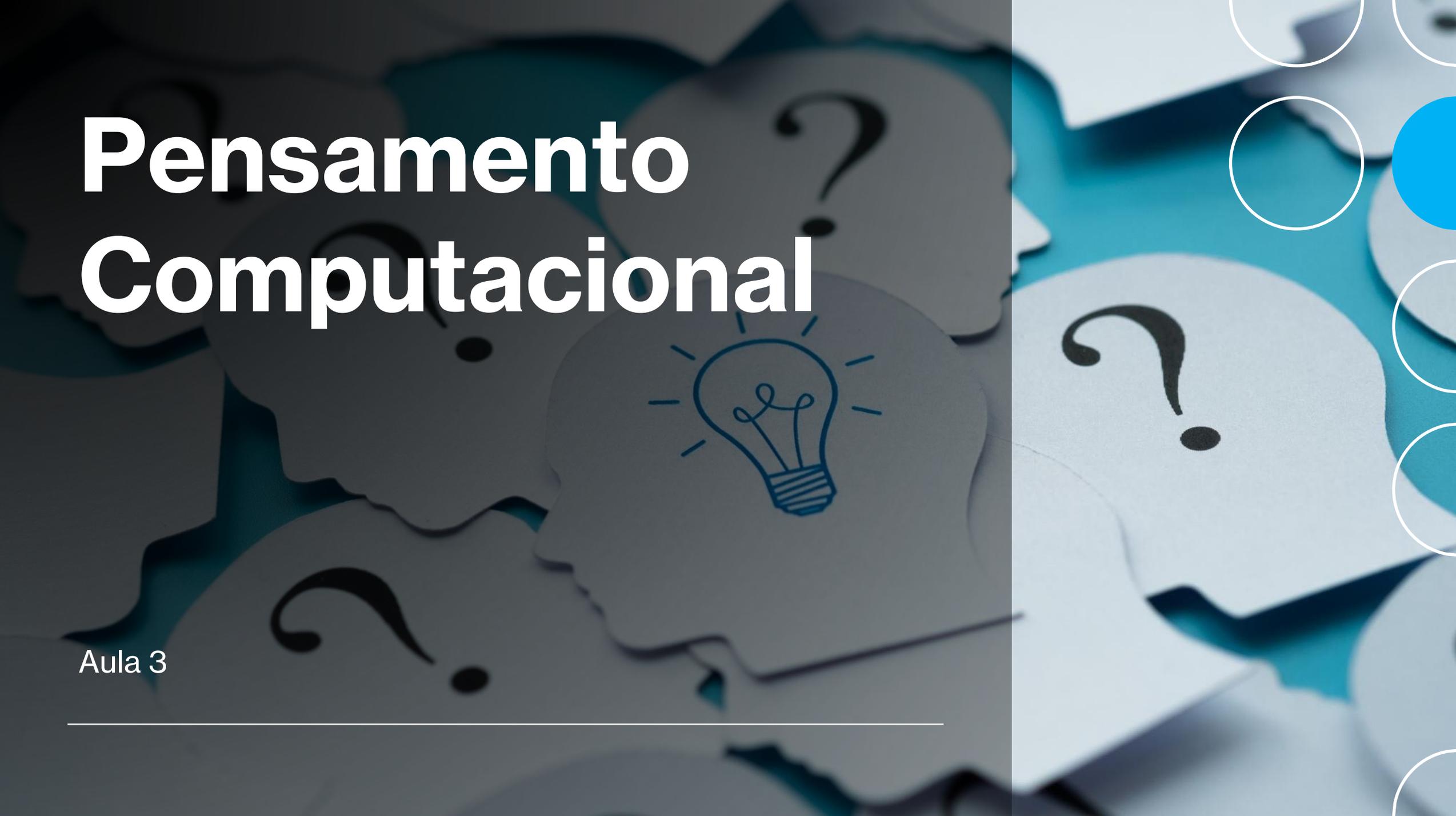


# Pensamento Computacional



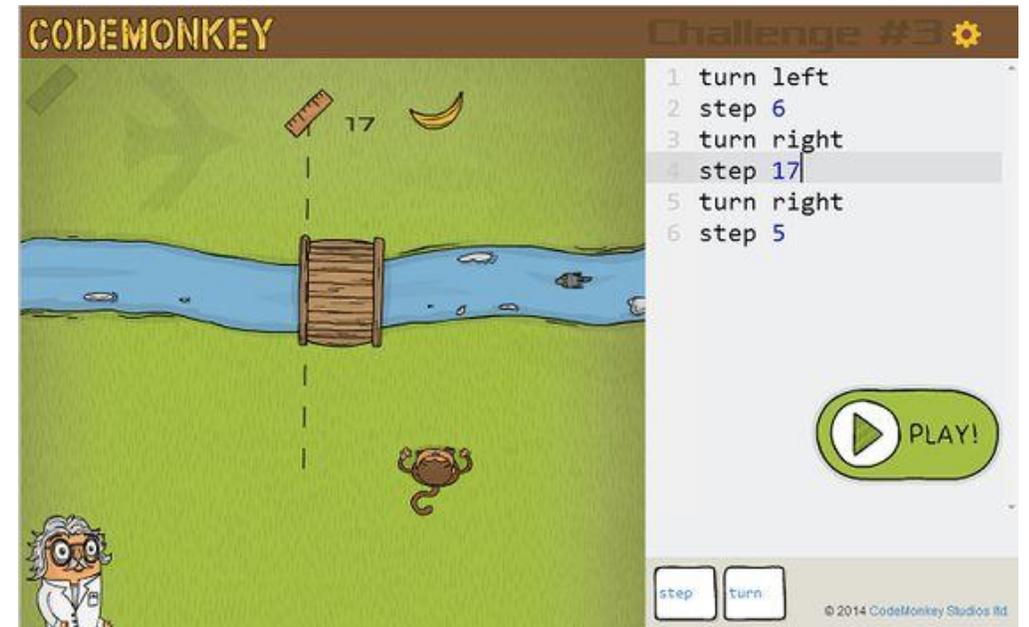
Aula 3

---

# Linguagens de Programação

## Programar:

- Processo de criação de um conjunto de instruções (*statements*), que indicam ao computador como efetuar uma dada tarefa.
- Pode ser feito usando um grande conjunto de linguagens de programação: Java, Python, C++, C#, etc.



# Porquê o Python?

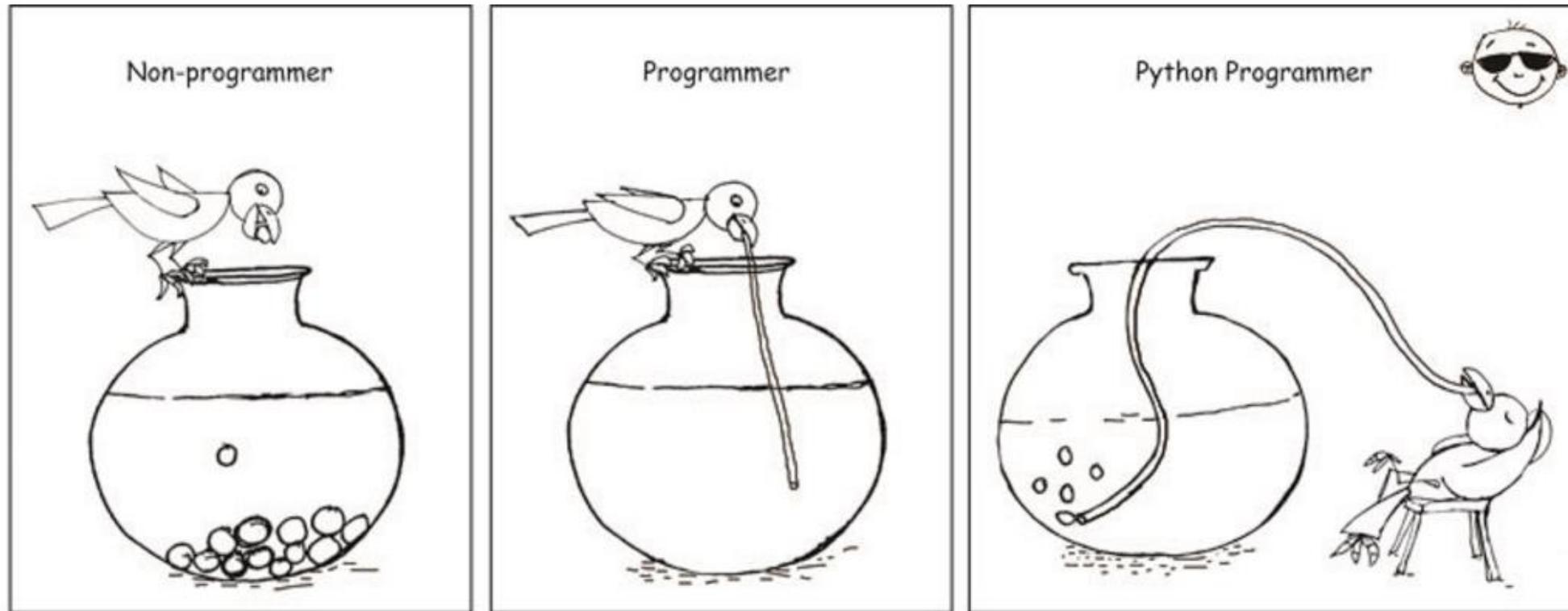
Uma das linguagens de programação mais fáceis de aprender!

- Simplicidade;
- Sintaxe elegante;
- Leitura fácil;
- Poderosa e flexível;
- Elevada produtividade;
- Comunidade alargada;
- Ecossistema alargado: muitas áreas de aplicação e muitas bibliotecas (>150k)

```
1. // C++ Programming Language
2. // Code For Printing To The Console
3. #include <iostream>
4. using namespace std;
5.
6. int main(){
7.     cout << "Hello World!";
8.     return 0;
9. }
```

```
1. // Python Programming Language
2. // Code For Printing To The Console
3. print("Hello World!")
```

# Porquê o Python?



# Porquê o Python?

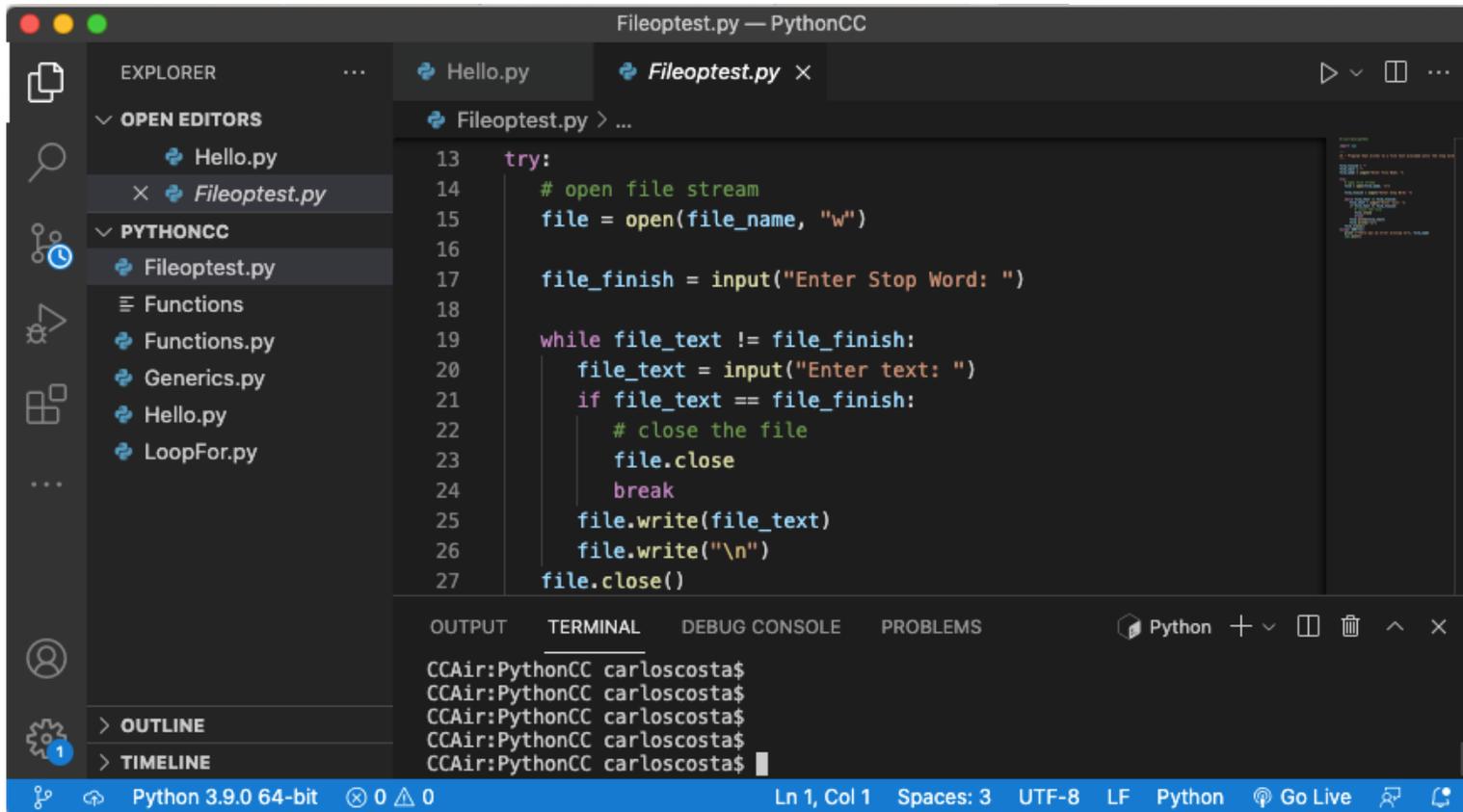
- Linguagem de programação de âmbito geral:
  - Desde aplicações de desktop até aplicações web e frameworks.
- *Free, open-source and high-level*
- Corre em qualquer plataforma.

# Instalar Python

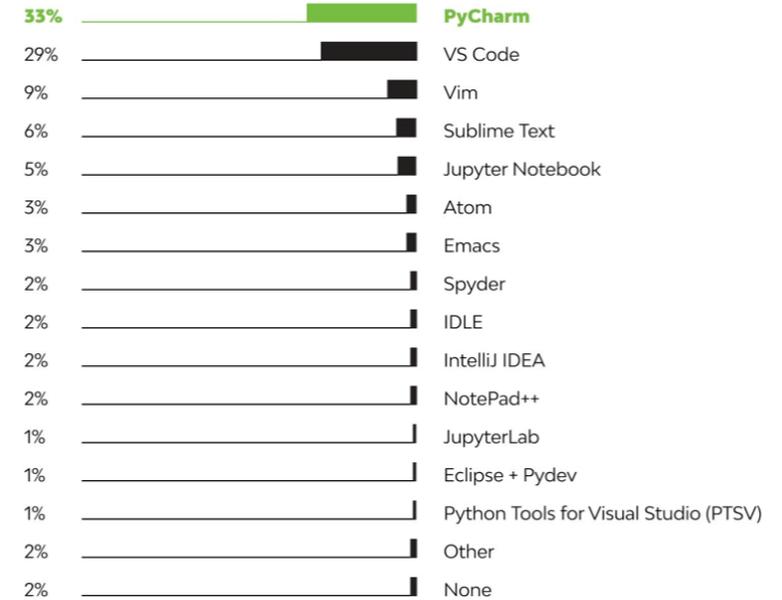
<https://www.python.org/downloads/>

The screenshot shows the Python.org website interface. At the top, there is a dark navigation bar with links for Python, PSF, Docs, PyPI, Jobs, and Community. Below this is the Python logo and a search bar with a 'Donate' button and a 'Socialize' link. A secondary navigation bar contains links for About, Downloads, Documentation, Community, Success Stories, News, and Events. The main content area features a prominent yellow button labeled 'Download Python 3.10.0' under the heading 'Download the latest version for macOS'. Below the button, there is text suggesting alternative OS options: 'Looking for Python with a different OS? Python for [Windows](#), [Linux/UNIX](#), [macOS](#), [Other](#)'. The background of the main content area is dark blue with a graphic of two yellow and white striped parachutes.

# Instalar IDE



## The Most Popular Python Editors and IDEs



source: jetbrains.com

# Python - linguagem interpretada

- O interpretador analisa/executa o programa linha a linha
- Existem 2 modos:

- **Interactivo:**

Executado pelo commando `python` ou `python3`

```
$ python3  
>>> 1 + 1  
2  
>>>
```

# Python - linguagem interpretada

- *Script*

- Crie um documento de texto com a extensão `.py` (ex. `meuprog.py`)
- Escreva as instruções que compõem o programa
- Guarde o ficheiro e execute-o mais tarde, sempre que precisar

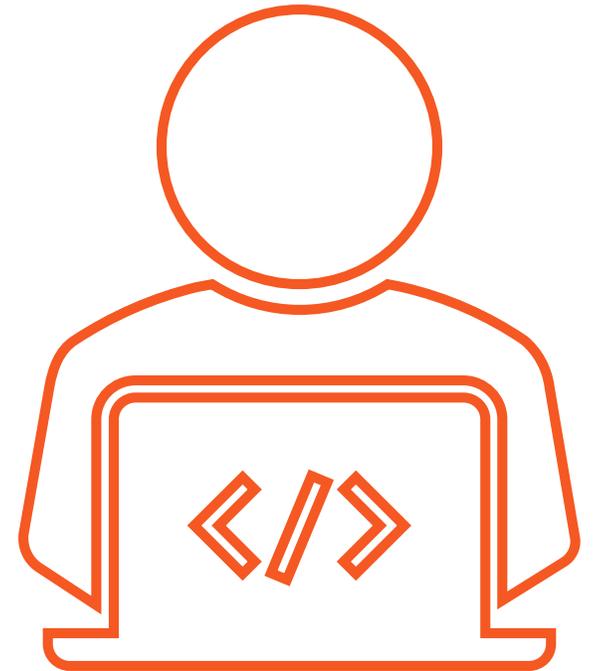
```
test.py
1 print ("The result is:")
2 print (24+18)
```

```
$ python3 test.py
The result is
42
```

Note: Em modo *script*, os resultados das operações não são escritos automaticamente. Precisa de chamar `print(...)`

# Desenhar um programa

- Os programas devem ser pensados/desenhados/estruturados antes de serem escritos.
- O conjunto de passos lógicos que irão permitir resolver a tarefa em questão – ou seja, o **algoritmo** – deve ser escrito antes de avançar.
- Ciclo de desenvolvimento de um programa:
  1. Estruturar o algoritmo;
  2. Escrever o código;
  3. Corrigir erros de sintaxe;
  4. Testar o programa;
  5. Corrigir erros lógicos.



# Entrada (*input*), processamento e Saída (*output*)

**Algoritmo** CalcularAreaRectangulo

**Var**

alt, lar, area: real

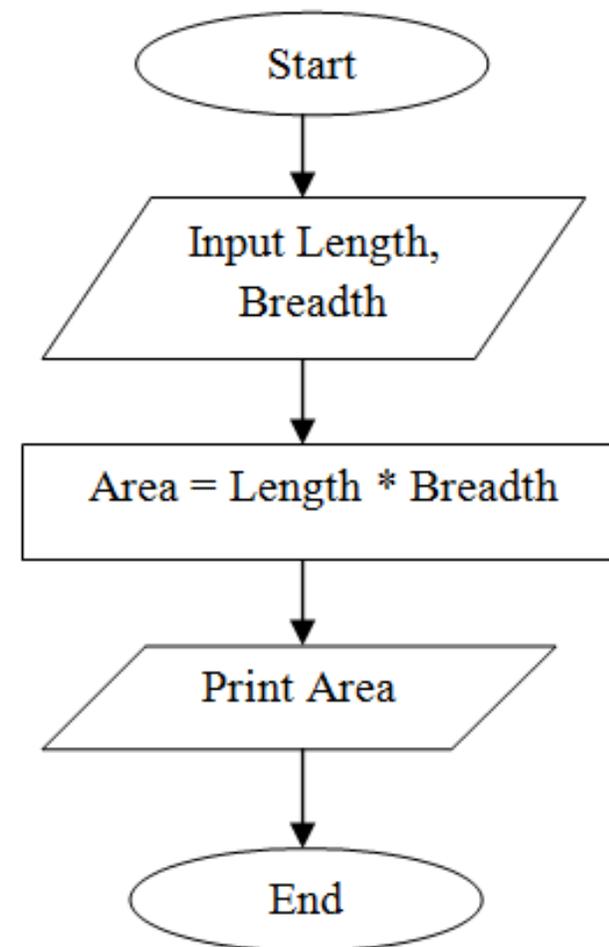
**Início**

Leia (alt, lar);

area  $\leftarrow$  (alt+lar) / 2;

Escreva (média);

**Fim**



# Python - Sintaxe

- Indentação

- Espaços ou tabulações;
- Todas as declarações de um bloco devem ser indentadas da mesma forma, incluindo quantidades.

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

NOK

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

OK

- Fim da declaração

- No final de cada declaração, bastará fazer 'Enter';
- Para aumentar a legibilidade, e em caso de uma linha muito longa, pode-se separar em várias linhas usando o *backslash* (\).

```
# OK  
msg = "Olá tudo bem?"  
  
# NOK  
msg  
=  
" Olá tudo bem"  
  
# OK  
msg \  
=  
" Olá tudo bem"
```

# Python - Sintaxe

- Para declarar várias instruções na mesma linha, usar ponto e vírgula (;)
- Comentários: `# Esta linha faz x`
- Linhas vazias serão ignoradas

```
# OK  
y = 3; x = 5; print(x+y)
```

# Python – *data types* simples

- Números
  - Integer (int): -5, 10, 77
  - Floating Point (float) : 3.1457, 0.34
  - Complex: 1+2j
- Booleans
  - Bool: True or False

```
>>> type(17)  
<class 'int'>
```

```
>>> type(3+5j)  
<class 'complex'>
```

# Strings

- Sequência de caracteres;
- Caracteres são *strings* de tamanho 1;
- São qualquer coisa escrita através do teclado.
- String no código
  - Devem estar entre aspas, simples (') ou duplas (");
  - Adicionar aspas em strings segue regras especiais:
    - \' ou \'”
    - “ ... ‘ ...” ou ‘ ... “ ... ‘
  - Aspas triplas permitem que as strings se expandam em linhas múltiplas.



# Exemplos de Strings

```
>>> print("Coding is fun") #OK
```

```
>>> print('Coding is fun') #OK
```

```
>>> print("Let's go coding") #OK
```

```
>>> print('Let's go coding') #Error
```

```
>>> print("Let\'s go coding") #OK
```

```
>>> print("She "likes" coding") #Error
```

```
>>> print("She \"likes\" coding") #OK
```

```
>>> print('She "likes" coding') #OK
```

```
>>> print("""Learn Python  
Programming""") #OK
```

```
>>> print('''Learn Python  
Programming''') #OK
```

# Valores, variáveis e atribuição

- **Valor:**
  - Unidade mais básica de um programa, por exemplo, uma letra ou número.
- **Variável:**
  - Nome que representa um valor guardado na memória do computador;
  - Permite guardar, aceder e manipular dados em memória.
- **Declaração de atribuição:**
  - Usada para criar a variável e atribuir-lhe um valor;
  - Formato genérico é: `variable = expression`
    - Expressão é o valor, a equação matemática ou a função que resulta num valor
    - Exemplos:  
`day = 5`  
`age = current_year - birth_year`

# Atribuição de nomes a variáveis

- **Regras**

1. O primeiro carater deve ser uma letra ou um *underscore* ( \_ );
2. Após o primeiro carater podem ser usadas letras, digitos ou *underscores*;
3. Não pode ser uma palavra chave (*keyword*) de Python;
4. Não pode conter espaços:
  - Usar, em alternativa, o *underscore*;
5. Sensível ao uso de maiúsculas ou minúsculas (*case sensitive*).

# Atribuição de nomes a variáveis

- **Recomendações**

1. Devem ser curtos mas descritivos;
2. O nome da variável deve refletir o seu objetivo
  - ex: `finaltax`, `approvalrate`
  - Ou como palavras separadas: `final_tax`, `approval_rate`

# Nomes (*keywords*) reservadas

- Keywords são palavras reservadas
  - Não podem ser usadas como nomes de variáveis ou como qualquer outra forma de identificação

<code>False</code>	<code>def</code>	<code>if</code>	<code>raise</code>
<code>None</code>	<code>del</code>	<code>import</code>	<code>return</code>
<code>True</code>	<code>elif</code>	<code>in</code>	<code>try</code>
<code>and</code>	<code>else</code>	<code>is</code>	<code>while</code>
<code>as</code>	<code>except</code>	<code>lambda</code>	<code>with</code>
<code>assert</code>	<code>finally</code>	<code>nonlocal</code>	<code>yield</code>
<code>break</code>	<code>for</code>	<code>not</code>	
<code>class</code>	<code>from</code>	<code>or</code>	
<code>continue</code>	<code>global</code>	<code>pass</code>	

# Exemplos de Atribuição

```
>>> pi = 3.14           #float
>>> r = 2               #int
>>> area = r**2 * pi   #float
>>> print (area)
12.56
```

```
>>> r = r + 1
>>> print (r)
3
```

```
>>> print (area)
12.56
```

```
>>> b = True           #bool
>>> print (b)
True
```

```
>>> i = 2+5j           #complex
>>> print(i.real, i.imag)
2.0 5.0
```

# Exemplos de Atribuição

```
#str
>>> message = "Hello Python world!"
>>> print(message)
Hello Python world!

>>> message = "Hello PC Class!"
>>> print(message)
Hello PC Class!

>>>
```

```
# simultaneous assignments
>>> name, age, height = "Maria", 21,
1.63

>>> #print the variables
>>> print (pi, r, area, message)
3.14 3 12.56 Hello PC Class!

>>>
```

# Operadores, expressões e declarações

- **Operadores** – símbolos especiais que representam cálculos: `+`, `-`, `*`, `/`, `**`, `%`, `<=`, `or`
- **Operandos** – valores combinados por operadores
  - Devem ser de tipos compatíveis para um dado operador;
  - O resultado depende do tipo dos operandos.
- **Expressão** – combinação de valores, variáveis e operadores.
- **Declaração** - uma unidade de código que o interpretador de Python consegue executar.

# Operadores Aritméticos

Operator	Example	Meaning	Result
+ (unary)	+a	Unary Positive	a
- (unary)	-a	Unary Negation	a with opposite sign
**	a ** b	Exponentiation	a raised to the power of b
*	a * b	Multiplication	Product of a and b
/	a / b	Division	Quotient when a is divided by b. The result always has type float.
%	a % b	Modulo	Remainder when a is divided by b
//	a // b	Floor Division (or Integer Division)	Quotient when a is divided by b, rounded to the next smallest whole number
+ (binary)	a + b	Addition	Sum of a and b
- (binary)	a - b	Subtraction	b subtracted from a

Precedência descendente

Mesma cor = mesma precedência

# Operadores de Atribuição Aumentados

Operator	Example	Equivalent To
Assignment =	$i = 1$	$i = 1$
Addition Assignment +=	$j += 1$	$j = j + 1$
Subtraction Assignment -=	$k -= 2$	$k = k - 2$
Multiplication Assignment *=	$m *= 2$	$m = m * 2$
Float Division Assignment /=	$n /= 2$	$n = n / 2$
Integer Division Assignment //=	$p //= 2$	$p = p // 2$
Modulus or Remainder Assignment %=	$q %= 2$	$q = q \% 2$
Exponent Assignment **=	$r **= 2$	$r = r ** 2$

Combinação do operador de atribuição com outros operadores aritméticos.

# Operadores e Precedências

- Expressão com múltiplos operadores: segue as regras de precedência do *slide* anterior
  - Use parêntesis para tornar os cálculos mais óbvios!
- Em *strings*:
  - O operador + concatena;
  - O operador \* indica uma repetição
  - 'Ah' \* 3 é equivalente a 'AhAhAh'
- Boa Prática:
  - Adicionar comentários, em linguagem natural, para explicar o que o programa está a fazer:
    - # This line calculates the area ...

# Conversão e Combinação de diferentes Tipos de Dados

- Pode juntar duas *strings* (concatenação)

```
>>> first_name = "Liliana"  
>>> last_name = "Ferreira"  
>>> full_name = first_name + " " + last_name  
>>> print (full_name)  
Liliana Ferreira
```

# Conversão e Combinação de diferentes Tipos de Dados

- Converter valores para tipos de dados diferentes
- Funções de conversão: `str`, `int`, `float`, ...

```
>>> age = 20
>>> print (full_name + " age: " + str(age))
```

```
>>> 100 + int("33")
133
```

```
>>> int(3.14) #down casting
3
```

# Leitura de Dados do Teclado

- A maioria dos programas precisa de ler dados do utilizador
- Python: função `input`
  - **sintaxe:** `variable = input(prompt)`
    - *prompt* é normalmente uma expressão (*string*) que indica ao utilizador que deve introduzir um valor
    - Devolve os dados como uma *string*

- Exemplos:

```
>>> name = input("What's your name? ")
```

```
What's your name? liliana
```

```
>>> name
```

```
'liliana'
```

```
# To get other types of values, you  
must convert
```

```
>> age = int(input("Enter your age: "))
```

```
Enter your age: 22
```

```
>>> age
```

```
22
```

```
>>> type(age)
```

```
<class 'int'>
```

# Enviar resultados (*Output*) para a consola

- Função **print** – Apresenta o texto resultado no ecrã

```
>>> print("Hello World")
```

- Para escrever em várias linhas – adicionar o carater ‘\n’

```
>>> print("Hello World\nThis is a message")
```

- Apresentar vários valores (separados por espaços em branco):

```
>>> print("speed =", v)
```

# Enviar resultados (*Output*) para a consola

- Enviar o resultado para um meio (*stream*) diferente (exemplo: ficheiro):

```
>>> fh = open("data.txt", "w")  
>>> print("Some text", file=fh)  
>>> fh.close()
```

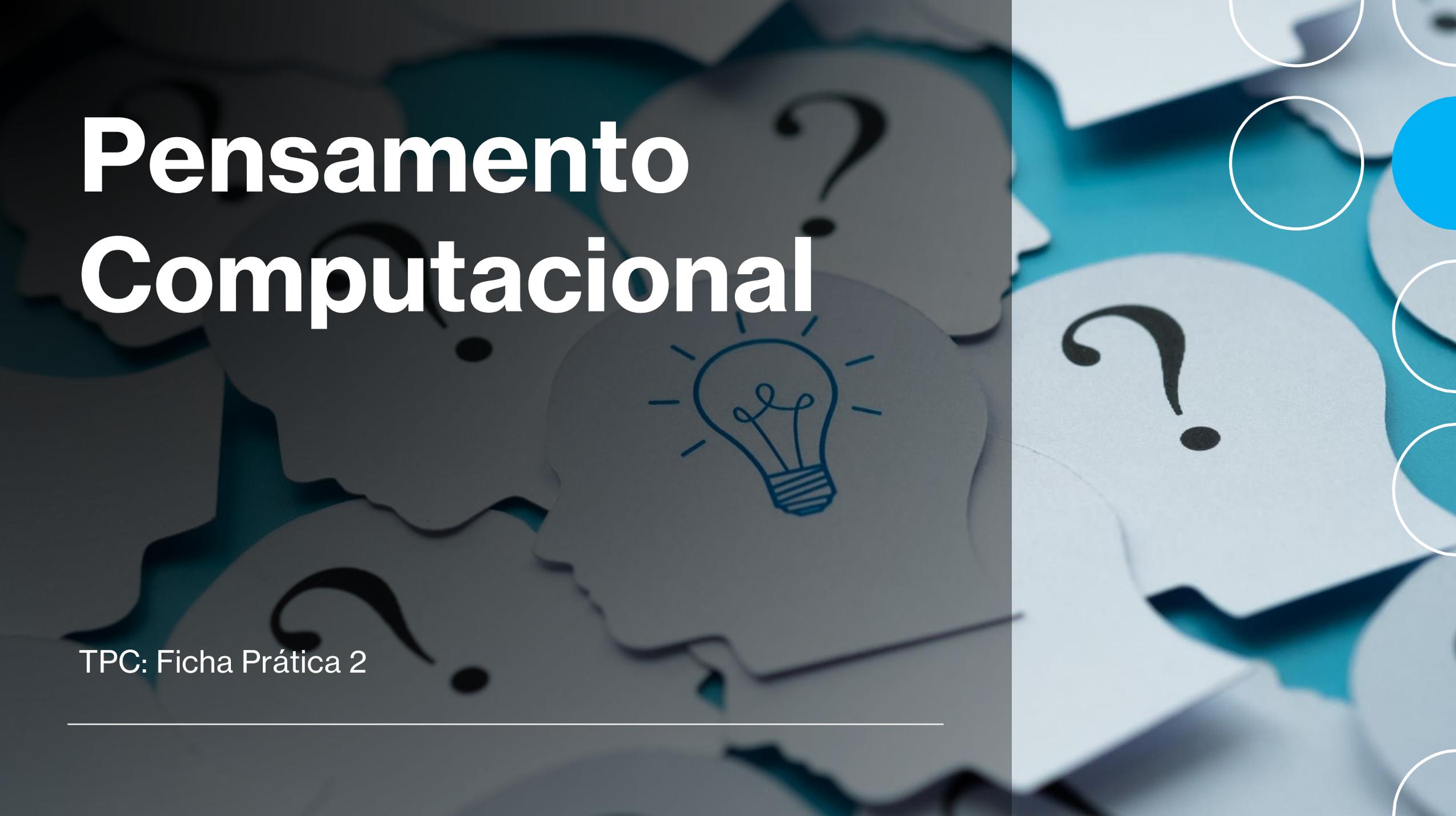
# Exemplo de um programa

```
# Get the user's first name.  
first_name = input('Enter your first name: ')  
  
# Get the user's last name.  
last_name = input('Enter your last name: ')  
  
# Print a greeting to the user.  
print('Hello', first_name, last_name)
```

## Output

```
Enter your first name: Liliana  
Enter your last name: Ferreira  
Hello Liliana Ferreira
```

# Pensamento Computacional



TPC: Ficha Prática 2

---