



ICC901 – Introdução à Programação de Computadores

IECo81 – Introdução à Ciência dos Computadores

IECo37 – Introdução à Programação de Computadores

Aula 01 – Variáveis e Estrutura Sequencial

Atualização: 6/mar/20



Você tem a liberdade de:



Compartilhar: copiar, distribuir e transmitir esta obra.

Remixar: criar obras derivadas.

Sob as seguintes condições:



Atribuição: você deve creditar a obra da forma especificada pelo autor ou licenciante (mas não de maneira que sugira que estes concedem qualquer aval a você ou ao seu uso da obra).



Uso não comercial: você não pode usar esta obra para fins comerciais.



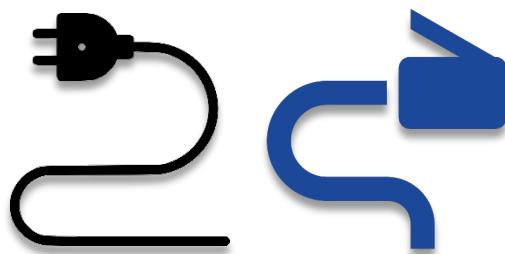
Compartilhamento pela mesma licença: se você alterar, transformar ou criar em cima desta obra, poderá distribuir a obra resultante apenas sob a mesma licença, ou sob uma licença similar à presente.



Conserve o laboratório



Os equipamentos são frágeis:
use-os com cuidado



Não mexa nos cabos



Não consuma alimentos ou bebidas.
Mantenha sua garrafa de água
tampada.

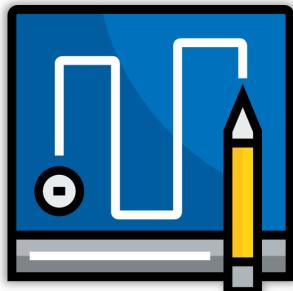
Antes de começar...



Está atento ao **calendário**?

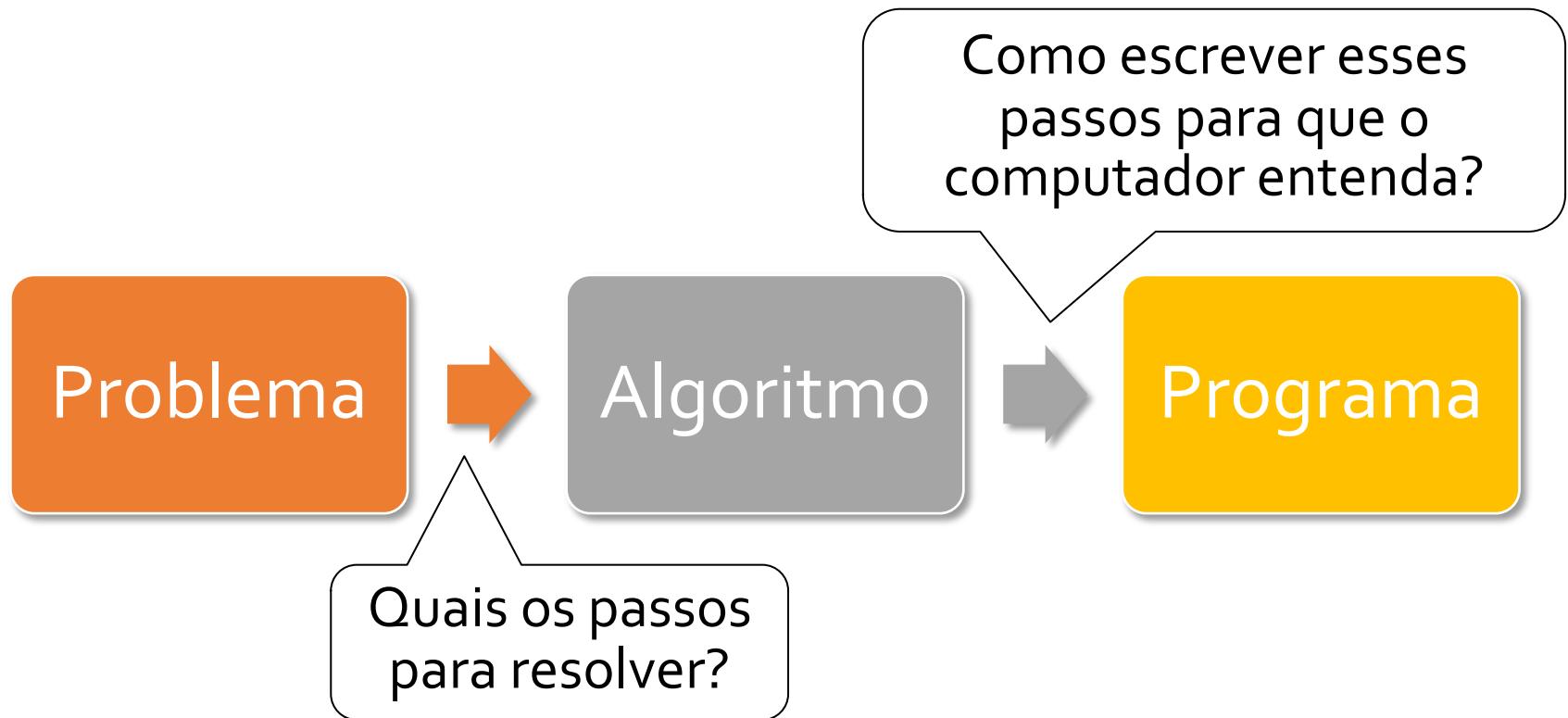


Este módulo tem **Peso 1** na avaliação



Como está sua **tática** de estudo?

Programar é resolver problemas



Conceitos Básicos de Programação

Algoritmo

- Descrição de um conjunto **ordenado** de comandos para a solução de um problema em um **tempo finito**.

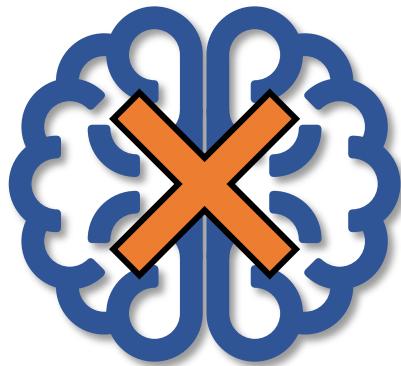
Linguagem de programação

- Estabelece **vocabulário**, **sintaxe** (formato) e **semântica** (significado) para que o algoritmo possa ser entendido por uma máquina.

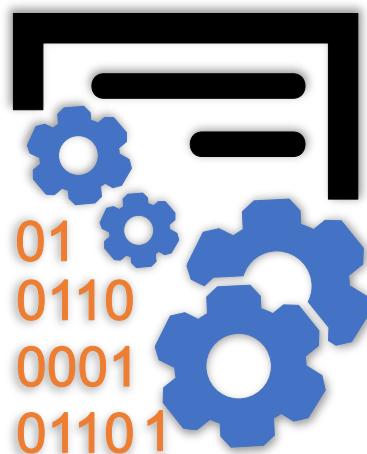
Programa

- É a **codificação** de um algoritmo em uma linguagem de programação.

Pense como o computador funciona



Computadores
não pensam

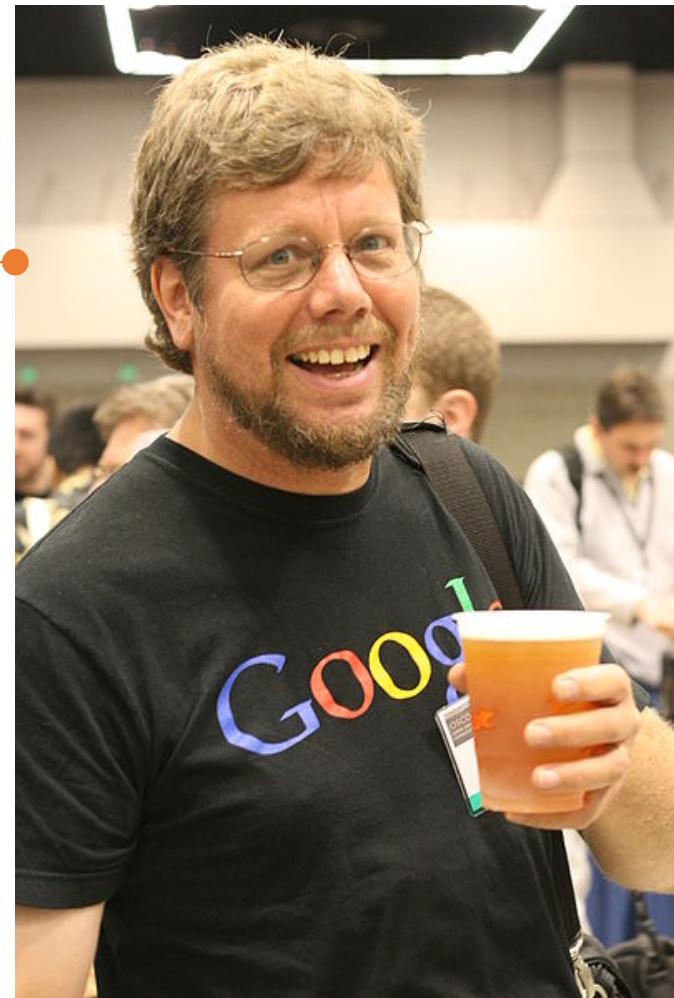


Eles **seguem**
instruções

O que é Python?

- Python é uma linguagem de programação criada por Guido van Rossum, na Holanda em 1990.
- Site oficial:

www.python.org



Versões do Python

:: Atenção

- ⚠ Neste curso, usaremos a versão **3.x** do Python.
- ⚠ Muitos livros e apostilas foram escritos para a versão **2.x** e anteriores.
- ⚠ A versão **3.x** possui comandos mais simples que **não são reconhecidos** pelas versões antigas.



Qual a relação entre Python e CodeBench?



Python

- É uma linguagem de programação, independente do ambiente escolar.



CodeBench

- É uma ferramenta de apoio ao ensino de programação, que pode ser em Python ou não.

Recomendações

1

- **Pense** antes de programar.

2

- Programas não são escritos para o computador entender. Servem para **explicar** a outros **humanos** o que o computador deve fazer.

3

- **Pratique!**

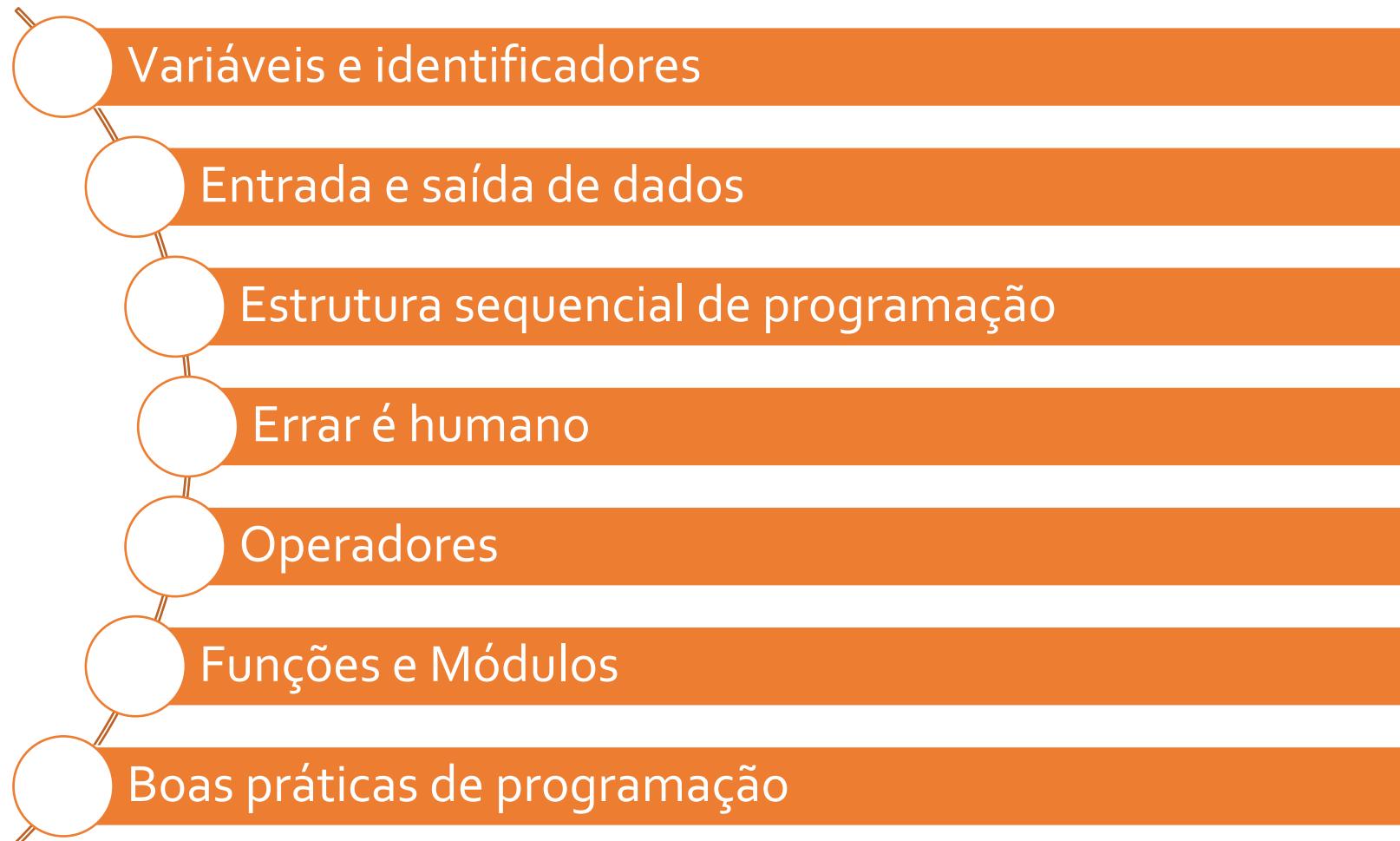
4

- **Teste** pequenos pedaços do programa.

5

- Use as mensagens de erro para corrigir seu código.

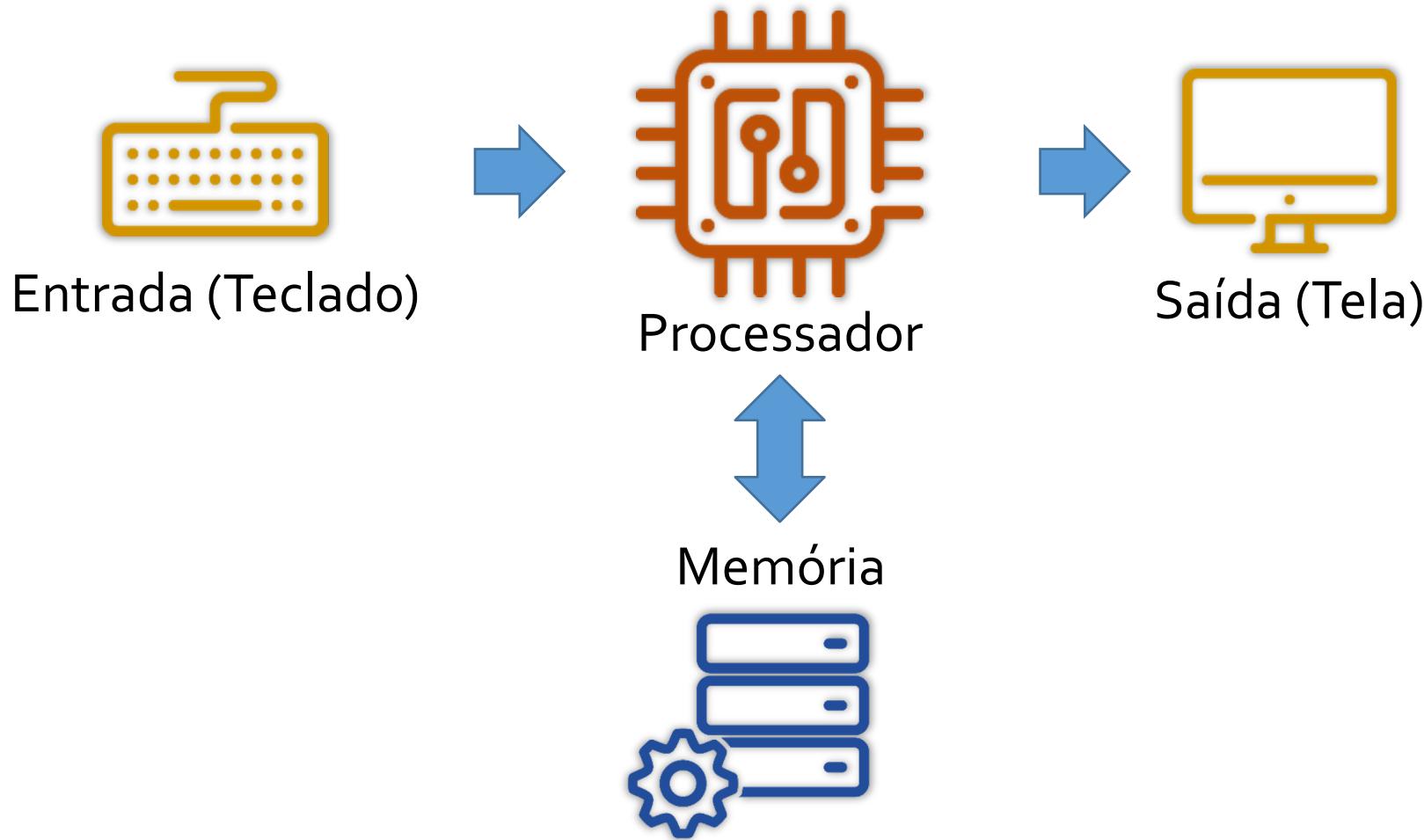
Conteúdo



Conteúdo



Componentes básicos de um computador



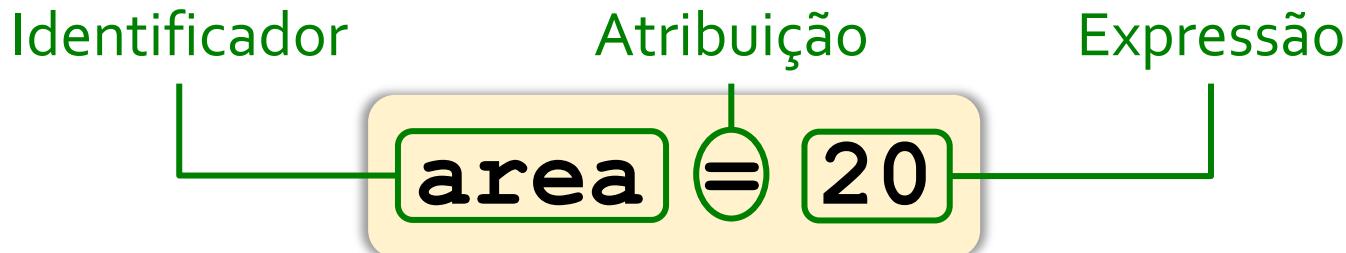
O que são variáveis?

- 💡 Variável é uma **região de memória** do computador.
- 💡 Toda variável contém um **valor**.
- 💡 Toda variável é apontada por um **identificador**.

Identificador	Conteúdo
moedas	12
nome	Maria
preco_pao	4.56
...	
condicao	Falso



Como declarar variáveis?



Identificador

- Nome dado aos **objetos** utilizados no programa (variáveis, constantes, funções, etc.)

Atribuição

- Comando que **instrui ao computador** que valor será guardado em uma variável

Expressão

- Pode ser um **valor** ou um conjunto de comandos que resulta em um valor

Como declarar variáveis? :: Exemplos

Arquivo Editar Buscar Executar Ferramentas

Python 3 main.py

```
var1 = 15
print("var1:", var1)

var2 = 12 * 4
print("var2:", var2)

var3 = var2
print("var3:", var3)
```

Console Shell

```
var1: 15
var2: 48
var3: 48
```

Arquivo Editar Buscar Executar Ferramentas

Python 3 main.py

```
var4 = var3 - 8 + 0.5
print("var4:", var4)

var5 = "Guido van Rossum"
print("var5:", var5)

var6 = var5 + " criou o
Python"
print("var6:", var6)
```

Console Shell

```
var4: 40.5
var5: Guido van Rossum
var6: Guido van Rossum
criou o Python
```

Qual a relação com variáveis no contexto matemático?

Matemática

Variáveis podem assumir o valor de **qualquer** elemento de um conjunto.



Algoritmos

Variáveis podem assumir apenas um **único** valor de cada vez, que muda ao longo da execução do programa.

Variáveis na memória do computador

Arquivo Editar Buscar Executar Ferramentas

Python 3 main.py

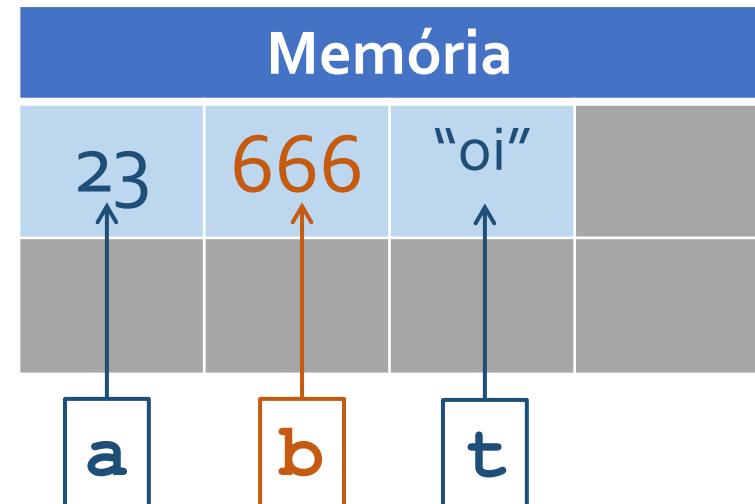
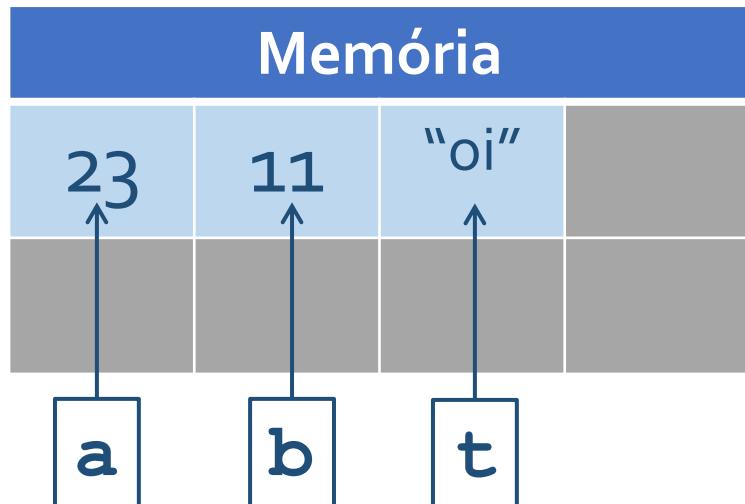
```
a = 23
b = 11
t = "oi"
```



Arquivo Editar Buscar Executar Ferramentas

Python 3 main.py

```
...
b = 666
```



Valor antigo de
b é perdido.

Regras para criar um identificador

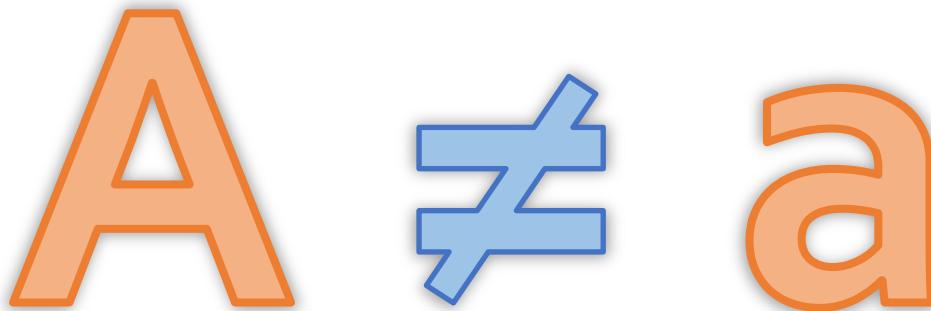
- 🛡️ O **início** do identificador deve ser obrigatoriamente uma **letra do alfabeto** ou o caractere **sublinhado** (_).
- 🛡️ Os demais caracteres podem conter **letras**, **números** ou o caractere **sublinhado** (_).
- 🛡️ Não use uma palavra-chave reservada:

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

Regras para criar um identificador

:: Cuidados

- 🛡 Não use espaços.
- 🛡 Letras maiúsculas e minúsculas são **diferentes**.
- 🛡 Os identificadores **Area** e **area** referem-se a variáveis distintas.



Regras para criar um identificador

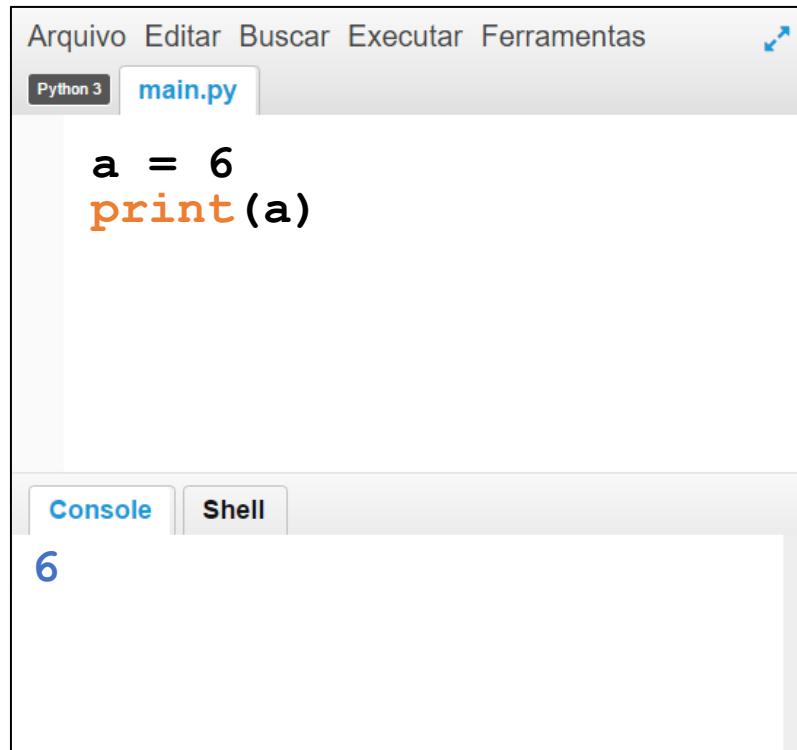
:: Exemplos

Nome	Válido	Comentário
dia1	✓	
diaDaSemana	✓	
dia da semana	✗	Contém espaços
dia_da_semana	✓	
dia#3	✗	Usa símbolo inválido
3o_dia	✗	Começa com número
_dia	✓	

Variáveis

:: Observações

- Se você declarar uma variável já existente, o conteúdo anterior será **perdido**.



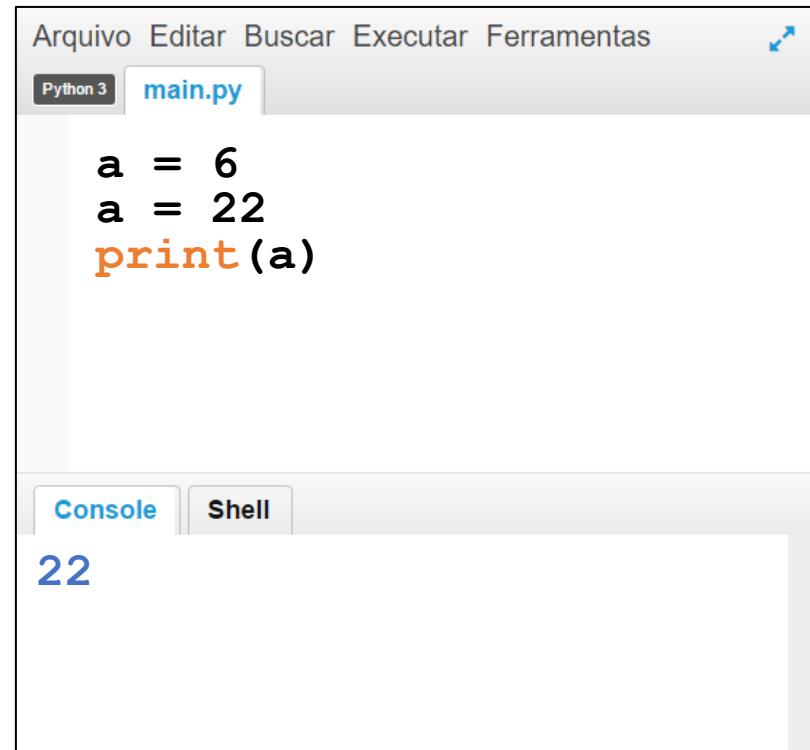
Arquivo Editar Buscar Executar Ferramentas

Python 3 main.py

```
a = 6
print(a)
```

Console Shell

6



Arquivo Editar Buscar Executar Ferramentas

Python 3 main.py

```
a = 6
a = 22
print(a)
```

Console Shell

22

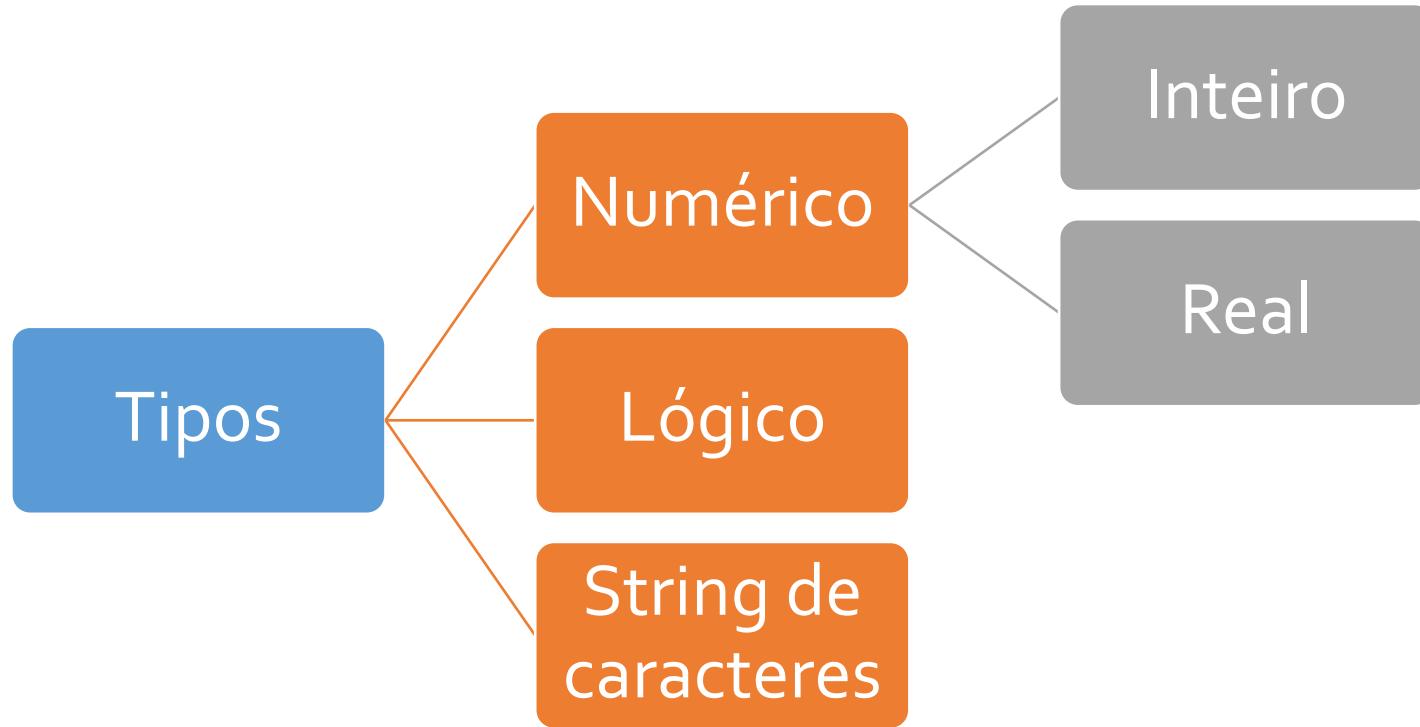
O valor das variáveis pode mudar

dívida	compra
0	?
0	100
100	100
100	200
300	200
300	300
600	300
600	0
600	0

- 1 **divida** = 0
- 2 **compra** = 100
- 3 **divida** = **divida** + **compra**
- 4 **compra** = 200
- 5 **divida** = **divida** + **compra**
- 6 **compra** = 300
- 7 **divida** = **divida** + **compra**
- 8 **compra** = 0
- 9 **print(divida)**

Tipos de variáveis

- O tipo define a **natureza dos dados** que a variável armazena, e as **operações** que podem ser realizadas.
- Tipos mais comuns no Python:



Tipos Numéricos

:: Classificação

Inteiros (int)

- São números sem a parte fracionária.
- Exemplos: 1 | 0 | -5 | 567

Reais (float)

- São números com parte fracionária.
- Também conhecidos como **ponto flutuante**.
- Exemplos: 1.0 | 3.1415 | 2.7182

Ponto é ponto, vírgula é vírgula

- Em Python, utilizamos o **ponto** para separar a parte inteira da parte fracionária de um número real:

1.03

.75

23.

- A **vírgula** é usada para **separar elementos** de uma listagem ou conjunto.
- Exemplo: separar os diversos argumentos da função **print()**

Tipos Numéricos

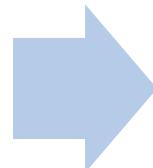
:: Exemplos

Número	Inteiro	Real
5	✓	
5.0		✓
4.3		✓
-2	✓	
100	✓	
1.333		✓

Representação de números reais

:: Limitações

O conjunto dos números reais é **infinito**, mas a memória do computador é um recurso **finito**.



Logo, somente **alguns** elementos do conjunto de números reais podem ser representados em um computador.

Representação de números reais

:: Exemplos de limitação

Arquivo Editar Buscar Executar Ferramentas

Python 3 main.py

```
print(10**5 + 7.1)
print(10**30 + 7.1)
print(10**(-30) + 7.1)
```

Quando números de
grandezas bem diferentes são
somados ou subtraídos, o
menor deles é **desprezado**.

Console Shell

100007.1
1e+30
7.1

Arquivo Editar Buscar Executar Ferramentas

Python 3 main.py

```
a = 5.4
b = 10**20
c = b
print((a + b) - c)
print(a + (b - c))
```

A propriedade
associativa nem
sempre é **válida**.

Console Shell

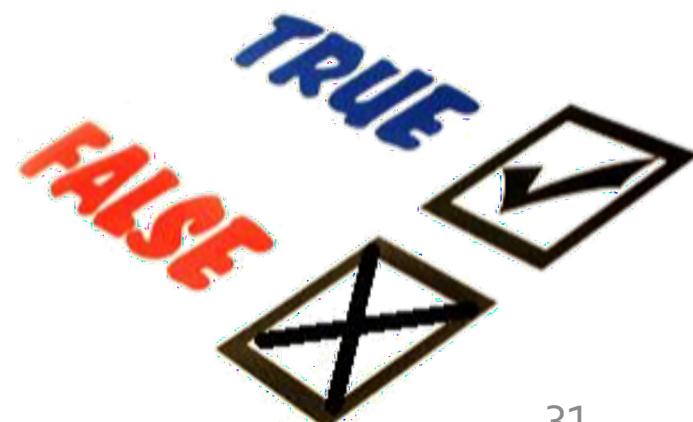
0.0
5.4

Tipo Lógico

- Uma variável do tipo lógico (ou booleano) assume apenas um entre dois valores possíveis:

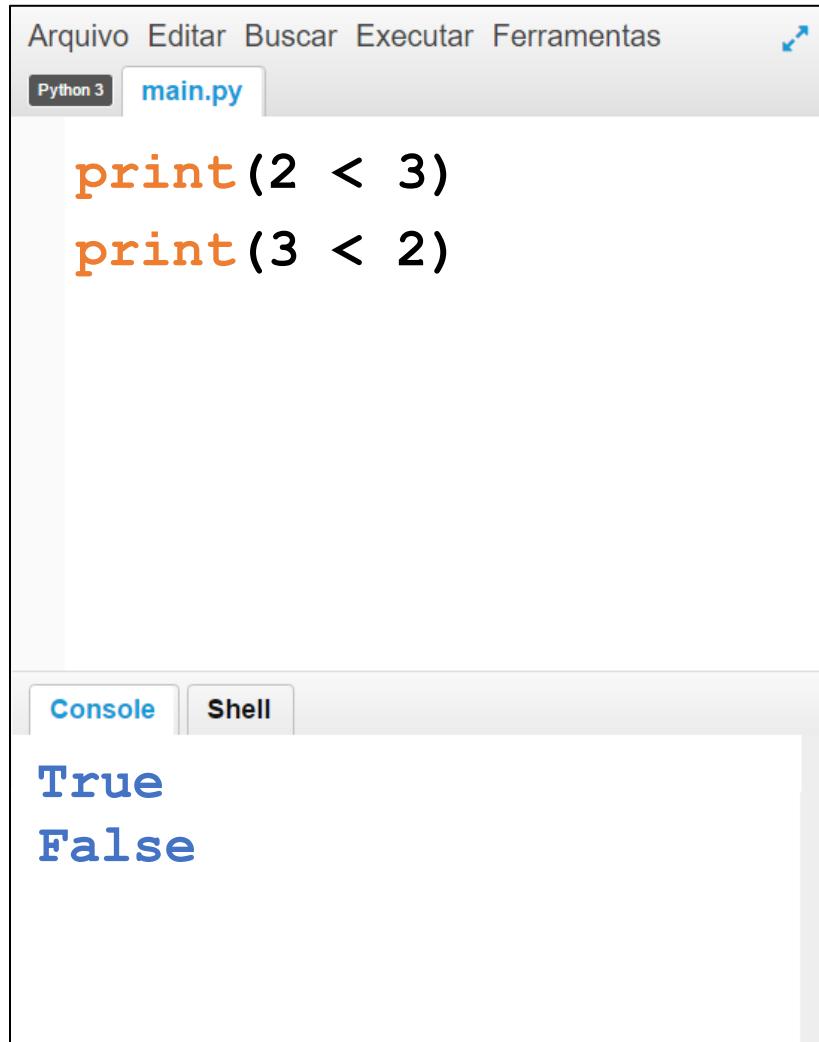
True
(verdadeiro)  **False**
(falso)

- Note que as iniciais **T** e **F** são escritas em **maiúsculas**.



Tipo Lógico

:: Exemplos



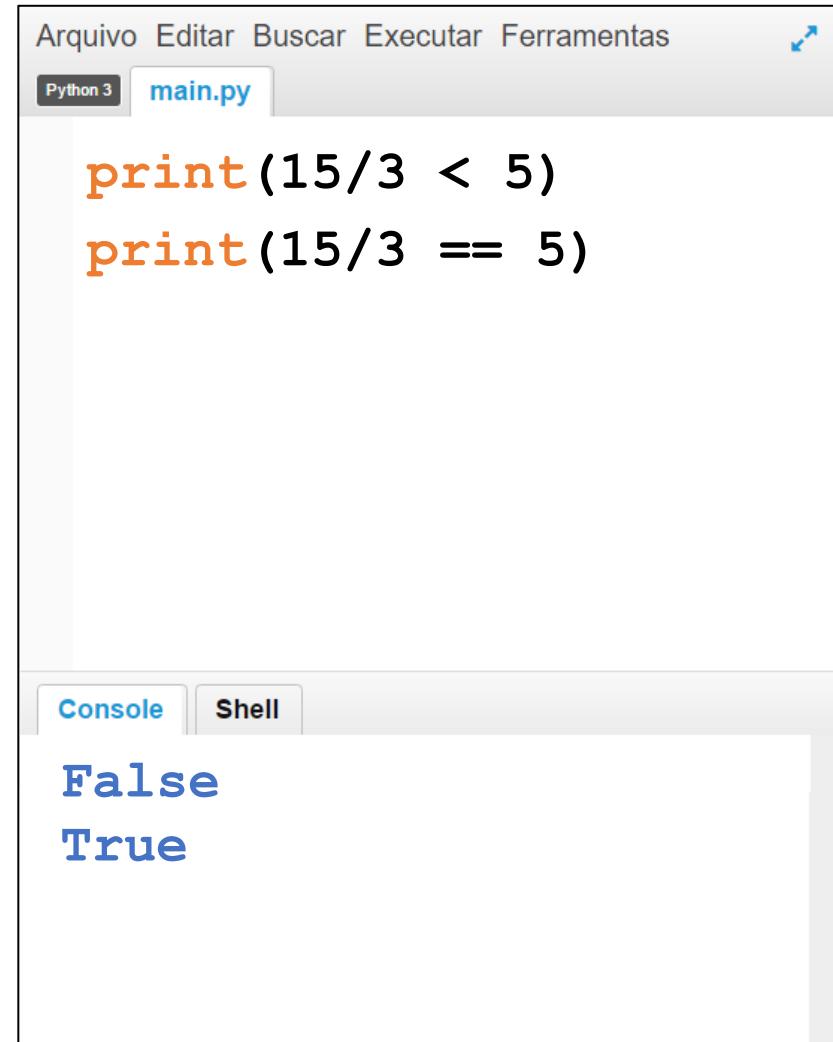
Arquivo Editar Buscar Executar Ferramentas

Python 3 main.py

```
print(2 < 3)
print(3 < 2)
```

Console Shell

True
False



Arquivo Editar Buscar Executar Ferramentas

Python 3 main.py

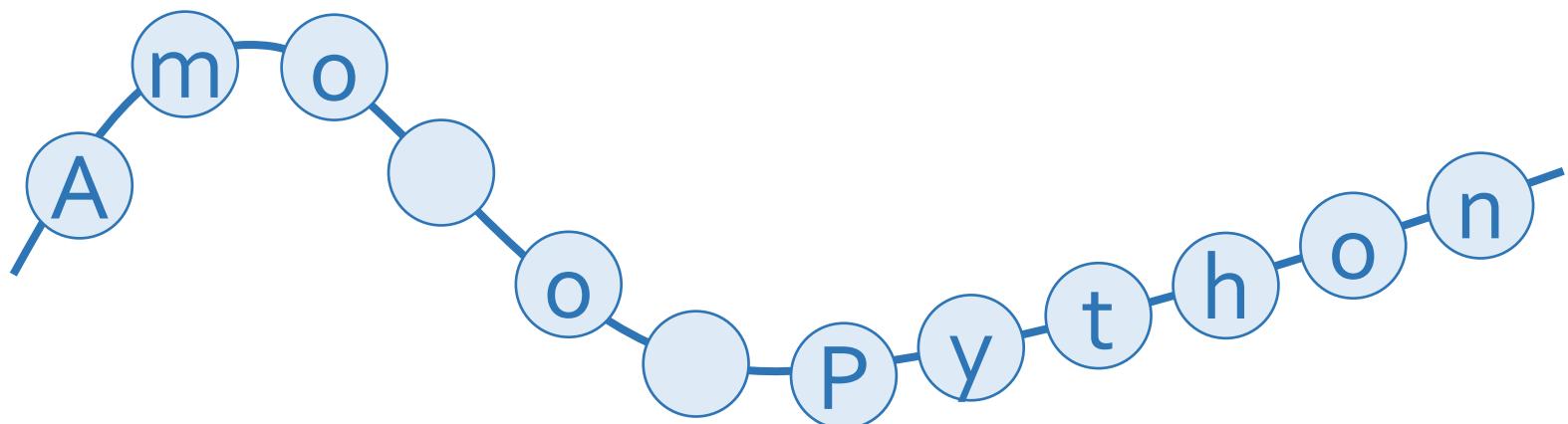
```
print(15/3 < 5)
print(15/3 == 5)
```

Console Shell

False
True

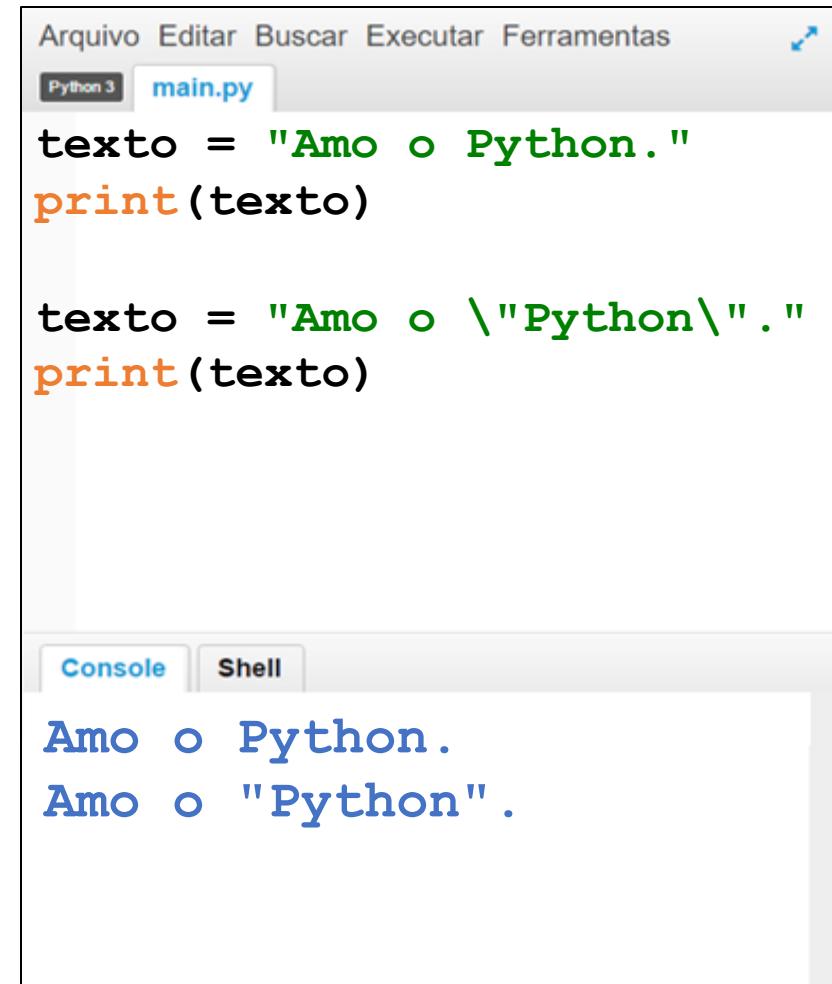
String de caracteres

- Uma **string** (= corda) é uma cadeia de caracteres.
- Uma cadeia de caracteres é um sequência de símbolos, tais como letras, números, sinais de pontuação, etc., que formam **textos** em geral.



String de caracteres

- As aspas ("") são usadas para **delimitar** os caracteres que fazem parte de uma string.
- Você pode usar **espaços** dentro de uma string.
- Se o símbolo de aspas fizer parte da string, use a expressão \".



The screenshot shows a Python code editor with a file named `main.py`. The code contains two `print` statements:

```
Arquivo Editar Buscar Executar Ferramentas
Python 3 main.py
texto = "Amo o Python."
print(texto)

texto = "Amo o \"Python\"."
print(texto)
```

The output window below shows the results of the `print` statements:

```
Console Shell
Amo o Python.
Amo o "Python".
```

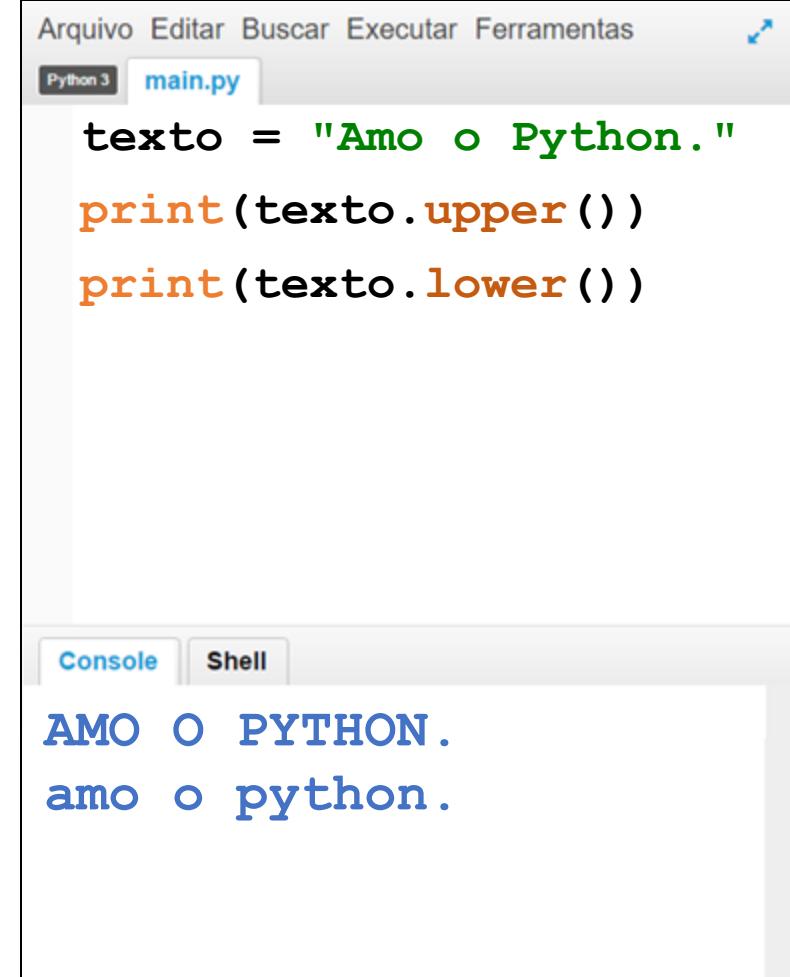
Maiúsculas e minúsculas

.upper()

- Ajusta todas as letras de uma string para **maiúsculas**

.lower()

- Ajusta todas as letras de uma string para **minúsculas**



The screenshot shows a Python code editor with a file named `main.py` containing the following code:

```
Arquivo Editar Buscar Executar Ferramentas
Python 3 main.py
texto = "Amo o Python."
print(texto.upper())
print(texto.lower())
```

The code defines a string `texto` with the value "Amo o Python.". It then prints the string in uppercase and lowercase. The output is displayed in the console tab, showing:

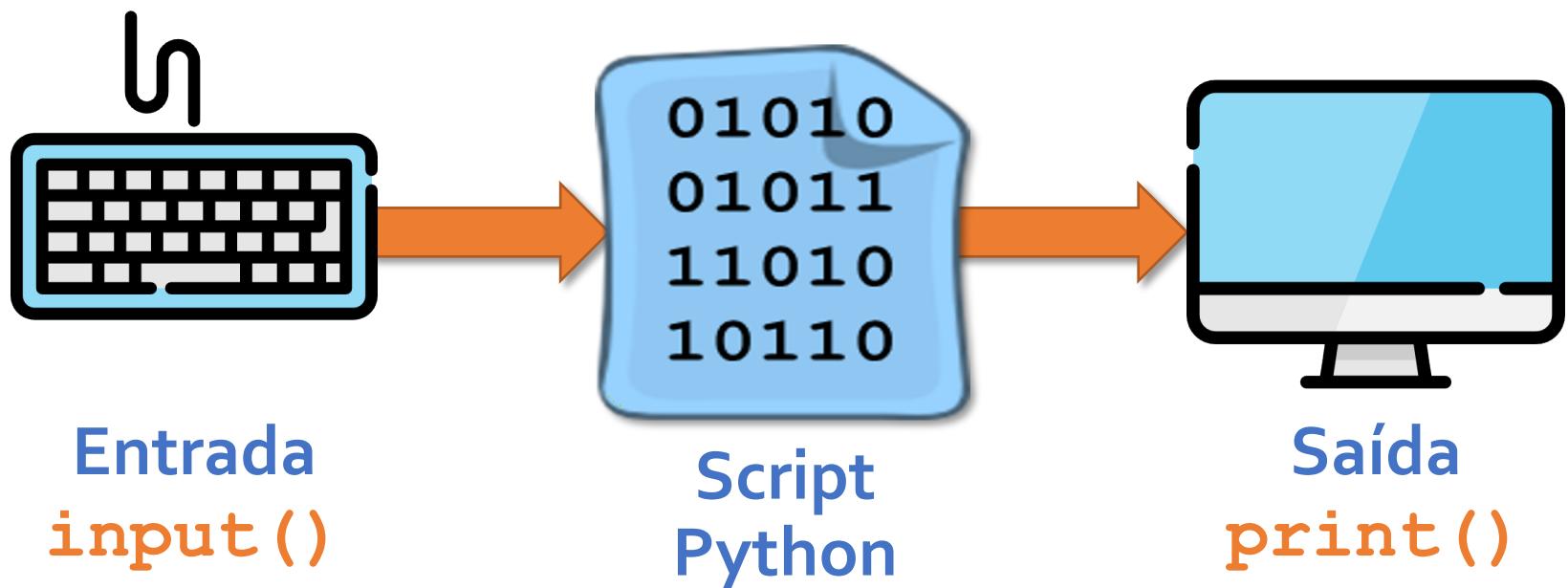
```
AMO O PYTHON.
amo o python.
```

Conteúdo



Entrada e Saída de Dados

:: Operação Básica



Programação

:: Papéis

Usuário

- Insere dados
- Testa o script



Programador

- Projeta e escreve o script



Comandos de Entrada e Saída

Comandos de Entrada

- Permitem que o usuário entre com novos valores no script.
- Exemplo:
 - Função `input()`

Comandos de Saída

- Permitem que o script exiba resultados no console.
- Exemplos:
 - Função `print()`

Saída de dados

:: **print()**

- 🛡 Exibe os dados informados no interior dos parênteses:

```
print("Oi")
```

```
x = 108
print(x)
print("Metade:", x/2)
```

```
texto = "ola mundo"
print(texto)
```

Entrada de dados pelo usuário

:: `input()`

Variável de entrada

Orientação ao usuário

```
var = input("Digite um numero: ")
```

1. Exibe, no console, um **texto de orientação ao usuário**.
2. O console **espera** até que o usuário digite um valor.
3. O valor digitado será armazenado na variável **var**.
4. O valor digitado é lido como uma **string**. Por isso, deve ser **convertido** ao tipo desejado (inteiro, float, lógico).

Entrada e Saída de Dados

:: Exemplo

```
var = input("Digite um numero: ")  
print("Voce digitou ", var)
```

1 2 3
4

1



Digite um numero:

2

15



3



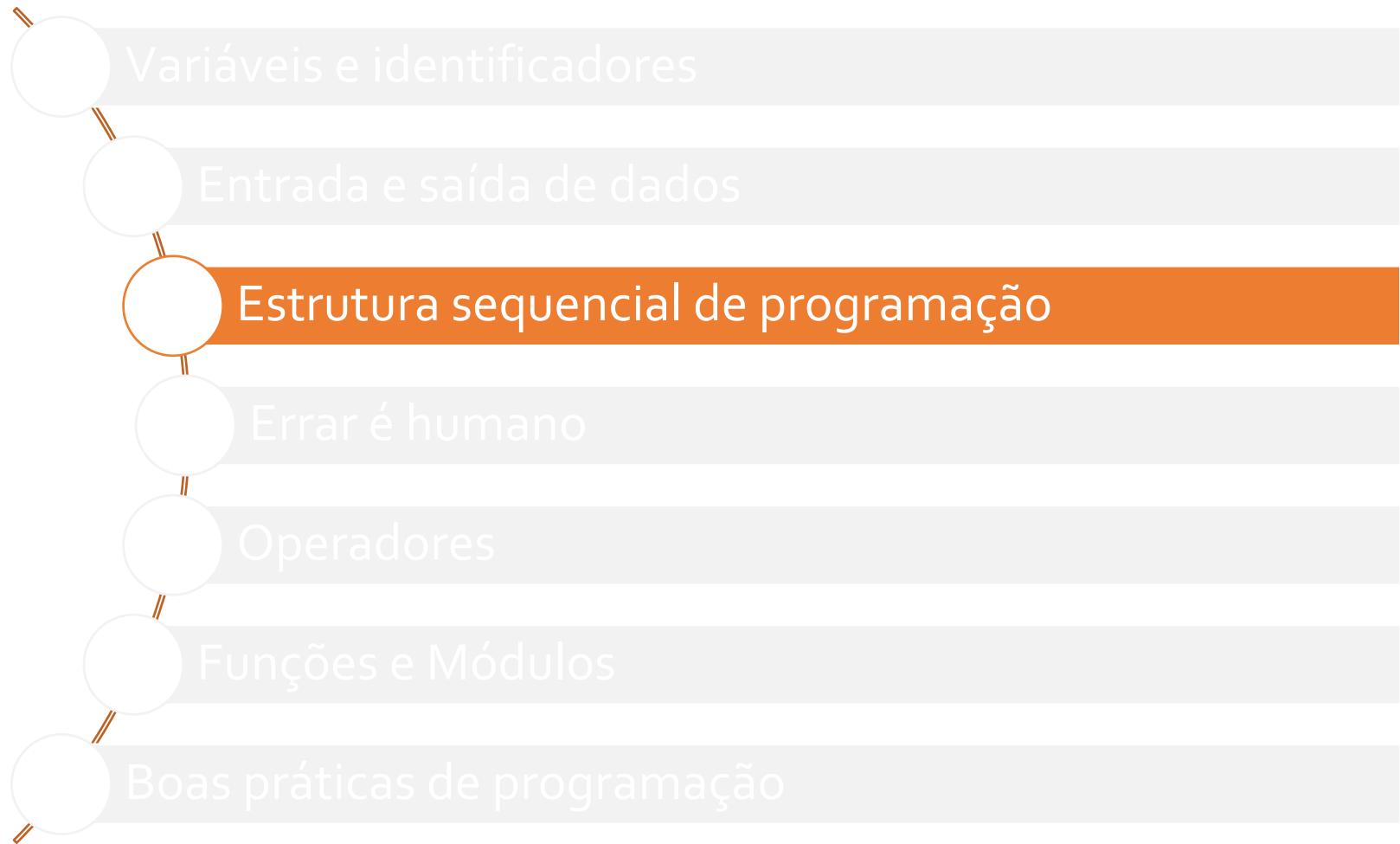
var = "15"

4



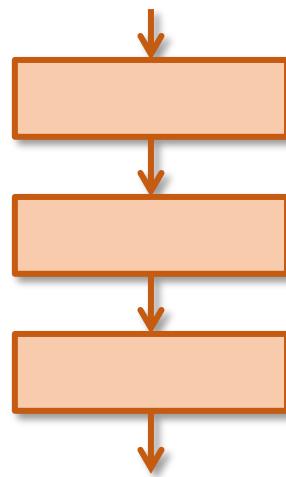
Voce digitou 15

Conteúdo

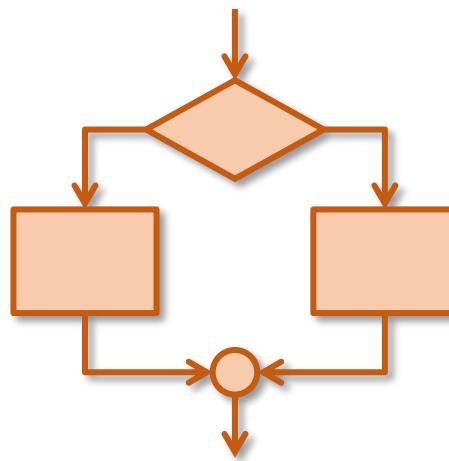


Estruturas de Programação

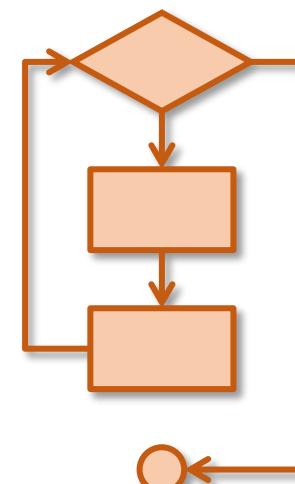
- Qualquer programa de computador pode ser escrito combinando-se os **três tipos básicos de estruturas de programação**:



Sequencial



Condicional



Repetição

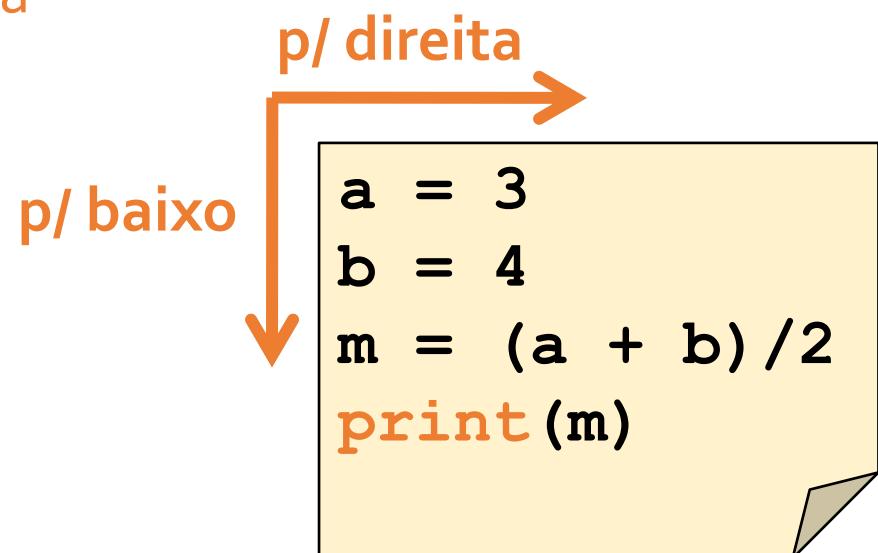
Estruturas de Programação

- 🛡 Teorema provado em 1966 por Corrado Böhm (1923-2017) e Giuseppe Jacopini (1936-2001) no artigo: *“Flow Diagrams, Turing Machines And Languages With Only Two Formation Rules”*.



Estrutura Sequencial

- 🛡 É a estrutura de programação mais simples.
- 🛡 O fluxo de comandos do algoritmo segue a mesma sequência linear da nossa escrita:
 - 🛡 De **cima para baixo**
 - 🛡 Da **esquerda para direita**

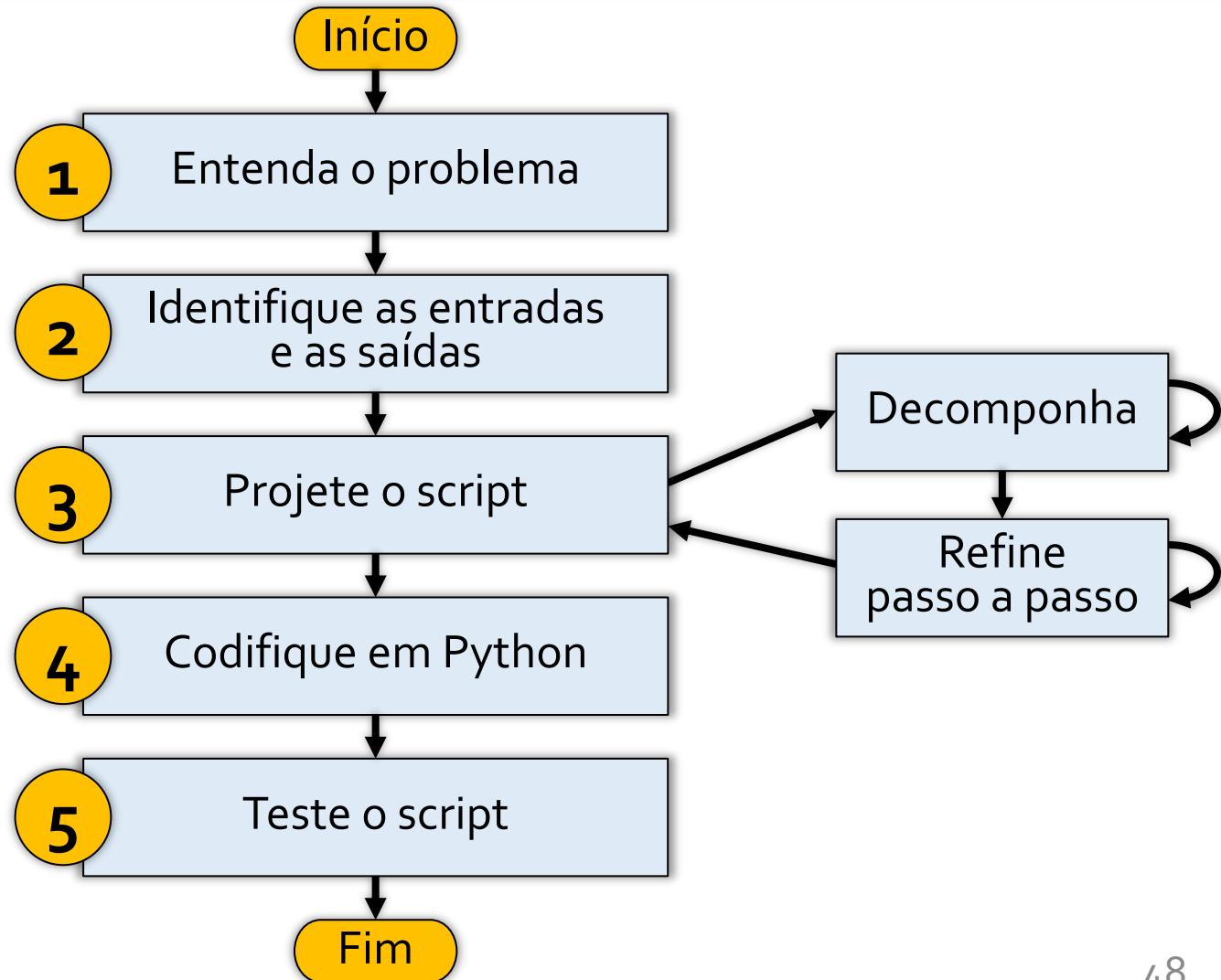


Problema 1

- Um estádio oferece três categorias de assentos:
 - Classe A – R\$ 50
 - Classe B – R\$ 30
 - Classe C – R\$ 20
- Escreva um programa que leia **quantos bilhetes de cada classe** foram vendidos.
- Como saída, determine a **renda** gerada pela venda dos ingressos.

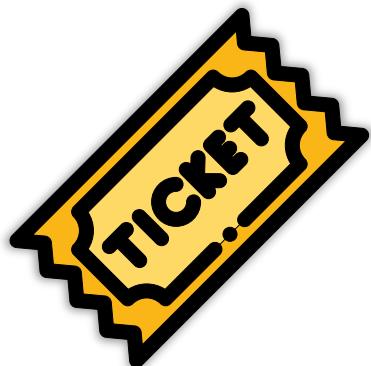


Resolução de Problemas Algorítmicos



Problema 1

1 – Entenda o problema



Quantos **tickets** foram vendidos?
Qual o valor de **cada um**?



Qual a **renda** gerada?

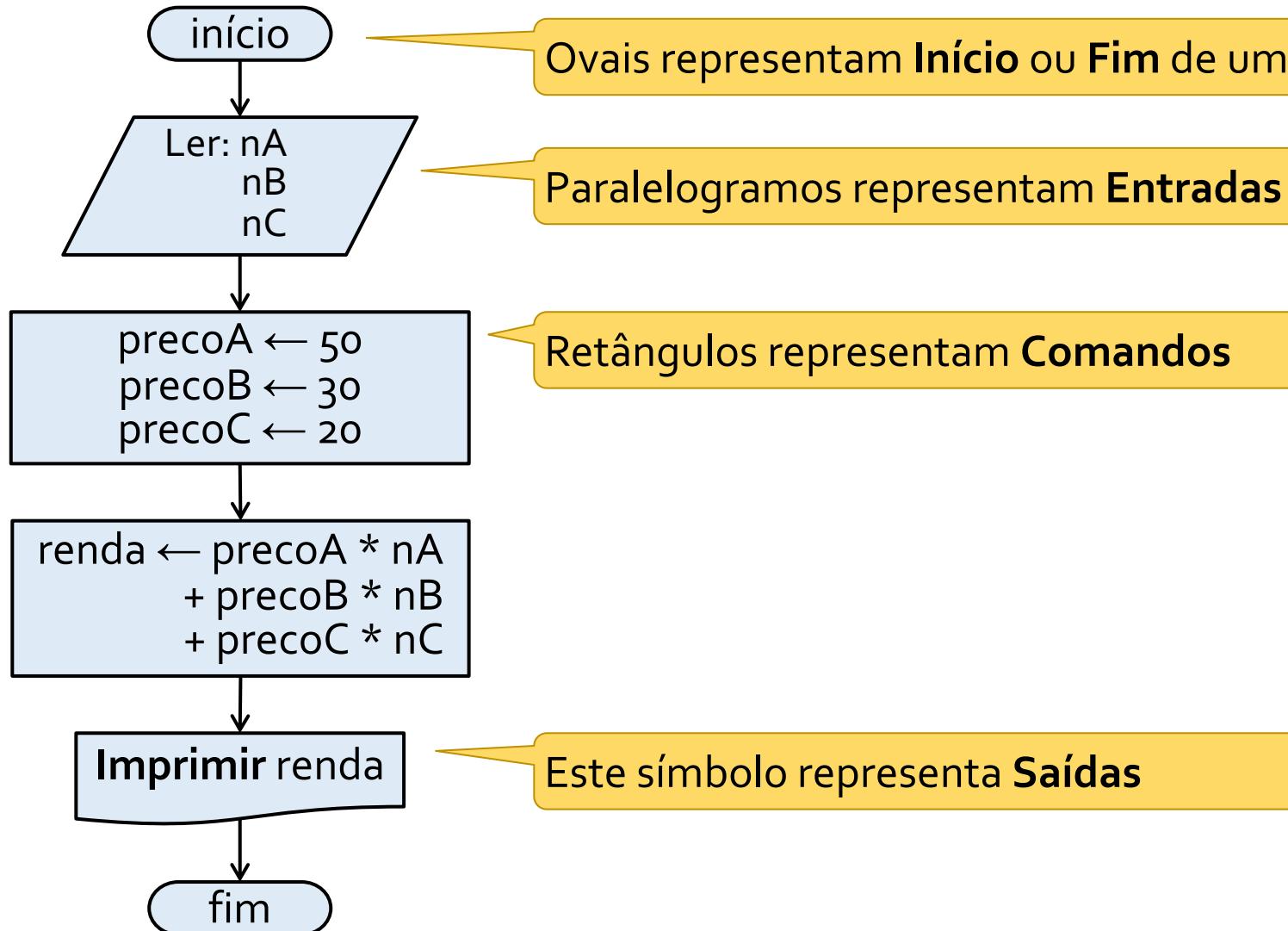
Problema 1

2 – Identifique as entradas e as saídas

	Grandeza	Unidade de medida	Faixa de valores
Entradas	Nº de bilhetes A	bilhetes	≥ 0
	Nº de bilhetes B	bilhetes	≥ 0
	Nº de bilhetes C	bilhetes	≥ 0
Saídas	Renda	R\$	≥ 0

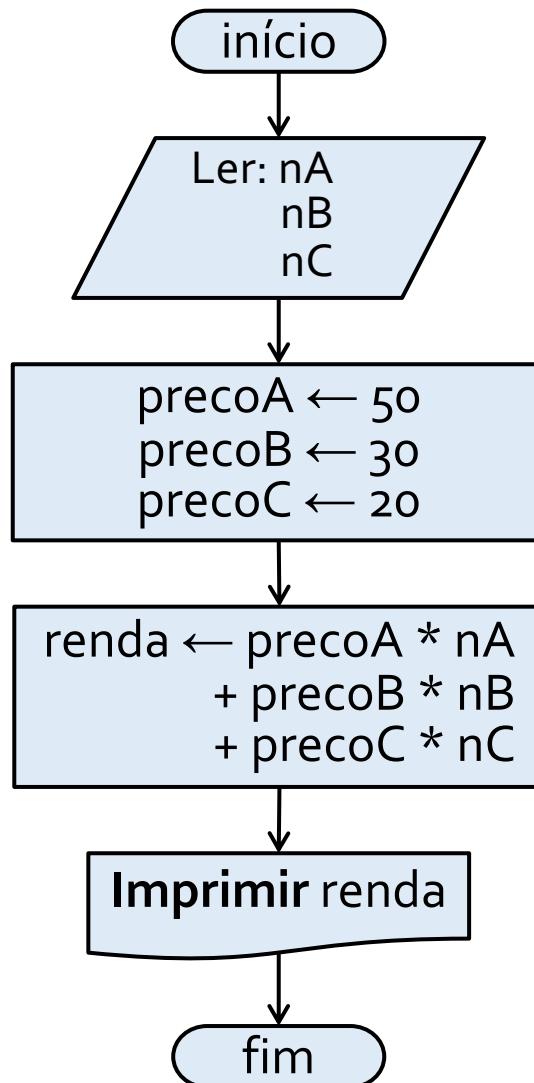
Problema 1

3 – Projete o script (fluxograma)



Problema 1

4 – Codifique em Python



```
nA = input("No. bilhetes A: ")  
nB = input("No. bilhetes B: ")  
nC = input("No. bilhetes C: ")
```

```
precoA = 50.0  
precoB = 30.0  
precoC = 20.0
```

```
renda = precoA * nA + precoB *  
nB + precoC * nC
```

```
print(renda)
```

Problema 1

5 – Teste o script

```
nA = input("No. bilhetes A: ")  
nB = input("No. bilhetes B: ")  
nC = input("No. bilhetes C: ")  
  
precoA = 50.0  
precoB = 30.0  
precoC = 20.0  
  
renda = precoA * nA + precoB * nB + precoC * nC  
  
print(renda)
```



Por que funciona
diferente do esperado?

Problema 1

6 – Corrija o script

Lembre-se de **converter** a
entrada, se ela for numérica

```
nA = int(input("No. bilhetes A: "))
nB = int(input("No. bilhetes B: "))
nC = int(input("No. bilhetes C: "))

precoA = 50.0
precoB = 30.0
precoC = 20.0

renda = precoA * nA + precoB * nB + precoC * nC

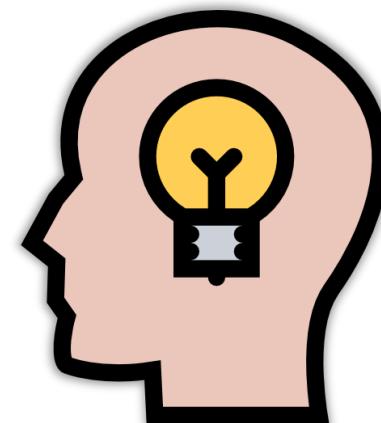
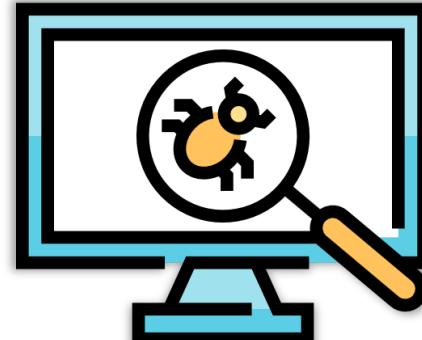
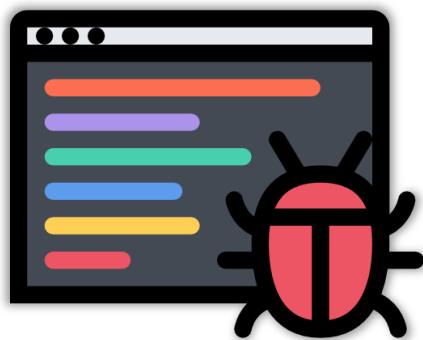
print(renda)
```

Conteúdo



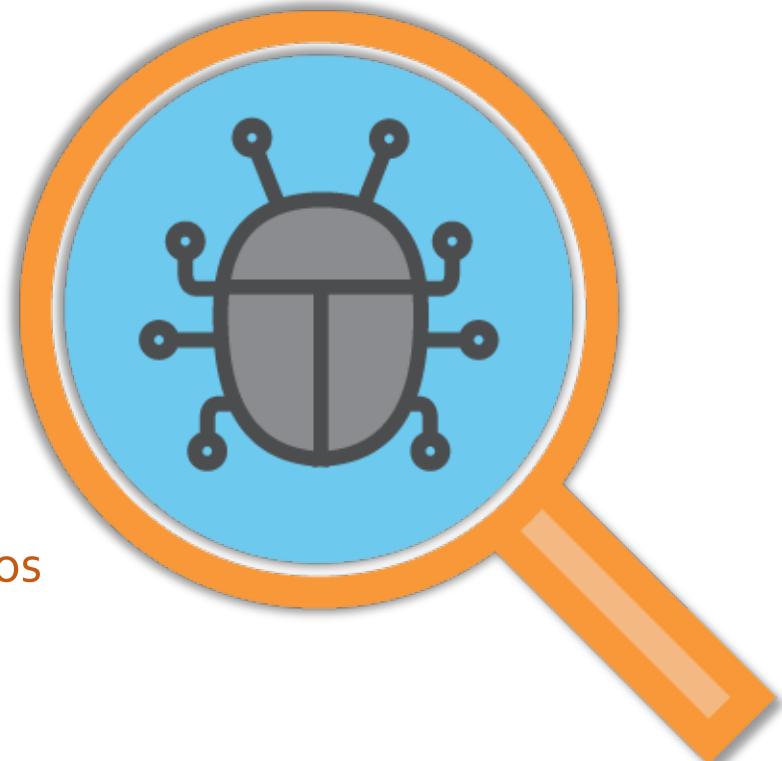
Todo mundo erra

- 🛡️ Lidar com erros **faz parte** da programação.
- 🛡️ Use as mensagens de erro **a seu favor** e aprenda com elas.
- 🛡️ Se você explorar os erros nos Labs de Codificação, terá bom desempenho nos Trabalhos Práticos.



Como lidar com erros de programação?

- 🛡 Não se pergunte: “por que o programa não funciona?”
- 🛡 Em vez disso, pergunte-se: “por que o programa está funcionando deste jeito?”
- 🛡 Estratégias:
 - 🛡 Não se limite ao exemplo: teste vários casos
 - 🛡 Imprima resultados intermediários
 - 🛡 Tente explicar o problema para outra pessoa
 - 🛡 Dê um tempo e tente de novo mais tarde



Erros de sintaxe

:: O que são?

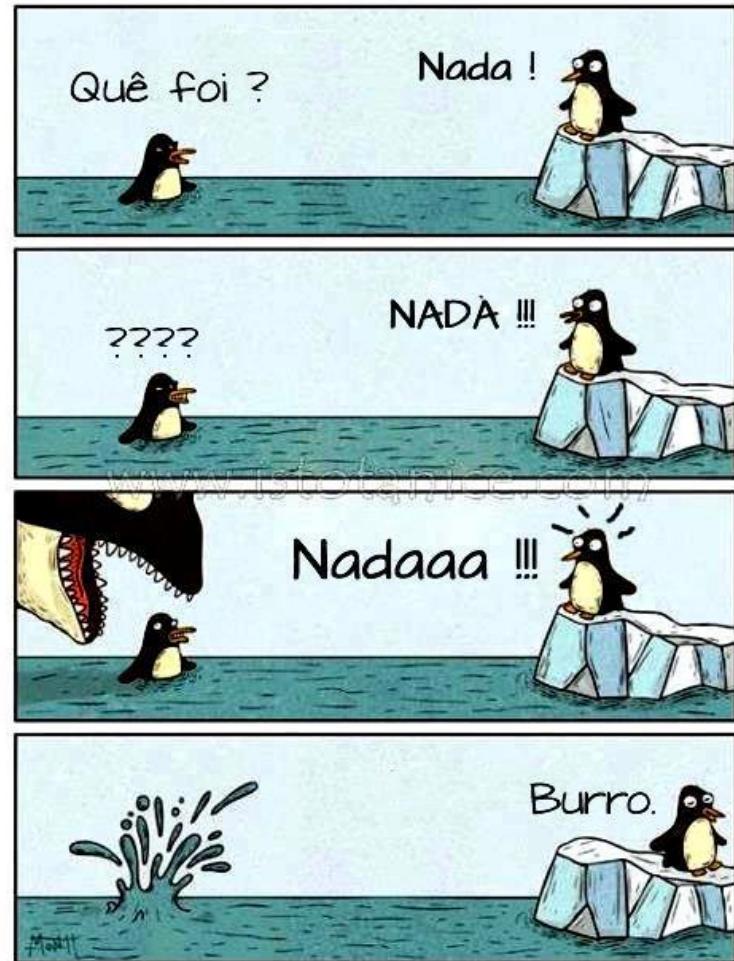
- 🛡 Os erros de sintaxe são violações das **regras de escrita** da linguagem.
- 🛡 Normalmente **são detectados** pelo Python.
- 🛡 O Python **informa** o tipo de erro: nome inválido, parêntese não fechado, etc.



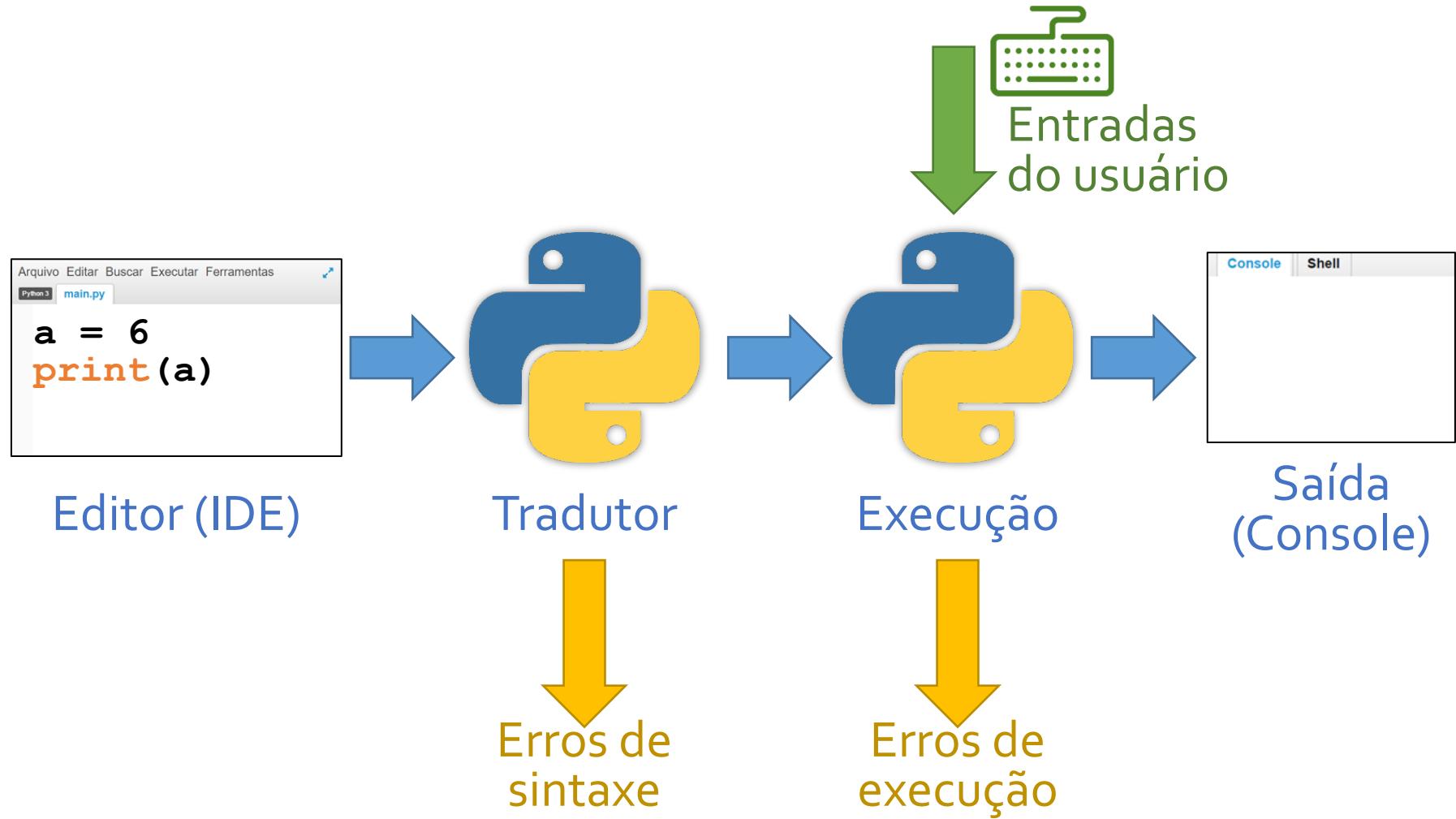
Erros de execução

:: O que são?

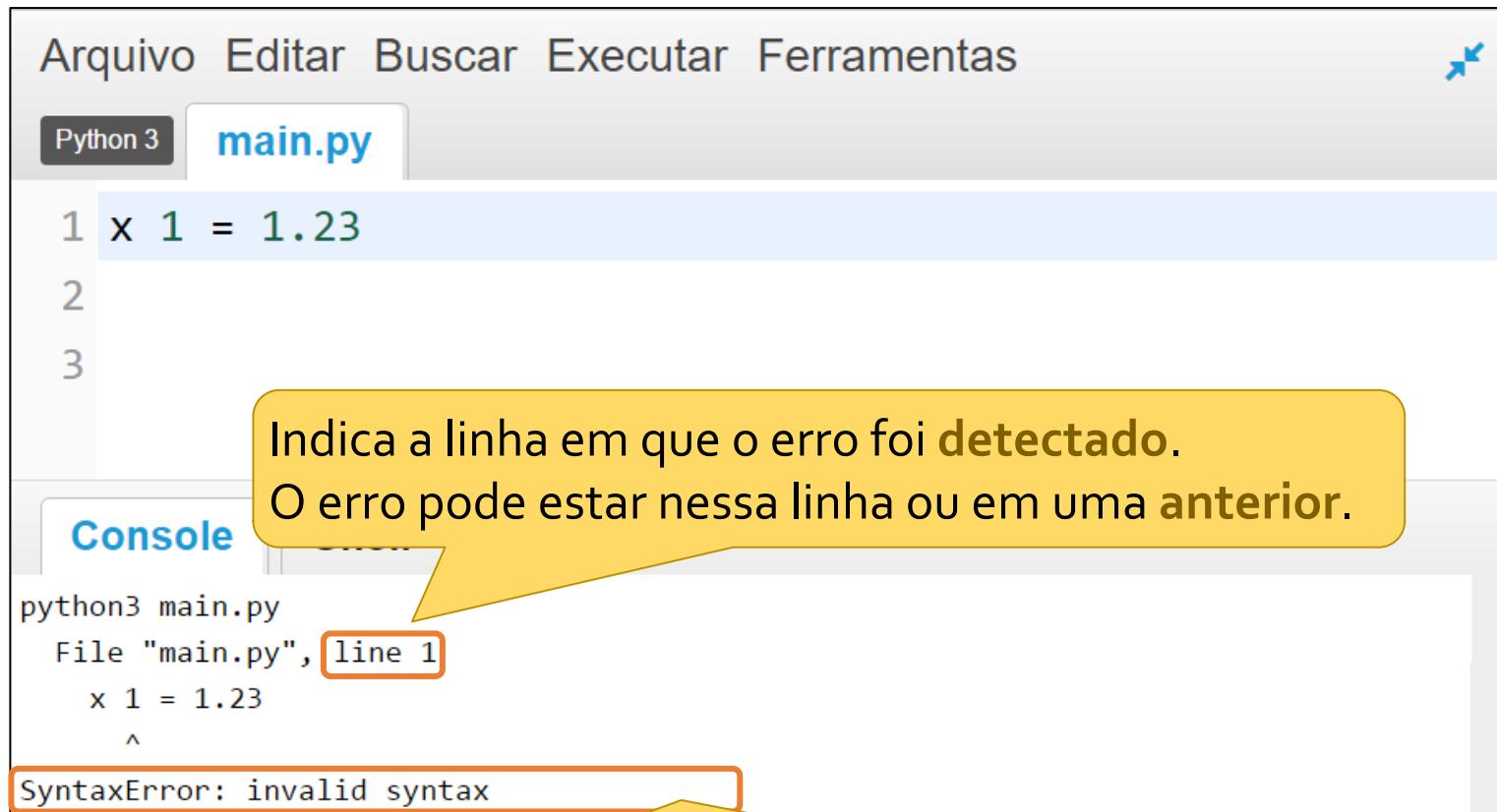
- Erros de execução são violações da **lógica** do problema.
- O **Python** não detecta esse erro.
- O **CodeBench** verifica que a resposta não corresponde ao esperado, mas **não sabe** dizer por quê.



Processo de programação em Python



Como interpretar mensagens de erro?



Arquivo Editar Buscar Executar Ferramentas ✖

Python 3 main.py

```
1 x 1 = 1.23
2
3
```

Console

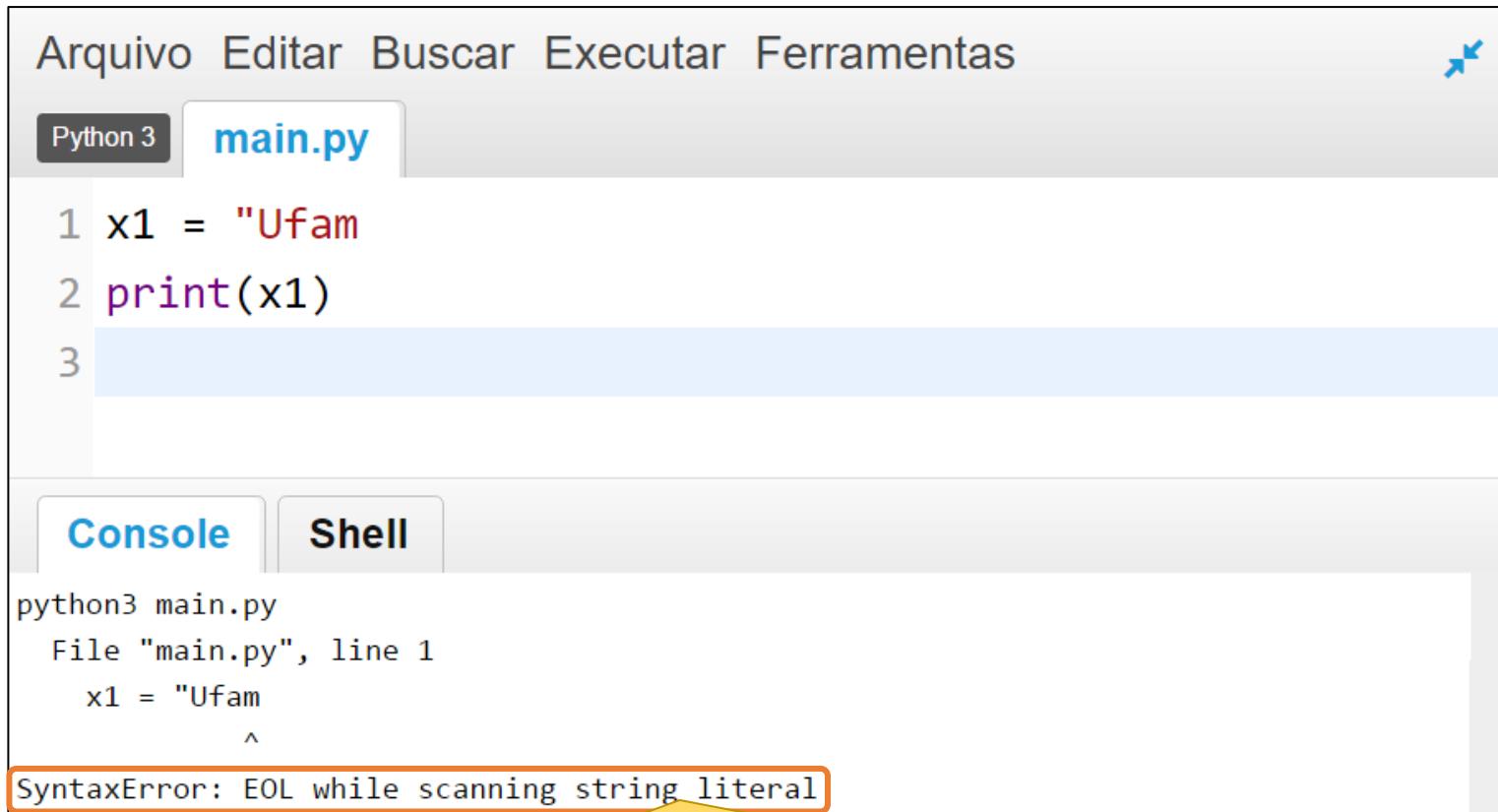
```
python3 main.py
  File "main.py", line 1
    x 1 = 1.23
      ^
SyntaxError: invalid syntax
```

Indica a linha em que o erro foi **detectado**.
O erro pode estar nessa linha ou em uma **anterior**.

Natureza do erro: “Syntax Error” (erro de sintaxe).

Explicação: o nome da variável é inválido, pois contém espaços.

Como interpretar mensagens de erro?



Arquivo Editar Buscar Executar Ferramentas ✖

Python 3 **main.py**

```
1 x1 = "Ufam
2 print(x1)
3
```

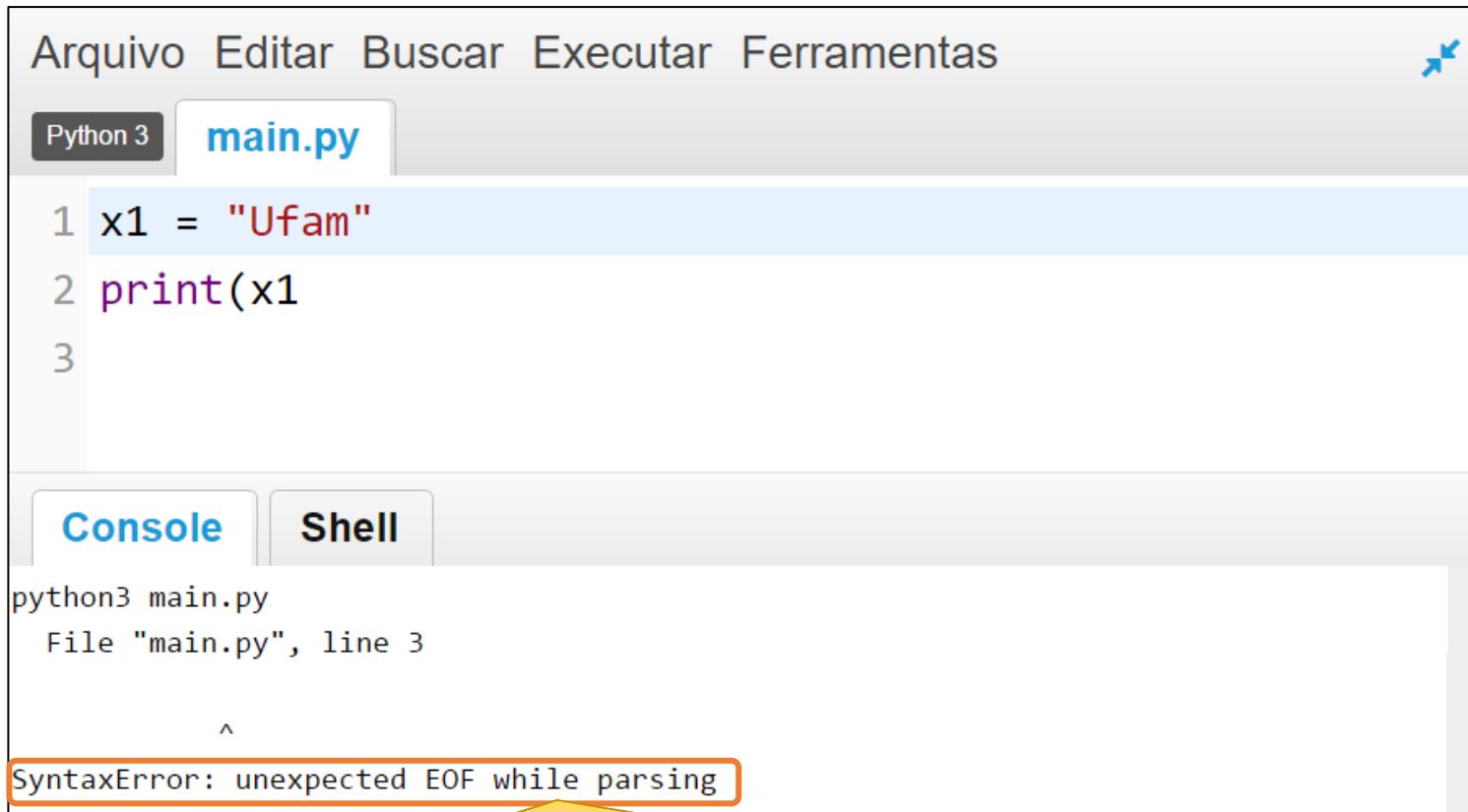
Console Shell

```
python3 main.py
  File "main.py", line 1
    x1 = "Ufam
          ^
SyntaxError: EOL while scanning string literal
```

Natureza do erro: "Syntax Error" (erro de sintaxe).

Explicação: o interpretador chegou ao fim da linha (*end of line*) sem fechar a string, pois faltou a aspa.

Como interpretar mensagens de erro?



Arquivo Editar Buscar Executar Ferramentas

Python 3 main.py

```
1 x1 = "Ufam"
2 print(x1
3
```

Console Shell

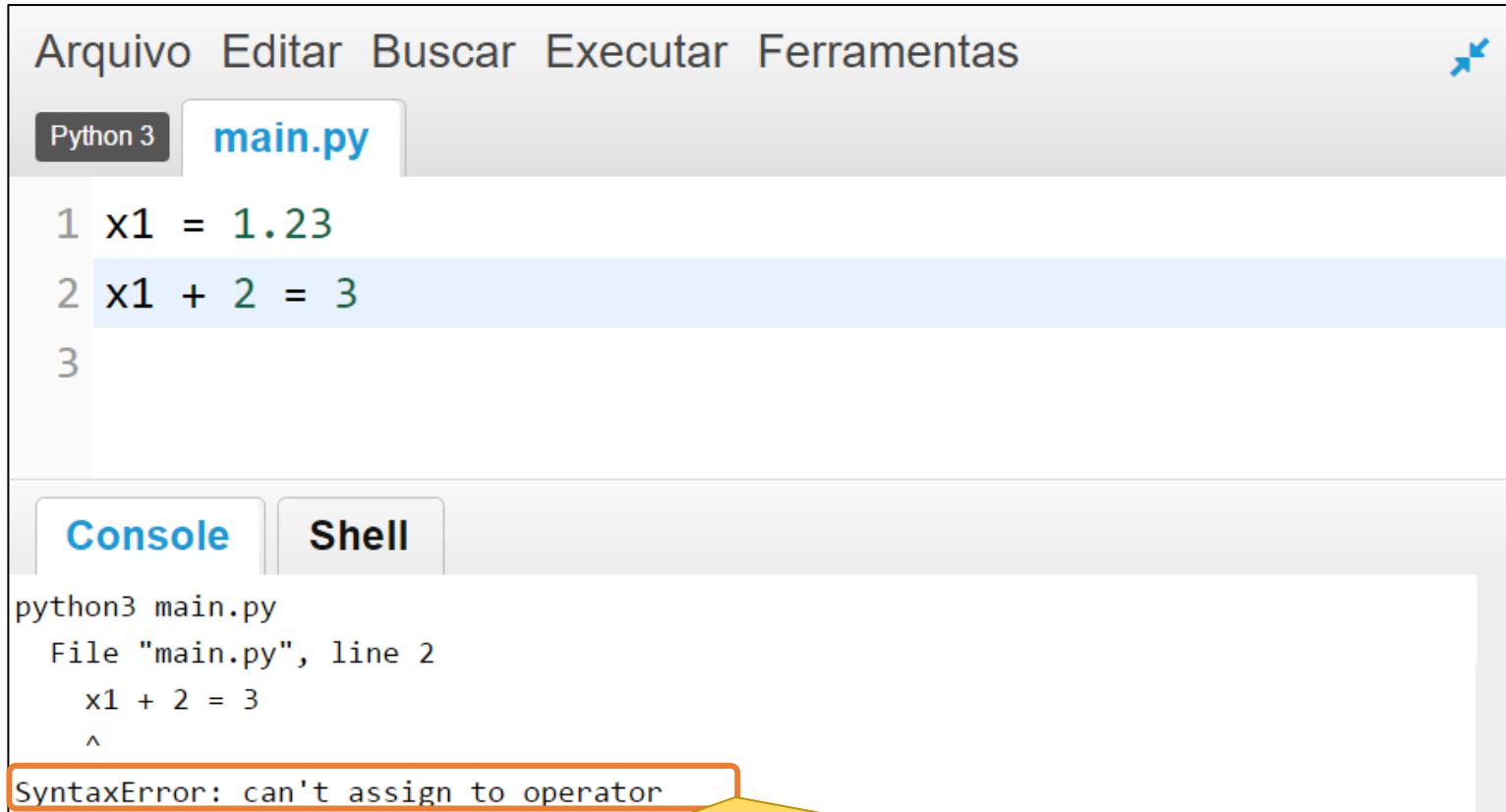
```
python3 main.py
  File "main.py", line 3

    ^
SyntaxError: unexpected EOF while parsing
```

Natureza do erro: “Syntax Error” (erro de sintaxe).

Explicação: o interpretador chegou ao fim do arquivo (*end of file*) sem fechar a análise (*parsing*) do comando, pois faltou o parêntese.

Como interpretar mensagens de erro?



Arquivo Editar Buscar Executar Ferramentas

Python 3 main.py

```
1 x1 = 1.23
2 x1 + 2 = 3
3
```

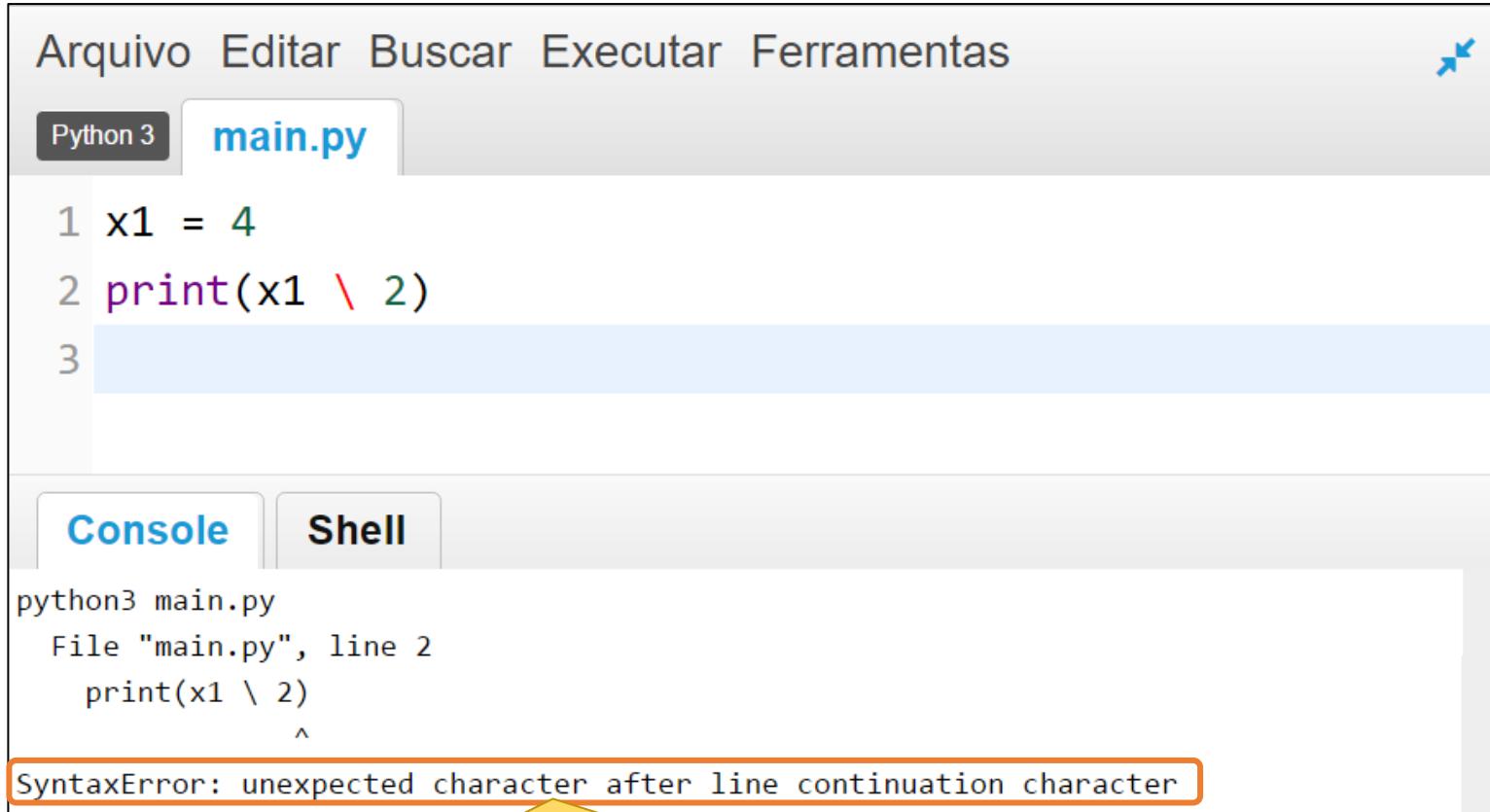
Console Shell

```
python3 main.py
  File "main.py", line 2
    x1 + 2 = 3
    ^
SyntaxError: can't assign to operator
```

Natureza do erro: “Syntax Error” (erro de sintaxe).

Explicação: não é possível atribuir (*assign*) um valor para uma operação (neste caso, a adição)

Como interpretar mensagens de erro?



Arquivo Editar Buscar Executar Ferramentas

Python 3 main.py

```
1 x1 = 4
2 print(x1 \ 2)
3
```

Console Shell

```
python3 main.py
  File "main.py", line 2
    print(x1 \ 2)
          ^
SyntaxError: unexpected character after line continuation character
```

The screenshot shows a Python code editor with a file named 'main.py' containing the following code:

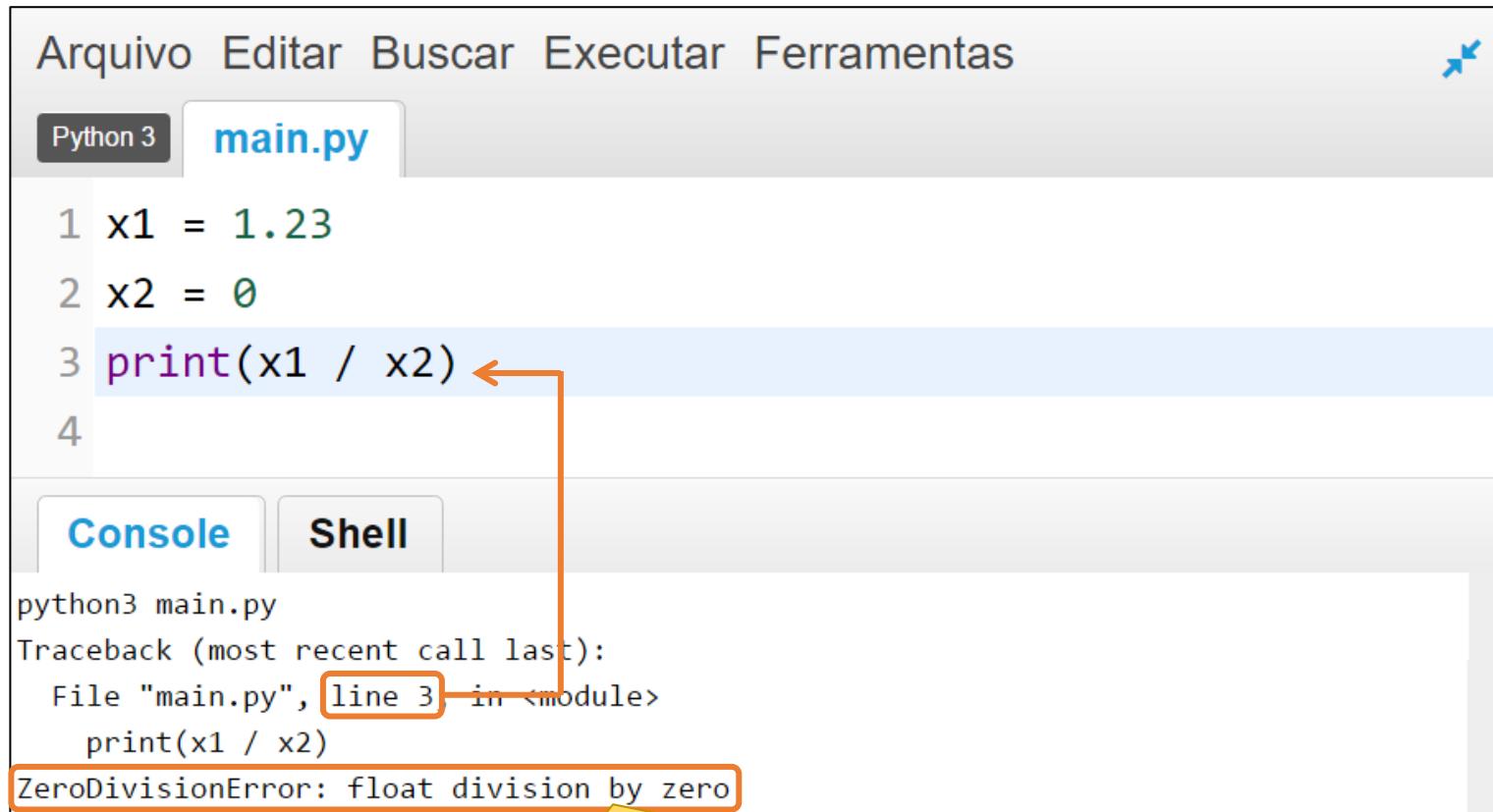
```
1 x1 = 4
2 print(x1 \ 2)
3
```

The line '2 print(x1 \ 2)' contains a syntax error. The editor highlights the character '\' in red. The error message 'SyntaxError: unexpected character after line continuation character' is displayed in the console output.

Natureza do erro: "Syntax Error" (erro de sintaxe).

Explicação: uso de caractere não esperado (contra-barra).

Como interpretar mensagens de erro?



Arquivo Editar Buscar Executar Ferramentas ✖

Python 3 main.py

```
1 x1 = 1.23
2 x2 = 0
3 print(x1 / x2) ←
4
```

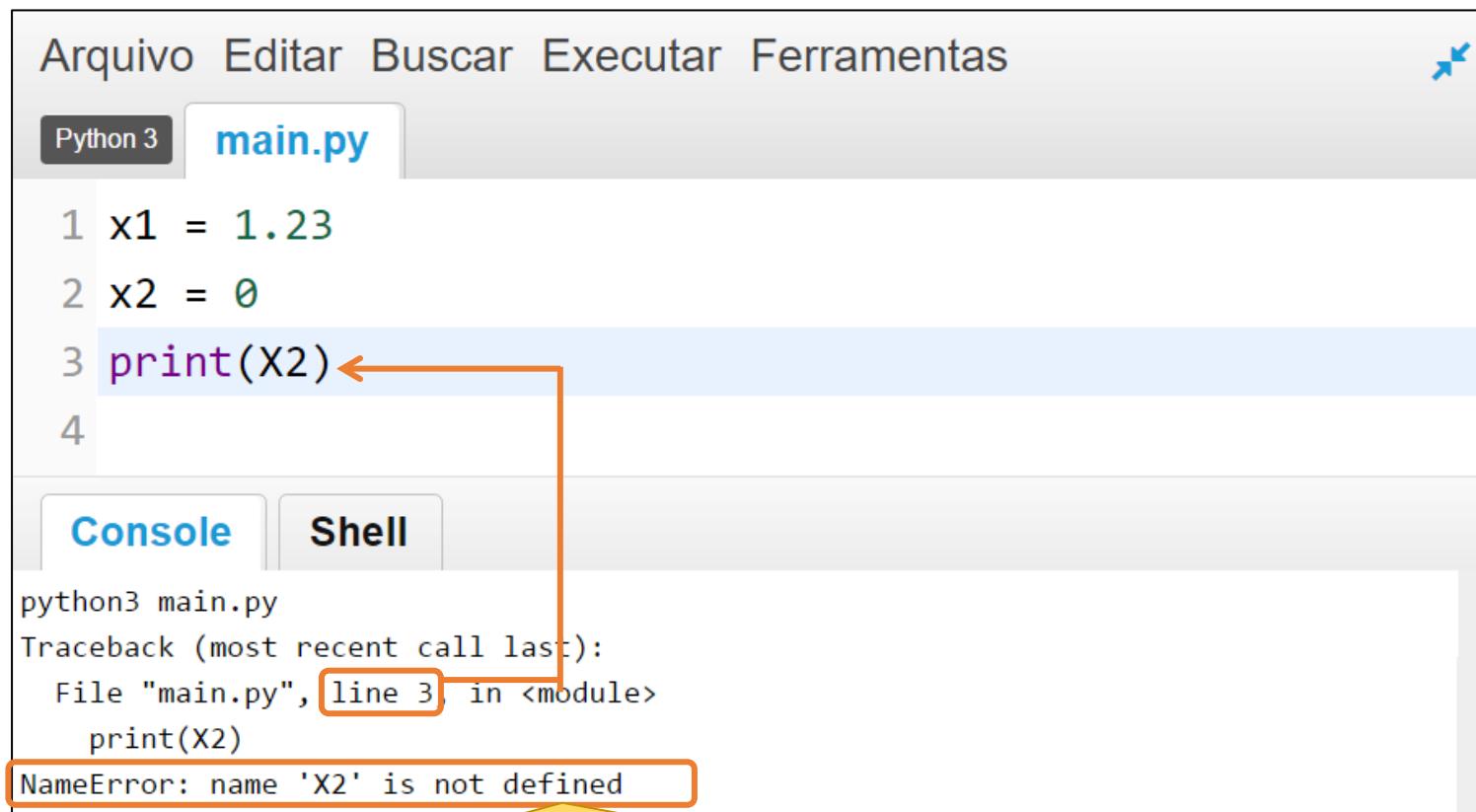
Console Shell

```
python3 main.py
Traceback (most recent call last):
  File "main.py", line 3 in <module>
    print(x1 / x2)
ZeroDivisionError: float division by zero
```

The screenshot shows a Python 3 code editor with a file named 'main.py'. The code contains four lines: 1. `x1 = 1.23`, 2. `x2 = 0`, 3. `print(x1 / x2)`, and 4. an empty line. The third line is highlighted with a light blue background and has an orange arrow pointing to it from the text below. Below the code editor is a terminal window showing the execution of `python3 main.py`. It displays a traceback where line 3 is highlighted with an orange box, and the error message `ZeroDivisionError: float division by zero` is also highlighted with an orange box.

Natureza do erro: “Zero Division Error”, ou seja, erro por divisão por zero.

Como interpretar mensagens de erro?



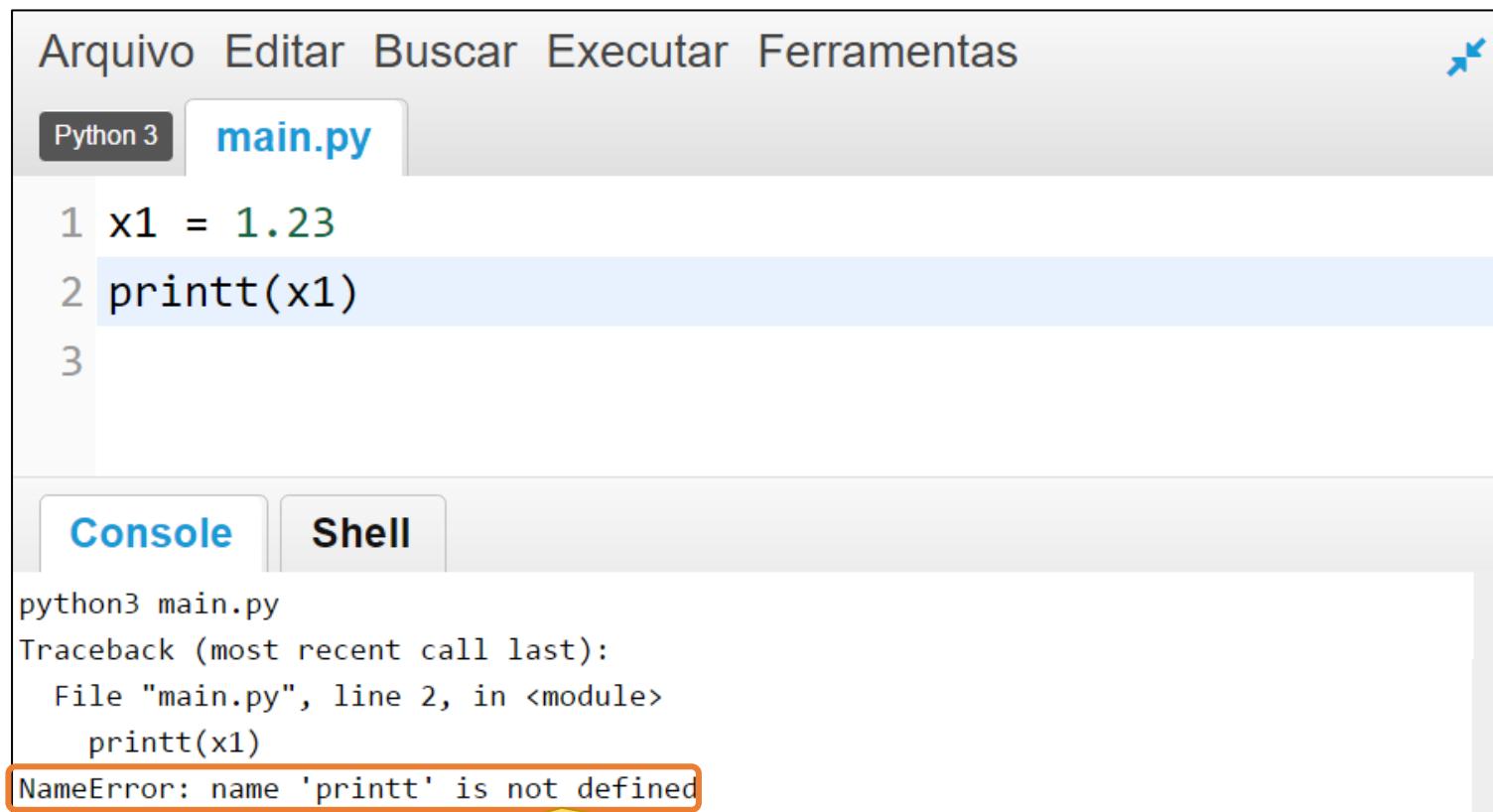
```
Arquivo Editar Buscar Executar Ferramentas
Python 3 main.py
1 x1 = 1.23
2 x2 = 0
3 print(X2)←
4

Console Shell
python3 main.py
Traceback (most recent call last):
  File "main.py", line 3, in <module>
    print(X2)
NameError: name 'X2' is not defined
```

Natureza do erro: “Name Error” (erro no nome de funções ou variáveis)

Explicação: erro por usar um nome de variável não definida previamente.

Como interpretar mensagens de erro?



Arquivo Editar Buscar Executar Ferramentas

Python 3 main.py

```
1 x1 = 1.23
2 printt(x1)
3
```

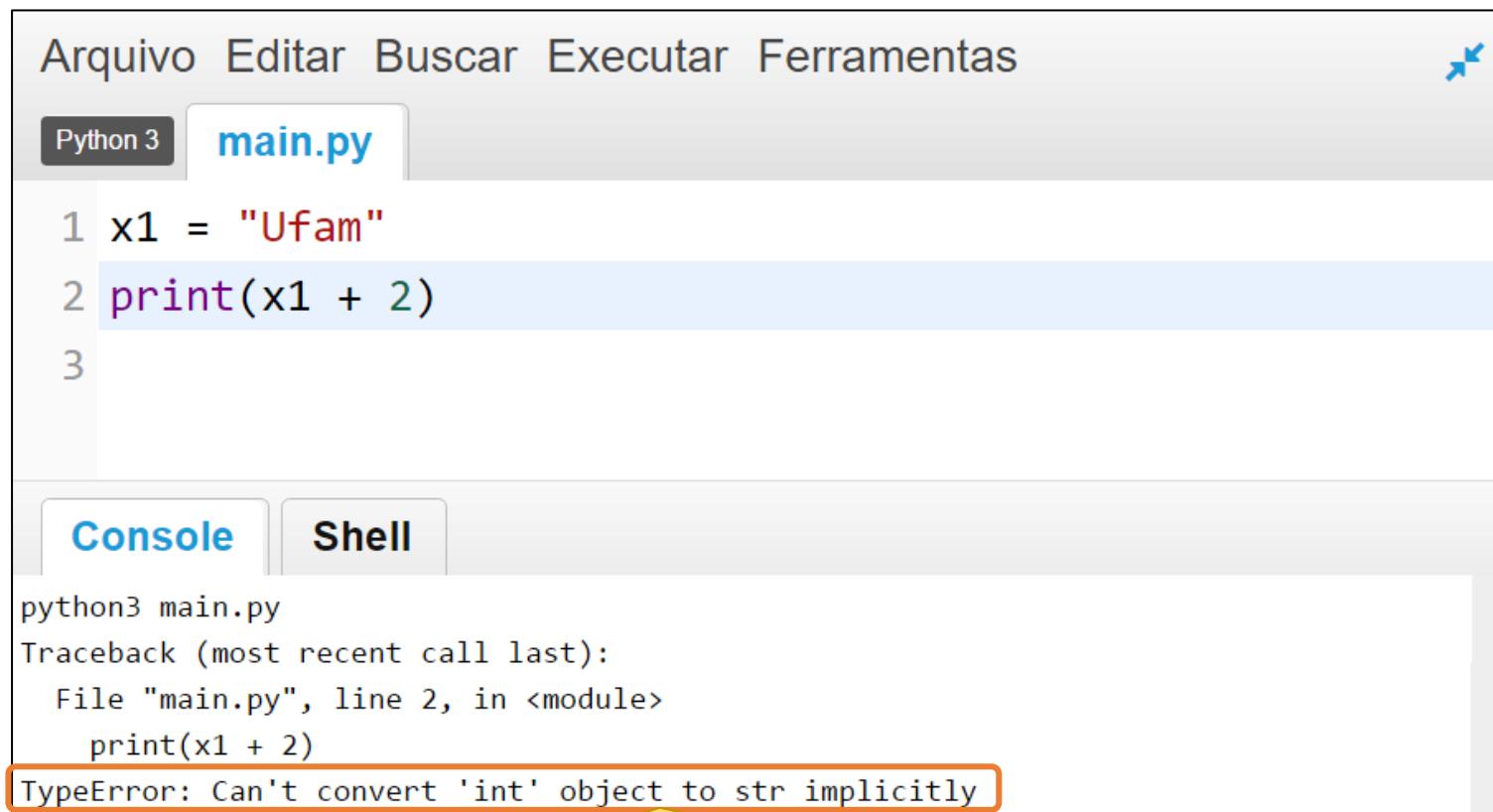
Console Shell

```
python3 main.py
Traceback (most recent call last):
  File "main.py", line 2, in <module>
    printt(x1)
NameError: name 'printt' is not defined
```

Natureza do erro: “Name Error” (erro no nome de funções ou variáveis)

Explicação: erro por usar uma função não existente.

Como interpretar mensagens de erro?



Arquivo Editar Buscar Executar Ferramentas

Python 3 main.py

```
1 x1 = "Ufam"
2 print(x1 + 2)
3
```

Console Shell

```
python3 main.py
Traceback (most recent call last):
  File "main.py", line 2, in <module>
    print(x1 + 2)
TypeError: Can't convert 'int' object to str implicitly
```

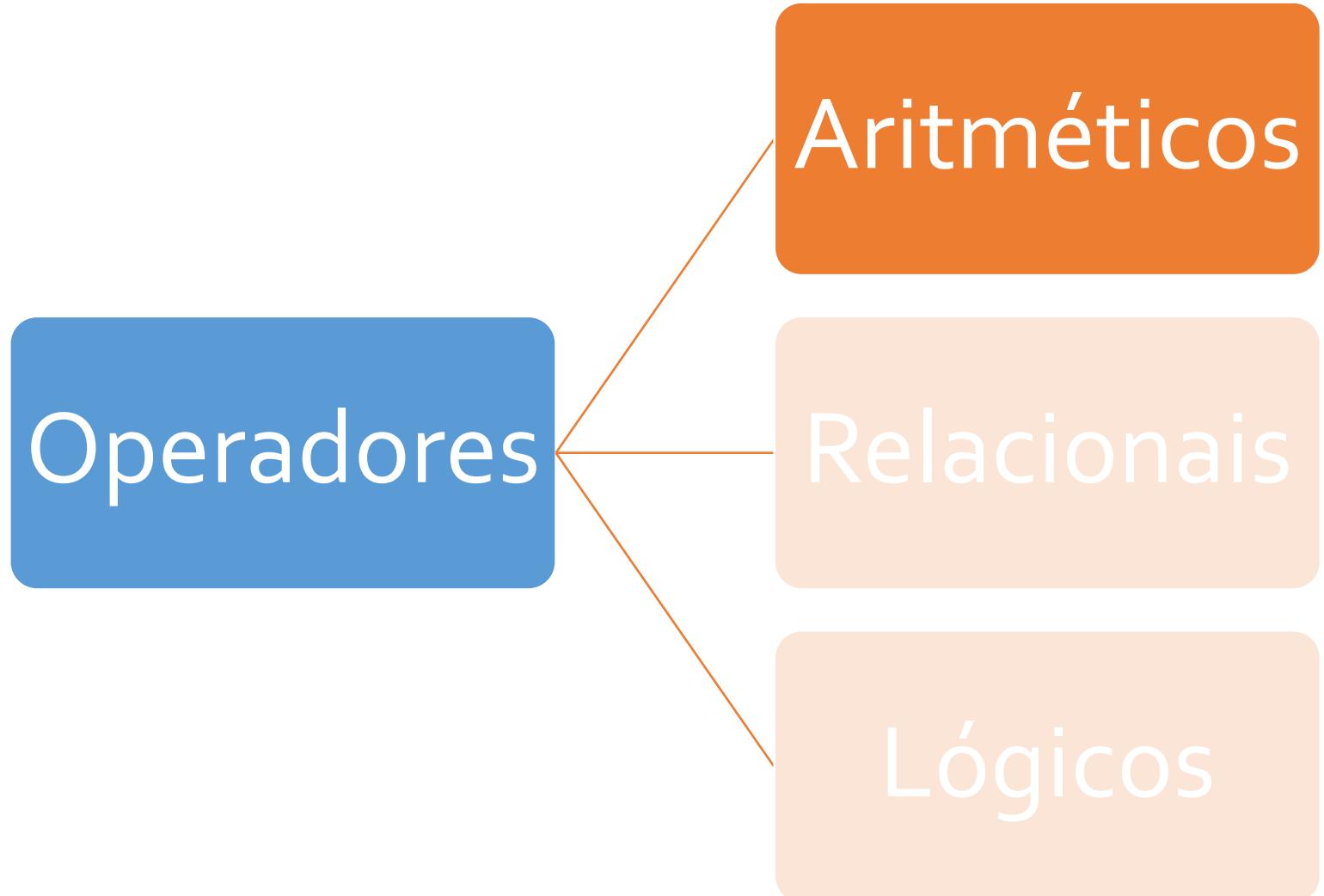
Natureza do erro: “Type Error” (erro de tipo de dado)

Explicação: erro por tentar somar um número com uma string.

Conteúdo



Tipos de operadores



Operadores Aritméticos

- 💡 São utilizados para realizar as operações aritméticas básicas.

Operador	Operação	Exemplo
+	Adição	$1 + y$
-	Subtração	$x - y$
*	Multiplicação	$8 * y$
/	Divisão real	$7 / 2 (= 3.5)$
//	Divisão inteira	$7 // 2 (= 3)$
%	Resto da divisão inteira	$7 \% 2 (= 1)$
**	Potenciação	$x ** 2$

Alguns editores de texto trocam o sinal de subtração por um travessão. Por isso, **digite** você mesmo o sinal de subtração, para evitar erros.



Operadores Aritméticos

- Entrada: 02 valores numéricos
- Saída: 01 valor numérico



Operadores Aritméticos

:: Formato

- Dois operandos devem estar **ligados** por um operador:



Exemplos:

$$\left[2a \right] \quad \times$$

$$\left[a^3 \right] \quad \times$$

$$\left[\sqrt{5} \right] \quad \times$$

$$\left[2 * a \right] \quad \checkmark$$

$$\left[a ** 3 \right] \quad \checkmark$$

$$\left[5 ** 0.5 \right] \quad \checkmark$$

Operadores Aritméticos

:: Precedência

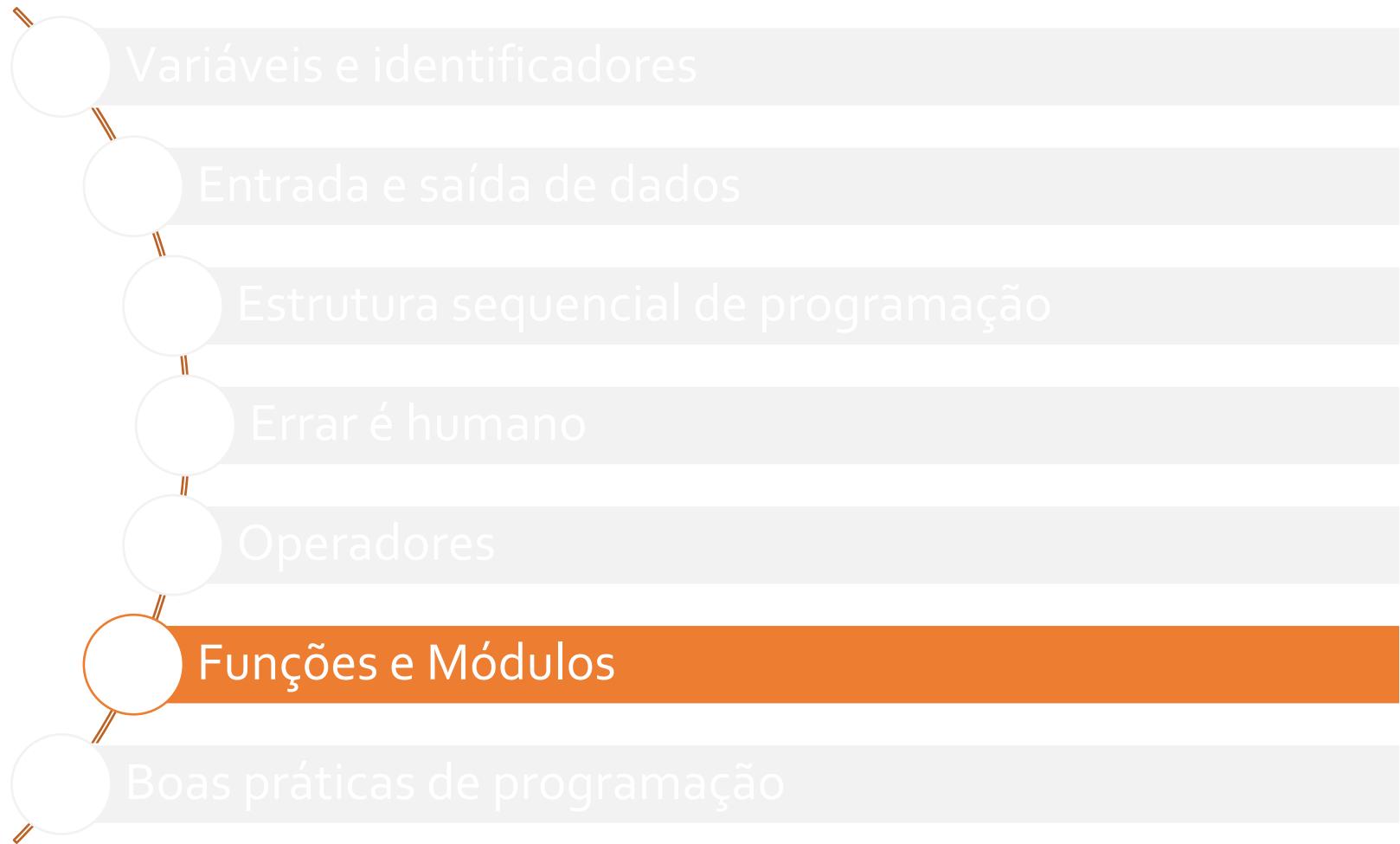
Precedência	Operador
1	Parênteses mais internos
2	Potenciação
3	Multiplicação, divisão, resto
4	Adição, subtração

Da esquerda
para a direita

Exemplos:	$4 * 3 ** 2$	$\rightarrow 36$
	$(4 * 3) ** 2$	$\rightarrow 144$
	$4 * 5 \% 3$	$\rightarrow 2$
	$4 * (5 \% 3)$	$\rightarrow 8$

Na dúvida, use
parênteses.

Conteúdo



Funções

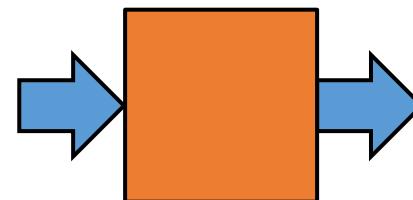
Na Matemática

- Relação entre dois conjuntos

$$f(x)$$

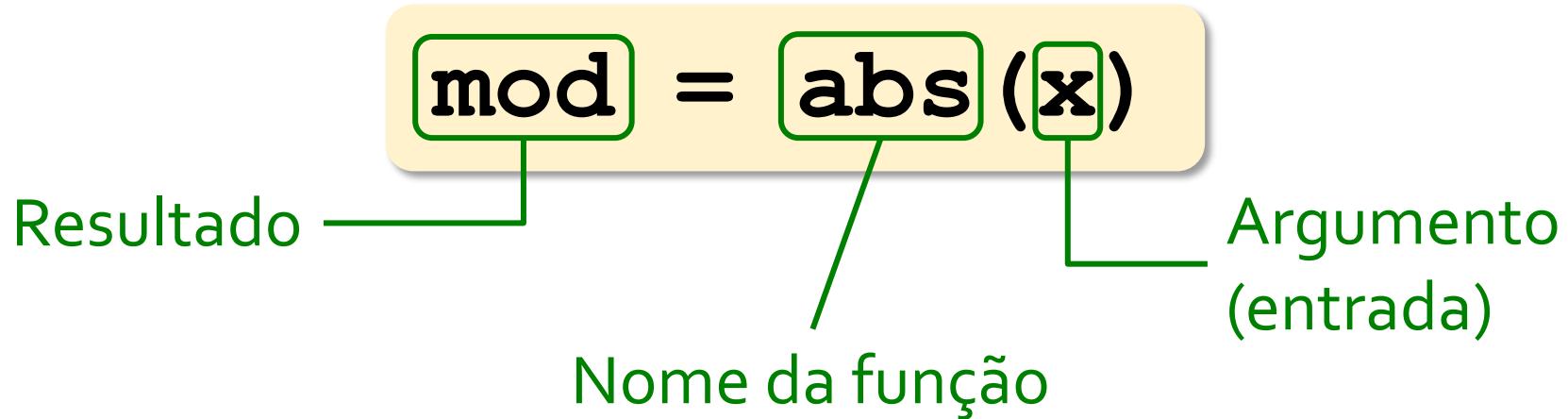
Em Programação

- Bloco de código que executa uma tarefa específica



Funções

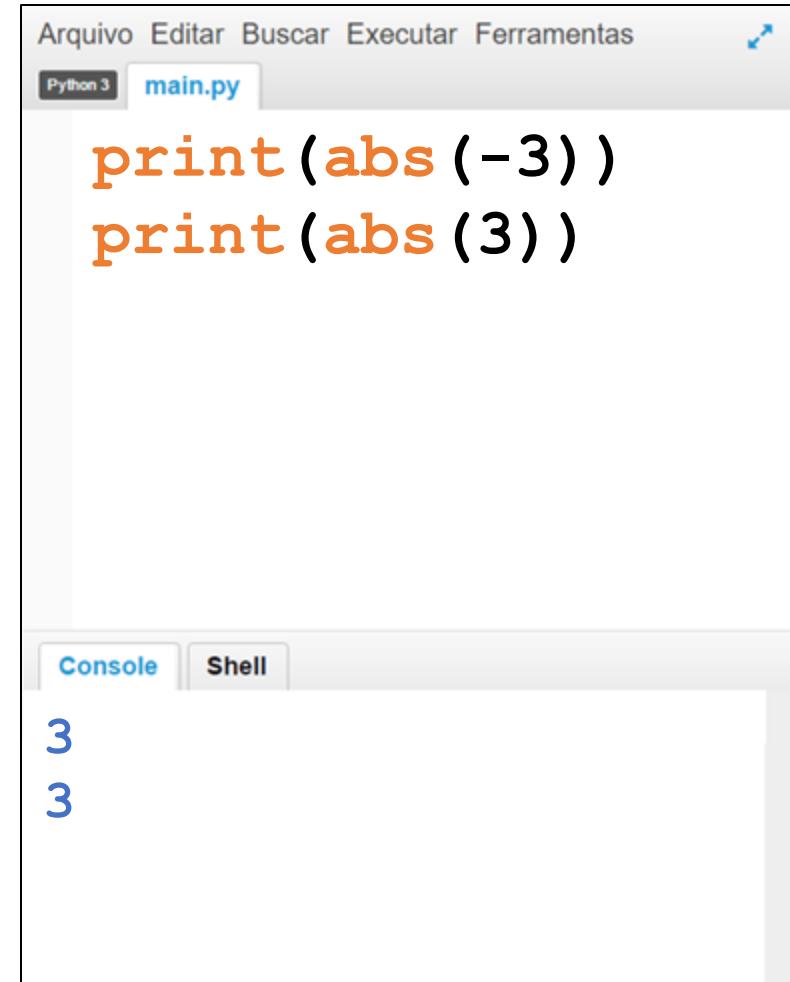
:: Nomenclatura



Funções

:: Valor Absoluto

- A função **abs (x)** fornece o valor absoluto de um número **x**.
- Em notação matemática, equivale a $|x|$, ou seja, a distância entre **x** e zero, na reta real.
- Possui **01** argumento.



The image shows a screenshot of a Python code editor. The menu bar includes 'Arquivo', 'Editar', 'Buscar', 'Executar', and 'Ferramentas'. The tab bar shows 'Python 3' and 'main.py'. The code in 'main.py' is:

```
print(abs(-3))
print(abs(3))
```

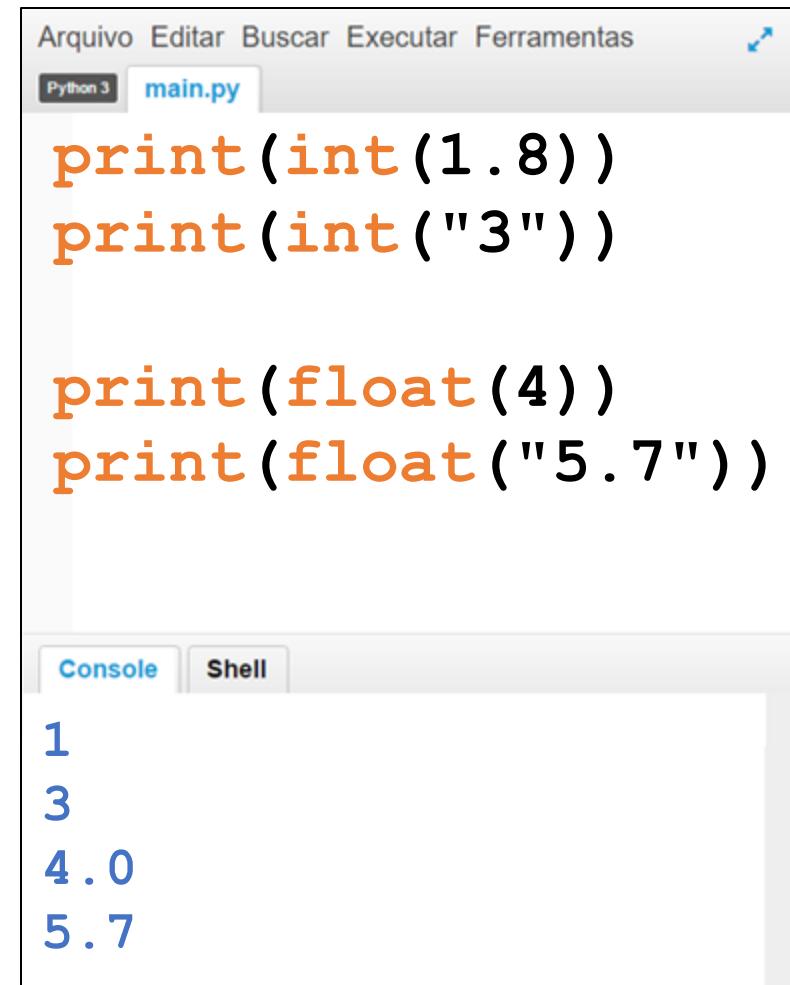
The 'Console' tab is active, showing the output:

```
3
3
```

Funções

:: Conversão de tipos

- A função **int(x)** converte um valor **x** (string ou real) em número inteiro.
- A função **float(y)** converte um valor **y** (string ou inteiro) em número real.
- Possuem **01** argumento.



The image shows a Python code editor and a terminal window. The code editor has a menu bar with 'Arquivo', 'Editar', 'Buscar', 'Executar', and 'Ferramentas'. A tab bar shows 'Python 3' and 'main.py'. The code in 'main.py' is:

```
print(int(1.8))
print(int("3"))

print(float(4))
print(float("5.7"))
```

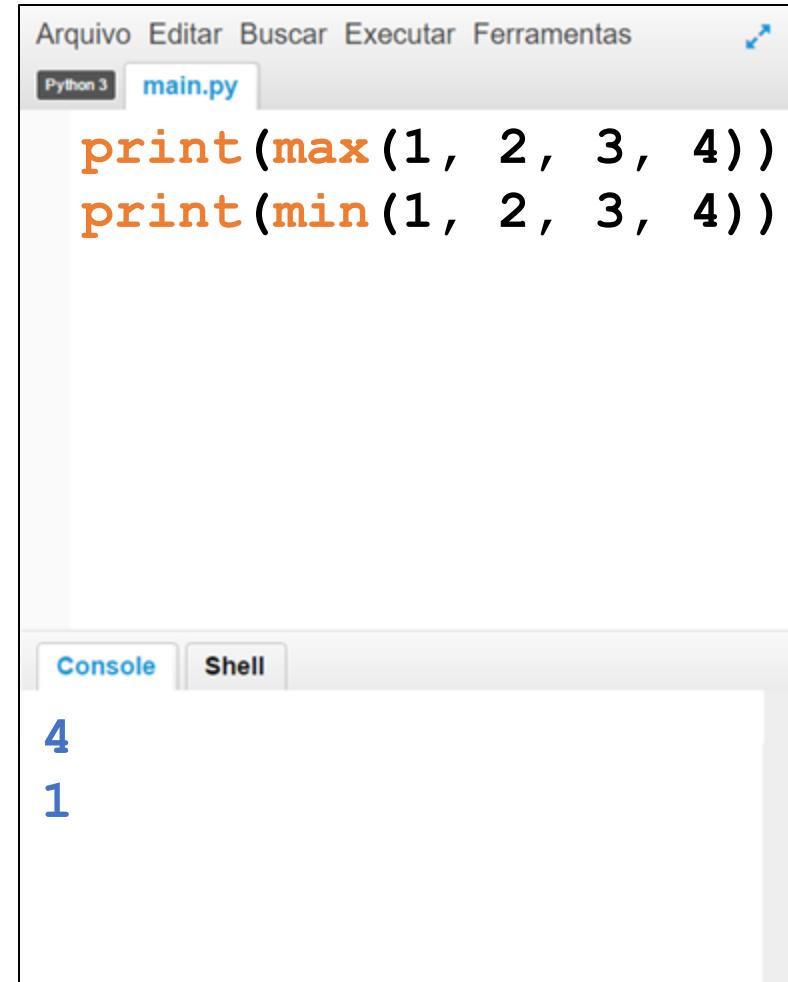
The terminal window below has tabs 'Console' and 'Shell'. The 'Console' tab is active, showing the output of the code:

```
1
3
4.0
5.7
```

Funções

:: Máximo e mínimo

- As funções **max()** e **min()** fornecem o maior e o menor valor entre uma lista de argumentos, respectivamente.
- Possuem quantidade variável de argumentos.



Arquivo Editar Buscar Executar Ferramentas

Python 3 main.py

```
print(max(1, 2, 3, 4))
print(min(1, 2, 3, 4))
```

Console Shell

4
1

Funções

:: Arredondamento

- 🛡 A função

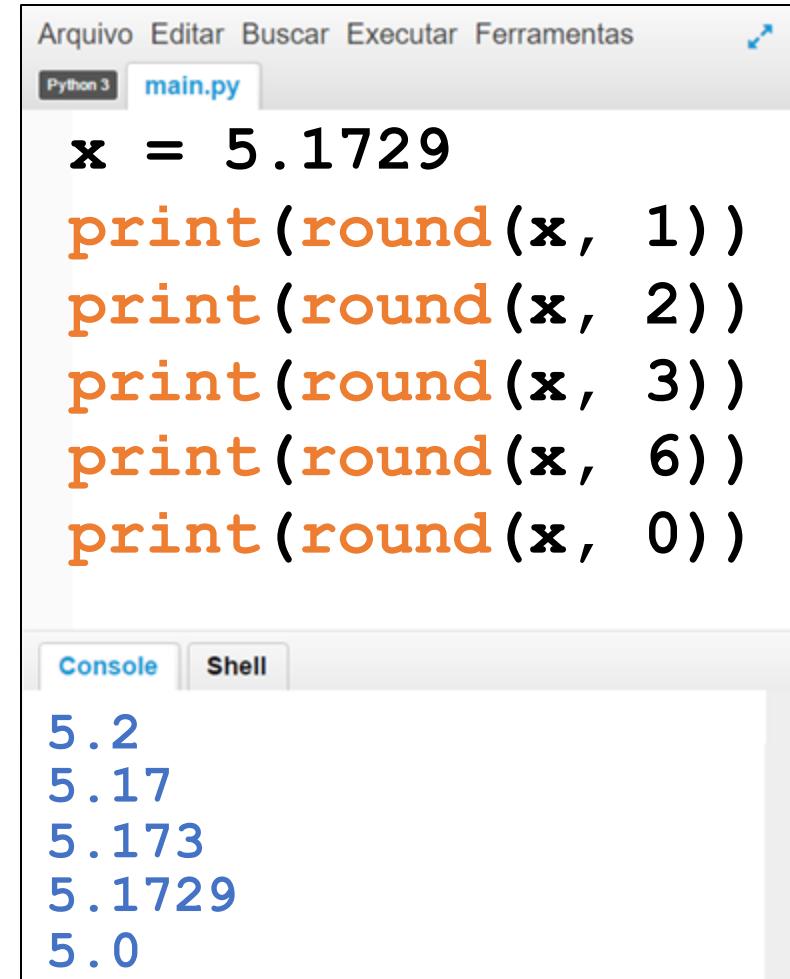
round(x, n)

aproxima um valor **x**
com **até n** casas
decimais.

- 🛡 Não completa com zeros
à direita.

- 🛡 Possui **02** argumentos.

- 🛡 O resultado é **float**.



The screenshot shows a Python code editor with a file named 'main.py' containing the following code:

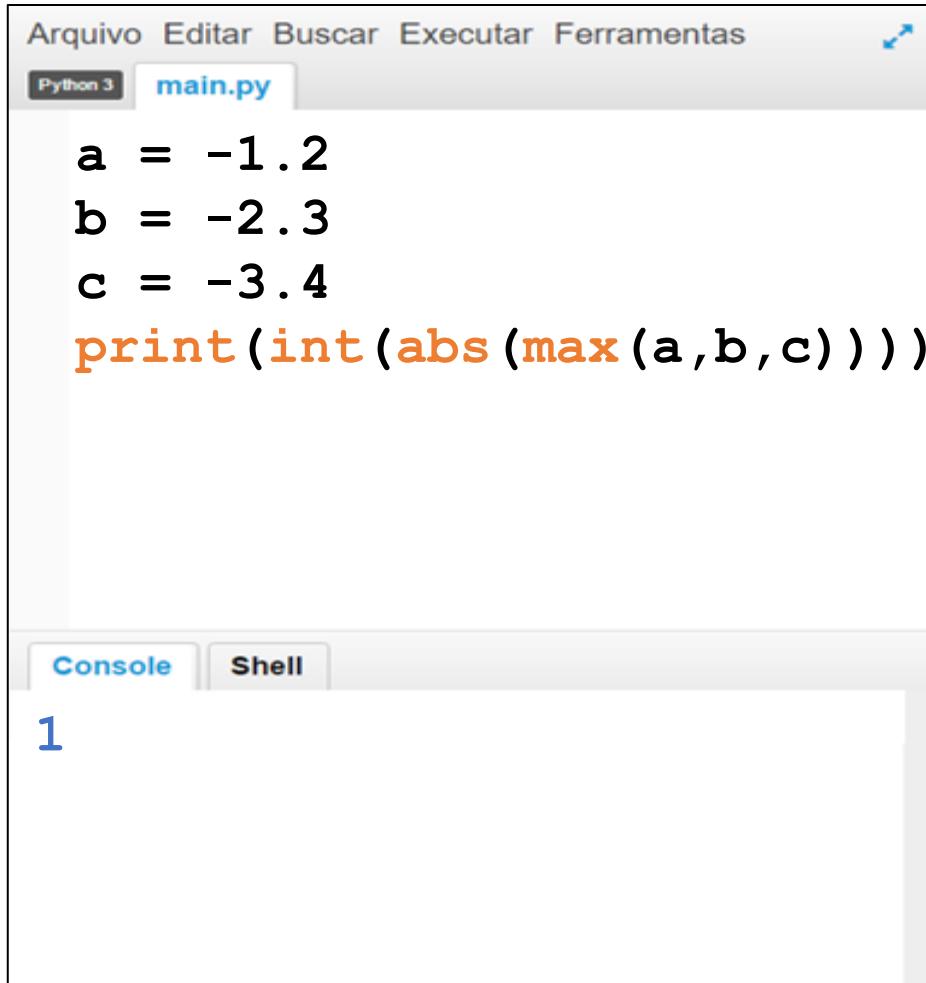
```
Arquivo Editar Buscar Executar Ferramentas
Python 3 main.py
x = 5.1729
print(round(x, 1))
print(round(x, 2))
print(round(x, 3))
print(round(x, 6))
print(round(x, 0))
```

The 'Console' tab is selected, showing the output of the code:

```
5.2
5.17
5.173
5.1729
5.0
```

Funções

:: Chamadas aninhadas



```
Arquivo Editar Buscar Executar Ferramentas
Python 3 main.py
a = -1.2
b = -2.3
c = -3.4
print(int(abs(max(a,b,c))))
```

Console Shell

1

- O **resultado** de uma função pode ser o **argumento** de outra função.
- Funcionamento análogo ao conceito de funções **compostas** na matemática:
$$g(f(x)) = (g \circ f)(x)$$

Módulos em Python

- 💡 O Python oferece poucas funções nativas, tais como **max ()** e **abs ()**.
- 💡 Outras funções são agrupadas em pacotes separados, conhecidos como **módulos**.
- 💡 O programador deve **importar** o módulo relacionado ao problema que deseja resolver.



Como importar um módulo Python?

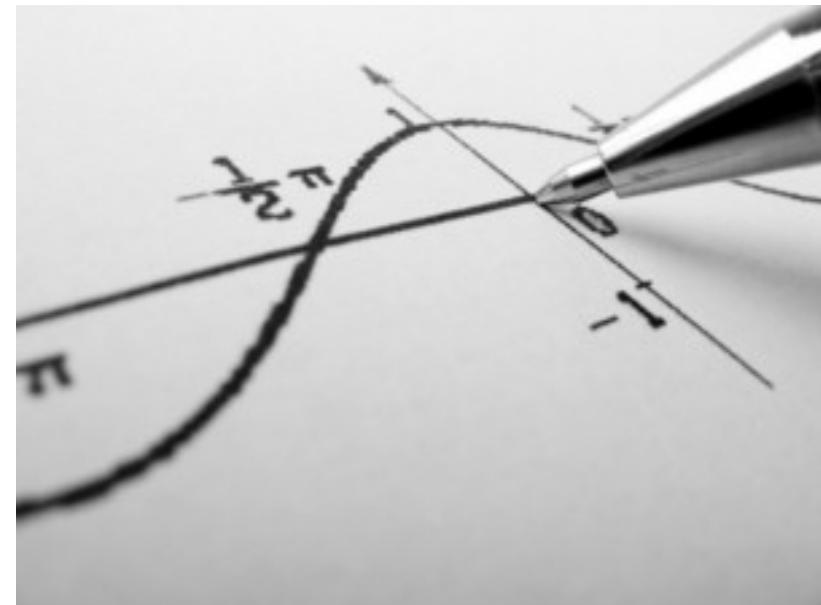
- Para importar as funções definidas em um módulo Python, use o seguinte comando:

```
from <nome_do_módulo> import *
```

Pode ser o **math**, o **numpy** ou outro que você necessitar.

Módulo **math**

- Contém diversas funções que podem ser usadas em cálculos matemáticos.



Módulo **math**

:: Funções matemáticas e constantes

exp (x)

- Calcula e^x

log (x)

- Logaritmo natural de x (base e)

log10 (x)

- Logaritmo de x na base 10

sqrt (x)

- Raiz quadrada de x

pi

- Valor da constante π

e

- Valor da constante neperiana

Módulo **math** :: Funções trigonométricas

sin (x)

- Calcula o seno de x

cos (x)

- Calcula o cosseno de x

tan (x)

- Calcula a tangente de x

asin (x)

- Calcula o arco-seno de x

acos (x)

- Calcula o arco-cosseno de x

atan (x)

- Calcula o arco-tangente de x

Entrada
deve ser
inserida
em
radianos

Saída é
dada em
radianos

Módulo **math** :: Funções trigonométricas

Funções trigonométricas do Python operam em radianos.

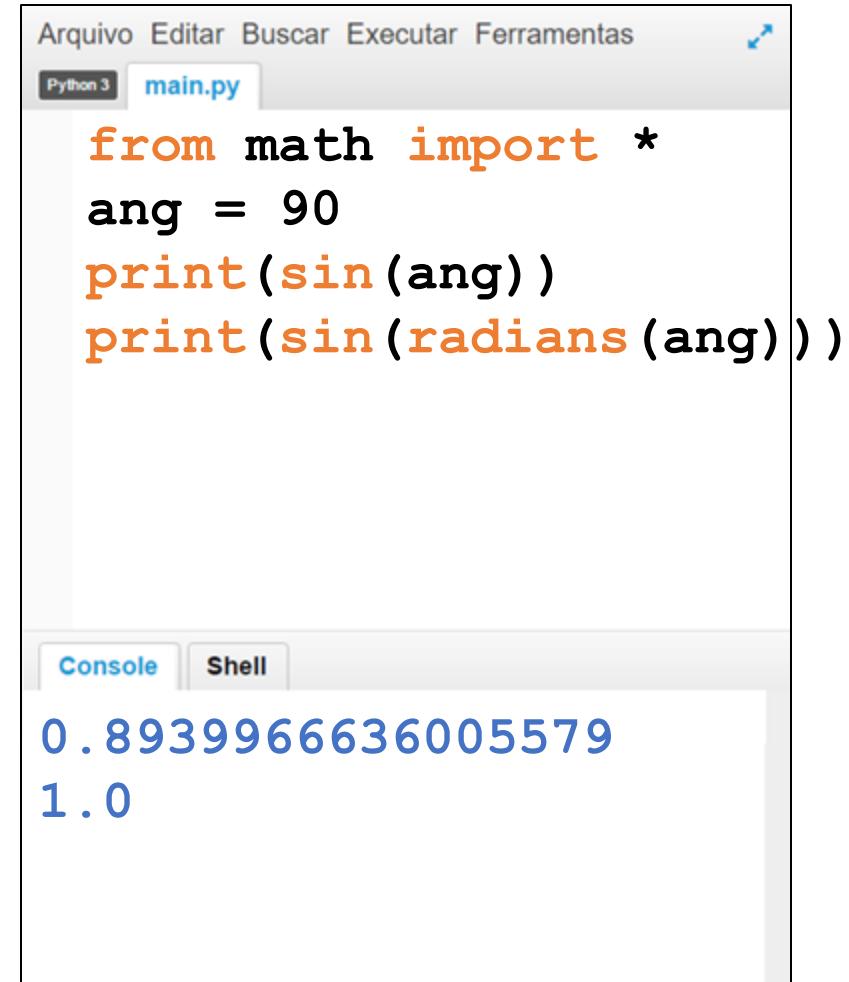
Funções úteis:

■ **radians ()**

- converte um ângulo de graus para radianos.

■ **degrees ()**

- converte um ângulo de radianos para graus.



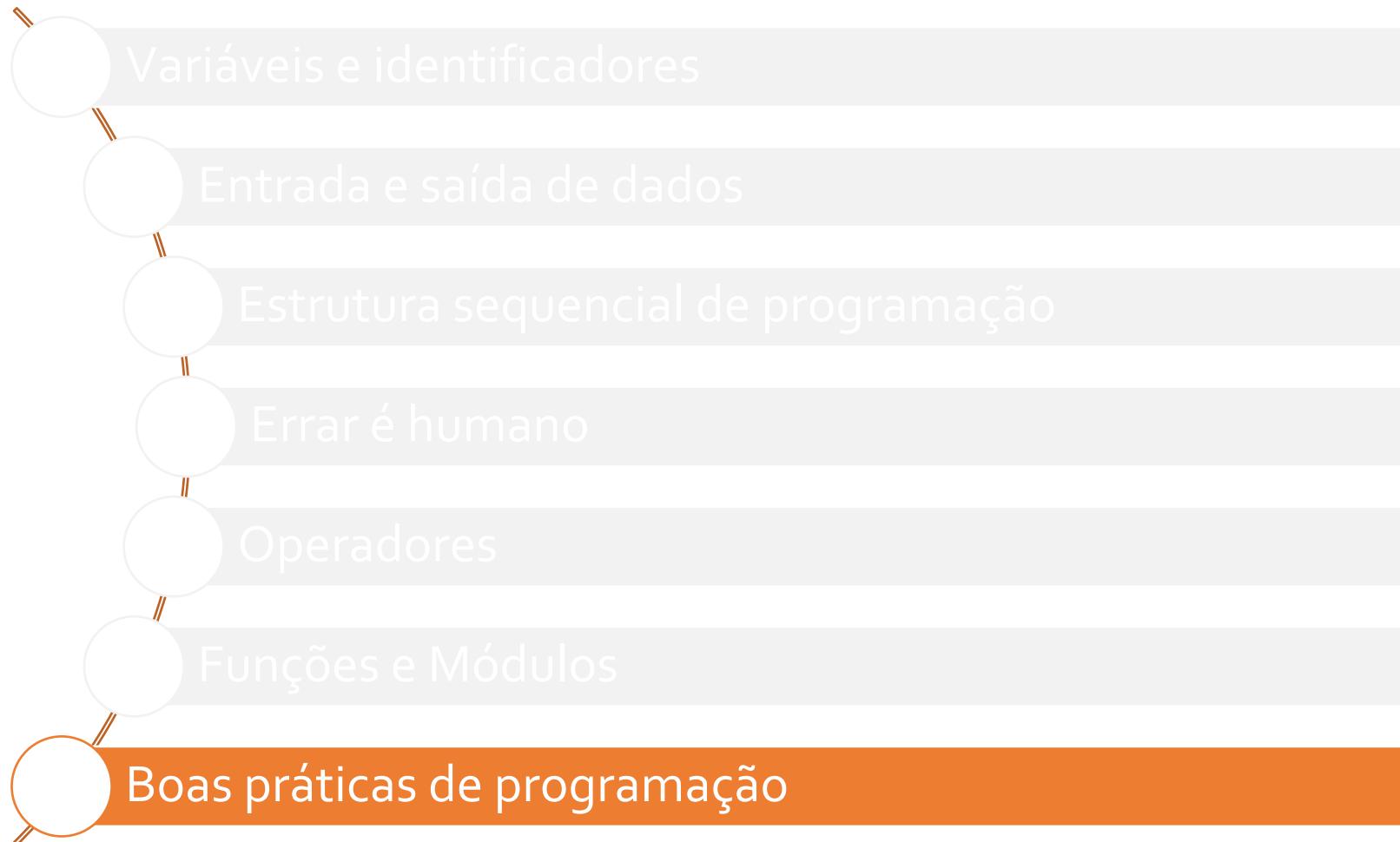
The image shows a screenshot of a Python code editor. The script file 'main.py' contains the following code:

```
from math import *
ang = 90
print(sin(ang))
print(sin(radians(ang)))
```

The 'Console' tab shows the output:

```
0.8939966636005579
1.0
```

Conteúdo



Boas práticas de programação

- 🛡 Códigos devem ser escritos para serem lidos por **seres humanos**.
- 🛡 Comentários ajudam a **entender** seu programa.
- 🛡 Os comentários são indicados por **#** e são **ignorados** pelo interpretador Python.



Boas práticas de programação

:: Comentários

- Escreva os comentários **no momento** em que estiver escrevendo o código.
- Os comentários devem **acrescentar** informação, e **não frasear** o comando:

```
# Multiplicacao de b por h:  
area = b * h
```



```
# Calculo da area do retangulo:  
area = b * h
```



Boas práticas de programação

:: Comentários

- Escreva um cabeçalho no início do script.
- Ajuda você a se lembrar mais tarde do que fez.

```
#-----  
# UNIVERSIDADE FEDERAL DO AMAZONAS  
# FULANO DA SILVA  
# DATA: 25/05/2017  
#  
# OBJETIVO: Calcular o volume de combustivel  
#           em um tanque cilindrico  
#-----
```

Boas práticas de programação

:: Identificadores

- Sempre use nomes descritivos e fáceis de lembrar:

`x = 1.3`



`raio = 2.2`



- Use sempre letras minúsculas em nomes de variáveis:

`raio = 1.3`

`Raio = 4.6`

`RAIO = 7.9`



`raio_interno = 1.3`

`raio_meio = 4.6`

`raio_externo = 7.9`



- Não utilize acentos nos nomes das variáveis. Pode funcionar em alguns sistemas, mas em outros, não.

`área = 1.3`



`area = 2.2`



Boas práticas de programação

:: Expressões

- 🛡 Use espaços em branco para melhorar a legibilidade.
- 🛡 Utilize **parênteses** para melhorar a compreensão e evitar erros, mesmo que não alterem a precedência.

```
H= (A**2+B**2) **0.5
```



```
H = ( (A ** 2) + (B ** 2) ) ** 0.5
```



Boas práticas de programação

- 🛡 Defina todas as variáveis no **início** de cada script, a fim de facilitar a manutenção do código.

```
nivel = 0.8      # nivel de combustivel (m)
altura = 2.3     # altura do tanque (m)
raio = 1.5       # raio da secao vertical (m)
volume = 0        # volume de combustivel (m3)
```

É como uma receita de bolo:

- Primeiro, **separe** os ingredientes
- Depois, prepare a massa

Referências bibliográficas



MENEZES, Nilo Ney Coutinho (2014). *Introdução à Programação com Python*, 2 ed. Editora Novatec.



HETLAND, Magnus Lie (2008). *Beginning Python: From Novice to Professional*. Springer eBooks, 2 ed. Disponível em: <http://dx.doi.org/10.1007/978-1-4302-0634-7>.

Dúvidas?

