

Programando en Python I

Enero 2023



VICEPRESIDENCIA
PRIMERA DEL GOBIERNO
MINISTERIO
DE ASUNTOS ECONÓMICOS
Y TRANSFORMACIÓN DIGITAL

SECRETARÍA DE ESTADO
DE DIGITALIZACIÓN
E INTELIGENCIA ARTIFICIAL

red.es

Centro de
Referencia Nacional
en Comercio Electrónico
y Marketing
CRN
Digital



UNIÓN EUROPEA

 Barrabés

 The Valley

"El FSE invierte en tu futuro"

Fondo Social Europeo

1. Introducción
2. Elementos
3. Programas y lenguajes de programación
4. Algoritmos
5. Operadores aritméticos
6. Errores de tecleo y excepciones
7. Tipos de datos
8. Variables y asignaciones
9. Datos de cadena de texto
10. Funciones predefinidas
11. Módulos e importación de funciones y variables
12. Métodos
13. De interés
14. Estructuras de control



Introducción



Introducción

¿Por qué Python?

- Lenguaje de muy alto nivel: expresar algoritmos de forma casi directa.
- Librerías adaptadas al *Machine Learning*.
- Otros lenguajes: C, C++, Java,...

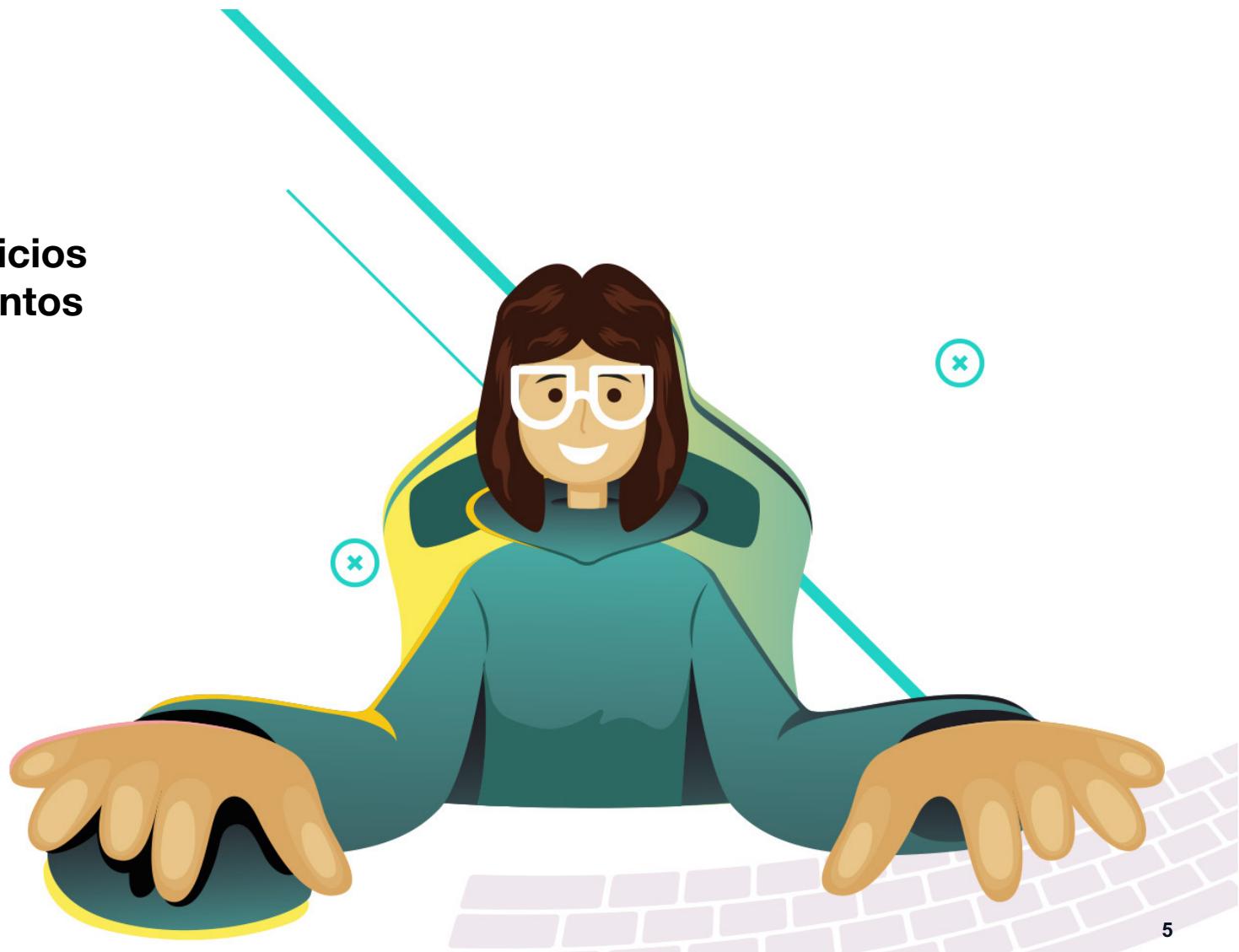


Aprender a programar

**Es imposible hacerlo
limitándose a leer un texto**

**Se necesitan muchos ejercicios
para asentar los conocimientos
y estrategias**

Mucho trabajo y práctica



Objetivo

Aprender a programar

Diseñar algoritmos y expresarlos como programas escritos en un lenguaje de programación para poder ejecutarlos en un computador.



Elementos



Elementos

Computador

<<Máquina electrónica, analógica o digital, dotada de una **memoria** de gran capacidad y de métodos de tratamiento de la información, capaz de resolver problemas matemáticos y lógicos mediante la utilización automática de programas informáticos>>.

Algoritmo

<<Conjunto ordenado de **operaciones sistemáticas** que permite hacer un cálculo y **hallar la solución** de un tipo de problemas>>.

Elementos

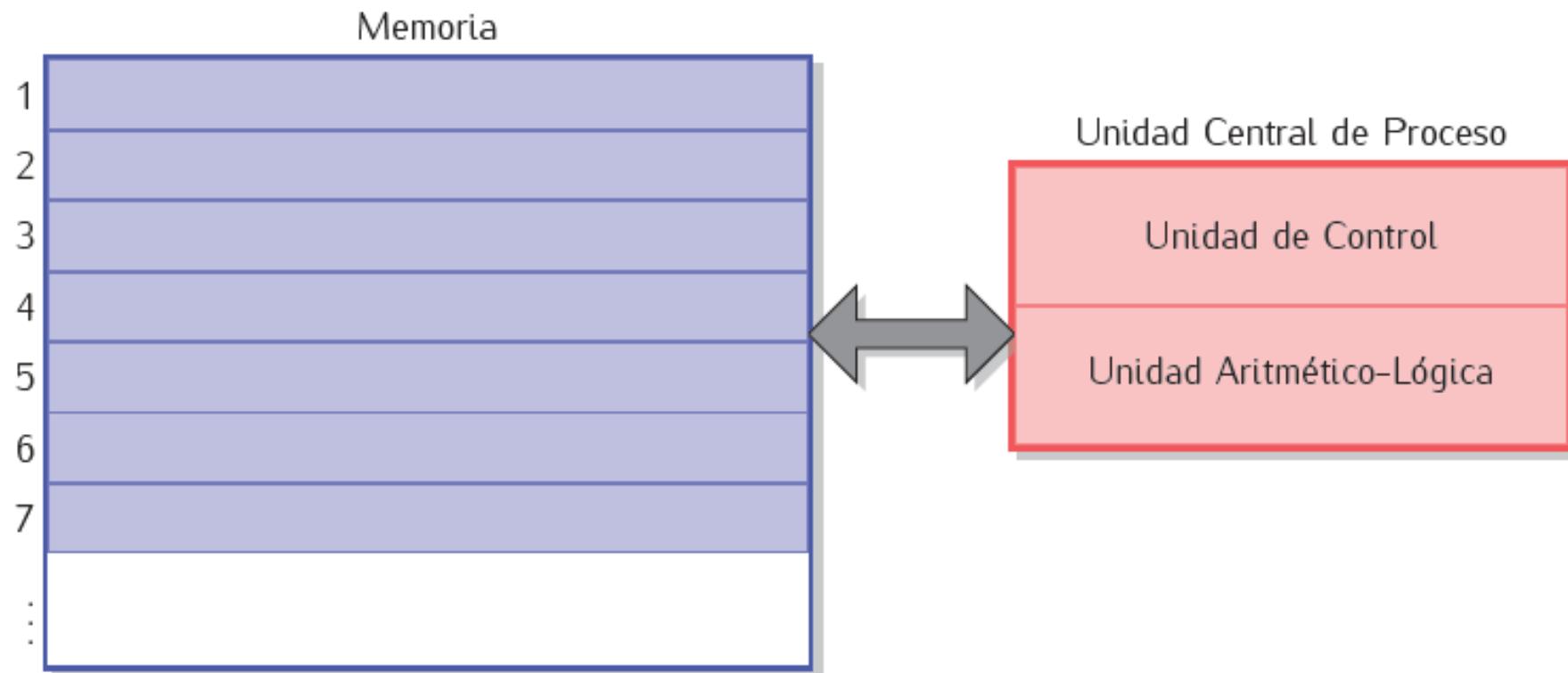
Memoria

Almacén de información (valores numéricos, textos, imágenes,...)

- **Unidad Aritmético-Logica (UAL):** Dispositivo encargado de efectuar operaciones matemáticas y lógicas.
- **Unidad de control:** Dispositivo que se encarga de transportar la información de la memoria a la UAL, controlar la UAL y depositar los resultados en la memoria.
- **Unidad central de proceso (CPU):** Unidad de control + UAL.

Elementos

Memoria



Elementos

Codificación de números

Cada posición de memoria permite almacenar una secuencia de unos y ceros de tamaño fijo.

- **Dispositivos binarios**
- Secuencias de N bits permiten representar 2^N números diferentes.
- 8 bits = 256 números

| Decimal | Binario |
|---------|---------|
| 0 | 0 |
| 1 | 1 |
| 2 | 10 |
| 3 | 11 |
| 4 | 100 |
| 5 | 101 |
| 6 | 110 |
| 7 | 111 |
| 8 | 1000 |
| 9 | 1001 |
| 10 | 1010 |

Elementos

Codificación de texto

ASCII: American Standard Code for Information Interchange

La tabla ASCII al completo

Estos son los valores (en base 10) de cada símbolo en la tabla ASCII:

| | | | | | | | | | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| 0 | nul | 1 | soh | 2 | stx | 3 | etx | 4 | eot | 5 | enq | 6 | ack | 7 | bel |
| 8 | bs | 9 | ht | 10 | nl | 11 | vt | 12 | np | 13 | cr | 14 | so | 15 | si |
| 16 | dle | 17 | dc1 | 18 | dc2 | 19 | dc3 | 20 | dc4 | 21 | nak | 22 | syn | 23 | etb |
| 24 | can | 25 | em | 26 | sub | 27 | esc | 28 | fs | 29 | gs | 30 | rs | 31 | us |
| 32 | sp | 33 | ! | 34 | " | 35 | # | 36 | \$ | 37 | % | 38 | & | 39 | , |
| 40 | (| 41 |) | 42 | * | 43 | + | 44 | , | 45 | - | 46 | . | 47 | / |
| 48 | 0 | 49 | 1 | 50 | 2 | 51 | 3 | 52 | 4 | 53 | 5 | 54 | 6 | 55 | 7 |
| 56 | 8 | 57 | 9 | 58 | : | 59 | ; | 60 | < | 61 | = | 62 | > | 63 | ? |
| 64 | @ | 65 | A | 66 | B | 67 | C | 68 | D | 69 | E | 70 | F | 71 | G |
| 72 | H | 73 | I | 74 | J | 75 | K | 76 | L | 77 | M | 78 | N | 79 | O |
| 80 | P | 81 | Q | 82 | R | 83 | S | 84 | T | 85 | U | 86 | V | 87 | W |
| 88 | X | 89 | Y | 90 | Z | 91 | [| 92 | \ | 93 |] | 94 | ^ | 95 | - |
| 96 | ' | 97 | a | 98 | b | 99 | c | 100 | d | 101 | e | 102 | f | 103 | g |
| 104 | h | 105 | i | 106 | j | 107 | k | 108 | l | 109 | m | 110 | n | 111 | o |
| 112 | p | 113 | q | 114 | r | 115 | s | 116 | t | 117 | u | 118 | v | 119 | w |
| 120 | x | 121 | y | 122 | z | 123 | { | 124 | | 125 | } | 126 | ~ | 127 | del |

Programas y lenguajes de programación

Programas y lenguajes de programación

Funcionamiento

1. Cada celda de la memoria almacena una secuencia de bits de tamaño fijo.
2. La CPU es capaz de ejecutar acciones especificadas mediante secuencias de instrucciones.
3. Una instrucción describe una acción muy simple: ‘suma esto con aquello’, ‘deja el resultado en tal dirección de memoria’, ‘haz una copia del dato de esta dirección en esta otra dirección’.
4. Las instrucciones también se almacenan en la memoria (secuencias de unos y ceros).

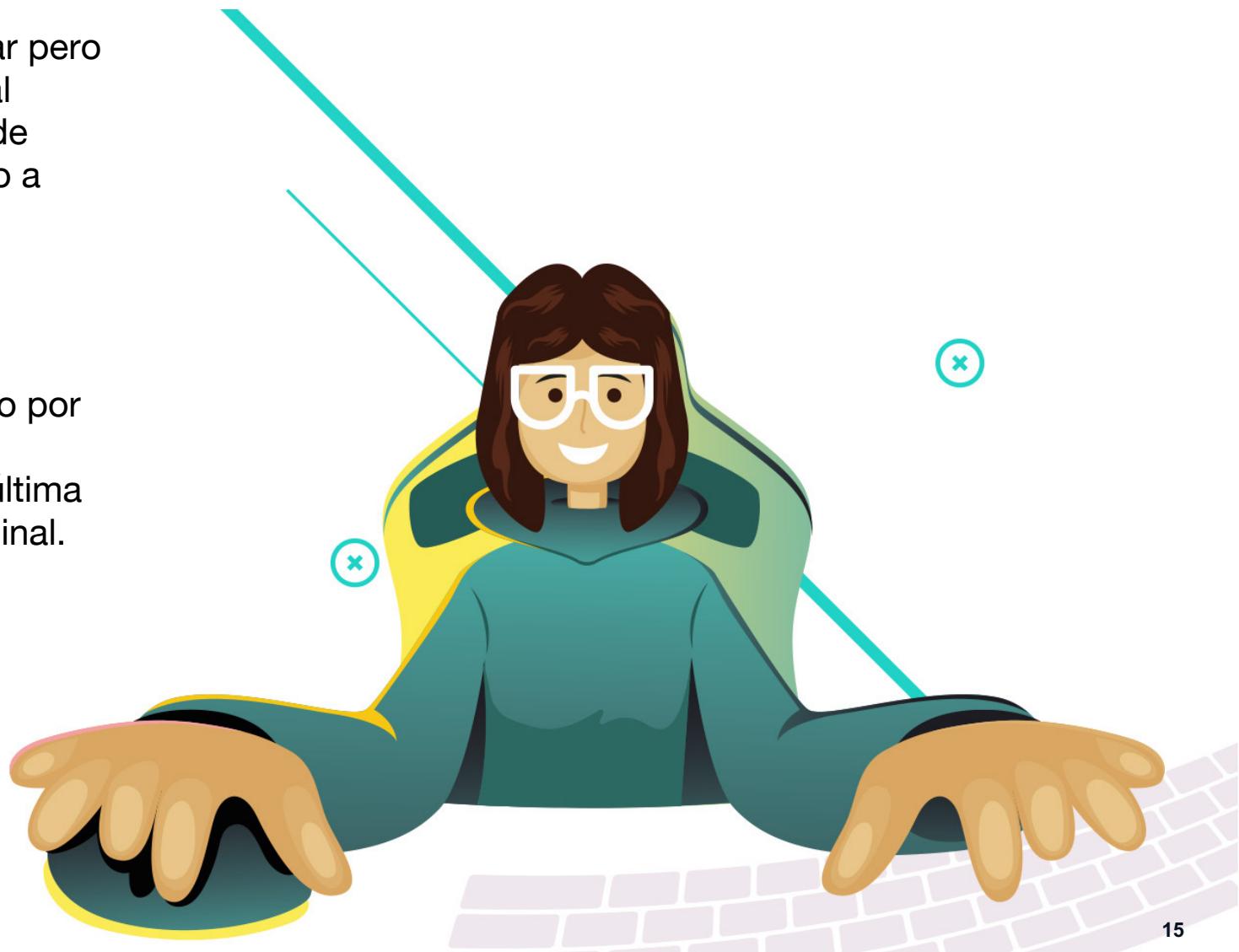
Programas y lenguajes de programación

Ejemplo

Si hay instrucción para multiplicar pero ninguna para elevar un número al cubo, construye una secuencia de instrucciones que calcula el cubo a partir de productos:

Solución:

1. Toma el número y multiplícalo por sí mismo.
2. Multiplica el resultado de la última operación por el número original.



Programas y lenguajes de programación

Programa

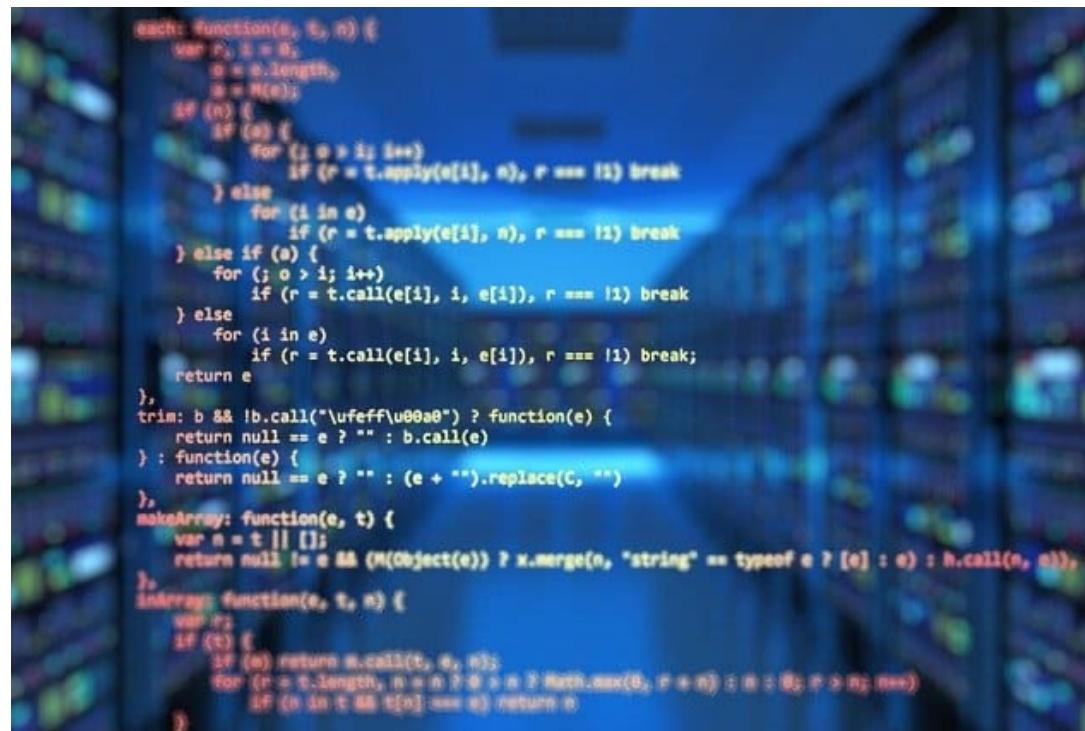
Es una secuencia de instrucciones.



Programas y lenguajes de programación

Programa en código máquina

Es una secuencia de instrucciones que el ordenador puede ejecutar.



```
each: function(e, t, n) {
    var r, i = 0;
    e = e.length;
    n = n || {};
    if (t) {
        if (e) {
            for (i = 0; i < e; i++)
                if (r = t.apply(e[i], n), r === i) break;
        } else
            for (i in e)
                if (r = t.apply(e[i], n), r === i) break;
    } else if (e) {
        for (i = 0; i < e; i++)
            if (r = t.call(e[i], i, e[i]), r === i) break;
    } else
        for (i in e)
            if (r = t.call(e[i], i, e[i]), r === i) break;
    return e
},
trim: b && !b.call("\ufffe\ufe0f") ? function(e) {
    return null == e ? "" : b.call(e)
} : function(e) {
    return null == e ? "" : (e + "").replace(C, "")
},
makeArray: function(e, t) {
    var n = t || [];
    return null != e && (Object(e)) ? x.merge(n, "string" == typeof e ? [e] : e) : b.call(n, e),
n
},
isArray: function(e, t, n) {
    var r;
    if (t) {
        if (e) return b.call(t, e, n);
        for (r = t.length, n = n || 0 > n ? Math.max(0, r + n) : 0; r > n; r++)
            if (b.in(t, r) === e) return e
    }
}
```

Programas y lenguajes de programación

Lenguaje de programación

Es cualquier sistema de notación que permite expresar programas.

Código de máquina

Codifica las secuencias de instrucciones como sucesiones de unos y ceros que siguen ciertas reglas.

Programas y lenguajes de programación

Programa en código máquina

| Memoria | |
|---------|-------------------------------------|
| 1 | 10101011 00001010 00001011 00001101 |
| 2 | 10101011 00001101 00001100 00001101 |
| 3 | 00001110 00001101 00000011 00001101 |
| 4 | 00000000 00000000 00000000 00000000 |
| 5 | |
| 6 | |
| 7 | |
| : | |

Programas y lenguajes de programación

Lenguajes de alto nivel

- **Independiente de un ordenador concreto.**
- **Muy legible.**
- **Permite la traducción automática de un programa al código máquina de cada CPU.**



Programas y lenguajes de programación

Python

- 1. Muy expresivo y compacto.**
- 2. Muy legible.**
- 3. Ofrece un entorno interactivo que facilita la realización de pruebas y ayuda a despejar dudas acerca de ciertas características del lenguaje.**
- 4. Se puede usar como lenguaje orientado a objetos.**
- 5. Posee un rico juego de estructuras de datos que se pueden manipular de modo sencillo.**



Programas y lenguajes de programación

Intérprete

Lee un programa escrito en un lenguaje de alto nivel instrucción a instrucción y, para cada una de ellas, efectúa una traducción a las instrucciones de código máquina equivalentes y las ejecuta inmediatamente.



Programas y lenguajes de programación

Versiones

Los productos software evolucionan añadiendo funcionalidad o corrigiendo errores presentes en una versión determinada.

- Una serie de números separados por puntos.
- 1.1, 1.2, 1.3,... cuando se añaden funcionalidades menores o se corrigen errores.
- 2.0,... cuando hay un cambio de funcionalidad importante.

The screenshot shows the Python Software Foundation website. At the top, there's a navigation bar with links for About, Downloads, Documentation, Community, Success Stories, News, and Events. Below the navigation is a search bar and a "Socialize" button. On the left, there's a sidebar with a tweet from @ThePSF. The tweet discusses a Diversity and Inclusion work group meeting on January 31st. It includes a link to register and mentions a "D&I Friendly Chat" on Hubilo.

Tweets from @ThePSF

Python Softw... @ThePSF · 21h

Join the PSF's Diversity and Inclusion work group on the 31st of January for a friendly chat! This will be the first in a series of quarterly meetings sharing news and information about the Python community around the world.

Register here, [events.hubilo.c...](#)
 D&I Friendly Chat

Python Documentation by Version

Python Documentation by Version

Some previous versions of the documentation remain available online. Use the list below to select a version to view.

For unreleased (in development) documentation, see [In Development Versions](#).

- [Python 3.11.1](#), documentation released on 6 December 2022.
- [Python 3.11.0](#), documentation released on 24 October 2022.
- [Python 3.10.9](#), documentation released on 6 December 2022.
- [Python 3.10.8](#), documentation released on 8 October 2022.
- [Python 3.10.7](#), documentation released on 6 September 2022.
- [Python 3.10.6](#), documentation released on 8 August 2022.

The screenshot shows the Python.org homepage with a dark blue header and footer. The top navigation bar includes links for Python, PSF, Docs, PyPI, Jobs, and Community. Below the header is the Python logo and a search bar with a 'Donate' button. A main content area features two code snippets demonstrating Python list comprehensions and the enumerate function. To the right, a section titled 'Compound Data Types' explains lists and provides a link to more information. At the bottom, there's a call-to-action with a 'Learn More' link.

Python

PSF

Docs

PyPI

Jobs

Community

python™

Donate

Search

GO

Socialize

About

Downloads

Documentation

Community

Success Stories

News

Events

```
# Python 3: List comprehensions
>>> fruits = ['Banana', 'Apple', 'Lime']
>>> loud_fruits = [fruit.upper() for fruit in
fruits]
>>> print(loud_fruits)
['BANANA', 'APPLE', 'LIME']

# List and the enumerate function
>>> list(enumerate(fruits))
[(0, 'Banana'), (1, 'Apple'), (2, 'Lime')]
```

Compound Data Types

Lists (known as arrays in other languages) are one of the compound data types that Python understands. Lists can be indexed, sliced and manipulated with other built-in functions. [More about lists in Python 3](#)

1 2 3 4 5

Python is a programming language that lets you work quickly and integrate systems more effectively. [» Learn More](#)

Get Started

Download

Docs

Jobs

python.org

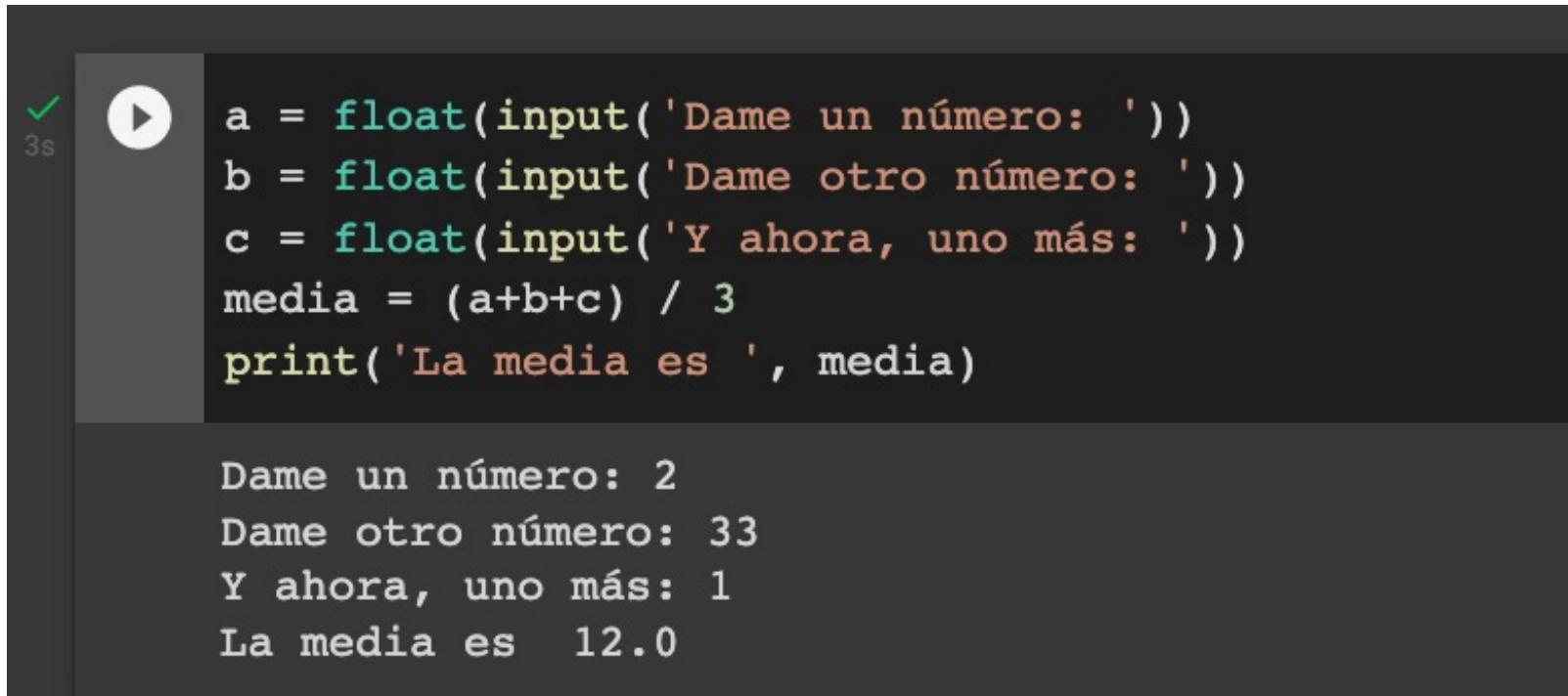


<https://jupytercon.com>

[**jupyter.org**](https://jupyter.org)

Programas y lenguajes de programación

Programa ejemplo en Python



The screenshot shows a dark-themed code editor window. On the left, there's a vertical toolbar with a green checkmark icon and a play button icon. Next to the play button is the text '3s'. The main area contains the following Python code:

```
a = float(input('Dame un número: '))
b = float(input('Dame otro número: '))
c = float(input('Y ahora, uno más: '))
media = (a+b+c) / 3
print('La media es ', media)
```

Below the code, the terminal output is displayed:

```
Dame un número: 2
Dame otro número: 33
Y ahora, uno más: 1
La media es 12.0
```

Programas y lenguajes de programación

Empresas que usan Python

- **Google** usa Python como uno de sus principales lenguajes de desarrollo.
- **Youtube** usa Python en sus sistemas de explotación.
- **Industrial Light & Magic** y **Pixar** recurren a Python en sus sistemas de producción cinematográfica.
- **BitTorrent** está escrito en Python
- Intel, Hewlett-Packard, NASA, JPMorgan Chase, etc.

Programas y lenguajes de programación

Empresas que usan Python

Python juega un papel clave en nuestra cadena de producción. Sin él, un proyecto de la envergadura de «Star Wars: Episodio II» hubiera sido muy difícil de sacar adelante. Visualización de multitudes, proceso de lotes, composición de escenas... Python es lo que lo une todo.

Tommy Brunette, director técnico senior de Industrial Light & Magic.

Python ha sido parte importante de Google desde el principio, y lo sigue siendo a medida que el sistema crece y evoluciona. Hoy día, docenas de ingenieros de Google usan Python y seguimos buscando gente diestra en este lenguaje.

Peter Norvig, director de calidad de búsquedas de Google Inc.

Programas y lenguajes de programación

Lenguajes de programación

List of Hello World Programs in 200 Programming Languages

by Scriptol.com

A complete collection of the smallest possible programs, in each existing programming language. This list is universal, comprised of programming languages and document formats.



A

4GL

```
message "Hello, World!" with style = popup;
```

Abap

```
WRITE 'Hello, World!'.
```

Abc

```
WRITE "Hello, World!"
```

Algoritmos



Algoritmos

¿Qué es?

Una secuencia de pasos orientada a la consecución de un objetivo.

1. Se pueden expresar en lenguaje natural.
2. Eventualmente, traducirlos a un lenguaje de programación.

Algoritmos

Calcular la media de tres números

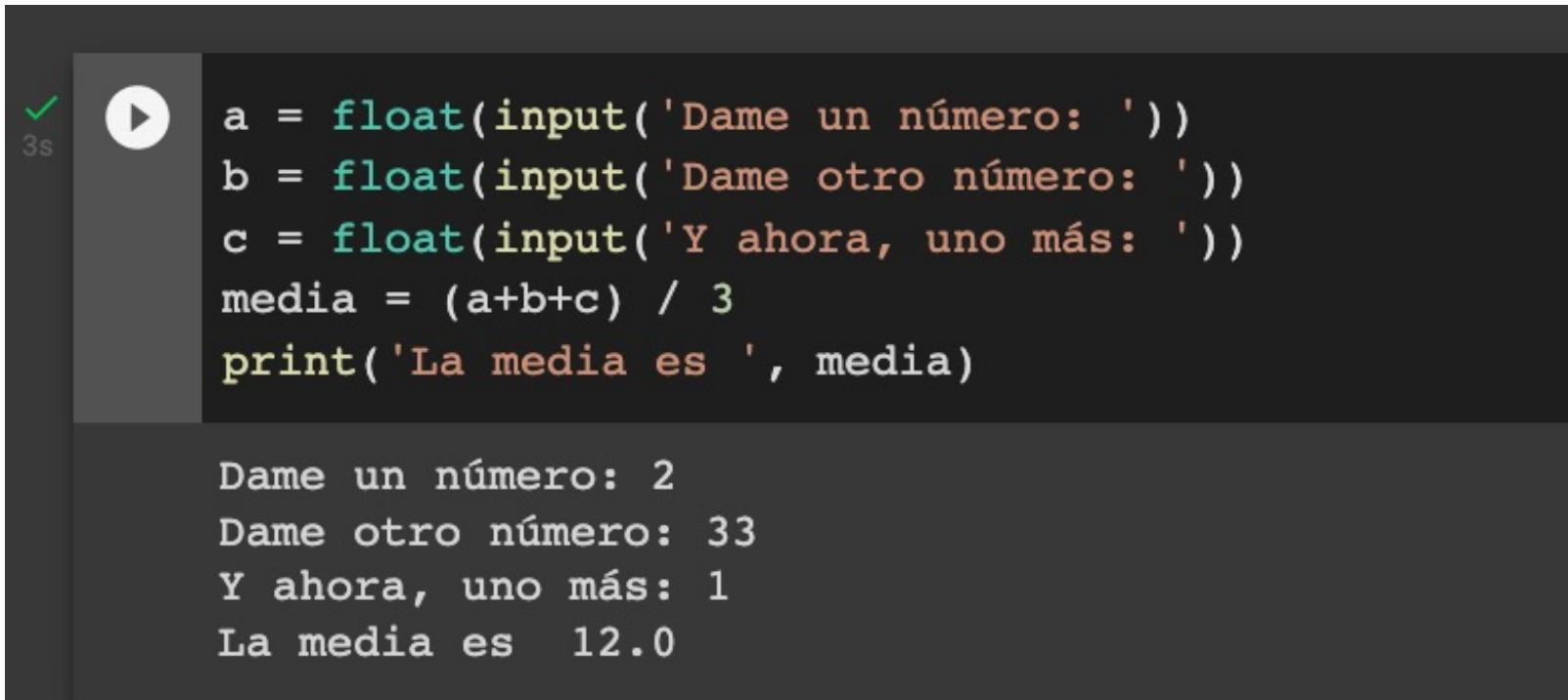
leídos de teclado:

1. Solicitar el valor del primer número.
2. Solicitar el valor del segundo número.
3. Solicitar el valor del tercer número.
4. Sumar los tres números y dividir el resultado por 3.
5. Mostrar el resultado.



Algoritmos

Traducido en Python



The screenshot shows a dark-themed code editor window. On the left, there's a vertical toolbar with a green checkmark icon and a play button icon. Next to the play button is the text '3s'. The main area contains the following Python code:

```
a = float(input('Dame un número: '))
b = float(input('Dame otro número: '))
c = float(input('Y ahora, uno más: '))
media = (a+b+c) / 3
print('La media es ', media)
```

Below the code, the terminal output is displayed:

```
Dame un número: 2
Dame otro número: 33
Y ahora, uno más: 1
La media es 12.0
```

Estudia este ejemplo

Primera receta:

1. Poner aceite en una sartén
2. Encender el fuego
3. Calentar el aceite
4. Coger un huevo
5. Romper la cáscara
6. Verter el contenido del huevo en la sartén
7. Aderezar con sal
8. Esperar a que tenga buen aspecto



Estudia este ejemplo

Primera receta:

1. Poner aceite en una sartén
2. Encender el fuego
3. Calentar el aceite
4. Coger un huevo
5. Romper la cáscara
6. Verter el contenido del huevo en la sartén
7. Aderezar con sal
8. Esperar a que tenga buen aspecto

- **Qué tipo de huevo utilizamos?: gallina? Rana?**
- **Cuánta sal?: una pizca? Un kilo?**
- **Cuánto aceite? Un cm³? Un litro?**
- **Cuál es el resultado?**



Estudia este ejemplo

Segunda receta:

Ingredientes: 10 cc de aceite de oliva, una gallina y una pizca de sal.

Método:

1. Esperar a que la gallina ponga un huevo
2. Poner aceite en una sartén
3. Encender el fuego
4. Calentar el aceite
5. Coger el huevo
6. Romper la cáscara
7. Verter el contenido del huevo en la sartén
8. Aderezar con sal
9. Esperar a que tenga buen aspecto

Presentación: depositar el huevo frito, sin aceite, en un plato.



Estudia este ejemplo

Segunda receta:

Ingredientes: 10 cc de aceite de oliva, una gallina y una pizca de sal.

Método:

1. Esperar a que la gallina ponga un huevo
2. Poner aceite en una sartén
3. Encender el fuego
4. Calentar el aceite
5. Coger el huevo
6. Romper la cáscara
7. Verter el contenido del huevo en la sartén
- **Cuán caliente el aceite?**
- **Cuánto tiempo hay que esperar?**
- **Estamos seguros de que la gallina pondrá el huevo?**
- Preparación: dejar que el huevo frite sin aceite, en un plato.



Estudia este ejemplo

Tercera receta:

Ingredientes: 10 cc de aceite de oliva, **un huevo de gallina** y una pizca de sal.

Método:

1. Poner aceite en una sartén
2. Encender el fuego a **medio gas**
3. Calentar el aceite **hasta que humee ligeramente**
4. Coger un huevo
5. Romper la cáscara **con el poder de la mente, sin tocar el huevo**
6. Verter el contenido del huevo en la sartén
7. Aderezar con sal
8. Esperar a que tenga buen aspecto

Presentación: depositar el huevo frito, sin aceite, en un plato.



Estudia este ejemplo

Tercera receta:

Ingredientes: 10 cc de aceite de oliva, **un huevo de gallina** y una pizca de sal.

Método:

1. Poner aceite en una sartén
2. Encender el fuego a **medio gas**
3. Calentar el aceite **hasta que humee ligeramente**
4. Coger un huevo
5. Romper la cáscara **con el poder de la mente, sin tocar el huevo** • **Es factible?**
6. Verter el contenido del huevo en la sartén
7. Aderezar con sal
8. Esperar a que tenga buen aspecto

Presentación: depositar el huevo frito, sin aceite, en un plato.



Estudia este ejemplo

Cuarta receta:

Ingredientes: 10 cc de aceite de oliva, un huevo de gallina y una pizca de sal.

Método:

1. Poner aceite en una sartén
2. Sintonizar una emisora musical en la radio
3. Encender el fuego a medio gas
4. Echar una partida al solitario
5. Calentar el aceite hasta que humee ligeramente
6. Coger un huevo
7. Romper la cáscara
8. Verter el contenido del huevo en la sartén
9. Aderezar con sal
10. Esperar a que tenga buen aspecto

Presentación: depositar el huevo frito, sin aceite, en un plato.



Estudia este ejemplo

Cuarta receta:

Ingredientes: 10 cc de aceite de oliva, un huevo de gallina y una pizca de sal.

Método:

1. Poner aceite en una sartén
2. Sintonizar una emisora musical en la radio
3. Encender el fuego a medio gas
4. Echar una partida al solitario
5. Calentar el aceite hasta que humee ligeramente
6. ~~• Coger un huevo~~
7. Romper la cáscara
8. Verter el contenido del huevo en la sartén
9. Aderezar con sal
10. Esperar a que tenga buen aspecto

Presentación: depositar el huevo frito, sin aceite, en un plato.



Algoritmos

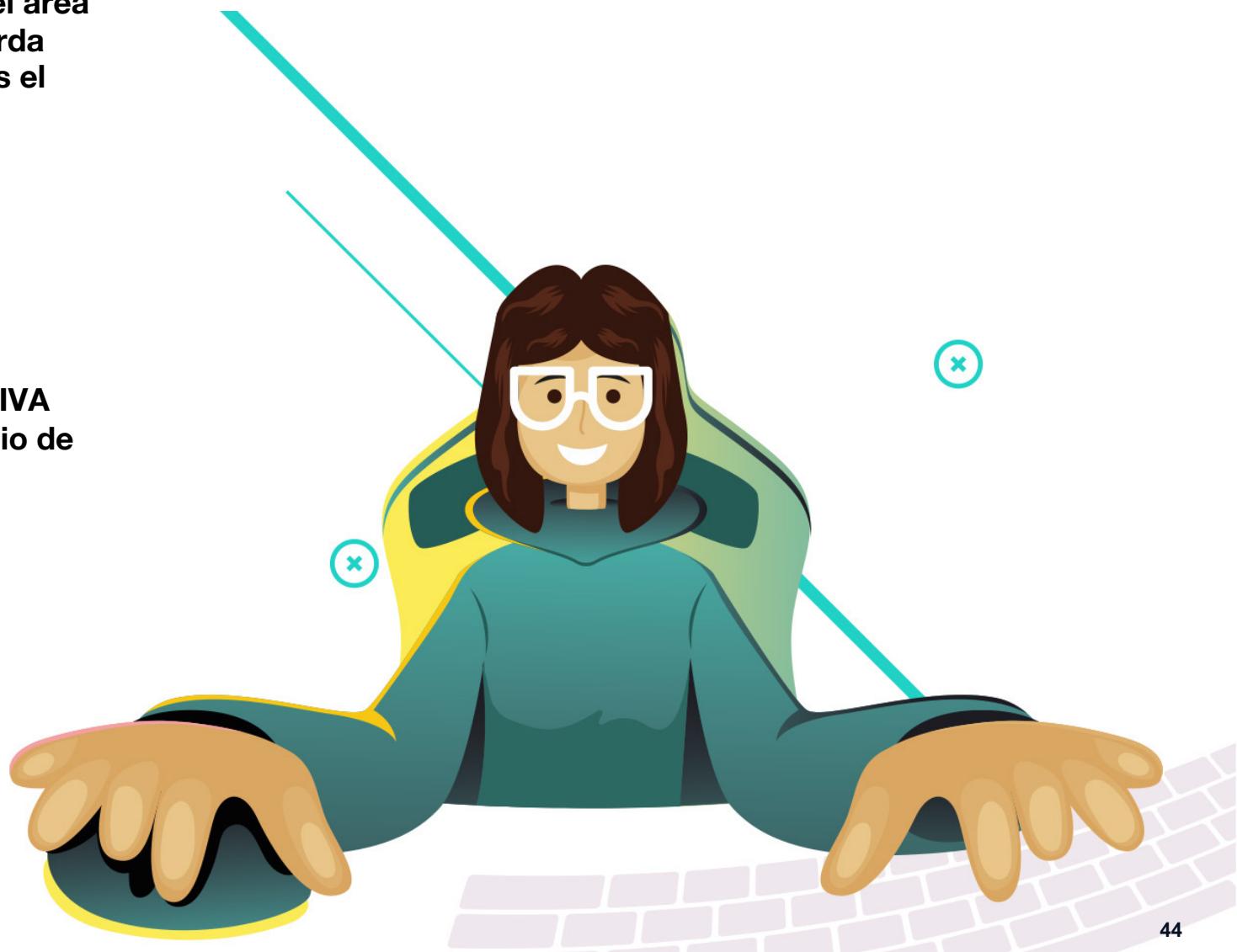
Características

1. Ha de tener cero o más **datos de entrada**.
2. Debe proporcionar uno o más **datos de salida** como resultado.
3. Cada paso del algoritmo ha de estar definido con **exactitud**, sin la menor ambigüedad.
4. Ha de ser **finito**, es decir, debe finalizar tras la ejecución de un número finito de pasos, cada uno de los cuales ha de ser ejecutable en tiempo finito.
5. Debe ser **efectivo**, es decir, cada uno de sus pasos ha de poder ejecutarse en tiempo finito con unos recursos determinados.

Ahora tú

- Diseña un algoritmo para calcular el área de un círculo dado su radio. Recuerda que el área de un círculo es π veces el cuadrado del radio.

- Diseña un algoritmo que calcule el IVA (21%) de un producto dado su precio de venta sin IVA.



Operadores aritméticos



Operadores aritméticos

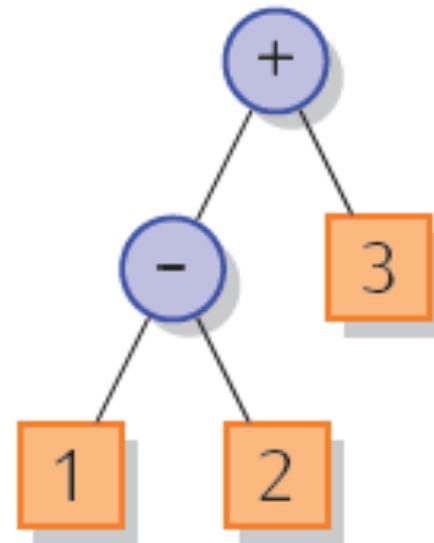
Características

1. Puedes introducir varias operaciones en una misma línea o expression.
2. El orden en el que se efectúan las operaciones es (en principio) de izquierda a derecha

Operadores aritméticos

Características

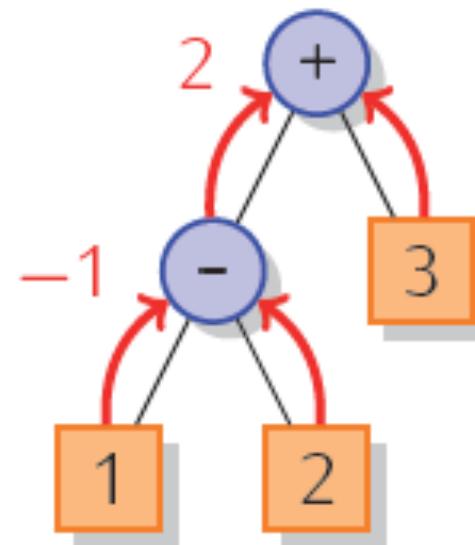
<<1 - 2 + 3>>



Operadores aritméticos

Características

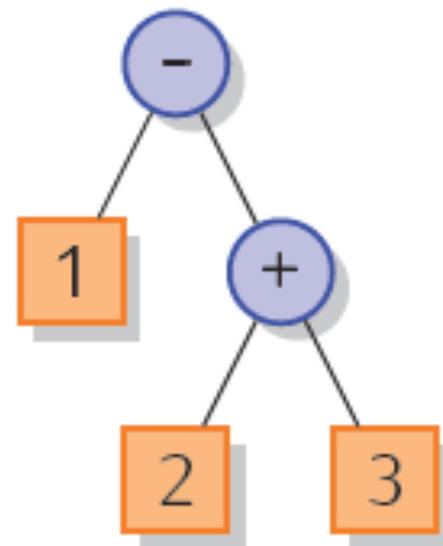
<<1 - 2 + 3>>



Operadores aritméticos

Características

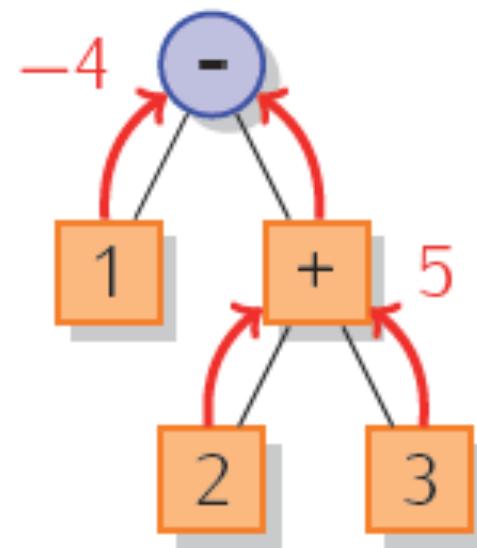
<<1 - (2 + 3)>>



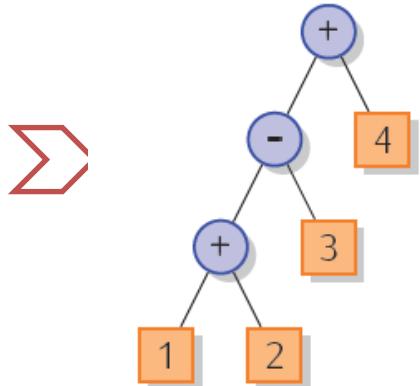
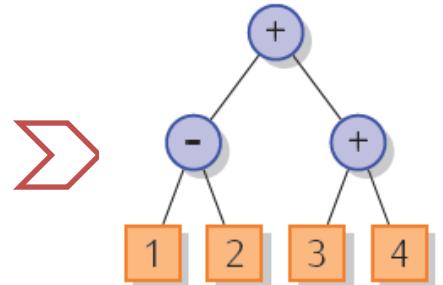
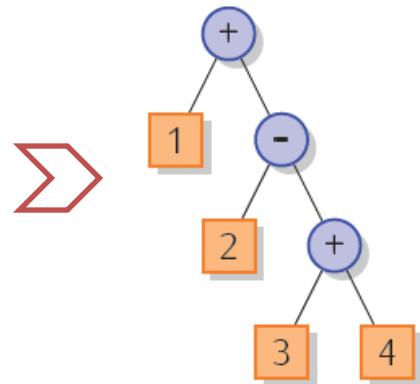
Operadores aritméticos

Características

<<1 - (2 + 3)>>



Ahora tú

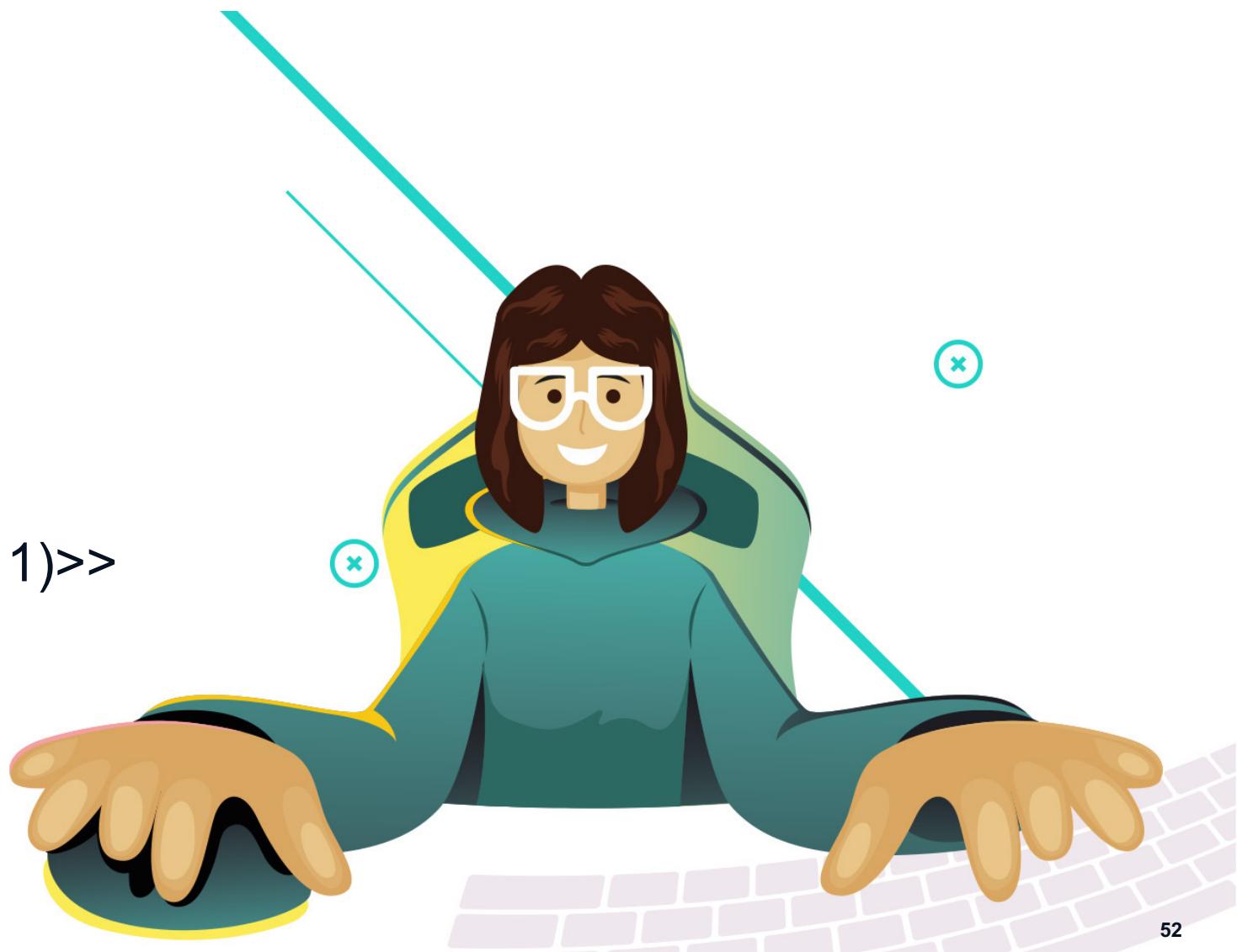


Ahora tú

➤ <<1 + 2 + 3 + 4>>

➤ <<1 - 2 - 3 - 4>>

➤ <<1 - (2 - (3 - 4) + 1)>>



Espacios en blanco

Características

1. No puedes poner espacios en medio de un número.
2. No puedes poner espacios al principio de la expresión

```
>>>     10 + 20 + 30<
File "<stdin>", line 1
  10 + 20 + 30
  ^
IndentationError: unexpected indent
```

```
>>> 10 + 20 + 30<
60
>>> 10+20+30<
60
>>> 10      +20      + 30<
60
>>> 10+ 20+30<
60
```

```
>>> 10 + 2    0 + 30<
File "<stdin>", line 1
  10 + 2    0 + 30
  ^
SyntaxError: invalid syntax
```

Operadores

Multiplicación, división y división entera

```
>>> 2 * 3  
6  
>>> 3 / 2  
1.5  
>>> 4 / 2  
2.0  
>>> 3 * 4 / 2  
6.0  
>>> 12 / 3 * 2  
8.0
```

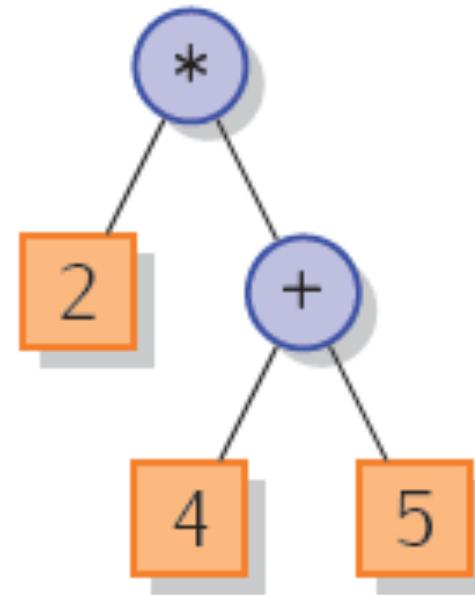
```
>>> 3 // 2  
1  
>>> 4 // 2  
2  
>>> -3 // 2  
-2
```

Operadores

Características

<<2 * (4 + 5)>> ??

<<2 * 4 + 5 >> ??



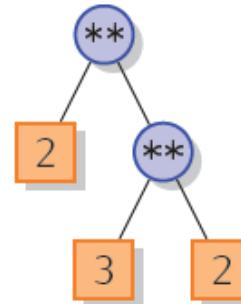
Operadores

Módulo y exponenciación

```
>>> 27 % 5↵  
2  
>>> 25 % 5↵  
0
```

```
>>> 2 ** 3↵  
8
```

Asociativa por la derecha

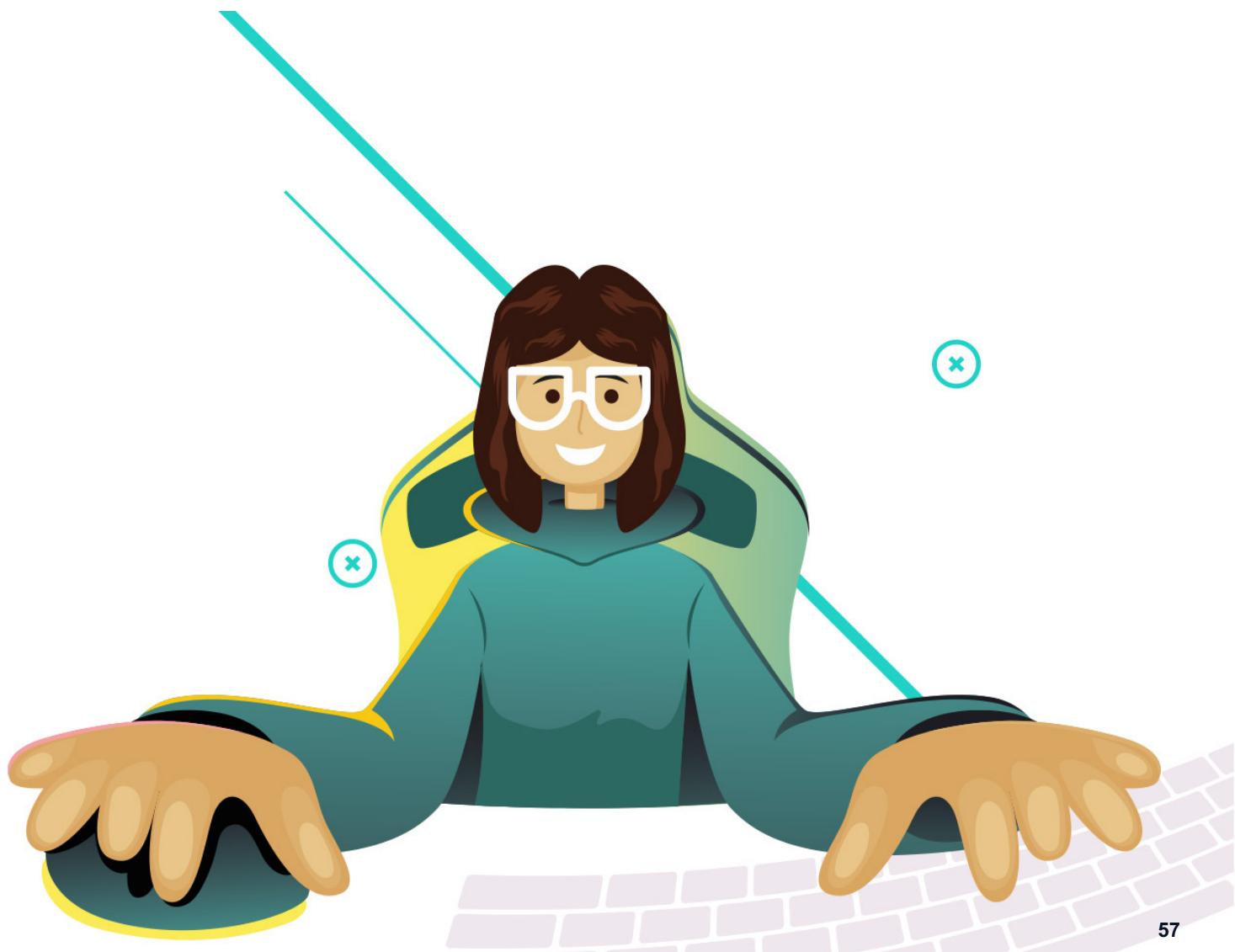


Ahora tú

- <<2 + 3 ** 2 * 5>>

- <<2 + ((3 ** 2) * 5)>>

- <<2 + 3 ** (2 * 5)>>



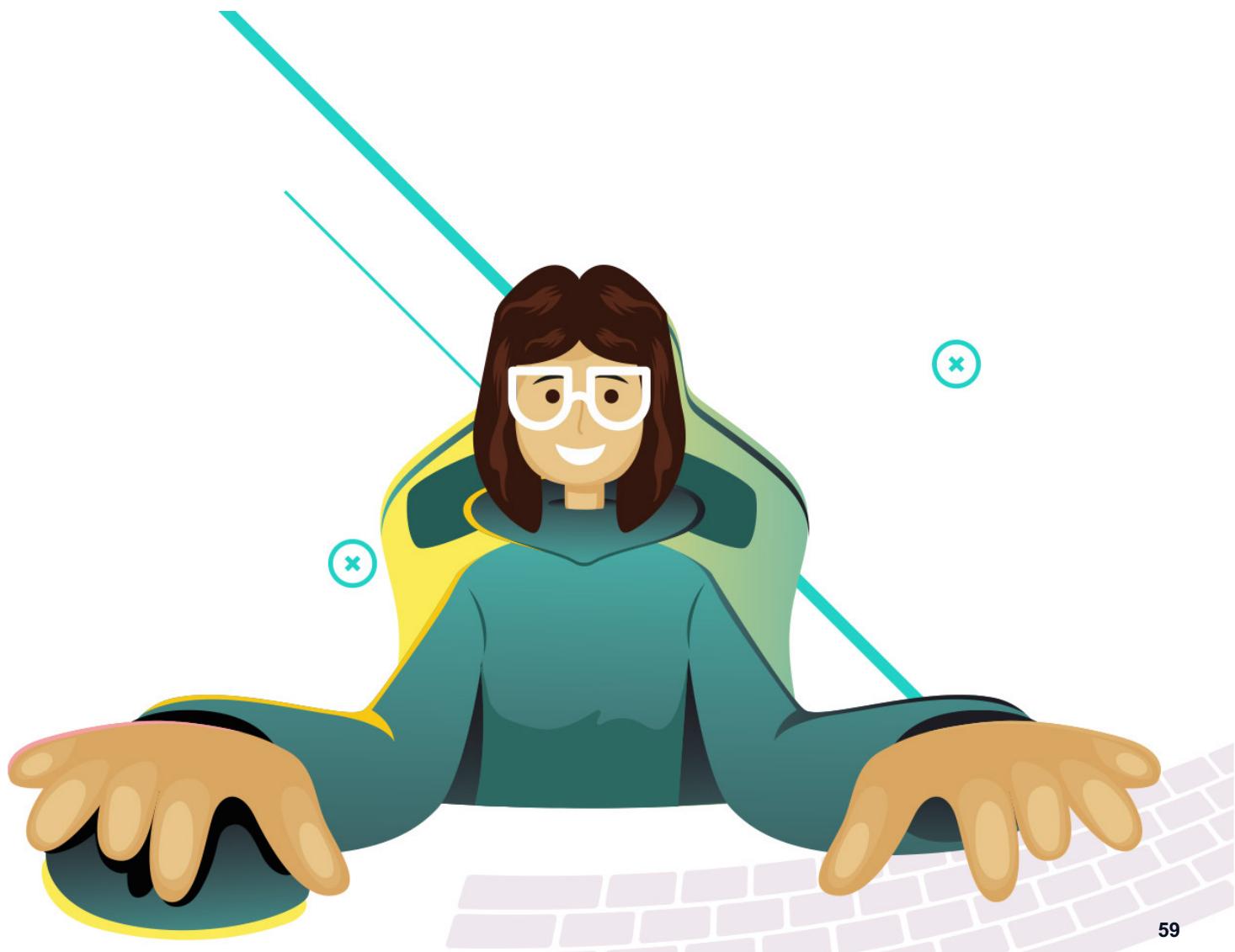
Operadores

Características

| Operación | Operador | Aridad | Asociatividad | Precedencia |
|------------------|-----------------|---------|------------------|-------------|
| Exponenciación | <code>**</code> | Binario | Por la derecha | 1 |
| Identidad | <code>+</code> | Unario | — | 2 |
| Cambio de signo | <code>-</code> | Unario | — | 2 |
| Multiplicación | <code>*</code> | Binario | Por la izquierda | 3 |
| División | <code>/</code> | Binario | Por la izquierda | 3 |
| División entera | <code>//</code> | Binario | Por la izquierda | 3 |
| Módulo (o resto) | <code>%</code> | Binario | Por la izquierda | 3 |
| Suma | <code>+</code> | Binario | Por la izquierda | 4 |
| Resta | <code>-</code> | Binario | Por la izquierda | 4 |

Ahora tú (opcional)

- <<2 + 3 + 1 + 2>>
- <<2 + 3 * 1 + 2>>
- <<(2 + 3) * 1 + 2>>
- <<(2 + 3) * (1 + 2)>>
- <<+---6>>
- <<-+-+6>>
- <<-3 / 2 - 1>>
- <<-3 // 2 - 1>>



Ahora tú (opcional)

a) $2 + (3 \cdot (6/2))$

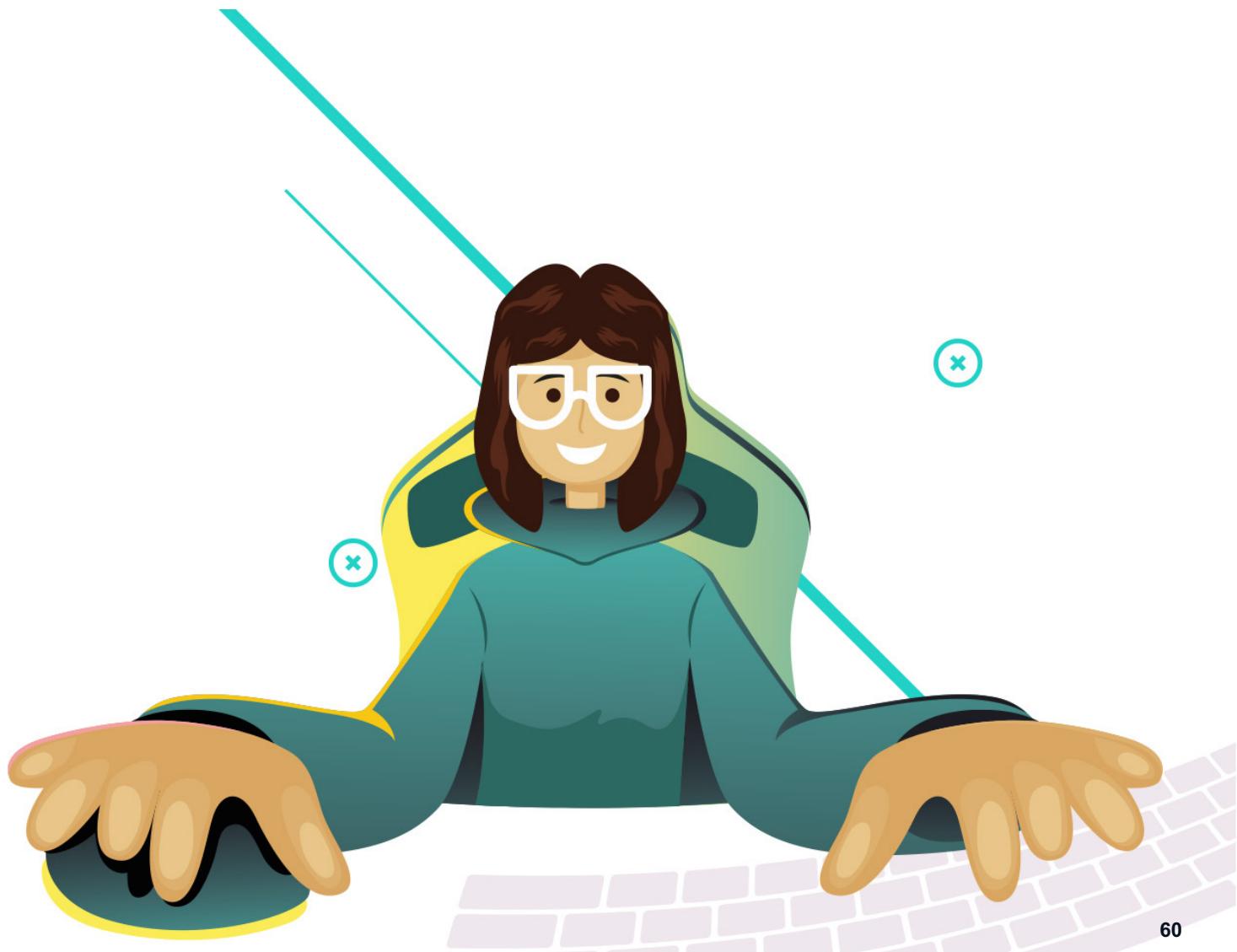
b) $\frac{4+6}{2+3}$

c) $(4/2)^5$

d) $(4/2)^{4+2^2}$

e) $(-3)^2$

f) $-(3^2)$



Errores de tecleo y excepciones



Errores de tecleo y excepciones

Ejemplos

```
>>> 1 + * 3
File "<stdin>", line 1
  1 + * 3
          ^
SyntaxError: invalid syntax
```

```
>>> 1 + 2)
File "<stdin>", line 1
  1 + 2)
          ^
SyntaxError: invalid syntax
>>> 2 + 3 %
File "<stdin>", line 1
  2 + 3 %
          ^
SyntaxError: invalid syntax
>>> 1 / 0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: int division or modulo by zero
```

Tipos de datos



Tipos de datos

Entero y flotante

1. Los enteros suelen ocupar menos memoria
2. Las operaciones entre enteros son, generalmente, más rápidas.

2e3.

3.2e-3.

.1

2.

Tipos de datos

Booleano

True / False

| and | | |
|-----------|---------|-----------|
| operандос | | resultado |
| izquierdo | derecho | |
| True | True | True |
| True | False | False |
| False | True | False |
| False | False | False |

| or | | |
|-----------|---------|-----------|
| operандос | | resultado |
| izquierdo | derecho | |
| True | True | True |
| True | False | True |
| False | True | True |
| False | False | False |

| Operación | Operador | Aridad | Asociatividad | Precedencia |
|------------|----------|---------|------------------|-------------|
| Negación | not | Unario | — | alta |
| Conjunción | and | Binario | Por la izquierda | media |
| Disyunción | or | Binario | Por la izquierda | baja |

| not | |
|----------|-----------|
| operando | resultado |
| True | False |
| False | True |

Tipos de datos

Operadores de comparación

| operador | comparación |
|----------|----------------------|
| != | es distinto de |
| == | es igual que |
| < | es menor que |
| <= | es menor o igual que |
| > | es mayor que |
| >= | es mayor o igual que |

```
>>> 2 < 1
False
>>> 1 < 2
True
>>> 5 > 1
True
>>> 5 >= 1
True
>>> 5 > 5
False
>>> 2 == 3
False
>>> 2 == 2
True
>>> 2.1 == 2.1
True
>>> True == True
True
>>> True == False
False
>>> 2 == 1+1
True
```

Resumen operadores

| Operación | Operador | Aridad | Asociatividad | Precedencia |
|-------------------|--------------------|---------|------------------|-------------|
| Exponenciación | <code>**</code> | Binario | Por la derecha | 1 |
| Identidad | <code>+</code> | Unario | — | 2 |
| Cambio de signo | <code>-</code> | Unario | — | 2 |
| Multiplicación | <code>*</code> | Binario | Por la izquierda | 3 |
| División | <code>/</code> | Binario | Por la izquierda | 3 |
| División entera | <code>//</code> | Binario | Por la izquierda | 3 |
| Módulo (o resto) | <code>%</code> | Binario | Por la izquierda | 3 |
| Suma | <code>+</code> | Binario | Por la izquierda | 4 |
| Resta | <code>-</code> | Binario | Por la izquierda | 4 |
| Igual que | <code>==</code> | Binario | — | 5 |
| Distinto de | <code>!=</code> | Binario | — | 5 |
| Menor que | <code><</code> | Binario | — | 5 |
| Menor o igual que | <code><=</code> | Binario | — | 5 |
| Mayor que | <code>></code> | Binario | — | 5 |
| Mayor o Igual que | <code>>=</code> | Binario | — | 5 |
| Negación | <code>not</code> | Unario | — | 6 |
| Conjunción | <code>and</code> | Binario | Por la izquierda | 7 |
| Disyunción | <code>or</code> | Binario | Por la izquierda | 8 |

Ahora tú (opcional)

```
>>> True == True != False↵
>>> 1 < 2 < 3 < 4 < 5↵
>>> (1 < 2 < 3) and (4 < 5)↵
>>> 1 < 2 < 4 < 3 < 5↵
>>> (1 < 2 < 4) and (3 < 5)↵
```



Variables y asignaciones



Variables y asignaciones

Variables

Para que el ordenador recuerde ciertos valores para usarlos más adelante.

```
>>> 2 * 3.14159265359 * 1.298373
8.157918156839218
>>> 3.14159265359 * 1.298373 ** 2
5.296010335524904
```

```
>>> pi = 3.14159265359
>>> r = 1.298373
>>> 2 * pi * r
8.157918156839218
>>> pi * r ** 2
5.296010335524904
```

Variables y asignaciones

Asignaciones

Acto de dar valor a una variable.

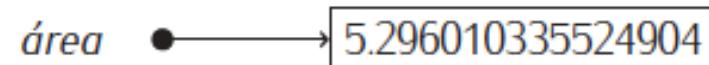
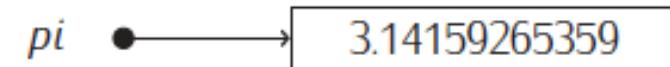
Al asignar un valor a una variable que no existía, Python reserva un espacio en la **memoria**, almacena el valor en él y crea una asociación entre el nombre de la variable y la dirección de memoria de dicho espacio.

Variables y asignaciones

Asignaciones

```
>>> pi = 3.14159265359
>>> r = 1.298373
>>> perímetro = 2 * pi * r
>>> área = pi * r**2
```

Variable = expresión



Variables y asignaciones

Asignaciones

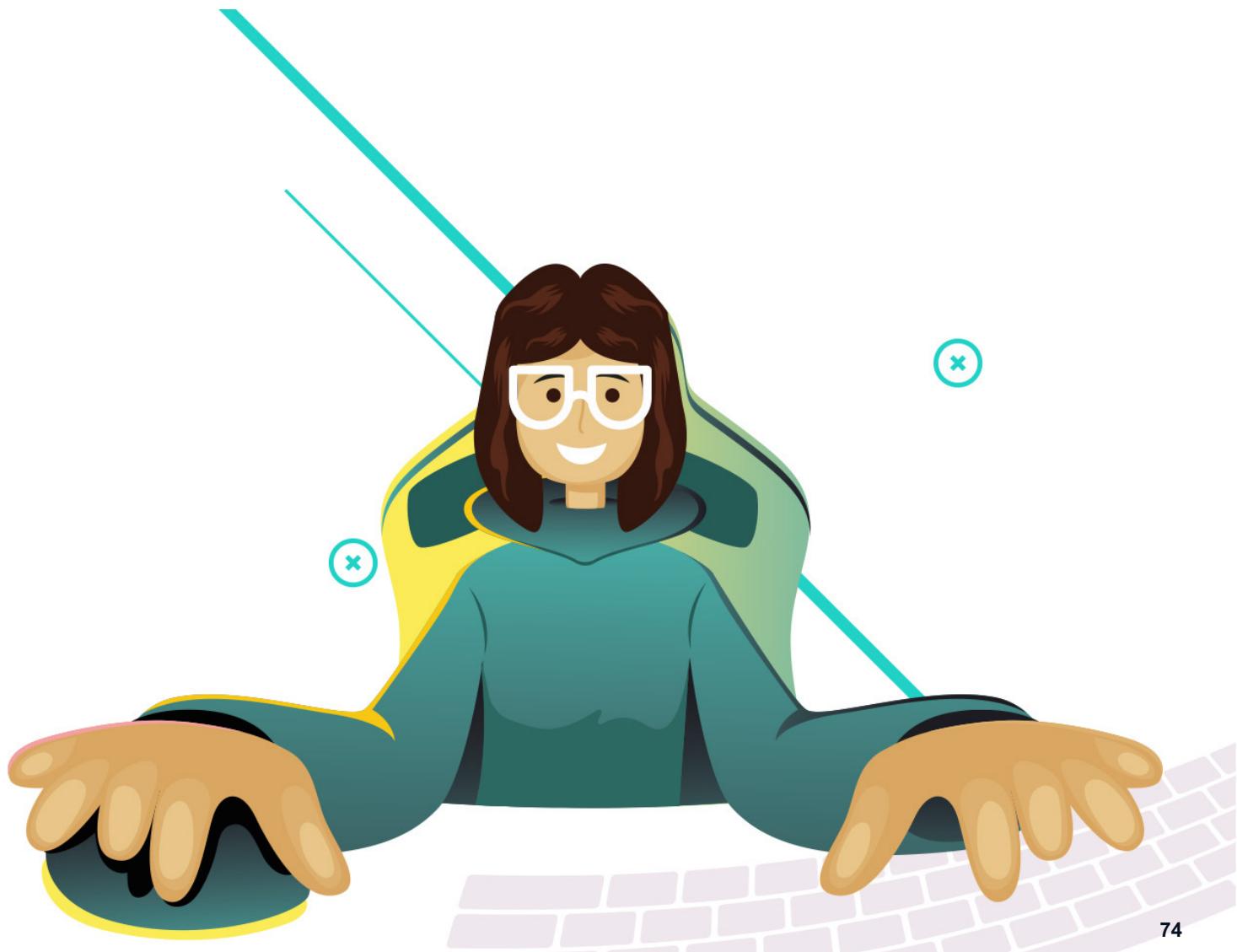
Variable = expresión

1. Evaluar la expresión a la derecha del símbolo =
 2. Guardar el valor resultante en la variable indicada a la izquierda del símbolo =
-
- Se puede asignar valor a una variable cuantas veces se quiera

== no es =

- » <<a = 10>>
- » <<a == 1>>

a?



Variables y asignaciones

Identificador

Es el nombre de una variable.

1. Letras minúsculas
2. Letras mayúsculas
3. Dígitos
4. El carácter de subrayado _

*El primer carácter no puede ser dígito.

*No puede coincidir con una palabra clave o reservada
(and, as, assert, break, class, continue, def, del, elif, else, except, False, finally, for, from, global, if, import, in, is, lambda, nonlocal, None, not, or, pass, raise, return, True, try, with, while, yield)

Variables y asignaciones

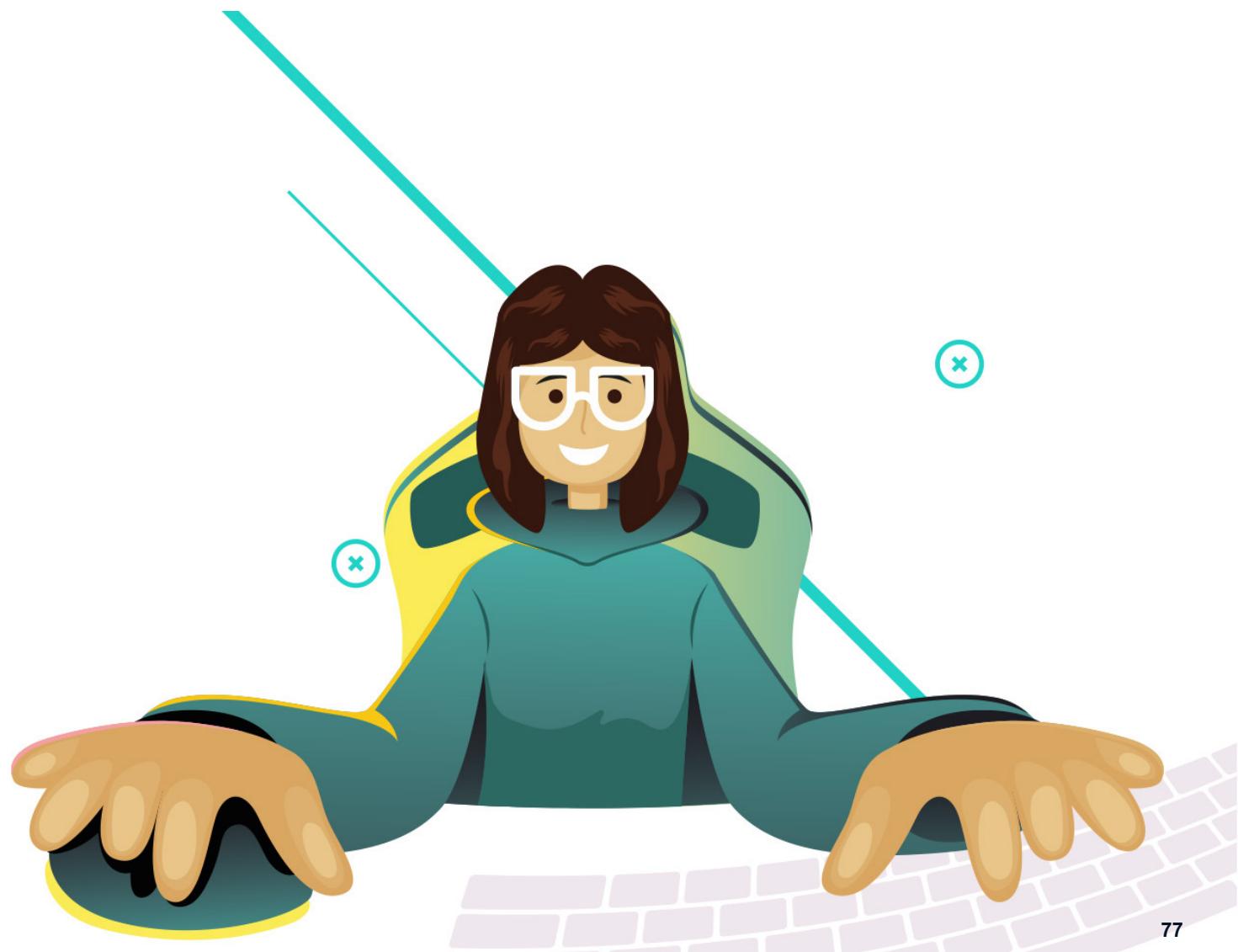
Identificador

- *edad media* son 2 identificadores porque tienen un espacio en medio.
- *edad_media*
- *edadMedia*

Escoge siempre nombres que guarden relación con los datos del problema

Son válidos los siguientes identificadores?

1. Identificador
2. Indice\dos
3. Dos palabras
- 4.
5. 12horas
6. hora12
7. desviación
8. año
9. from
10. var!
11. 'var'
12. import_from
13. UnaVariable
14. a(b)
15. 12
16. uno.dos
17. X
18. área
19. area-rect
20. x_____1
21. _____1
22. _x_
23. x_x



Variables y asignaciones

Iniciar

No se puede usar una variable a la que no se le ha asignado un valor.

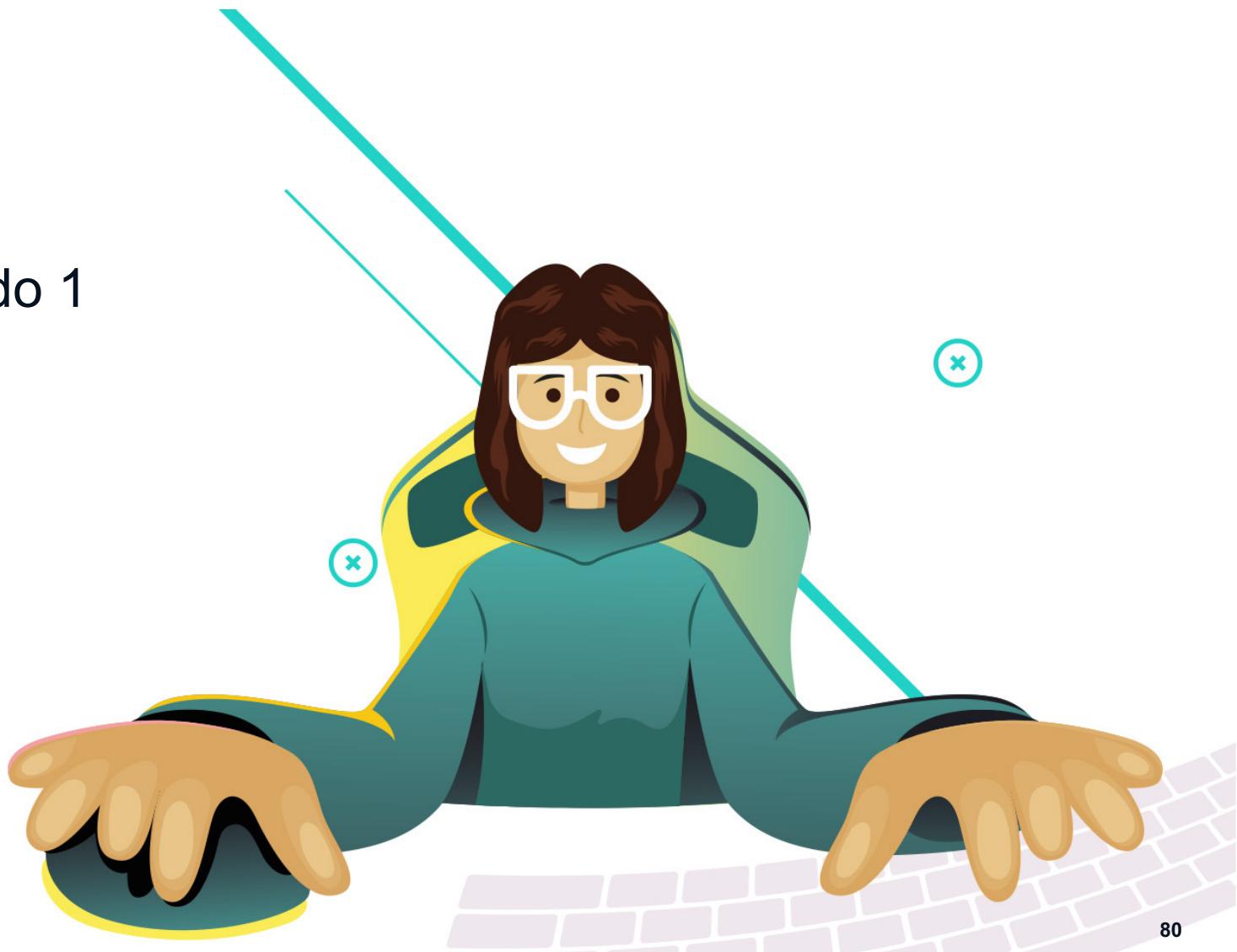
Más operadores

Para secuencias de bits

| En decimal | | En binario | |
|------------|-----------|----------------------|-----------|
| Expresión | Resultado | Expresión | Resultado |
| 5 & 12 | 4 | 00000101 & 00001100 | 00000100 |
| 5 12 | 13 | 00000101 00001100 | 00001101 |
| 5 ^ 12 | 9 | 00000101 ^ 00001100 | 00001001 |
| 5 << 1 | 10 | 00000101 << 00000001 | 00001010 |
| 5 << 2 | 20 | 00000101 << 00000010 | 00010100 |
| 5 << 3 | 40 | 00000101 << 00000011 | 00101000 |
| 5 >> 1 | 2 | 00000101 >> 00000001 | 00000010 |

Ejercicios

- Notebook 1. Apartado 1



Datos de cadena de texto



Datos de cadena de texto

¿Qué es una cadena?

Una cadena es una secuencia de caracteres (letras, números, espacios en blanco, signos de puntuación,...)

Va entre **comillas simples o dobles**.

'Esto es una cadena de texto'

"Y esto Tambien :)"

Datos de cadena de texto

Caracteres especiales

| Secuencia de escape para carácter de control | Resultado |
|--|--|
| \a | Carácter de «campana» (BEL) |
| \b | «Espacio atrás» (BS) |
| \f | Alimentación de formulario (FF) |
| \n | Salto de línea (LF) |
| \r | Retorno de carro (CR) |
| \t | Tabulador horizontal (TAB) |
| \v | Tabulador vertical (VT) |
| \ooo | Carácter cuyo código en octal es <i>ooo</i> |
| \xhh | Carácter cuyo código en hexadecimal es <i>hh</i> |

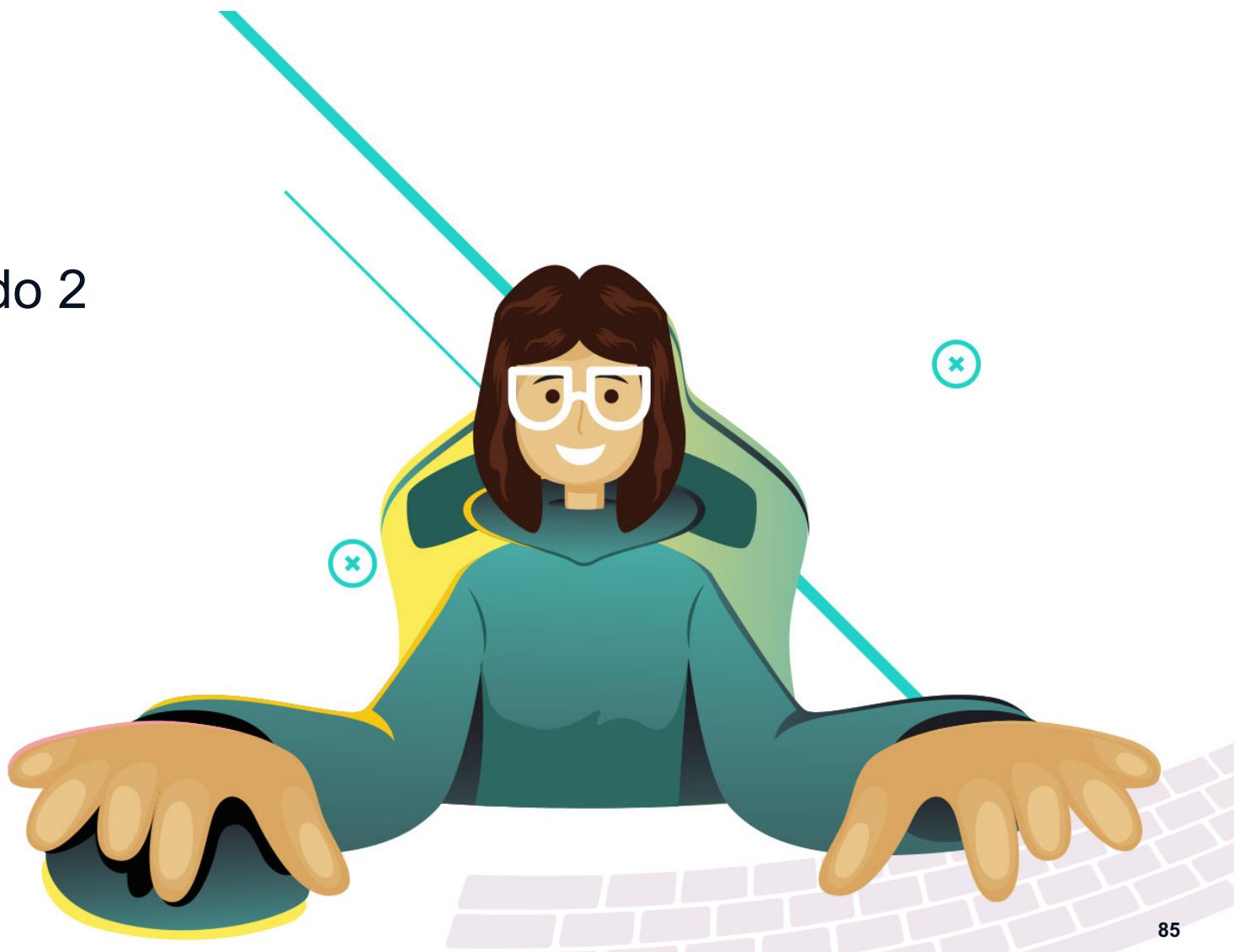
Datos de cadena de texto

Caracteres especiales

| Otras secuencias de escape | Resultado |
|----------------------------|---|
| \\" | Carácter barra invertida (\") |
| \' | Comilla simple (') |
| \\" | Comilla doble (") |
| \ y salto de línea | Se ignora (para expresar una cadena en varias líneas) |

Ejercicios

- Notebook 1. Apartado 2



Funciones predefinidas



Funciones predefinidas

¿Qué es una función?

Son métodos ya definidos que realizan una operación concreta como por ejemplo, calcular el valor absoluto de un número. El valor sobre el que se aplican es el **argumento** y éste va entre paréntesis.

abs(-3) devuelve el valor absoluto del número -3, que es 3.

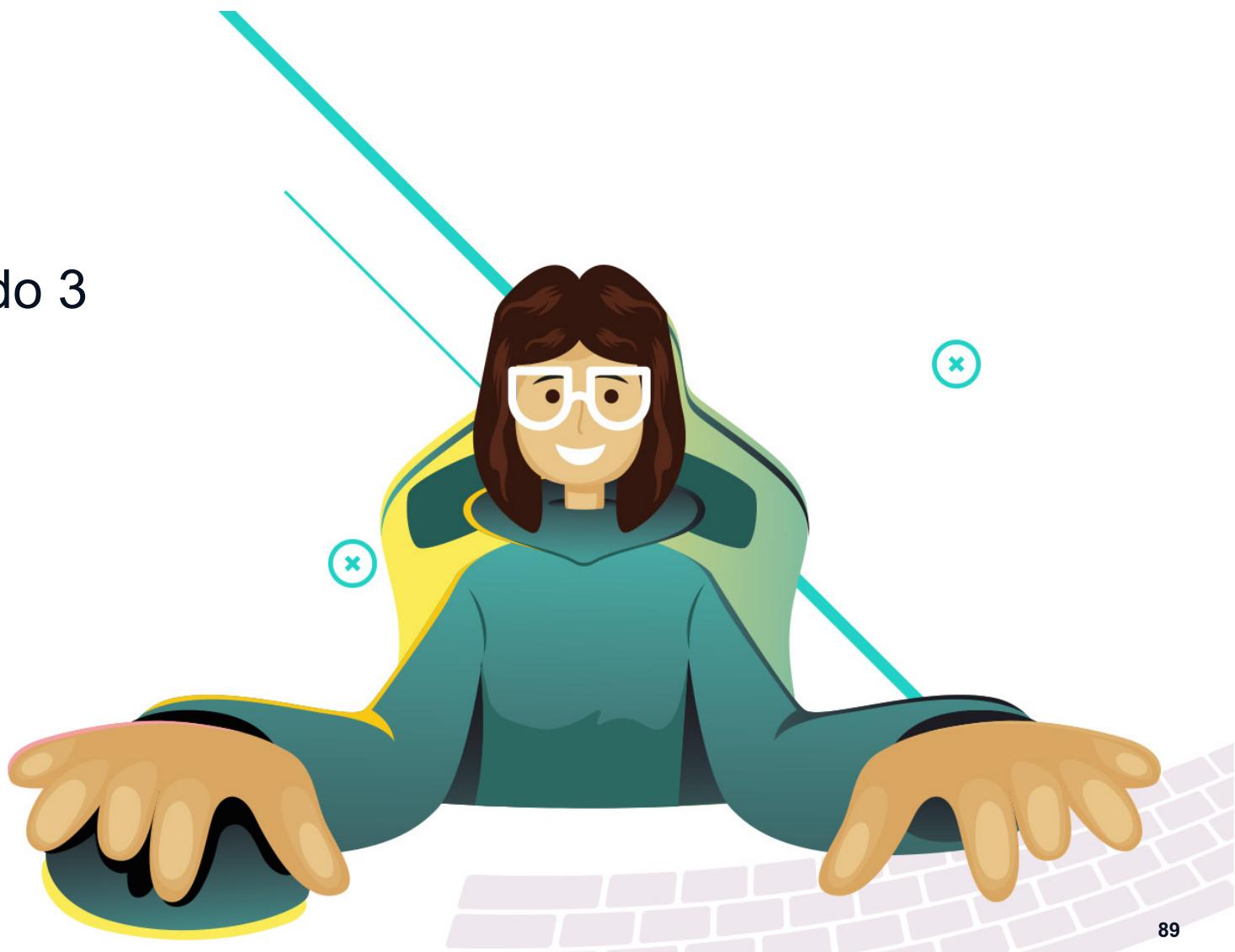
Funciones predefinidas

Algunas funciones

- `abs()`: calcular el valor absoluto
- `float()`: *conversion a flotante*
- `int()`: *conversion a entero*
- `str()`: *conversion a cadena/string*
- `round()`: *redondeo. A entero por defecto, o a un número de decimales indicado en un segundo argumento.*

Ejercicios

➤ Notebook 1. Apartado 3



Módulos e importación de funciones y variables

Módulos e importación de funciones y variables

Importar funciones

Python tiene disponibles funciones especiales, pero no se pueden usar si no las importamos antes.

Si usamos un asterisco, se importan todas las funciones de un modulo.

```
from math import sen  
from math import cos, pi, e  
from math import *
```

```
import math  
math.sen(2.3)
```

Módulos e importación de funciones y variables

Módulo math

- $\sin(x)$
- $\cos(x)$
- $\tan(x)$
- $\exp(x)$
- $\text{ceil}(x)$
- $\text{floor}(x)$
- $\log()$
- $\log10()$
- $\sqrt()$

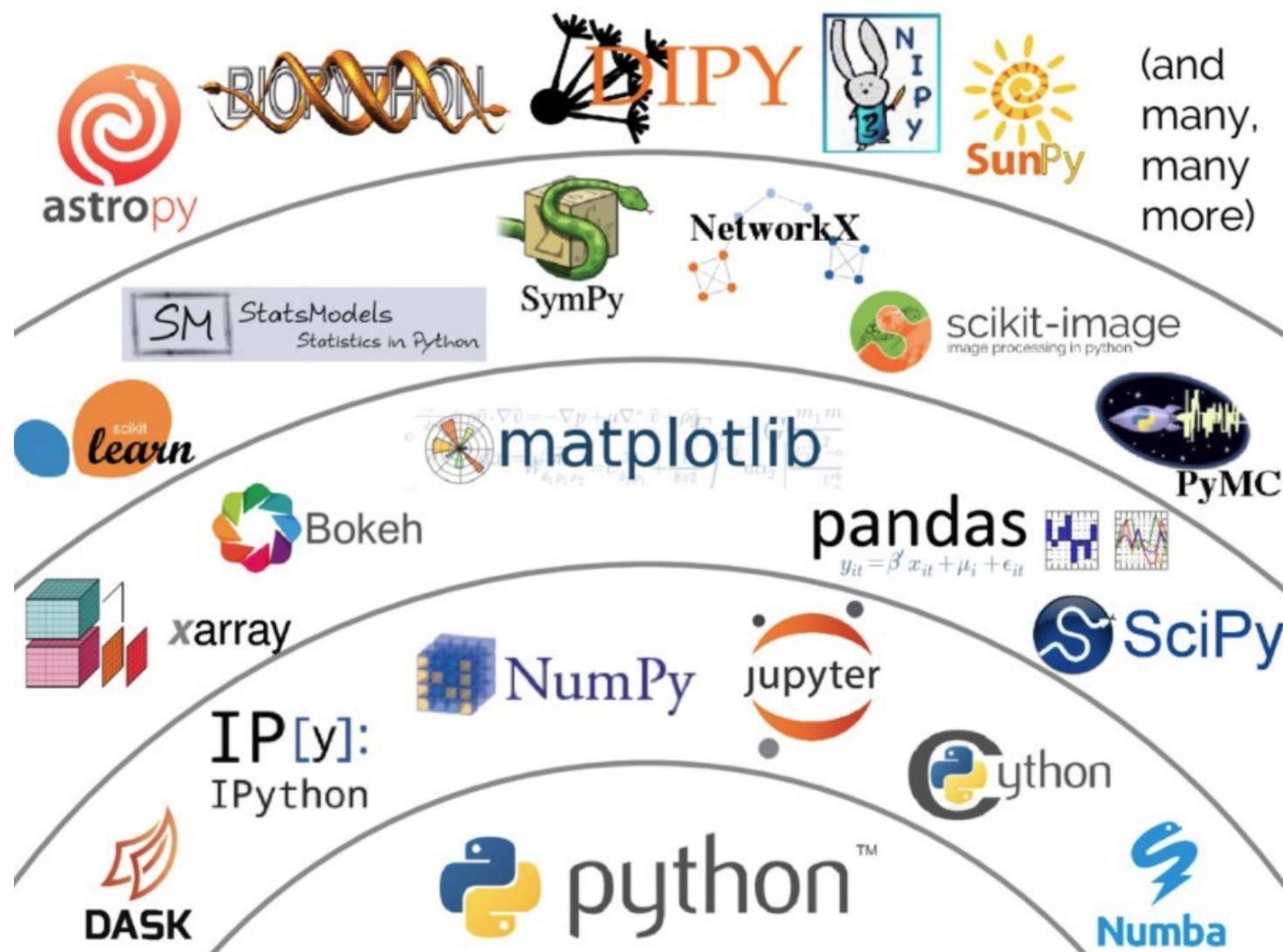
Módulos e importación de funciones y variables

Aproximaciones

Ten en cuenta que los números en python están acotados y por tanto el número pi, por ejemplo, será una aproximación del número real con un número finito de decimales.

Si por ejemplo calculas el seno de pi, verás que python no da exactamente cero.

Módulos e importación de funciones y variables



Módulos e importación de funciones y variables

Numpy

Install Documentation Learn Community About Us Contribute



The fundamental package for scientific computing with Python

GET STARTED

NumPy 1.24.0 released

POWERFUL N-DIMENSIONAL ARRAYS

Fast and versatile, the NumPy vectorization, indexing, and broadcasting concepts are the de-facto standards of array computing today.

NUMERICAL COMPUTING TOOLS

NumPy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more.

INTEROPERABLE

NumPy supports a wide range of hardware and computing platforms, and plays well with distributed, GPU, and sparse array libraries.

PERFORMANT

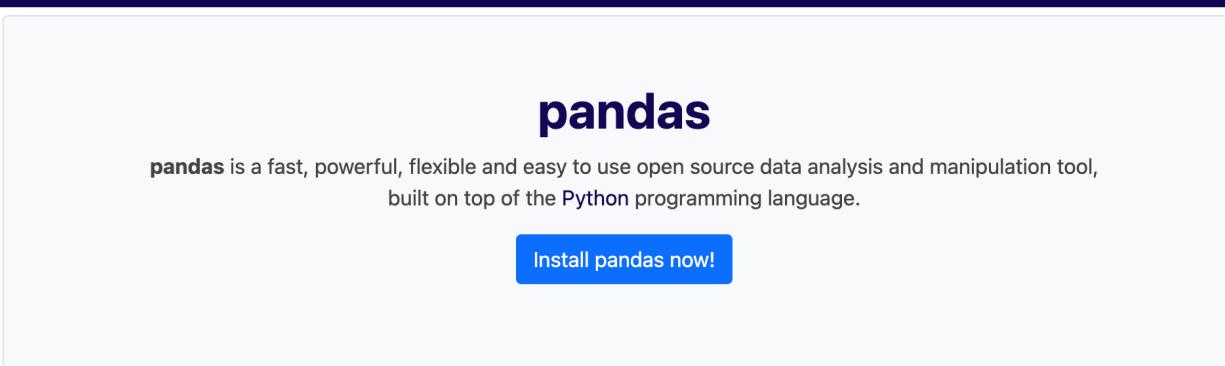
EASY TO USE

OPEN SOURCE

Numpy.org

Módulos e importación de funciones y variables

Pandas



The screenshot shows the official pandas documentation website. At the top, there's a dark header bar with the "pandas" logo on the left and navigation links for "About us", "Getting started", "Documentation", "Community", and "Contribute". Below the header, the main content area features the word "pandas" in a large, bold, dark blue font. A brief description follows: "pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language." A blue button labeled "Install pandas now!" is centered below the text. To the right, a sidebar highlights the "Latest version: 1.5.3" with a bulleted list of links: "What's new in 1.5.3", "Release date: Jan 19, 2023", "Documentation (web)", and "Download source code". Below this, social media links for Telegram and Twitter are shown, along with a link to "Get the book" which points to the O'Reilly book "Python for Data Analysis, 3rd Edition".

pandas

pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

Install pandas now!

About us ▾ Getting started Documentation Community ▾ Contribute

Latest version: 1.5.3

- What's new in 1.5.3
- Release date:
Jan 19, 2023
- Documentation (web)
- Download source code

Follow us

Get the book

pandas.pydata.org

Métodos



Métodos

¿Qué son?

Algunos tipos de datos permiten invocar unas funciones especiales que se denominan métodos.

Se utilizan con un . seguido del nombre del método y todo esto después de una variable.

Métodos

Métodos VS funciones

- `funcion(argumento1, argumento2, argumento3...)`
- `argumento1.metodo(argumento2, argumento3...)`

Métodos

Ejemplo

```
cadena = 'ejemplo de CADENA'  
cadena.lower()
```

Te devuelve 'ejemplo de cadena'

```
'un pequeño ejemplo'.replace('pequeño', 'gran')
```

Métodos

Método *format* para cadenas

- ‘los numeros {0} y {1} han sido interpolados’.format(1.23, 9.9999)
- ‘los numeros {0:.1f} y {1:.1f} han sido interpolados’.format(1.23, 9.9999)

Aunque hay muchas opciones, la f por ejemplo obliga a mostrar en flotante y el .1 a usar un decimal.

De interés



De interés

Datos de entrada por teclado

Para pedir datos de entrada que se introduzcan por teclado mientras se ejecuta un programa, se utiliza la función *input()*.

De interés

Mostrar datos e información

Para mostrar un mensaje de texto con información relevante durante la ejecución de un programa o incluso el valor de alguna variable, se usa la función *print()*.

De interés

Comentarios

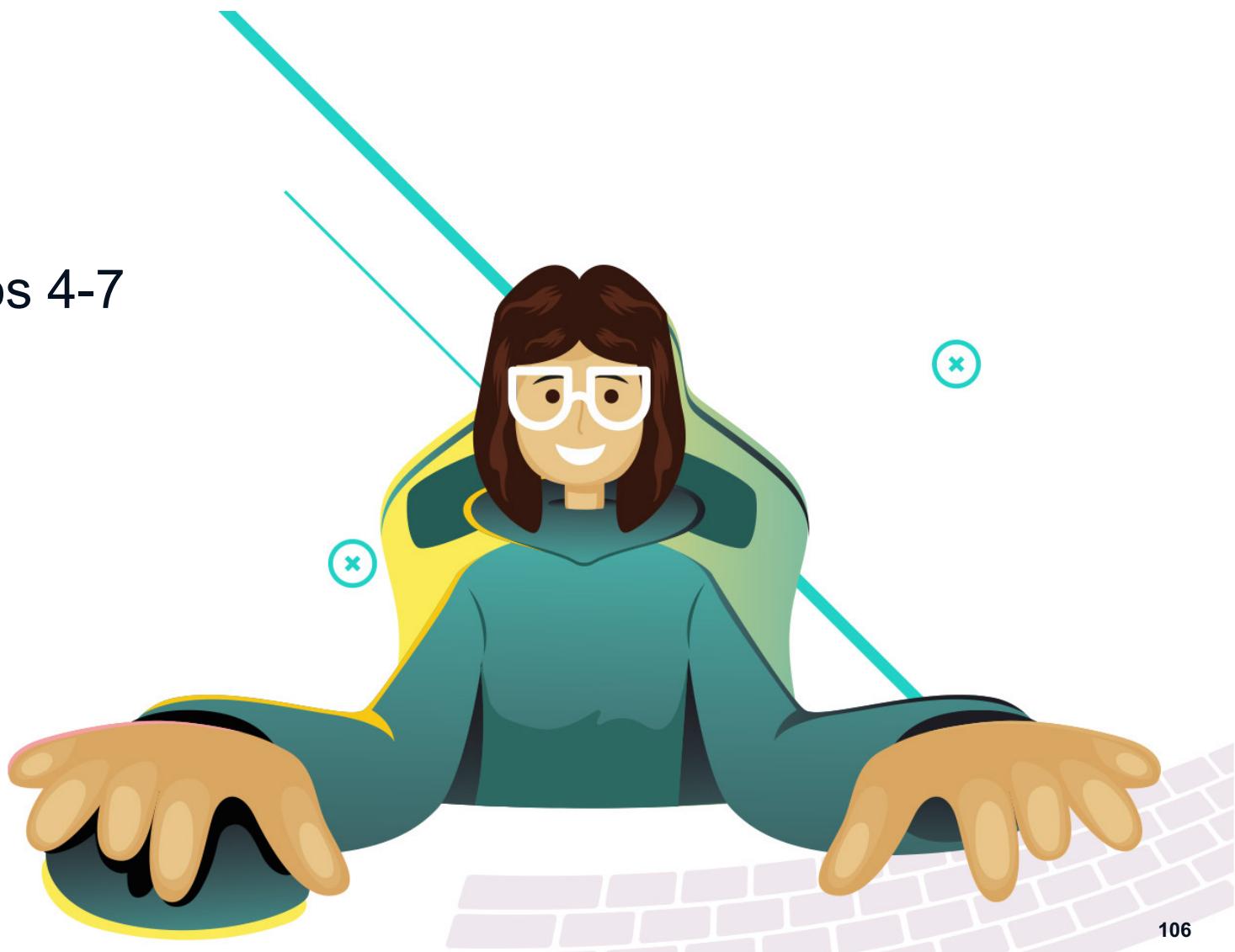
Puedes escribir comentarios que no quieras que sea en lenguaje de programación simplemente para explicar tu código y hacerlo más legible. Así podrás recordar lo que hacía cada segmento de código que escribiste en algún momento o facilitar que otra persona que lo lea lo entienda.

Para ello, tanto en las celdas de un notebook o en un programa de Python en un entorno, usa #.

```
# Calcular media  
resultado = (2 + 8 + 1) / 3  
# Mostrar resultado  
print(resultado)
```

Ejercicios

➤ Notebook 1. Apartados 4-7



Estructuras de control



Estructuras de control

Flujo de un programa

Un programa se forma con una serie de líneas de código que se ejecutan una tras otra siguiendo el orden en el que aparecen: el flujo del programa es secuencial.

Sim embargo, es posible alterar dicho flujo.

Los programas son capaces de tomar decisiones en función de datos o resultados intermedios, y en función de éstos,

1. Ejecutar ciertas sentencias y otras no.
2. Ejecutar ciertas sentencias más de una vez.

Estructuras de control

Sentencias condicionales: *if*

<<Al llegar a este punto, ejecuta esta(s) acción(es) sólo si esta condición es cierta.>>

```
1 if condición:  
2     acción  
3     acción  
4     ...  
5     acción
```

Estructuras de control

Sentencias condicionales: *if*

primer_grado.py

```
1 print('Programa para la resolución de la ecuación ax + b = 0.')
2
3 a = float(input('Valor de a: '))
4 b = float(input('Valor de b: '))
5
6 if a != 0:
7     x = -b / a
8     print('Solución:', x)
9
10 if a == 0:
11     if b != 0:
12         print('La ecuación no tiene solución.')
13     if b == 0:
14         print('La ecuación tiene infinitas soluciones.)
```

Estructuras de control

Sentencias condicionales: `else`

<<Si no, ejecuta estas otras acciones>>

```
1 if condición:  
2     acciones  
3 else:  
4     otras acciones
```

Estructuras de control

Sentencias condicionales: else

```
1 from math import sqrt # La función sqrt calcula la raíz cuadrada de un número.  
2  
3 print('Programa para la resolución de la ecuación a*x*x+b*x+c=0.')  
4  
5 a = float(input('Valor de a:'))  
6 b = float(input('Valor de b:'))  
7 c = float(input('Valor de c:'))  
8  
9 if a != 0:  
10     x1 = (-b + sqrt(b**2 - 4*a*c)) / (2 * a)  
11     x2 = (-b - sqrt(b**2 - 4*a*c)) / (2 * a)  
12     print('Soluciones: x1={0:.3f} y x2={1:.3f}'.format(x1, x2))  
13 else:  
14     if b != 0:  
15         x = -c / b  
16         print('Solución: x={0:.3f}'.format(x))  
17     else:  
18         if c != 0:  
19             print('La ecuación no tiene solución.')  
20         else:  
21             print('La ecuación tiene infinitas soluciones.')
```

Estructuras de control

Sentencias condicionales: else

```
1 mes = int(input('Dame un mes: '))
2
3 if 1 <= mes <= 3:
4     print('Inviero.')
5 else:
6     if mes == 4 or mes == 5 or mes == 6:
7         print('Primavera.')
8     else:
9         if not (mes < 7 or 9 < mes):
10            print('Verano.')
11        else:
12            if not (mes != 10 and mes != 11 and mes != 12):
13                print('Otoño.')
14            else:
15                print('Ningún año tiene {} meses.'.format(mes))
```

Estructuras de control

Sentencias condicionales: *elif*

```
1 if condición:  
2     ...  
3 elif otra condición:  
4     ...
```

Estructuras de control

Sentencias condicionales: *elif*

```
1 from math import pi
2
3 radio = float(input('Dame el radio de un círculo: '))
4
5 # Menú
6 print('Escoge una opción:')
7 print('a) Calcular el diámetro.')
8 print('b) Calcular el perímetro.')
9 print('c) Calcular el área.')
10 opción = input('Teclea a, b o c y pulsa el retorno de carro: ')
11
12 if opción == 'a': # Cálculo del diámetro.
13     diámetro = 2 * radio
14     print('El diámetro es {}.'.format(diámetro))
15 elif opción == 'b': # Cálculo del perímetro.
16     perímetro = 2 * pi * radio
17     print('El perímetro es {}.'.format(perímetro))
18 elif opción == 'c': # Cálculo del área.
19     área = pi * radio ** 2
20     print('El área es {}.'.format(área))
21 else:
22     print('Solo hay tres opciones: a, b o c.')
23     print('Tú has tecleado "{}".'.format(opción))
```

Estructuras de control

Bucles: while/for

Ejecutar un fragmento de código más de una vez.

Estructuras de control

Sentencias iterativas: *while*

<<Mientras se cumpla esta condición, repite estas acciones>>

```
1 while condición:  
2     acción  
3     acción  
4     ...  
5     acción
```

Estructuras de control

Sentencias iterativas: *while*

```
1 i = 0
2 while i < 3:
3     print(i)
4     i += 1
5 print('Hecho')
```

Estructuras de control

Sentencias iterativas: *while*

Bucles sin fin: CUIDADO! nunca acaba la ejecución del programa.

```
1 i = 0
2 while i < 10:
3     print(i)
```

Estructuras de control

Sentencias iterativas: *for*

<<Para todo elemento de una serie, hacer...>>

```
1 for variable in serie de valores:  
2     acción  
3     acción  
4     ...  
5     acción
```

Estructuras de control

Sentencias iterativas: *for*

```
1 número = int(input('Dame un número: '))
2
3 for potencia in [2, 3, 4, 5]:
4     print('{0} elevado a {1} es {2}'.format(número, potencia, número ** potencia))
```

Estructuras de control

Generar secuencias de valores: range

```
>>> list(range(2, 10))  
[2, 3, 4, 5, 6, 7, 8, 9]  
>>> list(range(0, 3))  
[0, 1, 2]  
>>> list(range(-3, 3))  
[-3, -2, -1, 0, 1, 2]  
>>> list(range(-10, -1))  
[-10, -9, -8, -7, -6, -5, -4, -3, -2]
```

```
>>> list(range(5))  
[0, 1, 2, 3, 4]
```

```
>>> list(range(10, 5, -1))  
[10, 9, 8, 7, 6]  
>>> list(range(3, -1, -1))  
[3, 2, 1, 0]  
>>> list(range(10, 1, -3))  
[10, 7, 4]
```

Estructuras de control

Roturas de bucles: *break*

Abortar la ejecución de un bucle desde cualquier punto del mismo.

```
1 creo_que_se_cumple_para_alguno = False
2 for elemento in conjunto:
3     if condición:
4         creo_que_se_cumple_para_alguno = True
5         break
6
7 if creo_que_se_cumple_para_alguno:
8     print('Se cumplió para alguno')
```

Estructuras de control

Bucles anidados

```
1 for i in range(0, 5):
2     for j in range(0, 3):
3         print(i, j)
```

Estructuras de control

Bucles anidados

```
1 for i in range(0, 5):
2     for j in range(0, i):
3         print(i, j)
```

Más

Optimización

Ejemplo: Si alguna operación se repite varias veces, puedes hacerla sólo una vez y guardarla en una variable.

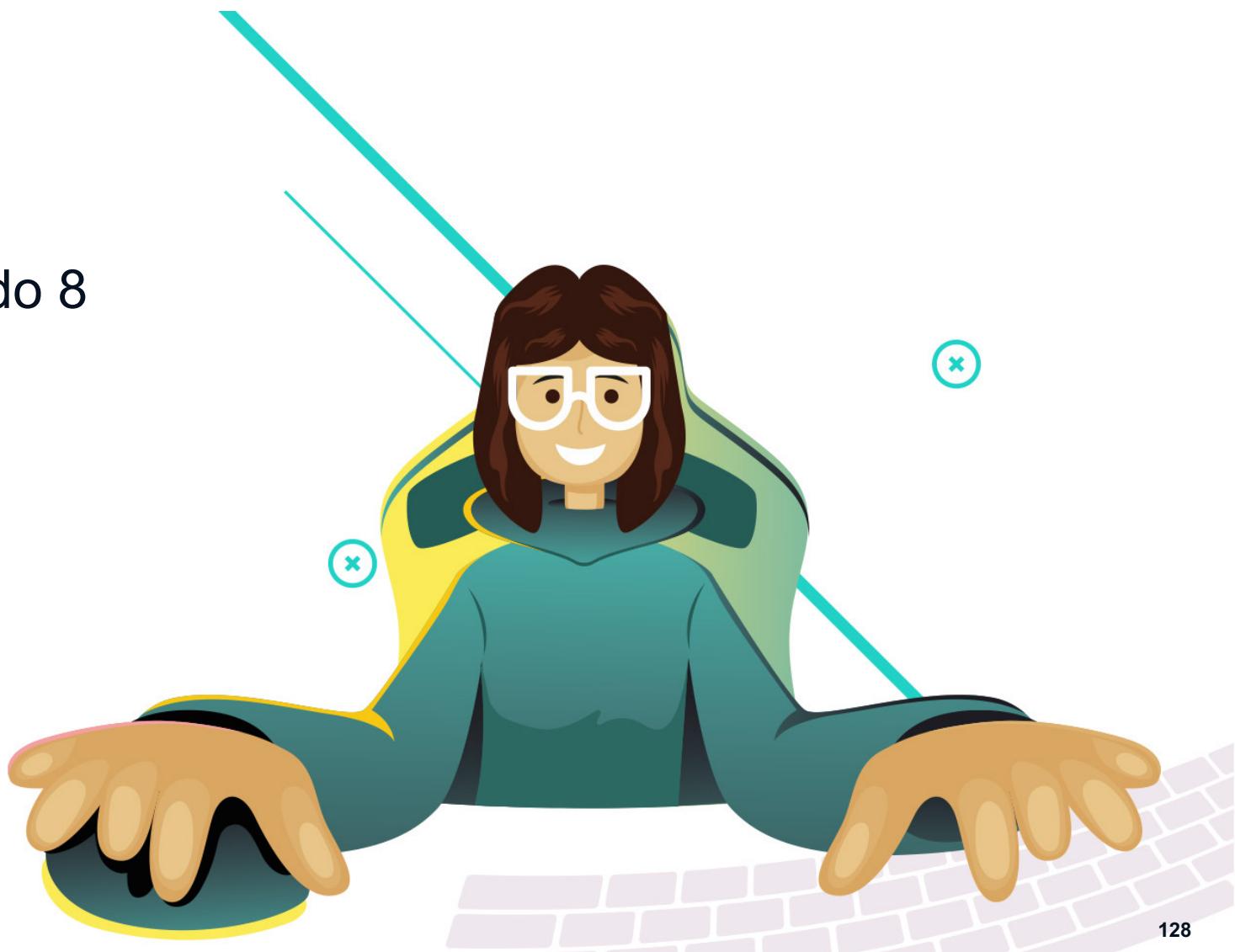
Más

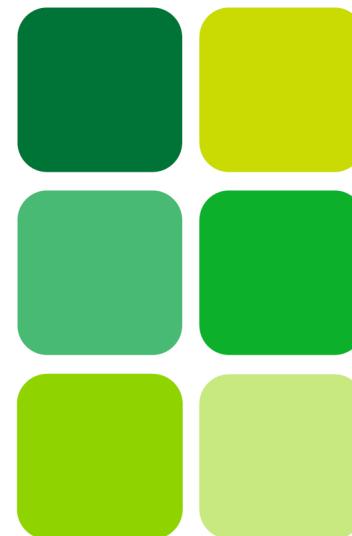
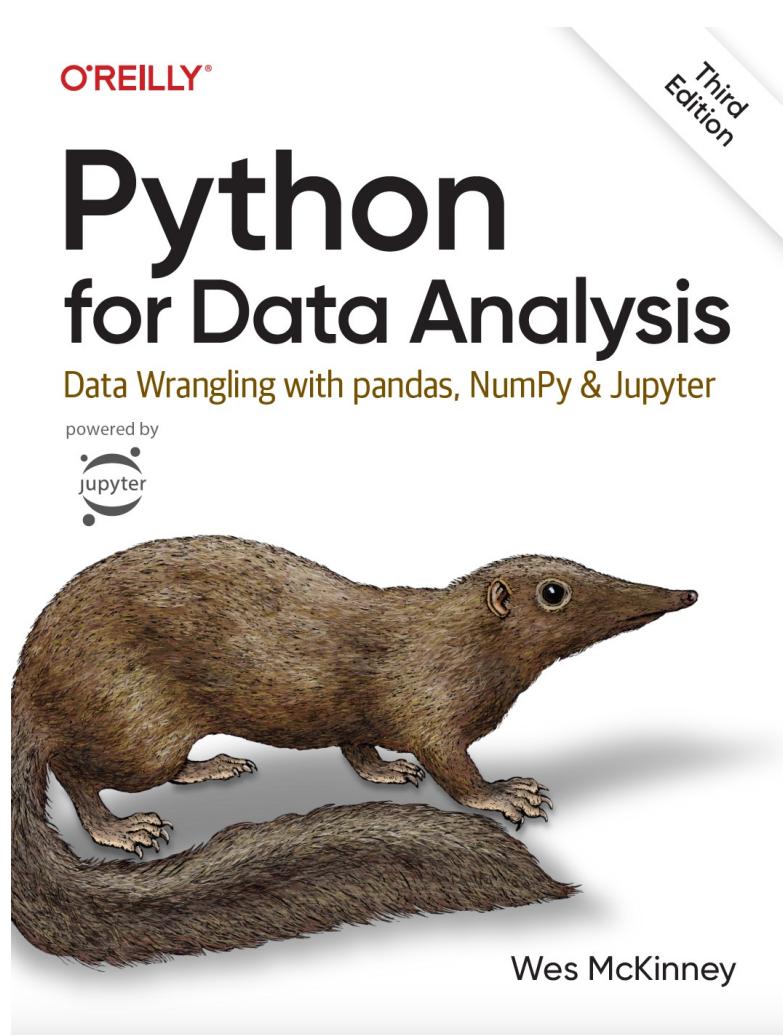
Solución

Cada problema de programación puede tener más de una solución. Es decir, podemos escribir un programa que realice lo mismo de muchas maneras.

Ejercicios

➤ Notebook 1. Apartado 8





Introducción a la programación con Python 3

Andrés Marzá Varó
Isabel Gracia Luengo
Pedro García Sevilla



www.sapientia.uji.es | 93

Contacto

Correo: a.cobo.aguilera@gmail.com

LinkedIn: [Aurora Cobo Aguilera](#)

GitHub: [AuroraCoboAguilera](#)

Google Scholar: [Aurora Cobo Aguilera](#)





GOBIERNO
DE ESPAÑA

VICEPRESIDENCIA
PRIMERA DEL GOBIERNO
MINISTERIO
DE ASUNTOS ECONÓMICOS
Y TRANSFORMACIÓN DIGITAL

SECRETARÍA DE ESTADO
DE DIGITALIZACIÓN
E INTELIGENCIA ARTIFICIAL

red.es

Centro de
Referencia Nacional
en Comercio Electrónico
y Marketing
CRN
Digital



UNIÓN EUROPEA

"El FSE invierte en tu futuro"

Fondo Social Europeo


Barrabés

 The Valley