- Editor avanzado
- Instalar la extensión Python



Ejecutar Python en VSCode



The image shows a screenshot of the Visual Studio Code (VS Code) interface. The editor window displays a Python file named `program.py` with the following code:

```
1 name = "Flavio"
2 print(name)
```

The interface includes a sidebar on the left with icons for Explorer, Search, Source Control, and Run and Debug. The top bar shows the file name `program.py` and the Python version `Python - Get Started`. The bottom panel is divided into sections: `OUTPUT`, `TERMINAL` (selected), `DEBUG CONSOLE`, and `PROBLEMS`. The `TERMINAL` section shows the output of the program, which is `Flavio`. A yellow circle highlights the Run button (a green play icon) in the top right corner of the editor window.

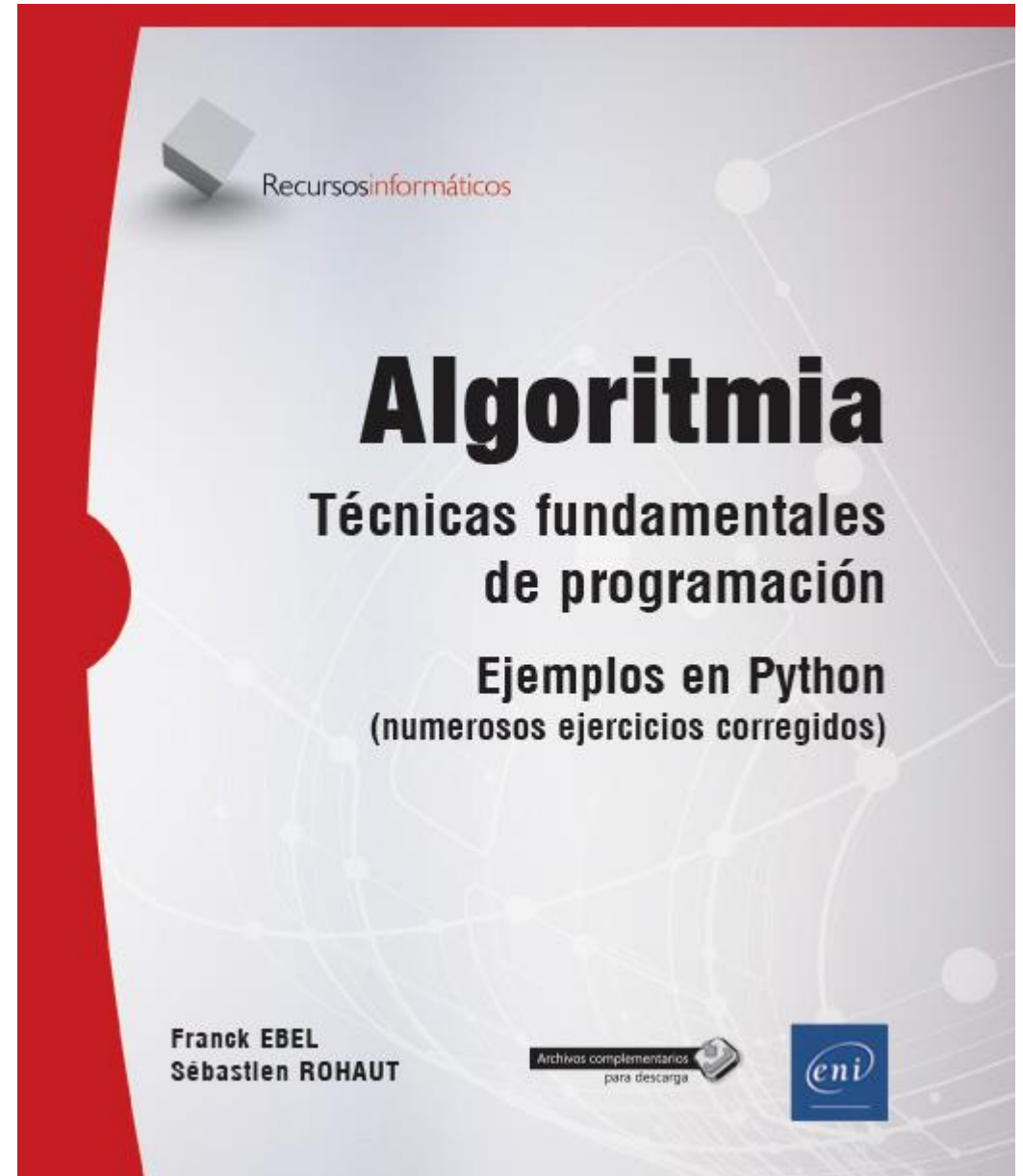
```
1 name = "Flavio"
2 print(name)
```

OUTPUT TERMINAL DEBUG CONSOLE PROBLEMS 1: Python

Welcome to fish, the friendly interactive shell
Type 'help' for instructions on how to use fish
+ ~ /usr/local/bin/python3 /Users/flavio/dev/python/program.py
Flavio
+ ~

Python 2 vs Python 3

- Python 3 aparece en 2008
- Python 2 con soporte hasta 2020



Variables

- Usamos el operador de asignación
- Declaración y asignación de variables
 - nombre="juan"
 - importe = 10
- Nombres válidos de variables
 - NOMBRE
 - _nombre
 - mi_nombre
 - Nombre123
- Nombres no válidos de variables
 - 12nombre
 - nombre!
 - Nombre%

The screenshot shows a Python IDE with a code editor and a console. The code editor contains the following lines:

```
1 # Declare a variable and initialize it
2 f = 0
3 print(f)
4 # re-declaring the variable works
5 f = 'guru99'
6 print(f)
```

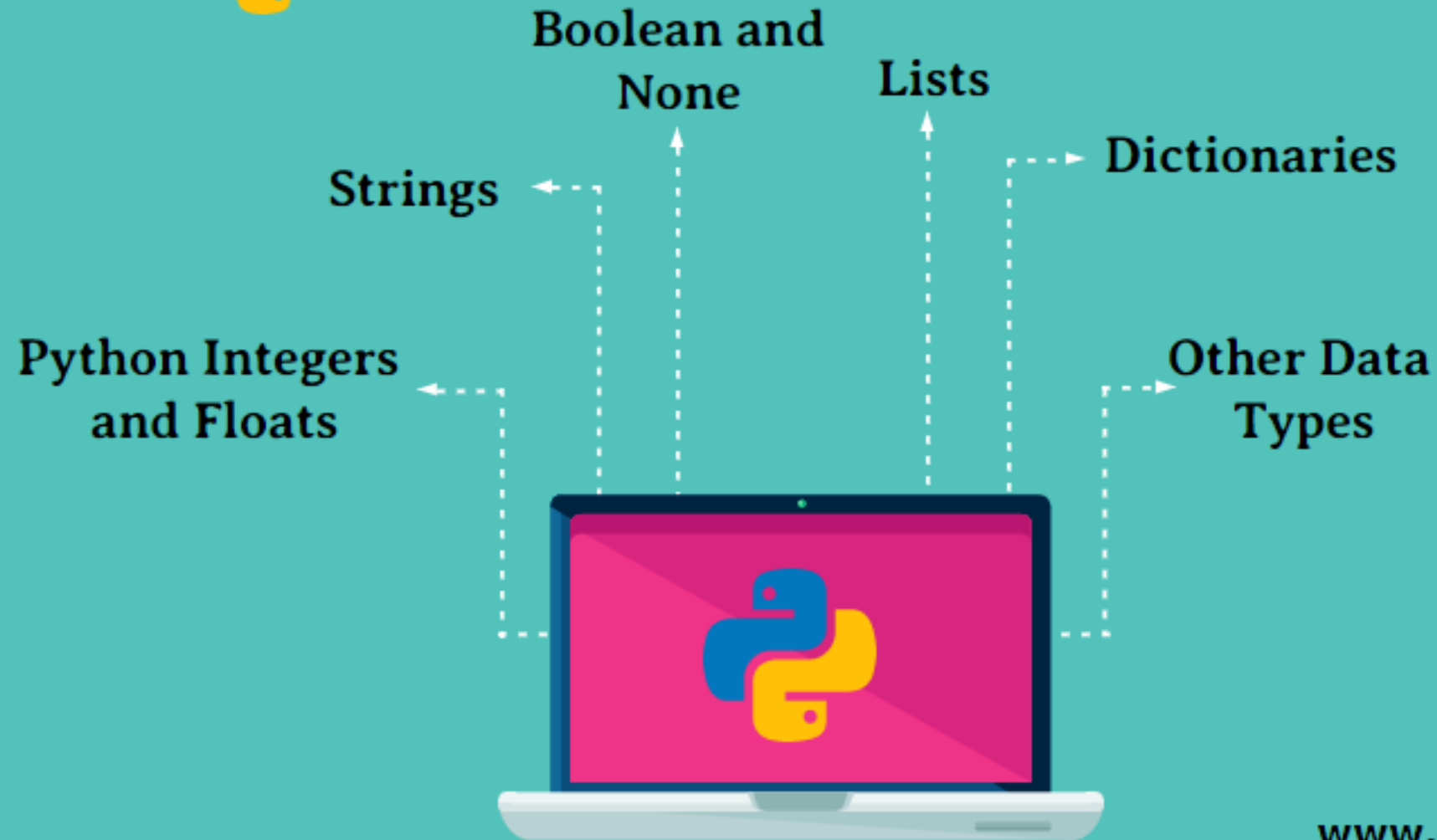
Lines 2 and 5 are highlighted with red boxes. A callout bubble points to line 5 with the text: "you can re-declare the variables, even-after if it is declared once. it works fine". The console output shows the results of running the code:

```
0
guru99
```

The output "0" and "guru99" are highlighted with red boxes.



Python Variable Types



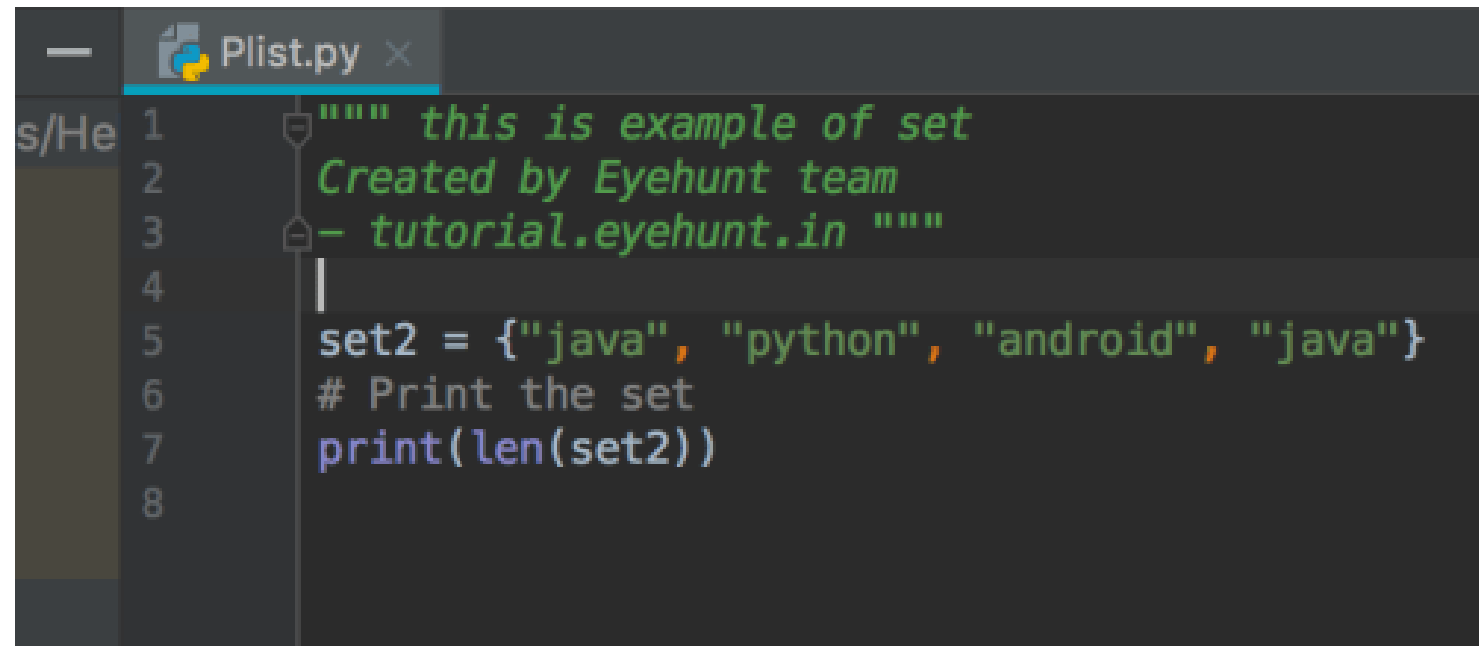
Expresiones vs Declaraciones

- Una expresión devuelve un valor
 - 5+6
- Una declaración es una operación sobre un valor
 - ciudad="Madrid"
 - print(ciudad)

	expression	value	type
literal	500	500	integer
	3.14	3.14	float
+	200 + 300	500	integer
	10.0 + 5.0	15.0	float

Comentarios

- En Python el comentario se indica con #
- La línea marcada por # se ignora
 - ciudad='Sevilla' #esto es un comentario



```
s/He 1 """ this is example of set
      2 Created by Eyehunt team
      3 - tutorial.eyehunt.in """
      4 |
      5 set2 = {"java", "python", "android", "java"}
      6 # Print the set
      7 print(len(set2))
      8
```

Indentado

- Python es un lenguaje indentado.
- Implica que el sangrado de código es importante para su intérprete
- El error por código mal indentado suele ser `IndentationError`

```
def foo():  
    print("Hi")  
    if True:  
        print("true")  
    else:  
        print("false")  
  
print("Done")
```

1st level indentation
foo() method statements

2nd level indentation
if and else block code

Code without indentation
Belongs to the source file

Tipos de datos

- Tipos de datos integrados
- Al declarar la variable y asignarle un valor, la variable toma el tipo de dato del valor
 - ciudad='Sevilla'
 - La variable ciudad representa un tipo de dato String
- Función type permite verificar el tipo de dato de una variable
 - `type(ciudad)==str #true`

Números

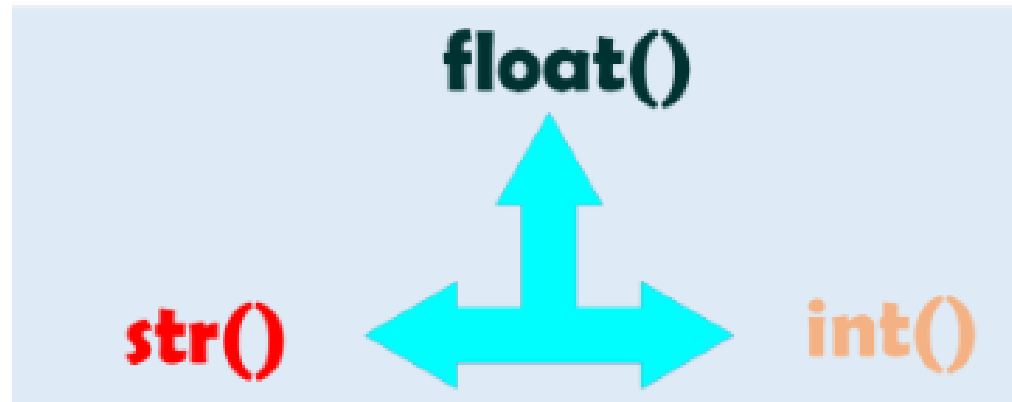
- Clase int para números enteros
- Clase float para números decimales

```
edad = 1  
type(edad) == int #True
```

```
fraccion = 0.1  
type(fraccion) == float #True
```

Casting

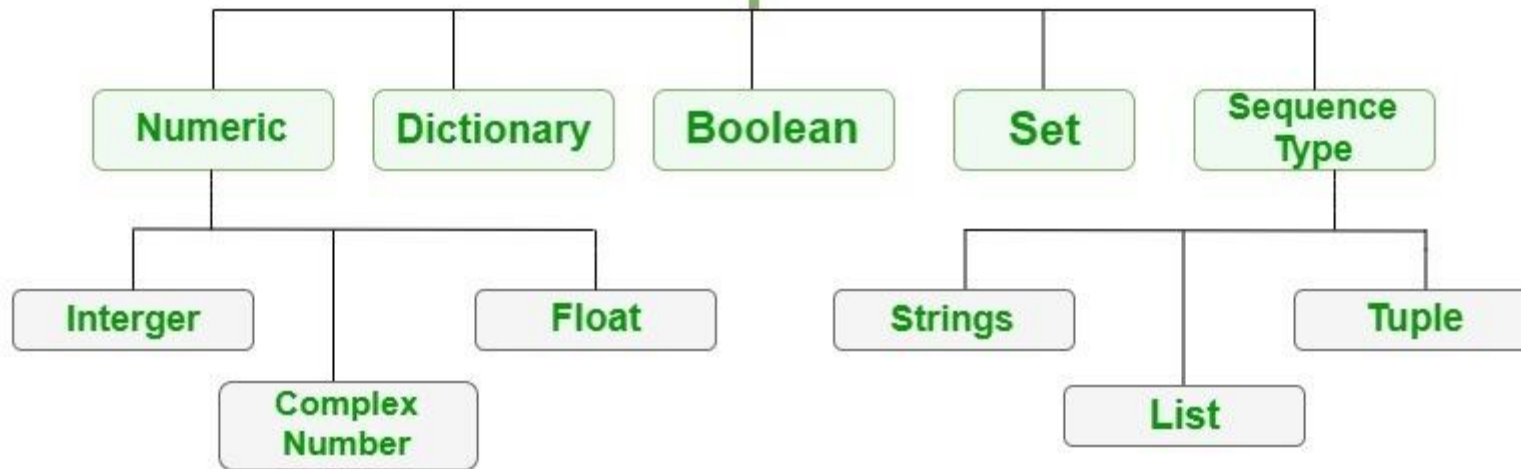
- Python intentará detectar el tipo de dato correcto
- De cualquier manera, es posible parsear, castear el tipo de dato
 - `unidades='10'`
 - `total=5*int(unidades)` #función int convierte a entero



Otros tipos de datos

- complex para números complejos
- bool para booleanos(verdadero/falso)
- list para listas
- tuple para tuplas
- range para rangos
- dict para diccionarios
- set para conjuntos(sets)

Python - Data Types



Operadores en Python

- Operador de asignación
- Operadores aritméticos
- Operadores de comparación
- Operadores lógicos
- Operadores bit a bit

OPERATORS IN PYTHON

Assingment Operator

`=, +=, -=, /=, *=`

Arithmetic Operator

`+, -, /, *, %, **`

Bitwise Operator

`&, |, ^, ~, <<, >>`

Logical Operator

- Logical AND
- Logical OR
- Logical NOT

Relational Operator

`>, >=, !=, <>, <, <=, ==`

Identity Operator

`&, |, ^, ~, <<, >>`

Operadores de asignación

- Se utiliza para asignar un valor a una variable
 - También para asignar una variable a otra

```
edad = 8  
otraVariable = edad
```

Operadores aritméticos

- Combinación de operadores aritméticos y de asignación

```
edad = 8  
edad += 1  
#edad es ahora 9
```

```
1 + 1 #2  
2 - 1 #1  
2 * 2 #4  
4 / 2 #2  
4 % 3 #1  
4 ** 2 #16  
4 // 2 #2
```


Operadores de comparación

- ==
- !=
- >
- <
- >=
- <=

```
a = 1
b = 2

a == b #False
a != b #True
a > b #False
a <= b #True
```

Operadores booleanos

- Not
- And
- Or

```
condicion1 = True  
condicion2 = False
```

```
not condicion1 #False  
condicion1 and condicion2 #False  
condicion1 or condicion2 #True
```

Operadores bit a bit

- Utilizados para trabajar con bits o números binarios
 - Se utilizan en raras ocasiones
- & realiza el binario AND
- | realiza el binario OR
- ^ realiza una operación binaria XOR
- ~ realiza una operación binaria NOT
- << operación shift left
- >> operación shift right

Operador ternario

- Permite definir una condicional abreviada

```
def es_adulto(edad):  
    if edad > 18:  
        return True  
    else:  
        return False
```

```
def es_adulto(edad):  
    return True if edad > 18 else False
```