



UNIDADE I

Introdução à Programação Estruturada

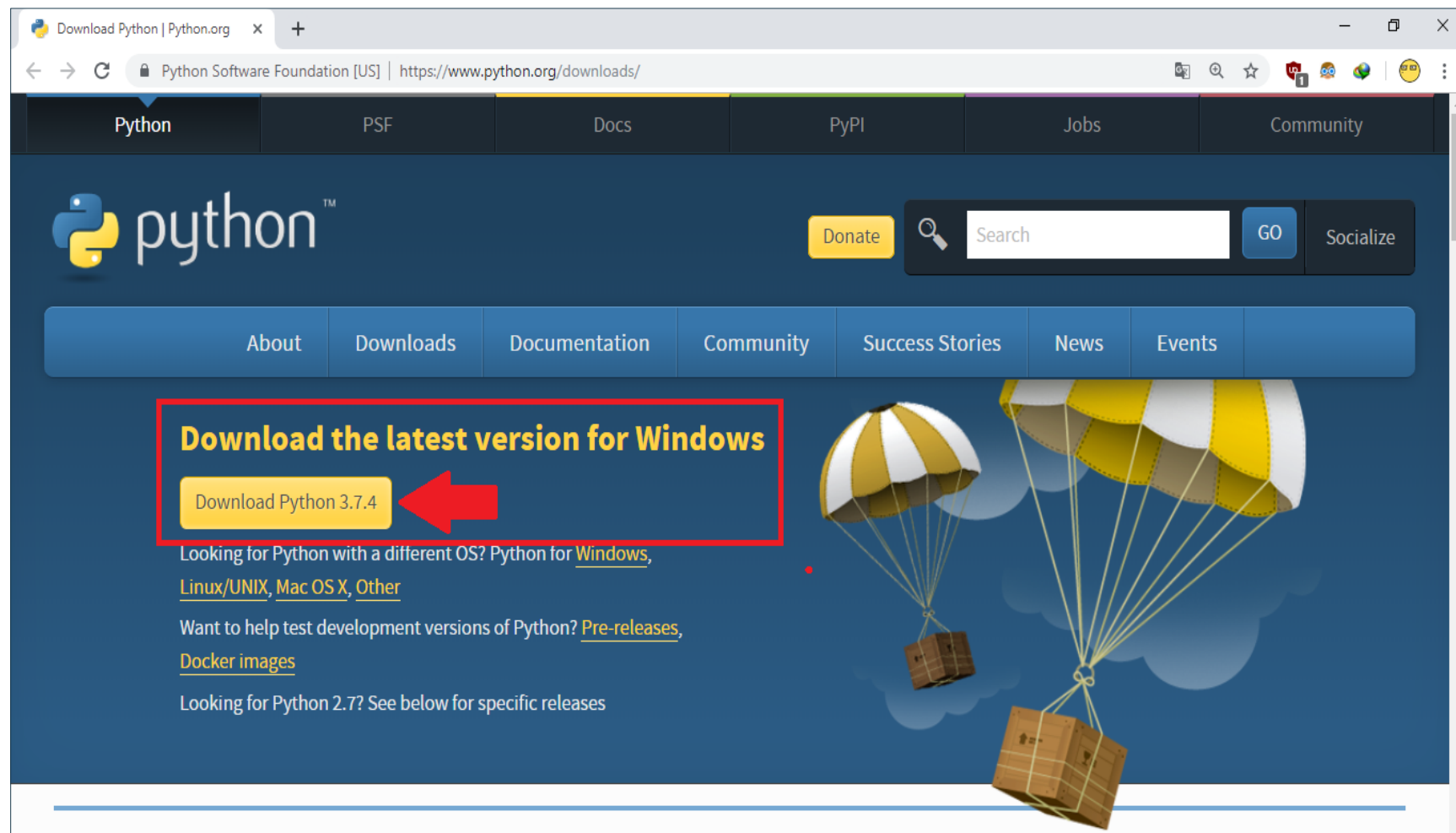
Prof. Me. Ricardo Veras

Instalação do Python

Para fazer o *download* do Python para Windows:

Entrar no *site* do Python:
<https://www.python.org>.

Clicar no *menu*:
Downloads.

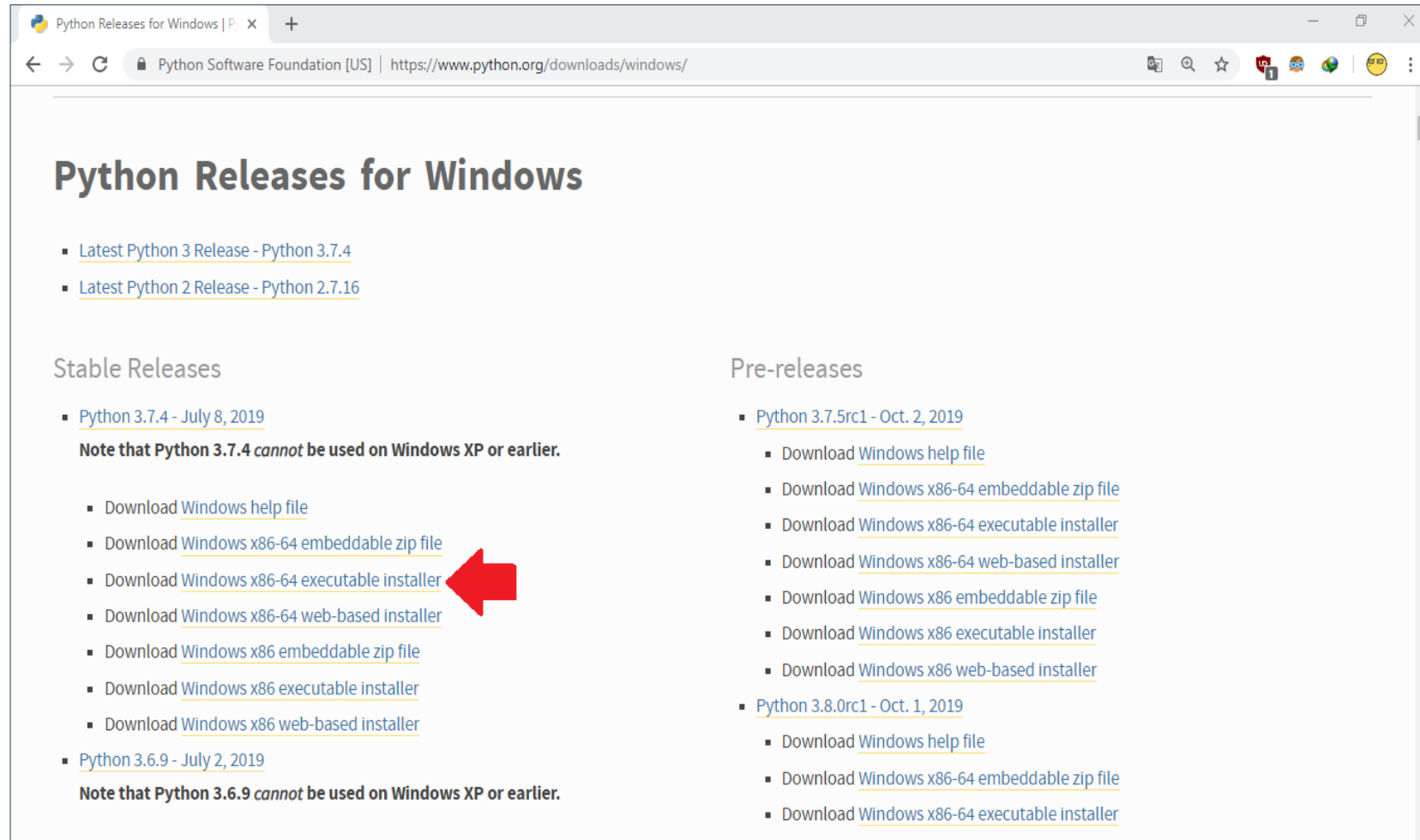


Fonte: <https://python.org.br/instalacao-windows/>

Instalação do Python

Download Python para Windows 32 bits ou 64 bits:

Caso queira instalar outras versões, basta clicar em *Downloads – Windows* da página inicial.



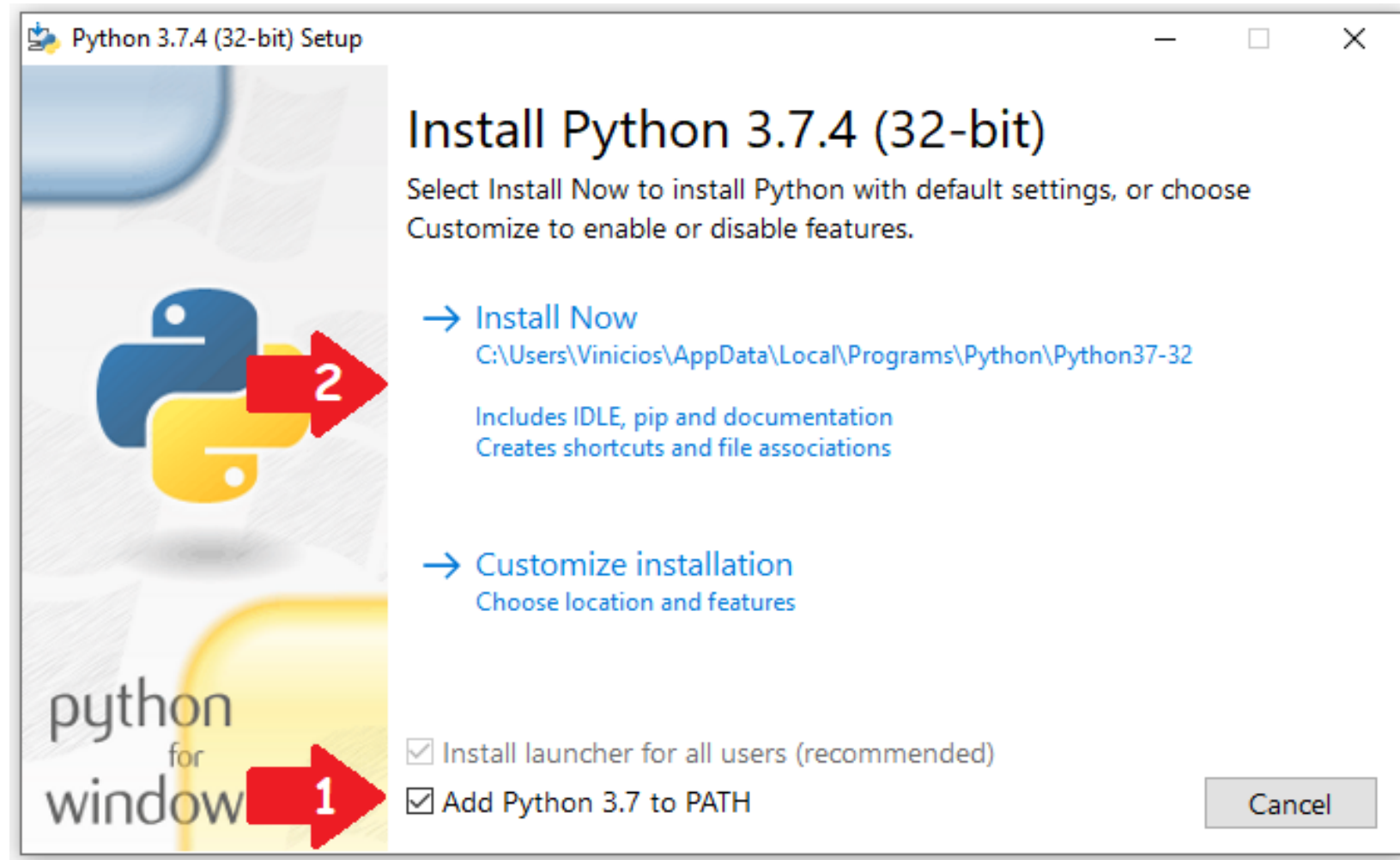
Fonte: <https://python.org.br/instalacao-windows/>

Instalação do Python

Para acionar o instalador do Python para Windows:

Após o *download*, dar um duplo clique no arquivo baixado para executá-lo:

1. Selecione a opção
“Add Python to PATH”;
2. Depois, clique em
“Install Now”.

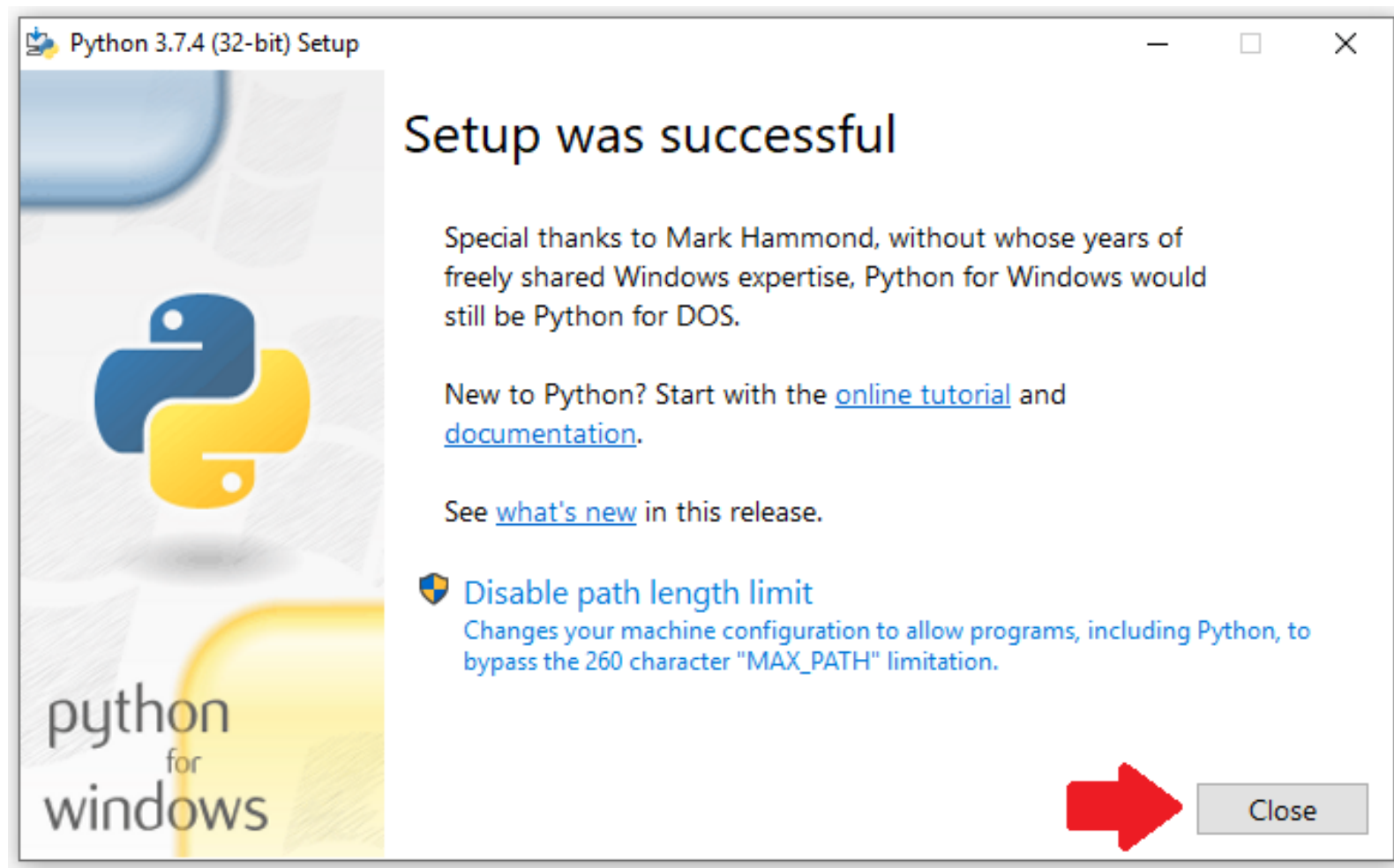


Instalação do Python

Instalação do Python para Windows concluída:

Deve-se aguardar a execução do instalador. Ao término do processo de instalação, a imagem ao lado aparece.

Com isso, Python estará instalado. Clicar em “Close” para encerrar.

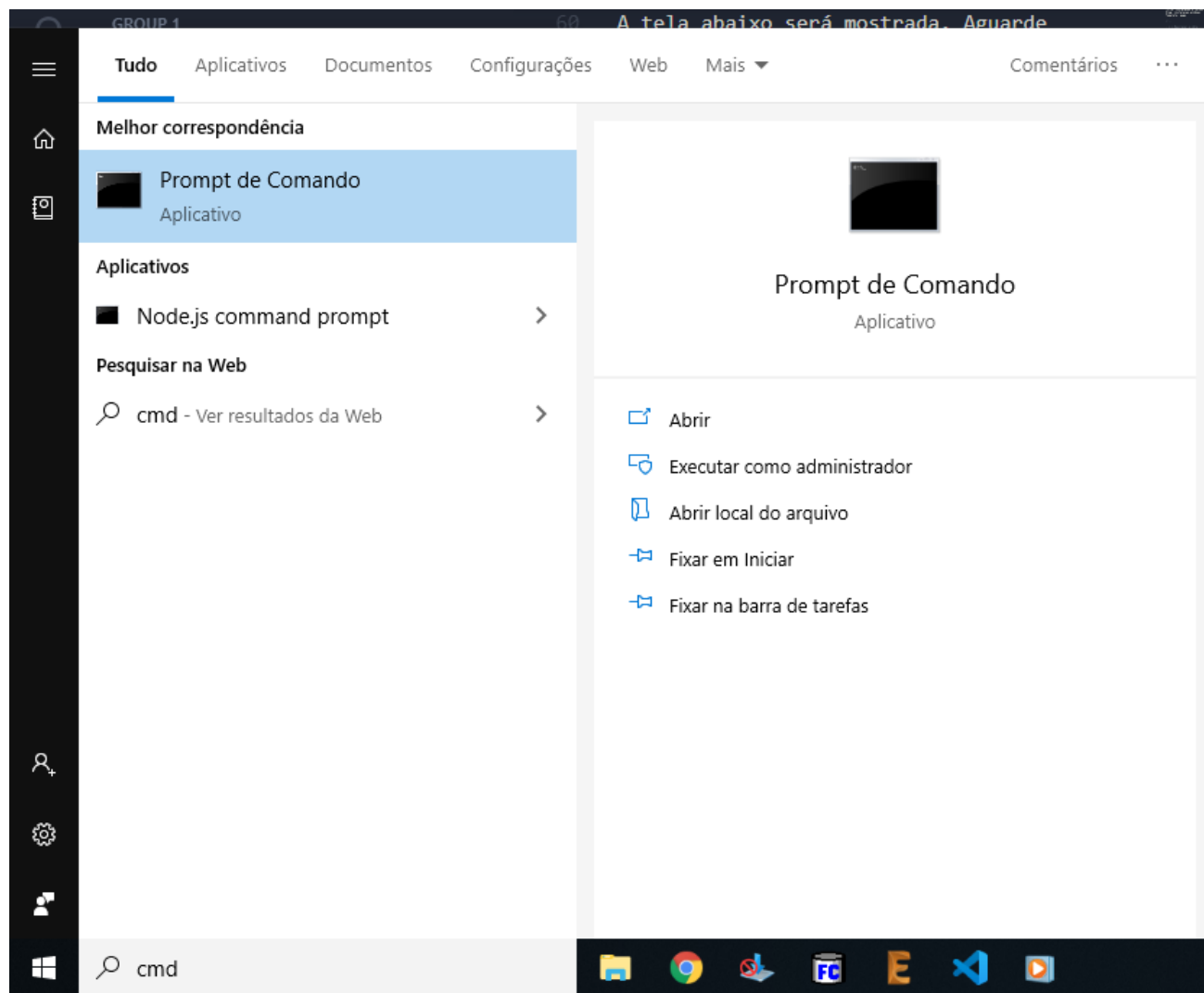


Fonte: <https://python.org.br/instalacao-windows/>

Instalação do Python

Abrindo o Prompt de Comando do Windows:

Para verificar se a instalação ocorreu com sucesso, abra o “Prompt de Comando”, digitando "cmd" na barra de pesquisa de arquivos.

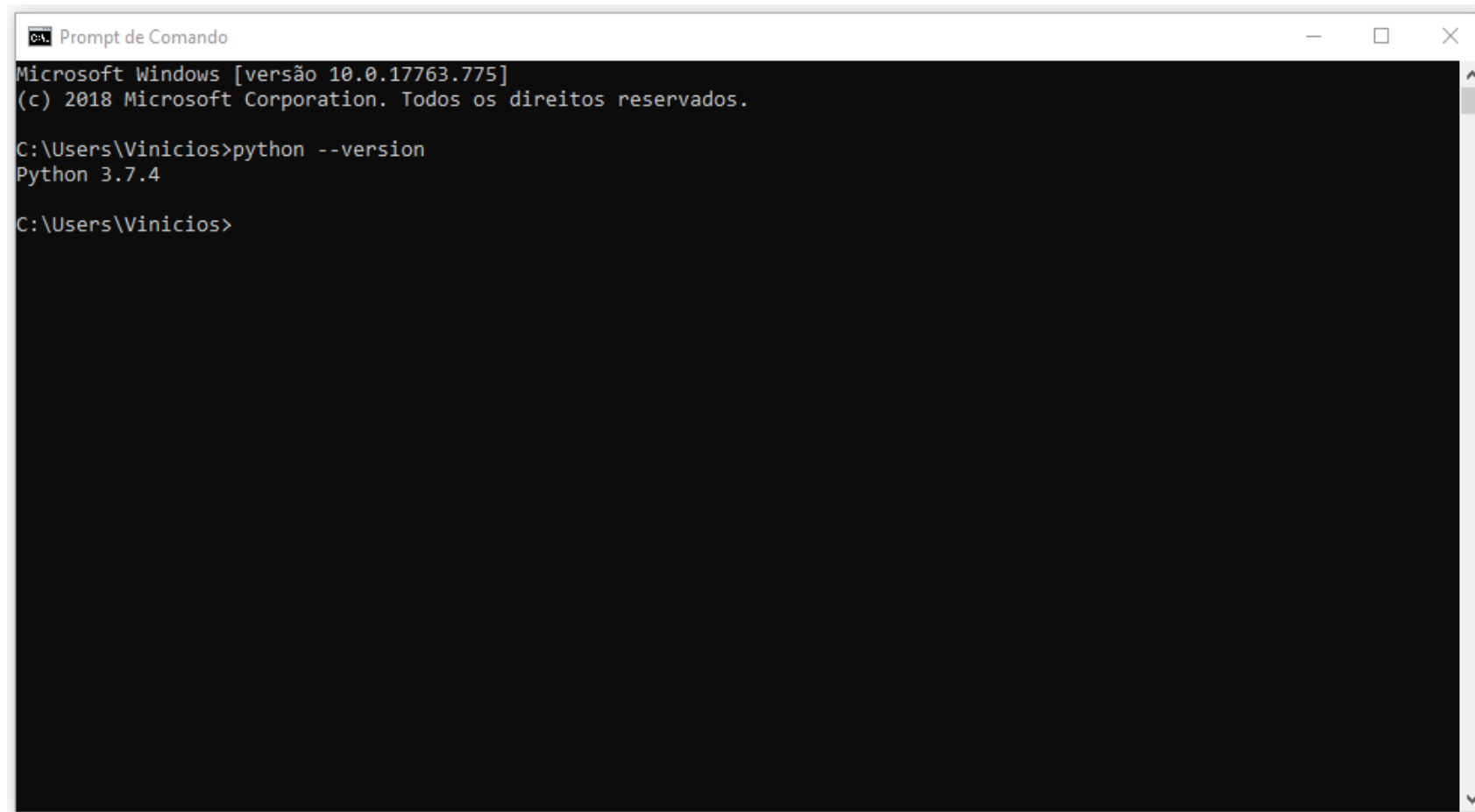


Instalação do Python

Verificando a instalação do Python no Windows:

Digite (no Prompt) o seguinte comando:

```
python --version
```



```
C:\> Prompt de Comando

Microsoft Windows [versão 10.0.17763.775]
(c) 2018 Microsoft Corporation. Todos os direitos reservados.

C:\Users\Vinicios>python --version
Python 3.7.4

C:\Users\Vinicios>
```

Fonte: <https://python.org.br/instalacao-windows/>

Instalação do Python em outras plataformas

Instalação do Python no MacOS X:

Primeiro, verifique se o micro já possui o Python instalado, abrindo o Terminal de Comandos do Mac OS.

1. Clique no ícone do *Spotlight* (um ícone de "Lupa" localizado no canto superior direito da tela). Obs.: pode-se abrir o *Spotlight* pressionando as teclas ⌘ Command + Espaço.
2. Digite "terminal" na caixa de pesquisa. Em seguida, o ícone do "Terminal" aparecerá nos resultados da busca.
3. Clique duas vezes sobre o ícone "Terminal" para abrir a ferramenta de Prompt de Comando.

Instalação do Python em outras plataformas

Instalação do Python no MacOS X (continuação):

4. Digite no terminal, de acordo com a versão instalada, os comandos:

- Para a versão 2 do Python:

```
$ which python
```

- Para a versão 3 do Python:

```
$ which python3
```

Se retornar algo como: `"/usr/bin/python"`, quer dizer que o Python já está instalado.

Do contrário, antes de instalar o Python, é necessário instalar o Xcode (que pode ser baixado na App Store), do pacote para "desenvolvimento em linha de comando" e dos "gerenciadores de pacotes" homebrew e pip.

Instalação do Python em outras plataformas

Instalação do Python no MacOS X (continuação):

Instalando o Xcode, o pip, o homebrew e, por fim, o Python:

- Para instalar as "ferramentas de linha de comando", digite no terminal:

```
$ xcode-select --install
```

- Para instalar o pip:

```
$ sudo easy_install pip
```

- Para atualizar o pip:

```
$ sudo pip install --upgrade pip
```

- Para instalar o homebrew:

```
$ ruby -e "$(curl -fsSL  
https://raw.githubusercontent.com/mxcl/homebrew/go)"
```

- Para instalar o Python 3:

```
$ brew install python3
```

Instalação do Python em outras plataformas

Instalação do Python no Linux:

Primeiro, verifique se o micro já possui o Python instalado, abrindo o Terminal de Comandos do Linux.

Pode-se abrir o Terminal de Comandos de duas formas: digitando a palavra "terminal" no campo de Busca de Aplicativos, ou teclando "ctrl+alt+t".

Digite no terminal, de acordo com a versão instalada, os comandos:

- Para a versão 2 do Python:

```
$ which python
```

- Para a versão 3 do Python:

```
$ which python3
```

Se retornar algo como: `"/usr/bin/python"`, quer dizer que o Python já está instalado. Do contrário, será necessário instalá-lo.

Instalação do Python em outras plataformas

Instalação do Python em Linux, por Gerenciadores de Pacotes:

Os gerenciadores de pacotes mais comuns são:

- apt-get (Debian, Ubuntu);
- yum (RedHat, CentOS).

Caso sua distribuição utilize um Gerenciador de Pacotes diferente, acesse a página de *downloads* do Python para mais informações.

Com apt-get:

- Para instalar o Python 3, digite no terminal:

```
$ sudo apt-get install python3
```

- Para instalar o gerenciador de pacotes Python pip:

```
$ sudo apt-get install python3-pip
```

Instalação do Python em outras plataformas

Instalação do Python em Linux, por Gerenciadores de Pacotes (continuação):

Com yum:

- Para instalar o Python 3, digite no terminal:

```
$ sudo yum install python3
```

- Para instalar o gerenciador de pacotes Python pip:

```
$ yum -y install python3-pip
```

Após a instalação do Python

Uma vez instalado o Python, seja qual for a plataforma de seu micro, abra um "Terminal de Comando" e digite:

```
python
```

Isso abrirá o Terminal do Python (com o prompt: ">>>")

Para testar um primeiro comando, digite:

```
>>> print("olá mundo!")
```

Interatividade

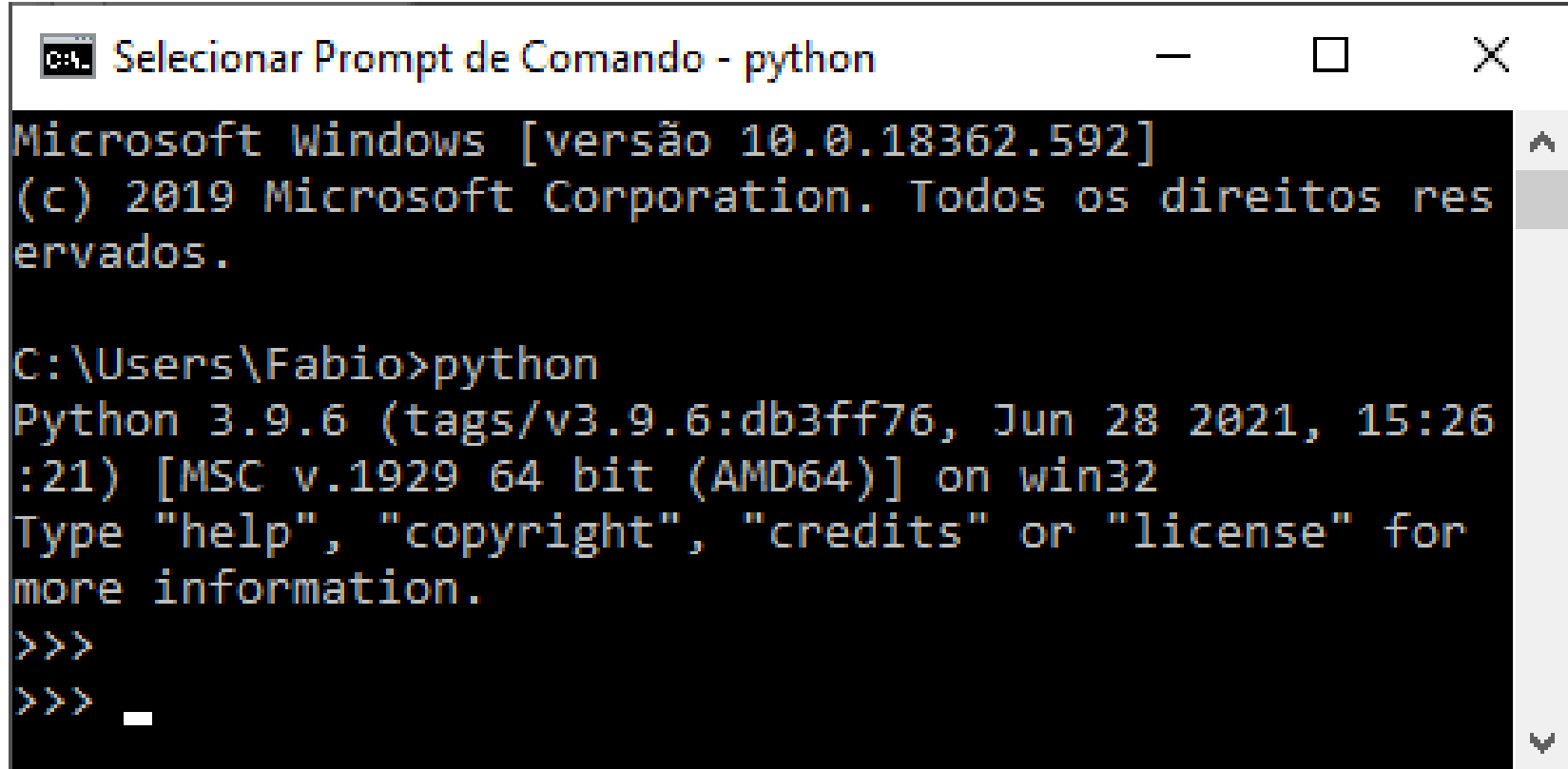
Qual é o símbolo do Prompt de Comando do "Terminal de Comandos" que aparece após acionarmos o Python?

- a) \$
- b) c:\>
- c) >>>
- d) python>
- e) >

Resposta

Qual é o símbolo do Prompt de Comando do "Terminal de Comandos" que aparece após acionarmos o Python?

- a) \$
- b) c:\>
- c) >>>
- d) python>
- e) >



```
Selecionar Prompt de Comando - python
Microsoft Windows [versão 10.0.18362.592]
(c) 2019 Microsoft Corporation. Todos os direitos reservados.

C:\Users\Fabio>python
Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> _
```

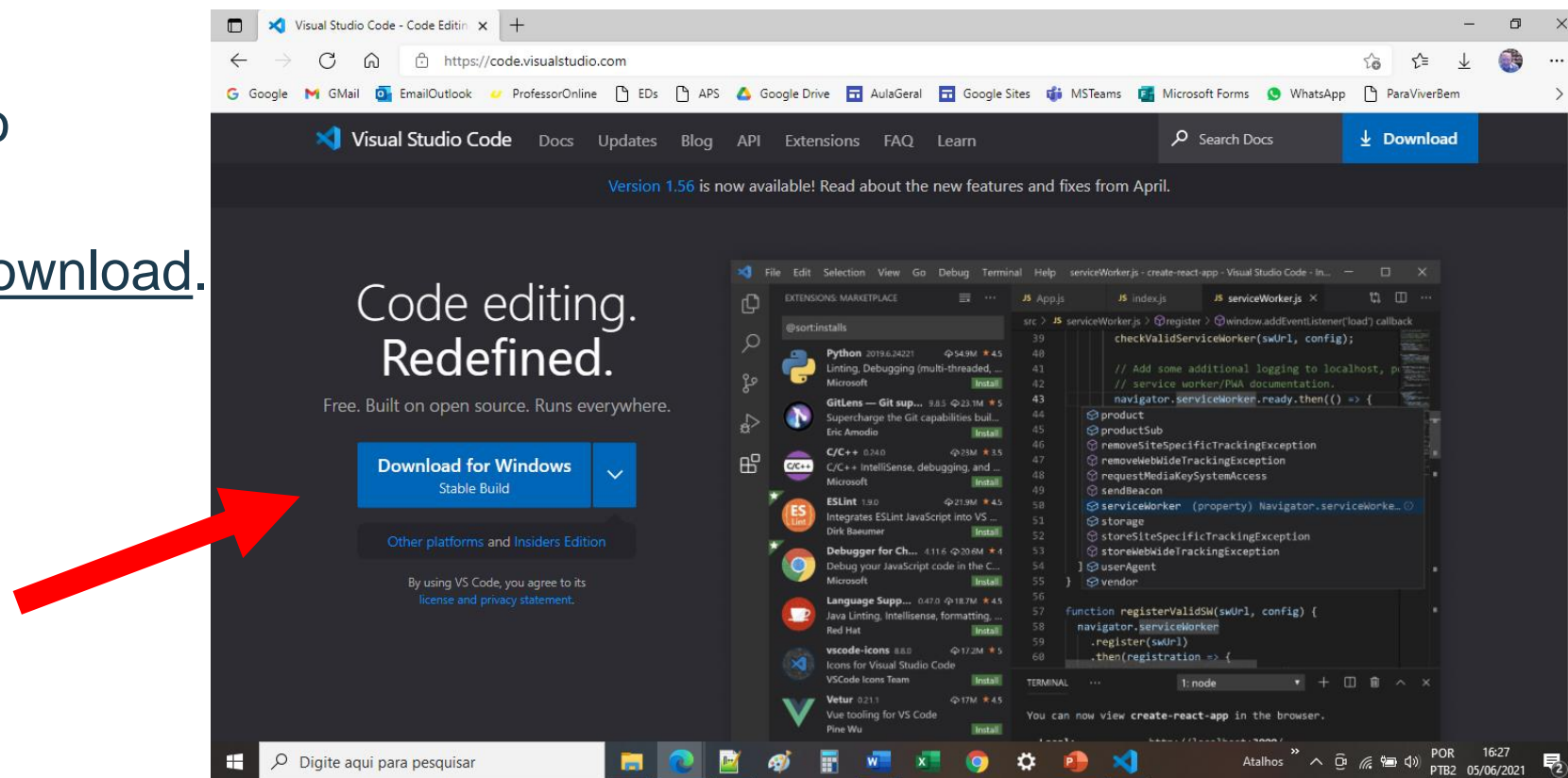
Fonte: Autoria própria.

Ambiente de desenvolvimento: Visual Studio Code (VSCode)

Além do Terminal do Python (*shell*), e do Python IDLE, pode-se utilizar uma IDE (*Integrated Development Environment* ou Ambiente de Desenvolvimento Integrado) para programar em Python.

O VSCode pode ser baixado no seguinte *link*:

<https://code.visualstudio.com/download>.

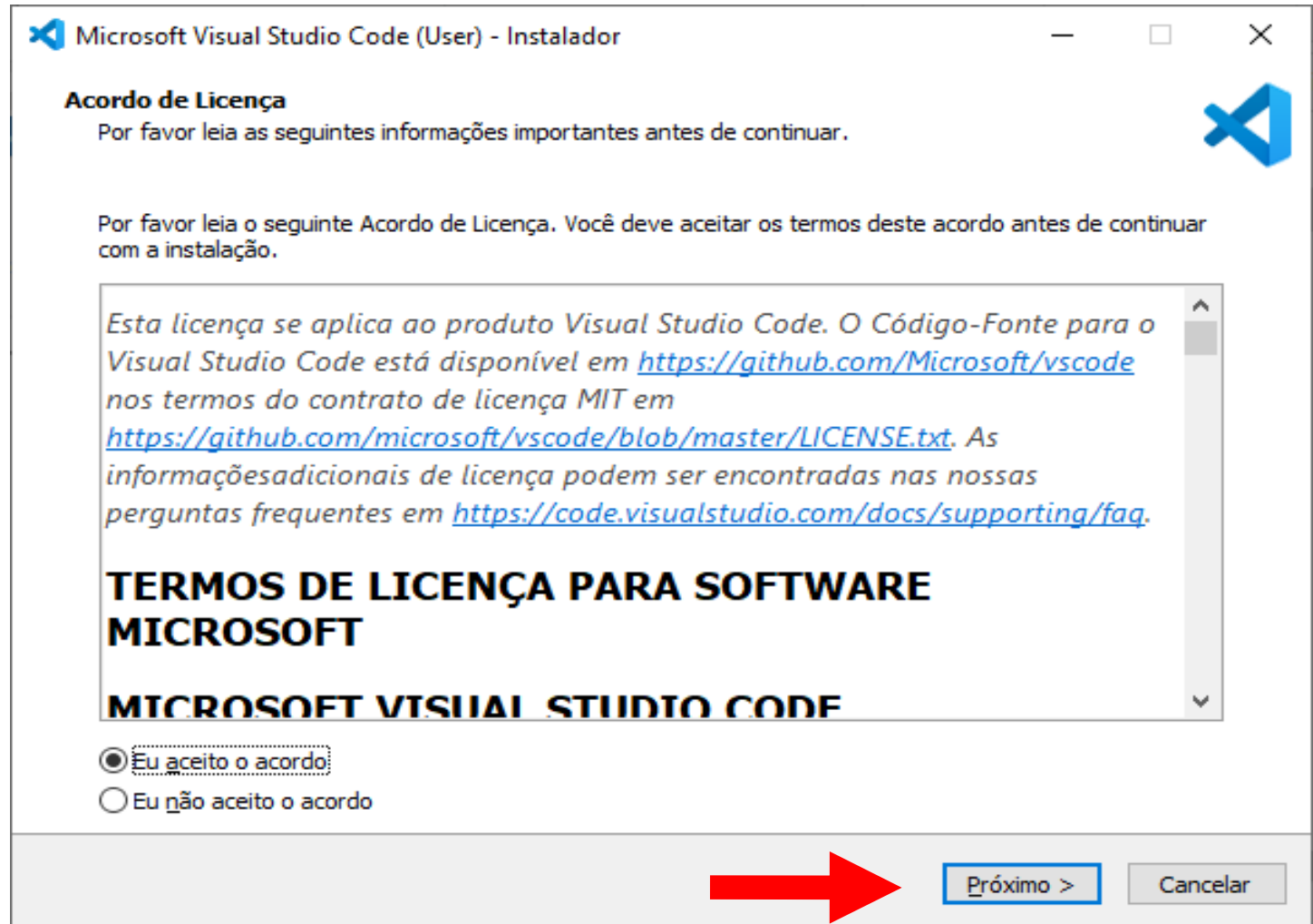


Fonte: <https://code.visualstudio.com/>. Acesso em: 15 jul. 2021.

Ambiente de desenvolvimento: Visual Studio Code (VSCode)

Após o *download*, execute o arquivo baixado (dando um duplo clique sobre ele).

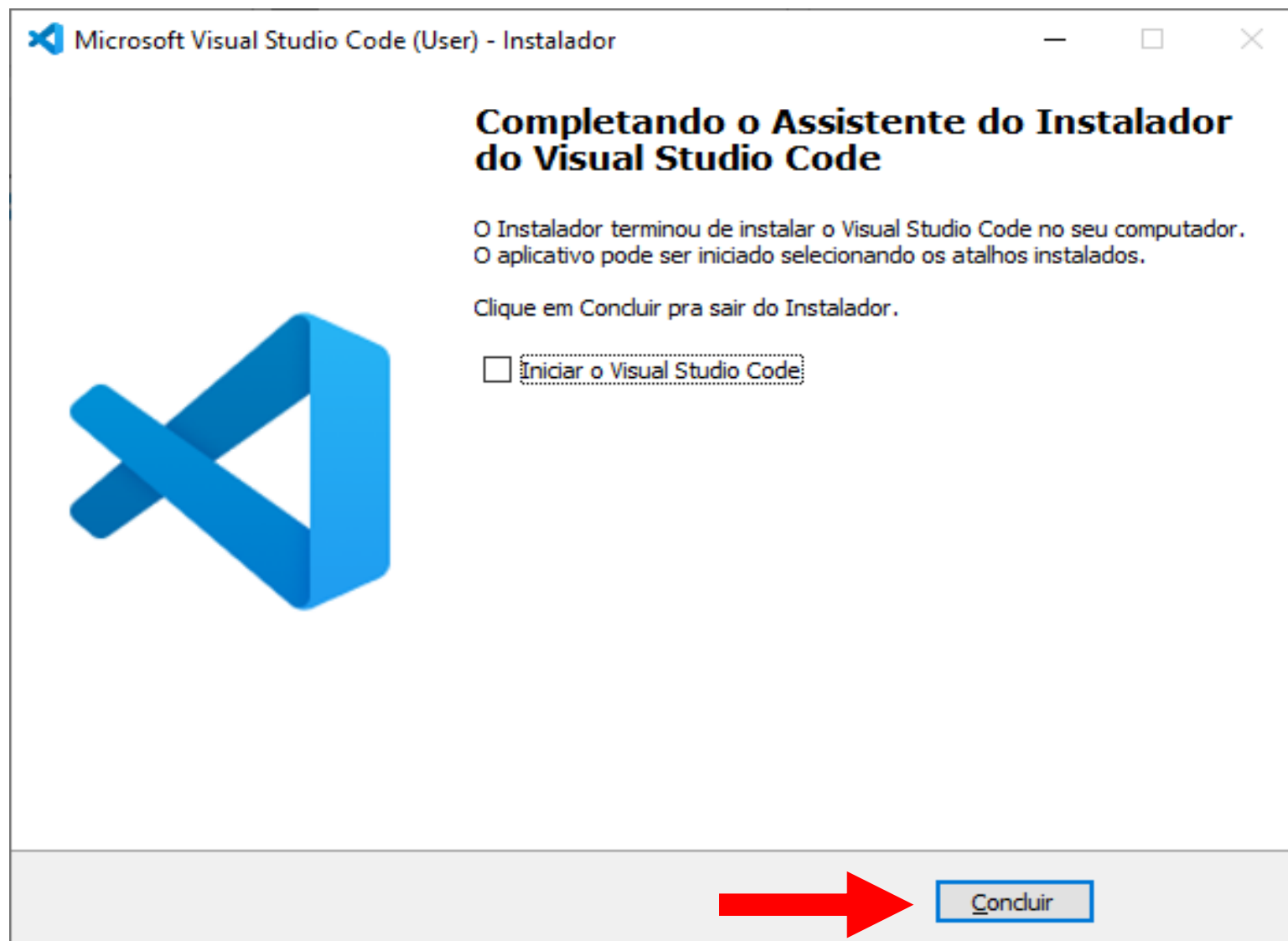
Aceite o Acordo de Licença e, a partir de então, clique sempre no botão "Próximo >".



Fonte: Autoria própria.

Ambiente de desenvolvimento: Visual Studio Code (VSCode)

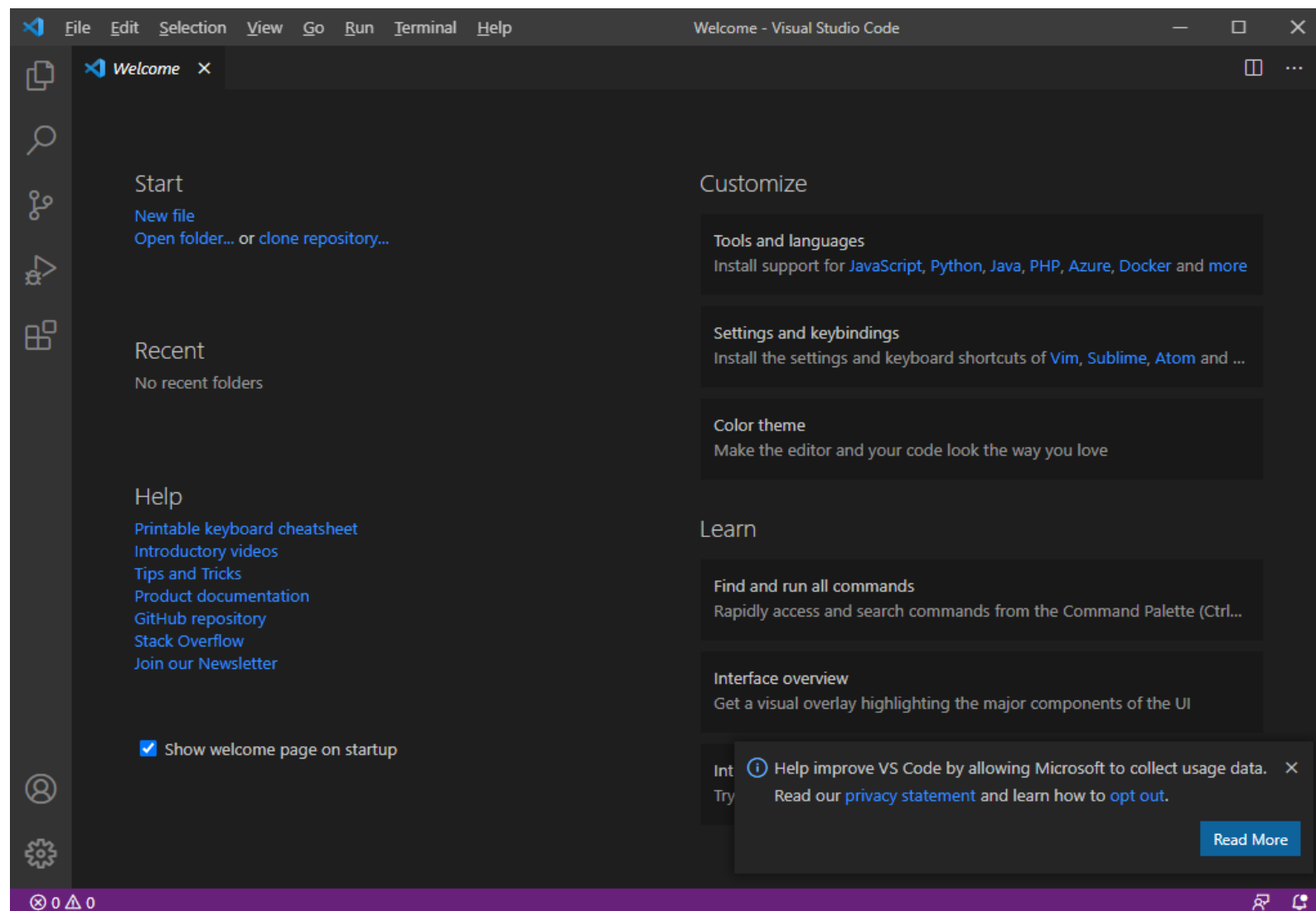
Ao final da instalação, aparecerá uma tela como esta, na qual deve-se clicar em "Concluir".



Fonte: Autoria própria.

Ambiente de desenvolvimento: Visual Studio Code (VSCode)

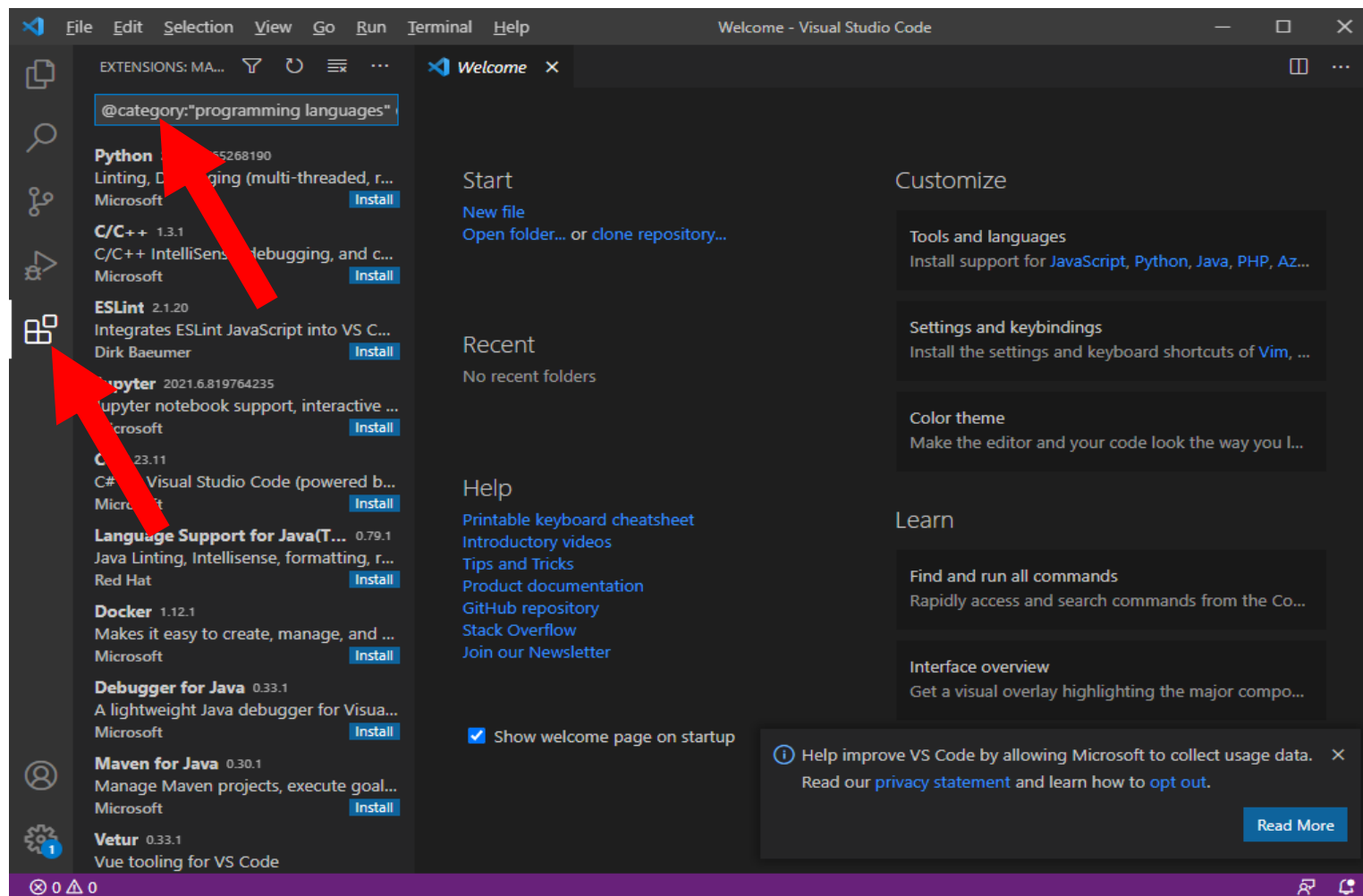
Após a finalização da instalação, acione o Visual Studio Code a partir do *menu* "Iniciar" do Windows.



Fonte: Autoria própria.

Ambiente de desenvolvimento: Visual Studio Code (VSCode)

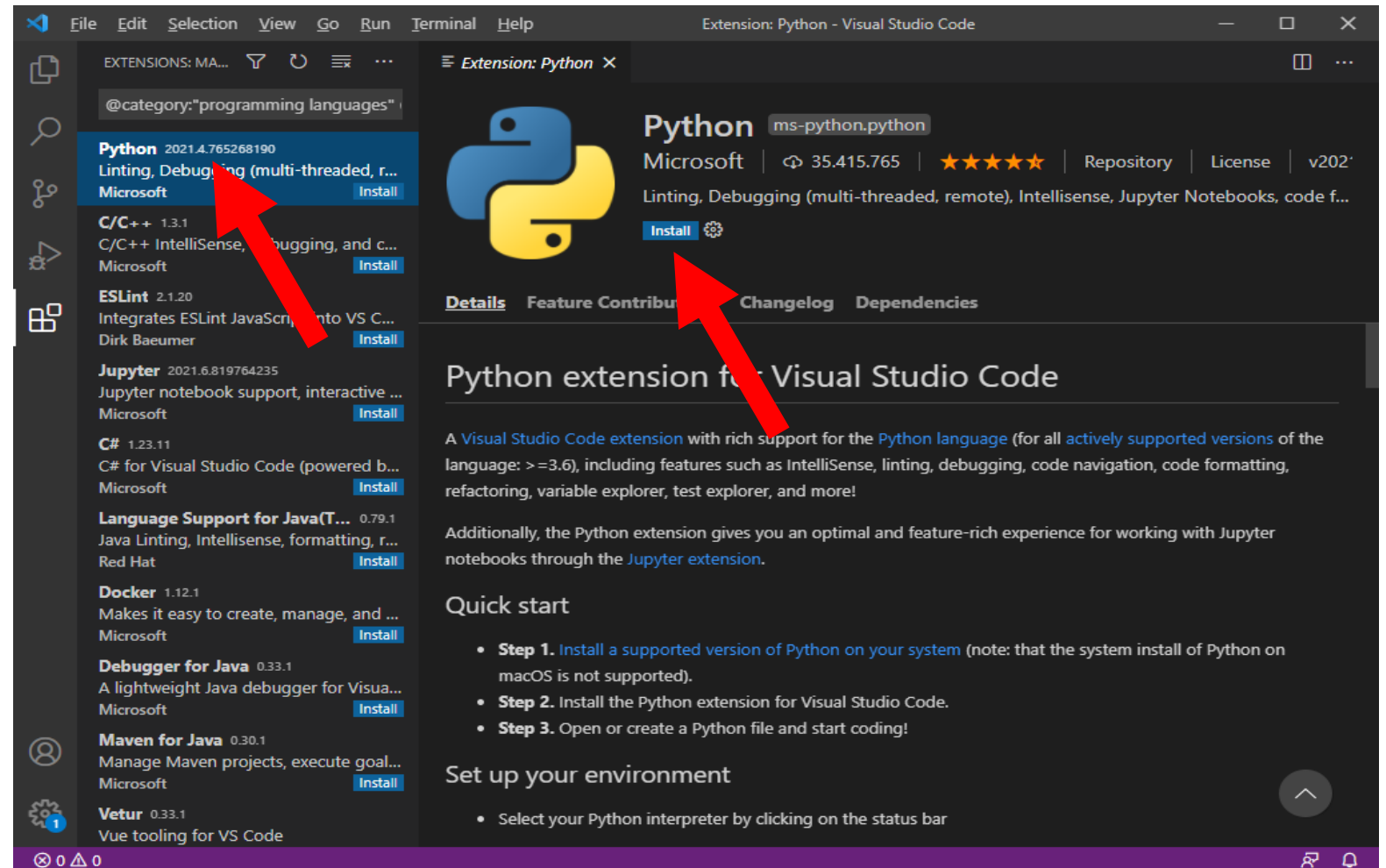
Algumas configurações iniciais são necessárias. Clique no ícone de "Extensions" (na lateral esquerda) e digite a palavra "python" no campo de busca que aparece no canto superior esquerdo.



Ambiente de desenvolvimento: Visual Studio Code (VSCode)

Selecione a 1ª opção de Python que aparecer e clique em "install".

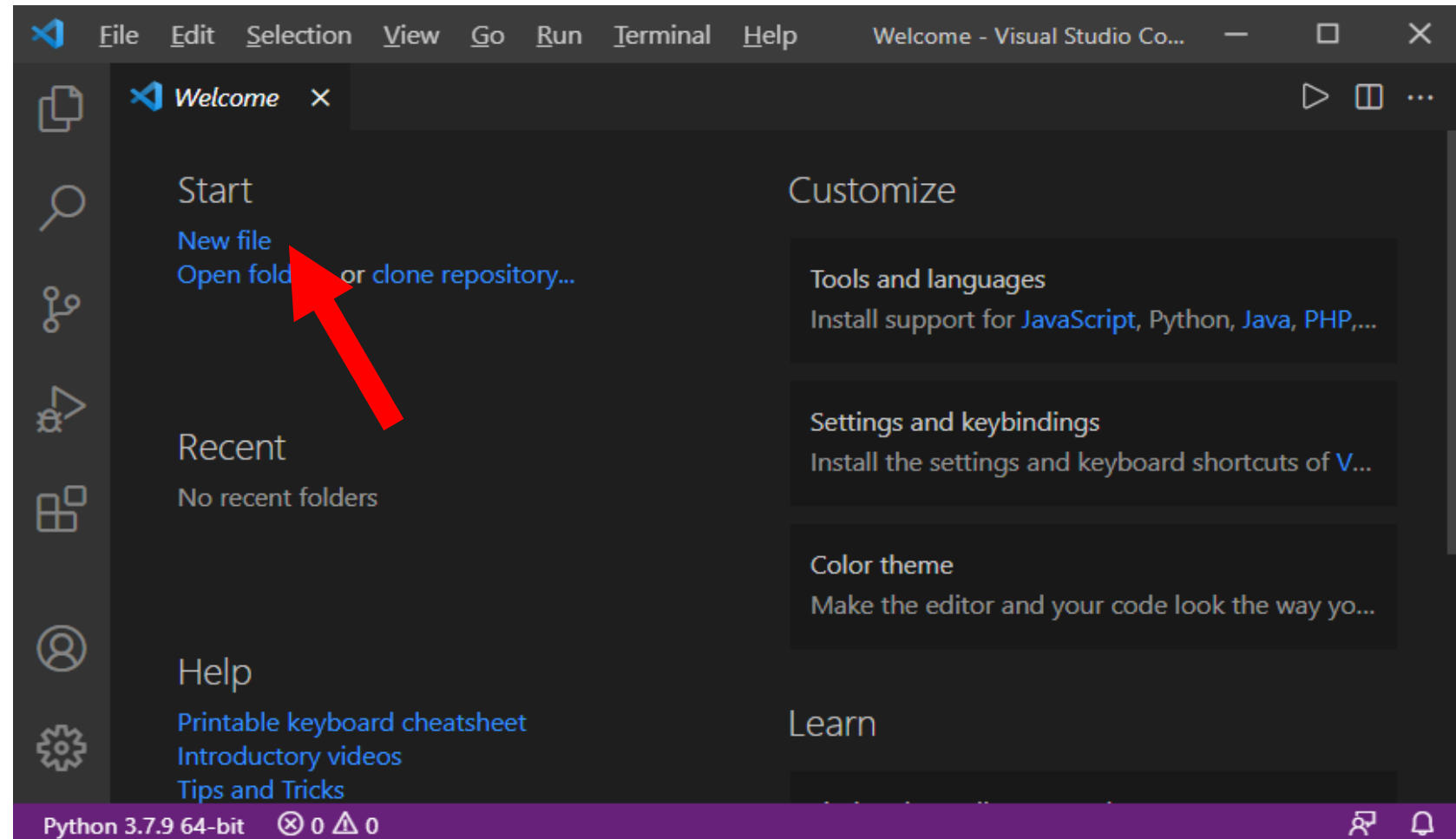
Obs.: é necessário que seu micro esteja conectado à internet.



Fonte: Autoria própria.

Ambiente de desenvolvimento: Visual Studio Code (VSCode)

Na página de *Welcome*, clique em "New File":

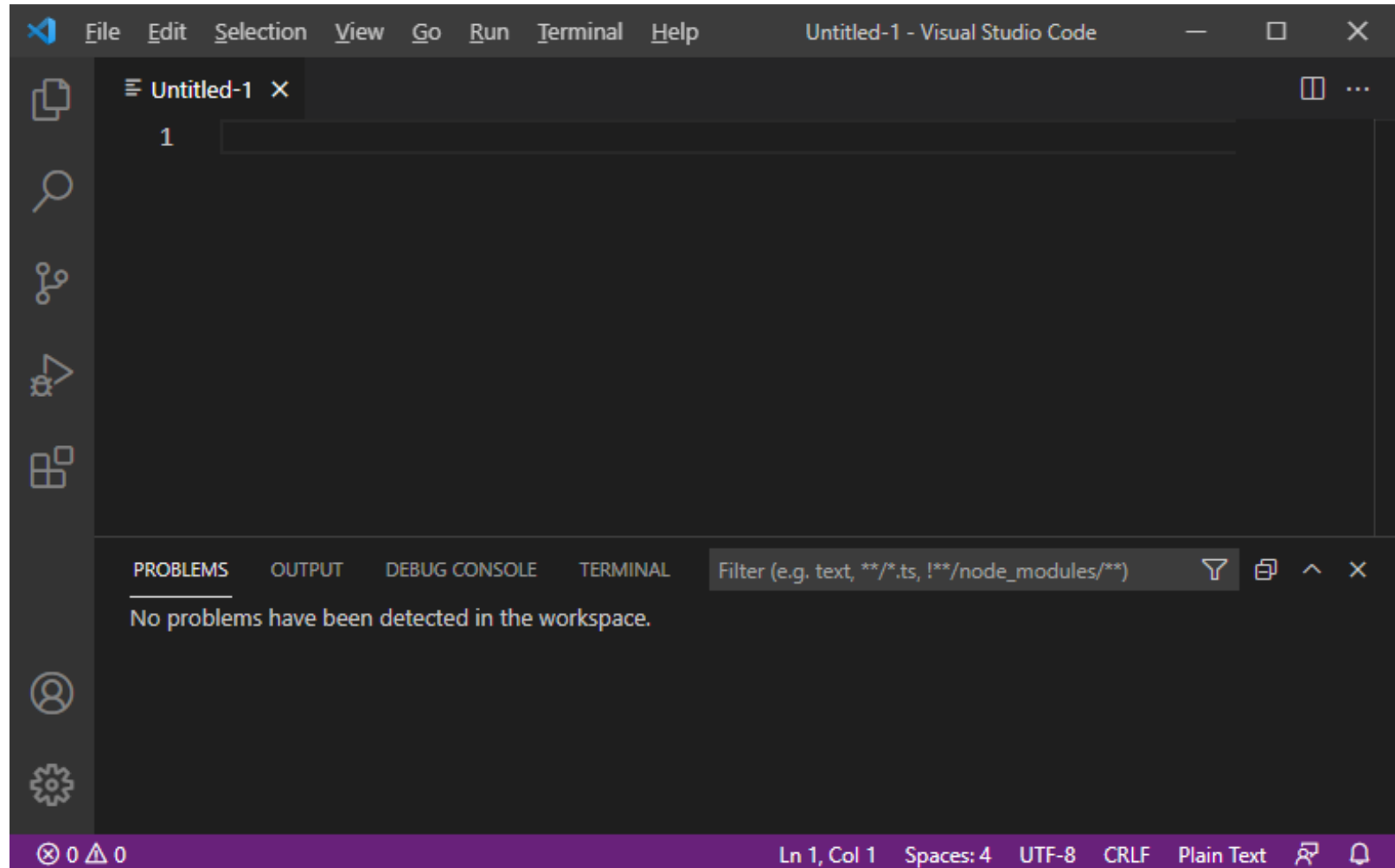


Fonte: Autoria própria.

Ambiente de desenvolvimento: Visual Studio Code (VSCode)

A partir daqui, podemos começar a programar em Python.

Obs.: para salvar um arquivo, deve-se colocar o nome entre aspas duplas, e com a extensão .py. Ex.: "programa.py".

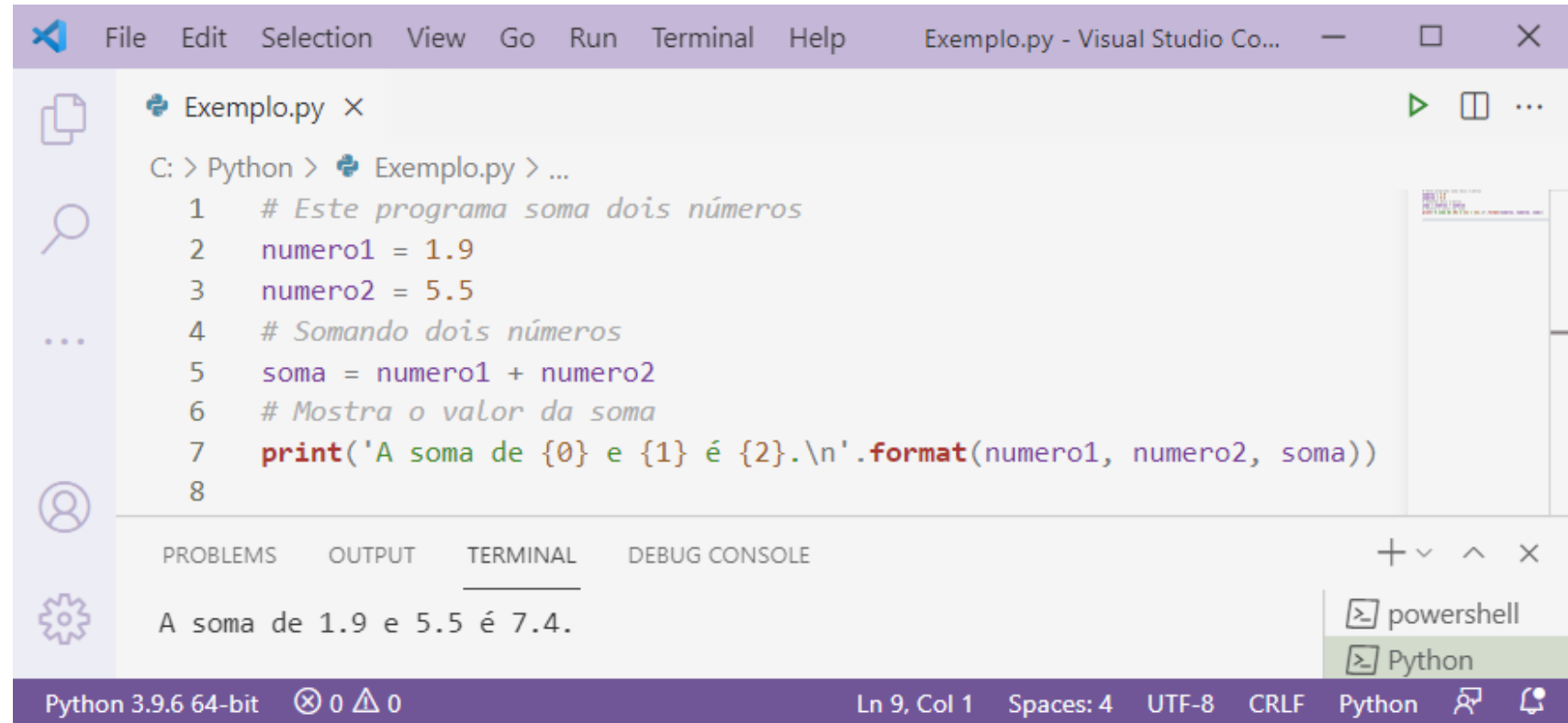


Fonte: Autoria própria.

Ambiente de desenvolvimento: Visual Studio Code (VSCode)

Exemplos de programas em Python:

Neste caso, executando no próprio VSCode.

A screenshot of the Visual Studio Code interface. The top menu bar includes File, Edit, Selection, View, Go, Run, Terminal, and Help. The title bar shows 'Exemplo.py - Visual Studio Co...'. The editor window displays a Python file named 'Exemplo.py' with the following code:

```
1  # Este programa soma dois números
2  numero1 = 1.9
3  numero2 = 5.5
4  # Somando dois números
5  soma = numero1 + numero2
6  # Mostra o valor da soma
7  print('A soma de {0} e {1} é {2}.\n'.format(numero1, numero2, soma))
8
```

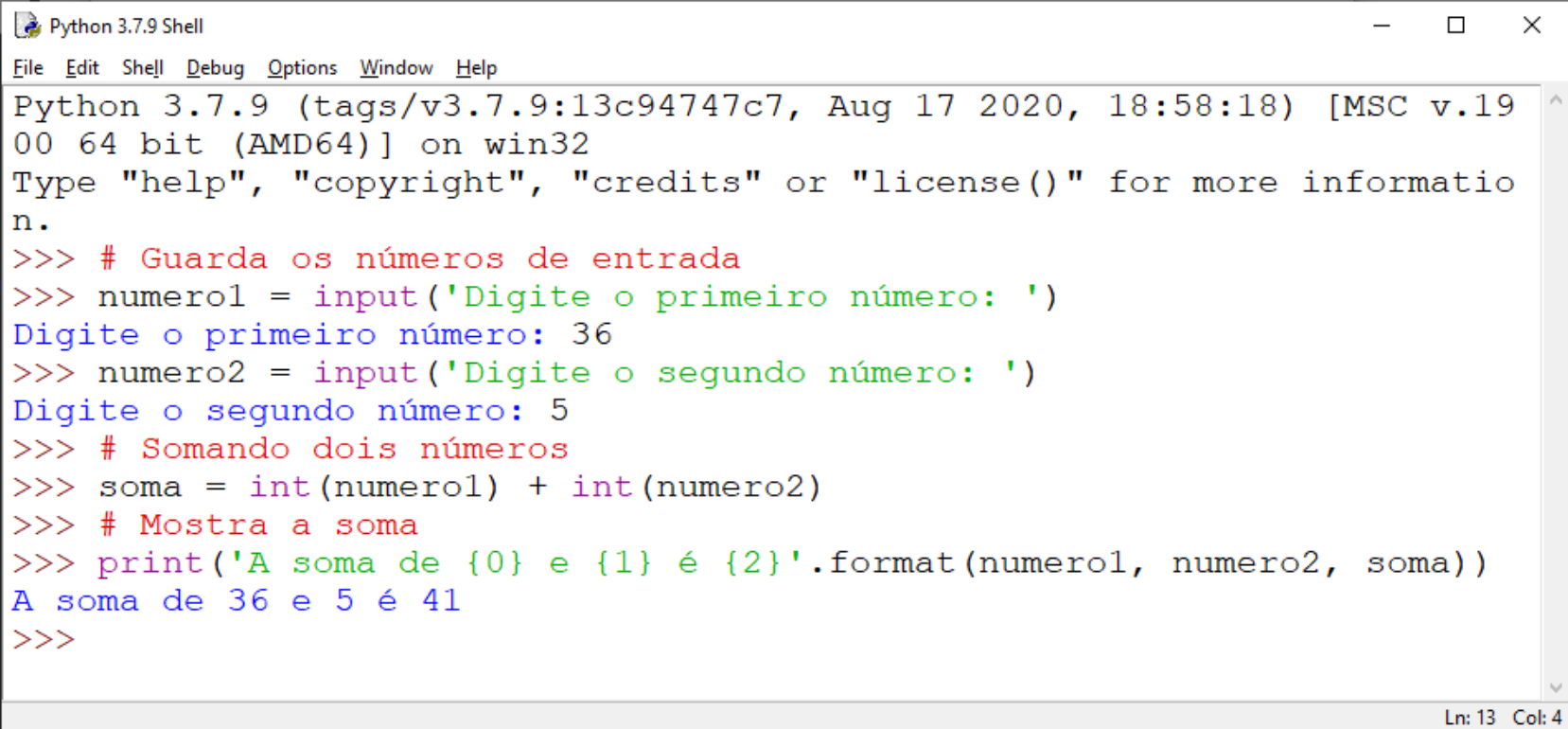
The bottom panel shows the 'TERMINAL' tab with the output: 'A soma de 1.9 e 5.5 é 7.4.'. The status bar at the bottom indicates 'Python 3.9.6 64-bit', '0 errors', '0 warnings', and the current cursor position 'Ln 9, Col 1'. The file encoding is 'UTF-8' and the line ending is 'CRLF'.

Fonte: Autoria própria.

Ambiente de desenvolvimento: Python shell

Exemplos de programas em Python (continuação):

Neste caso, executando linha a linha no Terminal do Python (no shell).



```
Python 3.7.9 Shell
File Edit Shell Debug Options Window Help
Python 3.7.9 (tags/v3.7.9:13c94747c7, Aug 17 2020, 18:58:18) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> # Guarda os números de entrada
>>> numero1 = input('Digite o primeiro número: ')
Digite o primeiro número: 36
>>> numero2 = input('Digite o segundo número: ')
Digite o segundo número: 5
>>> # Somando dois números
>>> soma = int(numero1) + int(numero2)
>>> # Mostra a soma
>>> print('A soma de {0} e {1} é {2}'.format(numero1, numero2, soma))
A soma de 36 e 5 é 41
>>>
```

Fonte: Autoria própria.

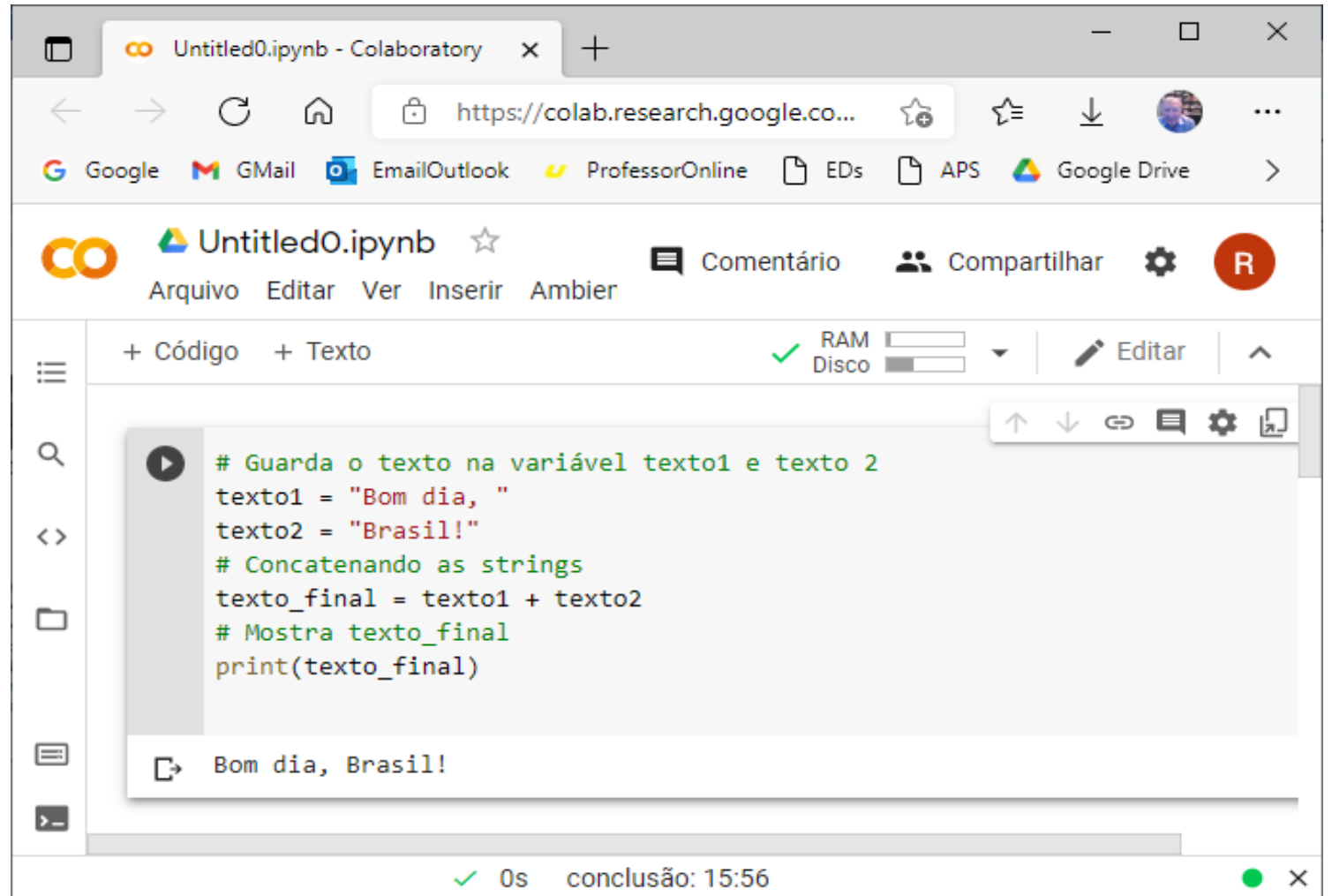
Ambiente de desenvolvimento: Google Colaboratory

Exemplos de programas em Python (continuação):

Neste outro caso, executando um programa no Google CoLab.

Site:

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



The screenshot displays a web browser window with a single tab titled 'Untitled0.ipynb - Colaboratory'. The address bar shows the URL 'https://colab.research.google.co...'. Below the browser window, the Google Colaboratory interface is visible. The top bar includes the Google Colab logo, the file name 'Untitled0.ipynb', and navigation links: 'Arquivo', 'Editar', 'Ver', 'Inserir', and 'Ambier'. To the right of these links are buttons for 'Comentário', 'Compartilhar', and a settings gear icon, followed by a red circle with a white 'R'. Below the top bar, there is a section for '+ Código' and '+ Texto'. On the right side of this section, there is a green checkmark, a 'RAM' indicator, a 'Disco' indicator, and an 'Editar' button. The main area of the notebook contains a code cell with the following Python code:

```
# Guarda o texto na variável texto1 e texto 2
texto1 = "Bom dia, "
texto2 = "Brasil!"
# Concatenando as strings
texto_final = texto1 + texto2
# Mostra texto_final
print(texto_final)
```

Below the code cell, the output is displayed: 'Bom dia, Brasil!'. At the bottom of the notebook, there is a status bar showing a green checkmark, '0s', and 'conclusão: 15:56'.

Fonte: <https://colab.research.google.com/>. Acesso em: 15 jul. 2021.

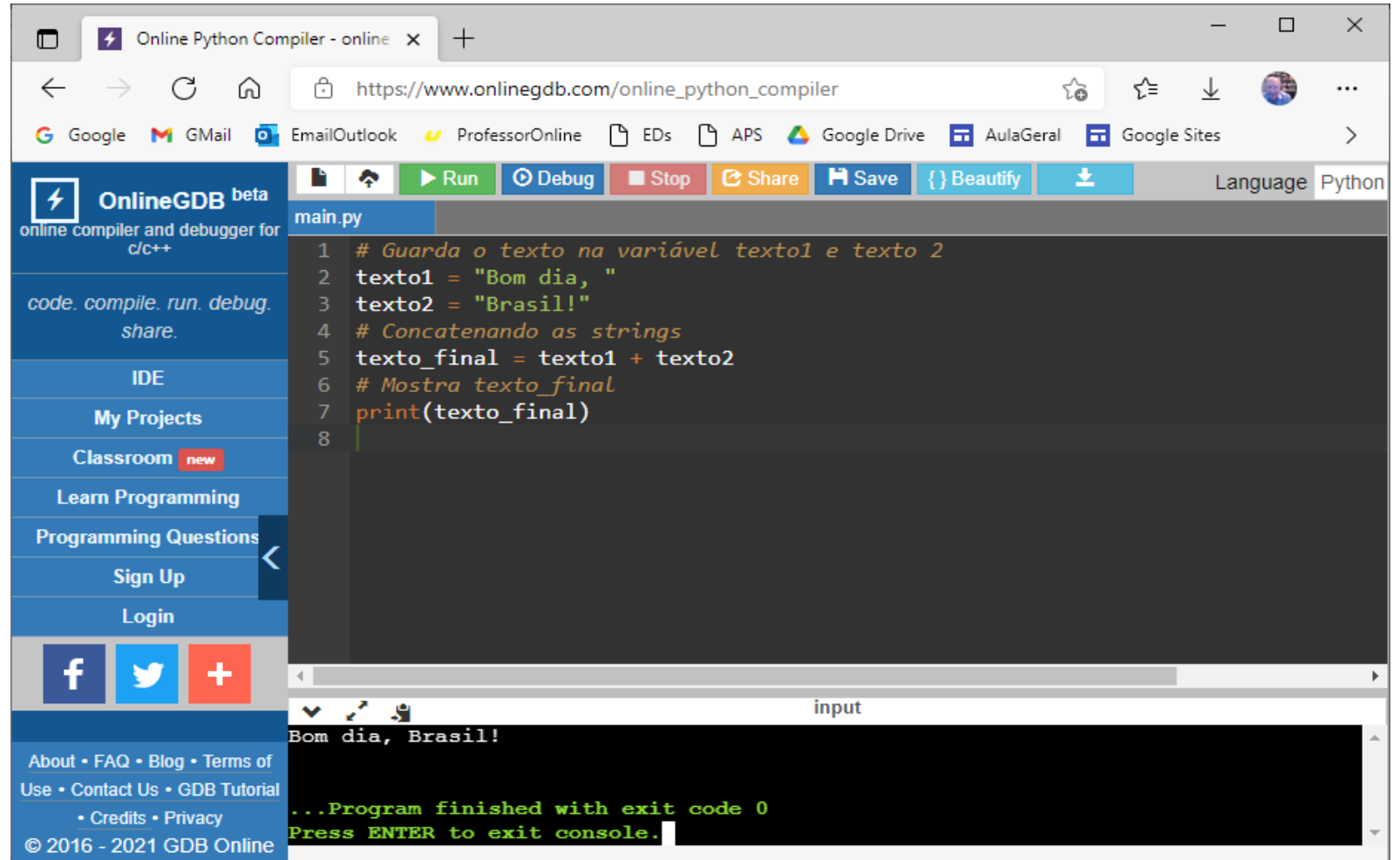
Ambiente de desenvolvimento: OnlineGDB

Exemplos de programas em Python (continuação):

Neste outro caso, executando um programa no OnlineGDB.

Site:

https://www.onlinegdb.com/online_python_compiler.



The screenshot shows the OnlineGDB website interface. The browser address bar displays `https://www.onlinegdb.com/online_python_compiler`. The page features a sidebar on the left with navigation links: "code. compile. run. debug. share.", "IDE", "My Projects", "Classroom new", "Learn Programming", "Programming Questions", "Sign Up", and "Login". Below these are social media icons for Facebook, Twitter, and a generic share icon. At the bottom of the sidebar are links for "About", "FAQ", "Blog", "Terms of Use", "Contact Us", "GDB Tutorial", "Credits", and "Privacy", along with the copyright notice "© 2016 - 2021 GDB Online". The main area of the page has a toolbar with buttons for "Run", "Debug", "Stop", "Share", "Save", "Beautify", and a "Language" dropdown set to "Python". The code editor displays a Python script named "main.py" with the following code:

```
1 # Guarda o texto na variável texto1 e texto 2
2 texto1 = "Bom dia, "
3 texto2 = "Brasil!"
4 # Concatenando as strings
5 texto_final = texto1 + texto2
6 # Mostra texto_final
7 print(texto_final)
8
```

Below the code editor is an "input" field and a console output area. The input field contains the text "Bom dia, Brasil!". The console output shows the result of the program execution:

```
...Program finished with exit code 0
Press ENTER to exit console.
```

Fonte: https://www.onlinegdb.com/online_python_compiler/. Acesso em: 15 jul. 2021.

Interatividade

De acordo com o que vimos até agora, o que é um IDE?

- a) É um tipo de linguagem que pode interagir com o Python.
- b) É um comando da linguagem Python.
- c) É um *site*, de onde se pode baixar o programa Python.
- d) É o arquivo instalador de programas.
- e) É um ambiente integrado que auxilia no desenvolvimento de programas.

Resposta

De acordo com o que vimos até agora, o que é um IDE?

- a) É um tipo de linguagem que pode interagir com o Python.
- b) É um comando da linguagem Python.
- c) É um *site*, de onde se pode baixar o programa Python.
- d) É o arquivo instalador de programas.
- e) É um ambiente integrado que auxilia no desenvolvimento de programas.

Conceito de variáveis

Variável é um **espaço de memória** que contém, ou pode conter, um valor.

Uma variável possui:

- Um **nome** – que identifica a variável na memória;
- Um **valor** – uma informação guardada naquele espaço de memória;
- Um **tipo** – que indica o tipo da informação guardada naquela variável.

- Ex.: **num = 4**

...significa que, após a sua execução, a variável “num”
receberá o valor 4.

Nomes de variáveis

Uma variável pode receber qualquer nome, mas algumas regras de nomenclatura devem ser seguidas. Sendo assim, o nome de uma variável:

- Não pode conter espaço(s);
- Não pode conter acentuação;
- Não pode conter símbolos (com a exceção do *_ Underline*);
- Não pode iniciar com um número;
- Não pode ser uma palavra reservada.

Nomes <u>válidos</u>	Nomes que <u>não</u> podem ser dados a variáveis
notaP1	1aNota
NomeAluno	peso do paciente
valor_teste	<i>Print</i>
_taxa01	valorPerc%
Media	AçãoRealizada
xpto	Média do Semestre
iddPessoa	int

Os tipos das variáveis

- O "tipo da variável" define o tipo do valor que pode ser guardado na variável.

Na linguagem Python trabalhamos com 5 tipos (primitivos) de dados:

- **int** (para as variáveis numéricas inteiras – valores que indicam quantidades);
- **float** (para as variáveis numéricas de ponto flutuante – ex.: 3.1415 – valores que indicam medidas);
- **bool** (para as variáveis lógicas – que recebem valores lógicos *True* ou *False*);
 - **complex** (para as variáveis complexas – ex.: $3 + 2j$);
 - **str** (para as variáveis alfanuméricas: letras, palavras, frases – *String*).

Valores das variáveis

Ao lidarmos com as variáveis, muitas vezes, programamos valores iniciais para elas.

Vejamos como, dependendo do tipo da variável, devemos atribuir valores a elas:

- Para as variáveis do tipo **int** fazemos :: variável = valorInteiro

Ex.: idade_aluno = 21

- Para as variáveis do tipo **float** fazemos :: variável = valorReal

Ex.: altura_aluno = 1.83

Obs.: perceba que, na programação, devemos utilizar o ponto "." (e não a vírgula ",") como o separador decimal de valores numéricos.

- Para as variáveis do tipo **bool** :: valores lógicos: *True* (que quer dizer "Verdadeiro") e *False* (que quer dizer "Falso"):

Obs.: percebam que esses valores começam com a Letra Maiúscula.

Ex.: alunoAprovado = True

Valores das variáveis

- Para as variáveis do tipo **complex** :: que representam números complexos:

Ex.: `valor_complexo = 3 + 2j`

Obs.: percebam que a linguagem **Python** não se utiliza da letra "i" para representar o "valor imaginário" ("raiz quadrada de menos-um"), mas sim a letra "j";

- Para as variáveis do tipo **str** :: que representam as letras, palavras, frases – uma *String*:

A linguagem *Python* aceita que se utilize aspas simples ou duplas para definir um valor *String* a uma variável;

Ex.: `nome_aluno = "Fulano de Tal"`

`# com aspas duplas`

... ou ainda...:

`nome_cliente = 'Beltrano dos Anzóis'`

`# com aspas simples`

Constantes numéricas

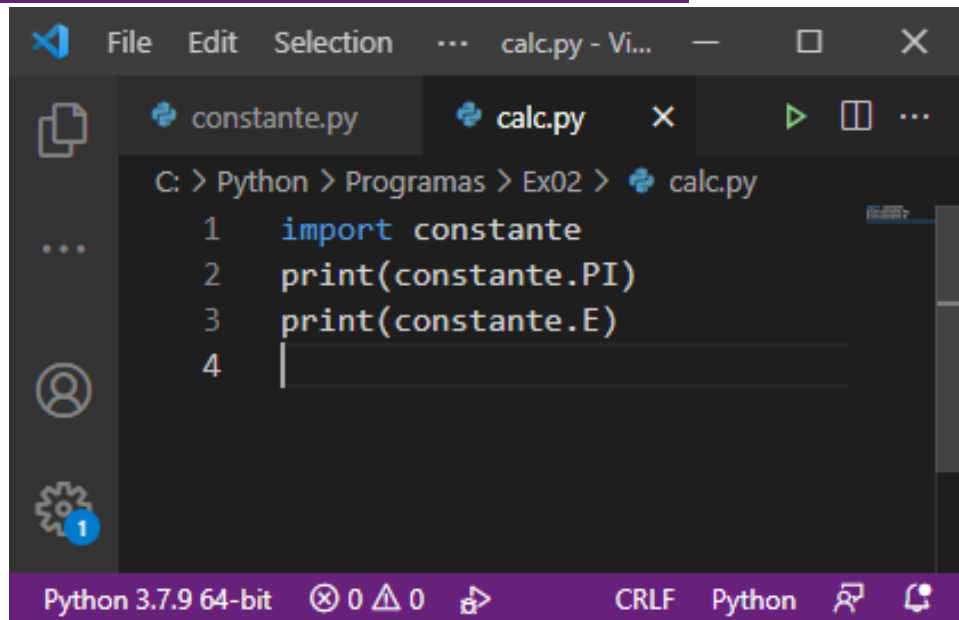
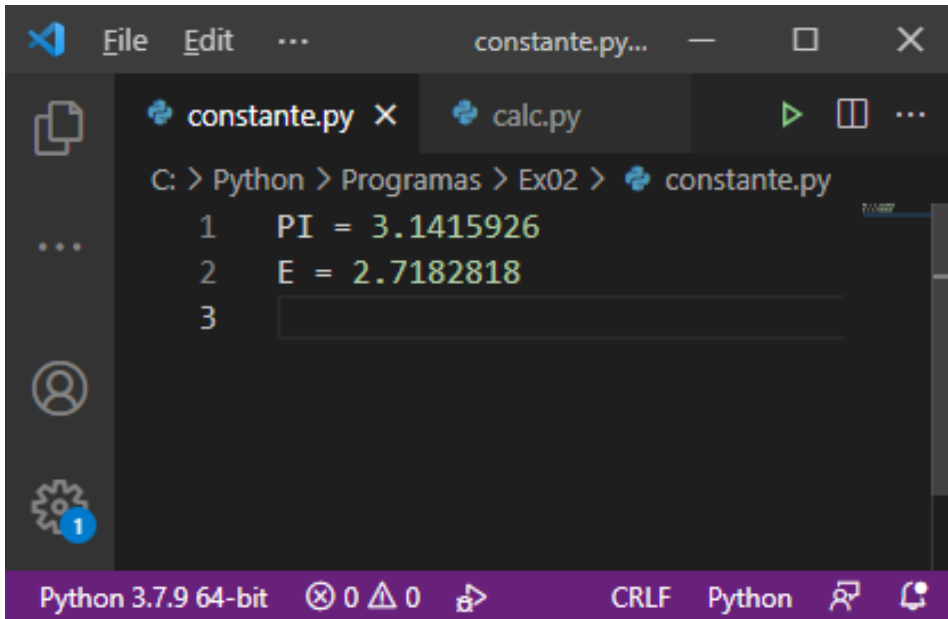
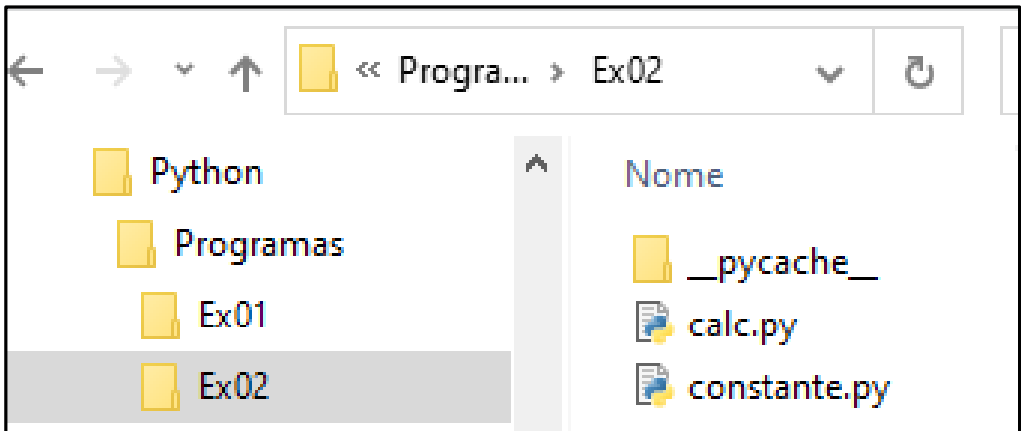
- Uma constante é um valor guardado em um espaço de memória (uma "variável"), mas que não pode ser modificado.
- Em Python, ainda não é possível gerar constantes, ou seja, áreas de memória que recebem valores que não podem sofrer qualquer alteração.
- No entanto, quando queremos “representar” uma constante, costuma-se identificá-la com um nome em "CAIXA-ALTA", ou seja, todas as letras em Maiúsculo.

Ex.:

- **PI = 3.1415926**
- **E = 2.7182818**

Constantes numéricas

Uma forma de criarmos uma constante:



Fontes: Autoria própria.

Operadores matemáticos

Os operadores matemáticos:

- Os operadores matemáticos são processados de acordo com as regras matemáticas.


Tabela dos Operadores Matemáticos (ou Aritméticos):

OPERADOR	OPERAÇÃO
+	Adição
-	Subtração
*	Multiplicação
/	Divisão (com o resultado fracionado)
//	"Div" (resultado inteiro da divisão)
%	"Mod" (resto da divisão)
**	Exponenciação ou Potenciação

Operadores matemáticos

Observações sobre algumas operações matemáticas:

Diferença entre os operadores /, // e % (dividido, div e mod).

```
C: > Python >  contas.py >...  
1  # diferença entre / , // e %  
2  x = 7 / 3 # 7 dividido por 3  
3  print(x) # imprimirá 2.3333333333333333  
4  x = 7 // 3 # 7 div 3  
5  print(x) # imprimirá 2  
6  x = 7 % 3 # 7 mod 3  
7  print(x) # imprimirá 1
```

Operadores matemáticos

Ordem de precedência das operações matemáticas:

- As operações seguem a mesma ordem das realizadas em cálculos matemáticos.

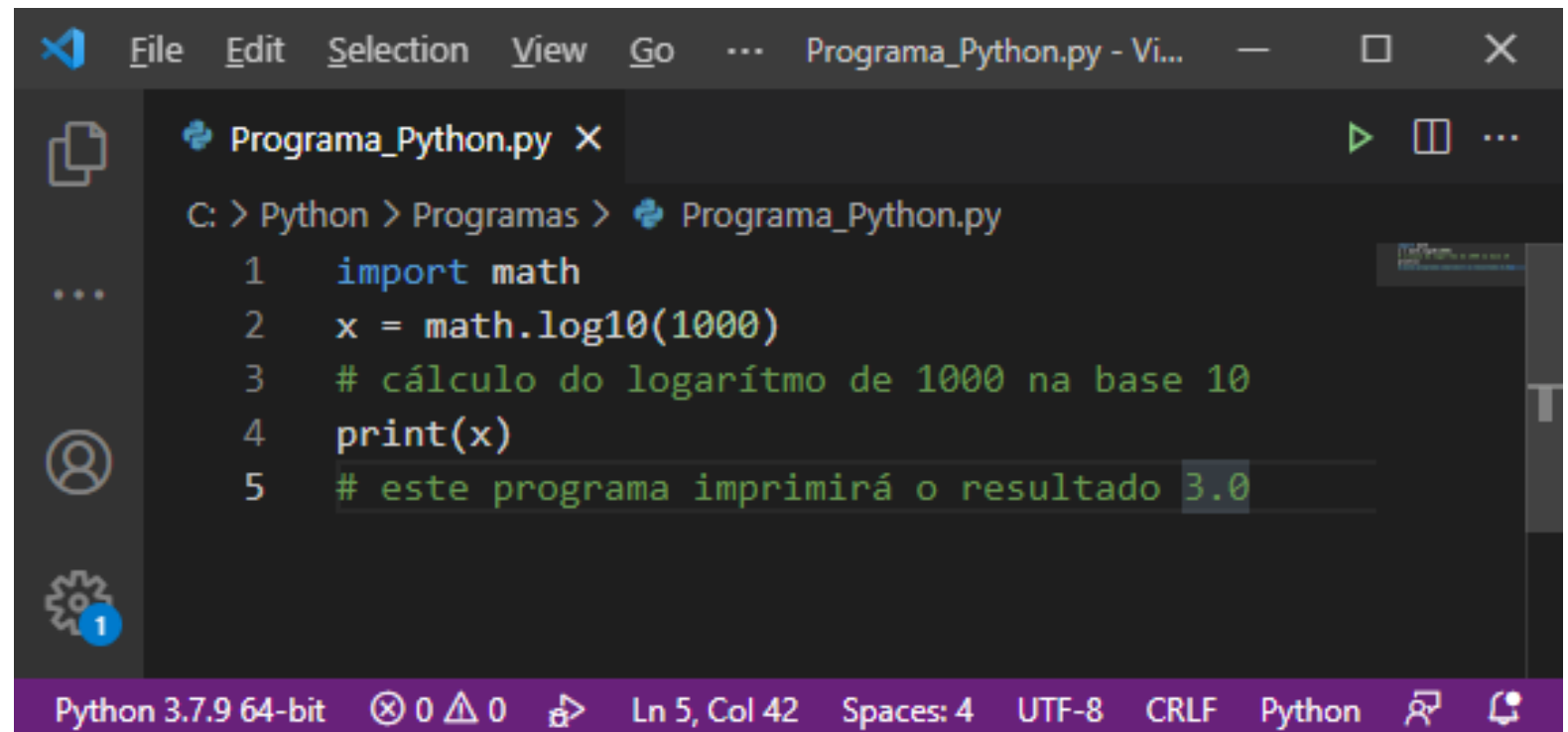
HIERARQUIA	OPERADORES
1º	()
2º	Funções
3º	**
4º	*, /, //, %
5º	+, -

- Operadores de uma mesma linha hierárquica são executados à medida em que aparecem na expressão (da esquerda para a direita).

Funções matemáticas

Funções matemáticas (da Biblioteca “*math*” do *Python*):

- Para que possamos utilizar essas funções, é necessário “importar” a **Biblioteca de Funções Matemáticas** do *Python* ao programa (usando: ***import math***);
- Devemos chamar as funções, sempre colocando o comando “***math.***” antes da função que se deseja executar.



```
File Edit Selection View Go ... Programa_Python.py - Vi...
Programa_Python.py X
C: > Python > Programas > Programa_Python.py
1 import math
2 x = math.log10(1000)
3 # cálculo do logarítmo de 1000 na base 10
4 print(x)
5 # este programa imprimirá o resultado 3.0
Python 3.7.9 64-bit 0 0 Ln 5, Col 42 Spaces: 4 UTF-8 CRLF Python
```

Funções matemáticas

Funções matemáticas (da Biblioteca “*math*” do *Python*):

Tabela com algumas das Funções da Biblioteca “*math*”:

FUNÇÃO	OPERAÇÃO
pow(x, y)	Calcula a potência: x elevado a y (na versão 3 do Python, o operador ** pode substituir esta função, pois o mesmo já trabalha com números reais).
e, pi	Constantes matemáticas: e = 2,7182 pi = 3,1415
sqrt(x)	Calcula a raiz quadrada de x.
cos(x)	Calcula o cosseno de x (em radianos).
sin(x)	Calcula o seno de x (em radianos).
tan(x)	Calcula a tangente de x (em radianos).
log10(x)	Calcula o logaritmo de x na base 10.
log(x)	Calcula o logaritmo de x na base e (log neperiano).

Interatividade

Analizando o código a seguir, qual será o valor das variáveis x e y após a execução deste programa?

$a = 11$

$b = 6$

$c = 5$

$x = a // c + a \% b$

$y = a + b / 2 * c$

- a) x receberá 2.2 e y receberá 70.
- b) x receberá 3 e y receberá 42.5.
- c) x receberá 7 e y receberá 26.
- d) x receberá 3 e y receberá 26.
- e) x receberá 4.03 e y receberá 11.6.

Resposta

Analizando o código a seguir, qual será o valor das variáveis x e y após a execução deste programa?

$a = 11$

$b = 6$

$c = 5$

$x = a // c + a \% b$

$y = a + b / 2 * c$

- a) x receberá 2.2 e y receberá 70.
- b) x receberá 3 e y receberá 42.5.
- c) **x receberá 7 e y receberá 26.**
- d) x receberá 3 e y receberá 26.
- e) x receberá 4.03 e y receberá 11.6.

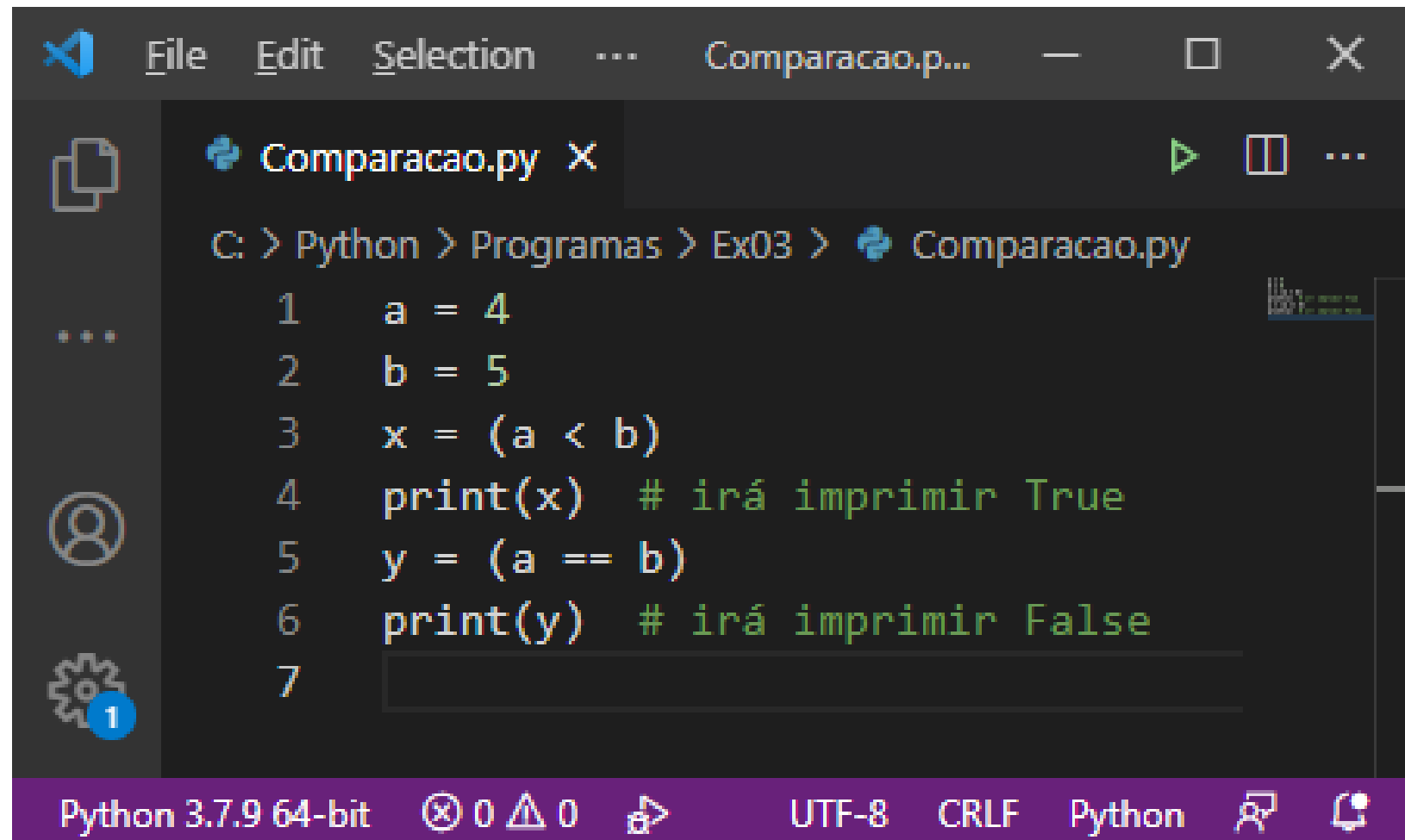
Operadores relacionais

- Os Operadores Relacionais são utilizados para realizar as Comparações Lógicas.
- O resultado de uma Comparação com esses operadores sempre será um Valor Lógico, ou seja, ou **True** (Verdadeiro) ou **False** (Falso).

OPERADOR	OPERAÇÃO	SÍMBOLO MATEMÁTICO
==	Igual a	=
>	Maior que	>
<	Menor que	<
!=	Diferente de	≠
>=	Maior ou igual a	≥
<=	Menor ou igual a	≤

Operadores relacionais

Exemplo de uma Operação Relacional (de Comparação):



The image shows a screenshot of a Python IDE window titled 'Comparacao.p...'. The window contains a Python script named 'Comparacao.py' with the following code:

```
C: > Python > Programas > Ex03 > Comparacao.py
1  a = 4
2  b = 5
3  x = (a < b)
4  print(x)  # irá imprimir True
5  y = (a == b)
6  print(y)  # irá imprimir False
7
```

The IDE interface includes a menu bar with 'File', 'Edit', and 'Selection'. The status bar at the bottom indicates 'Python 3.7.9 64-bit', '0 0', 'UTF-8', 'CRLF', and 'Python'.

Fonte: Autoria própria.

Operadores lógicos

- Realizam operações de Lógica Booleana simples (Negação) ou composta (E OU) cuja resposta é, também, um valor lógico (***True*** ou ***False***).

Em Python, existem 3 Operadores Lógicos básicos:

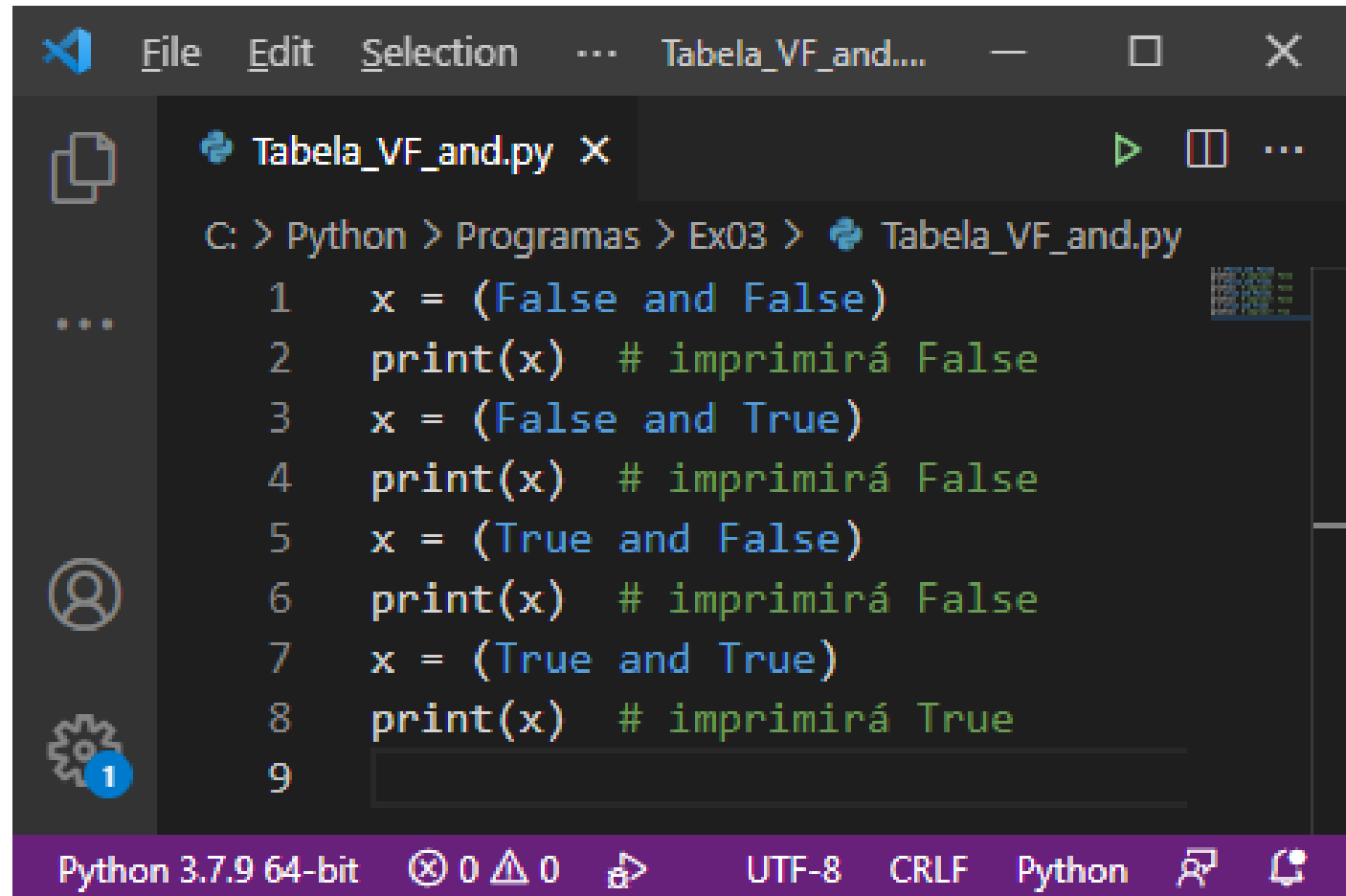
OPERADOR PYTHON	OPERAÇÃO
<i>not</i>	Negação
<i>and</i>	E
<i>or</i>	OU

- Esses operadores são utilizados, juntamente, com outras operações lógicas, realizando as operações de Lógica Matemática, e que servirão para as estruturas que funcionam a partir de comparações (as quais iremos estudar em aulas futuras).

Operadores lógicos

A Tabela Verdade do operador “***and***” (E):

X	Y	X <i>and</i> Y
F	F	F
F	V	F
V	F	F
V	V	V



```
File Edit Selection ... Tabela_VF_and...
Tabela_VF_and.py
C: > Python > Programas > Ex03 > Tabela_VF_and.py
1 x = (False and False)
2 print(x) # imprimirá False
3 x = (False and True)
4 print(x) # imprimirá False
5 x = (True and False)
6 print(x) # imprimirá False
7 x = (True and True)
8 print(x) # imprimirá True
9
```

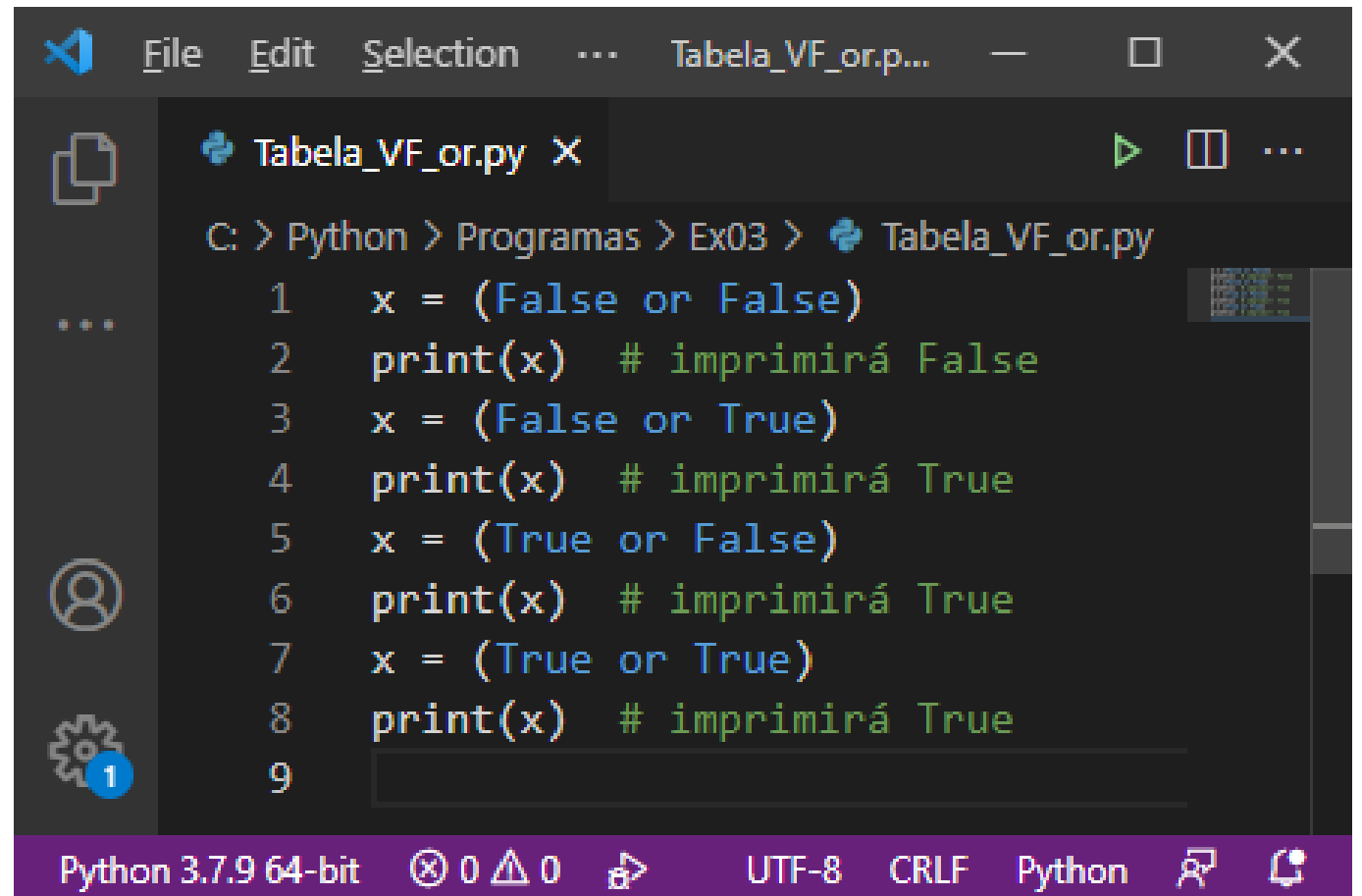
Python 3.7.9 64-bit 0 0 UTF-8 CRLF Python

Fonte: Autoria própria.

Operadores lógicos

A Tabela Verdade do operador “**or**” (OU):

X	Y	X <i>or</i> Y
F	F	F
F	V	V
V	F	V
V	V	V



```
File Edit Selection ... Tabela_VF_or.p...  
Tabela_VF_or.py X  
C: > Python > Programas > Ex03 > Tabela_VF_or.py  
1 x = (False or False)  
2 print(x) # imprimirá False  
3 x = (False or True)  
4 print(x) # imprimirá True  
5 x = (True or False)  
6 print(x) # imprimirá True  
7 x = (True or True)  
8 print(x) # imprimirá True  
9
```

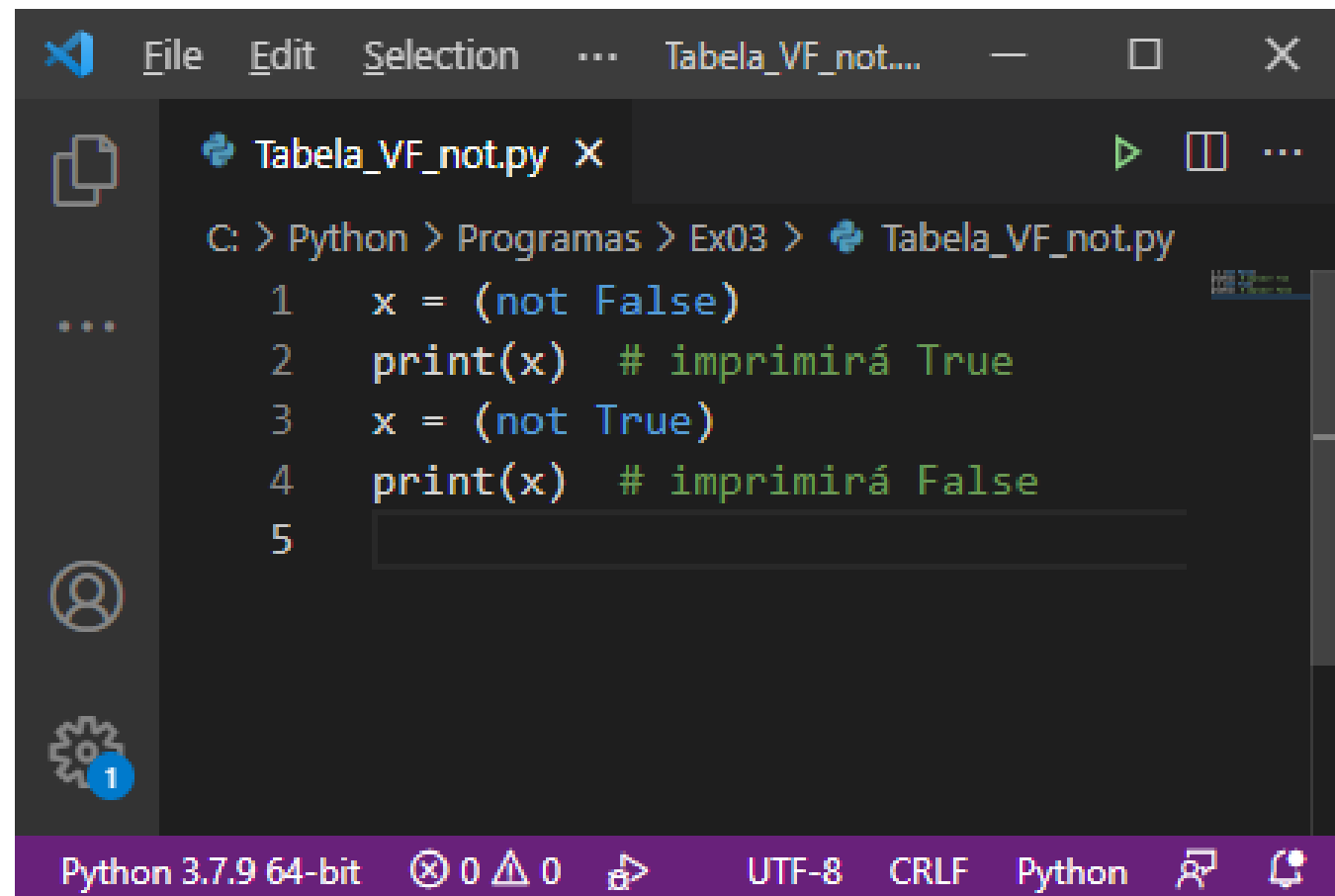
Python 3.7.9 64-bit 0 0 UTF-8 CRLF Python

Fonte: Autoria própria.

Operadores lógicos

A Tabela Verdade do operador “**not**” (Negação):

X	<i>not X</i>
V	F
F	V



The screenshot shows a Python IDE window titled 'Tabela_VF_not...'. The code in the editor is as follows:

```
C: > Python > Programas > Ex03 > Tabela_VF_not.py
1  x = (not False)
2  print(x)  # imprimirá True
3  x = (not True)
4  print(x)  # imprimirá False
5
```

The status bar at the bottom indicates 'Python 3.7.9 64-bit', '0 0' errors/warnings, 'UTF-8' encoding, 'CRLF' line endings, and 'Python' mode.

Fonte: Autoria própria.

Operadores de atribuição

Operadores de Atribuição:

- A Atribuição de Valores é a inserção de uma informação em uma variável;
- O Operador de Atribuição é o sinal de igual (=);
- Sempre, o elemento do lado **esquerdo** do Operador receberá o valor definido no lado **direito** do Operador.

Ex.:

`x = 3` # a variável `x` receberá o valor 3

- A atribuição múltipla – atribuição de um conjunto de valores a um conjunto de variáveis em uma mesma linha de comando.
Obs.: ambos (variáveis e valores) precisam, obrigatoriamente, possuir a mesma quantidade de elementos.

Ex.:

`x, y = 20, 30`

`x` receberá o valor 20 e `y` receberá o valor 30

Operadores de atribuição

Operações Recursivas:

- São operações que alteram o valor da variável, de acordo com o valor que ela possuía anteriormente.

Ex.: `x = 4`

```
x = x + 5    # x recebe 5 adicionado ao valor anterior do próximo x (que era 4)  
print(x)    # este comando imprimirá o valor 9
```

- Os Operadores Compostos e Recursivos são formados pela junção de um operador aritmético com o Operador de Atribuição.

Ex.:

```
x = 6  
x += 2    # equivalente ao comando: x = x + 2  
print(x)    # este comando imprimirá o valor 8
```

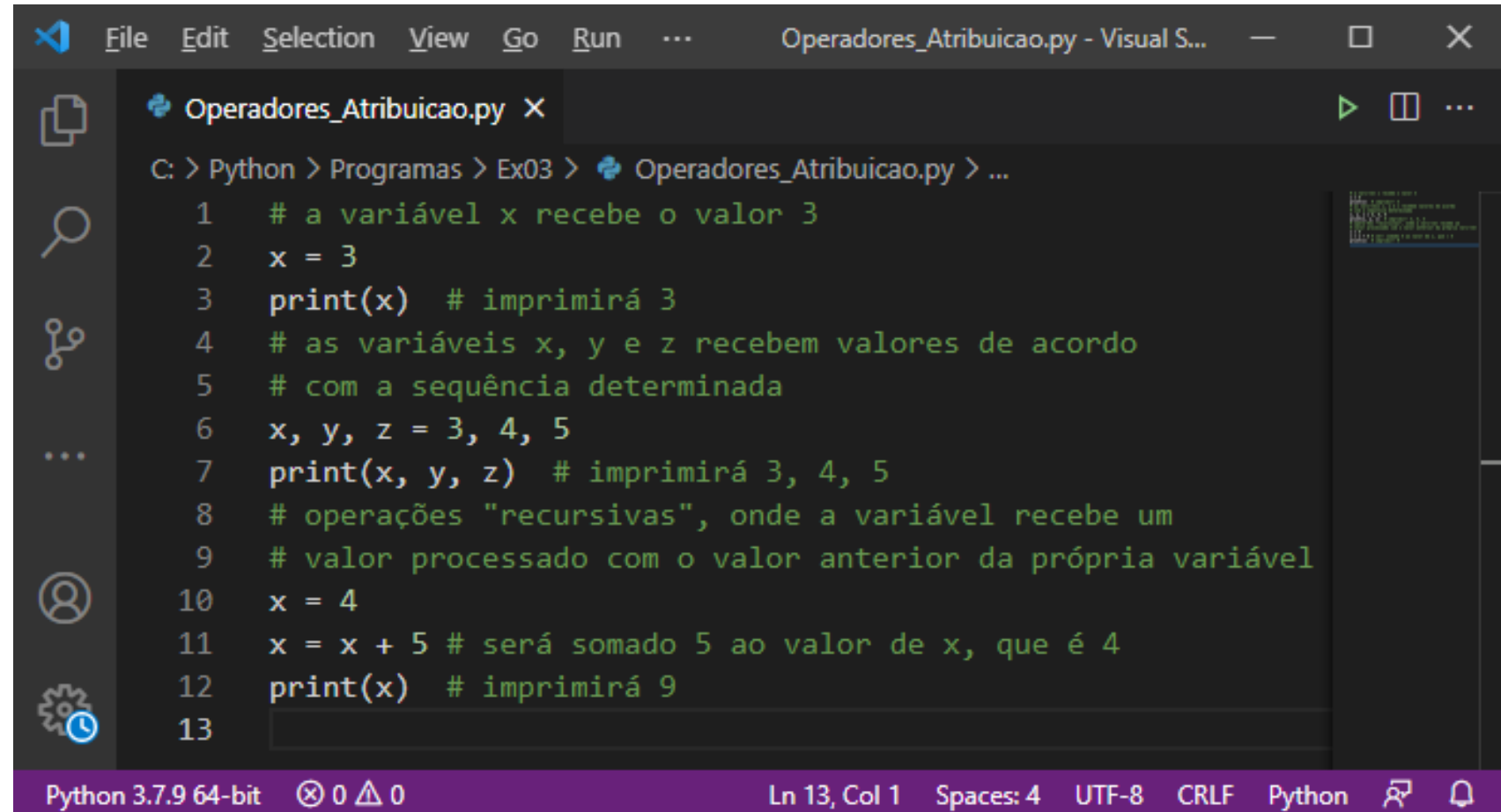
Operadores de atribuição

Operadores Compostos e Recursivos:

OPERADOR	EXEMPLO	EQUIVALÊNCIA
+=	x += 1	x = x + 1
-=	y -= 1	y = y - 1
*=	c *= 2	c = c * 2
/=	d /= 2	d = d / 2
**=	e **= 2	e = e ** 2
//=	f //= 4	f = f // 4
%=	g %= 6	g = g % 6

Operadores de atribuição

Na imagem a seguir, temos um exemplo de um programa com as operações de atribuição (de acordo com o que vimos anteriormente):



```
File Edit Selection View Go Run ... Operadores_Atribuicao.py - Visual S...
Operadores_Atribuicao.py X
C: > Python > Programas > Ex03 > Operadores_Atribuicao.py > ...
1  # a variável x recebe o valor 3
2  x = 3
3  print(x) # imprimirá 3
4  # as variáveis x, y e z recebem valores de acordo
5  # com a sequência determinada
6  x, y, z = 3, 4, 5
7  print(x, y, z) # imprimirá 3, 4, 5
8  # operações "recursivas", onde a variável recebe um
9  # valor processado com o valor anterior da própria variável
10 x = 4
11 x = x + 5 # será somado 5 ao valor de x, que é 4
12 print(x) # imprimirá 9
13
```

Python 3.7.9 64-bit 0 0 Ln 13, Col 1 Spaces: 4 UTF-8 CRLF Python

Fonte: Autoria própria.

Precedência dos operadores

Ordem geral de precedência dos operadores vistos (Matemáticos, Relacionais e Lógicos):

HIERARQUIA	OPERADORES
1º	(...)
2º	Funções
3º	**
4º	*, /, //, %
5º	+, -
6º	<=, <, >, >=, ==, !=
7º	<i>not</i>
8º	<i>and</i>
9º	<i>or</i>
10º	=, %=, /=, //=, -=, +=, *=, **=

Expressões lógicas

- As linguagens de programação usam Expressões Lógicas, por exemplo, para avaliar os desvios no fluxo de comandos (Lógica Booleana).
- Obs.: esses “desvios” são importantes na programação estruturada e serão detalhados nas próximas unidades.

Ex.:

`a = 5`

`b = 10`

`c = (a != b) or (a > b)`

variável “c” resulta em *True* (verdadeiro)

`d = (a > 2) and (b < 8)`

variável “d” resulta em *False* (Falso)

`e = not (a == 5)`

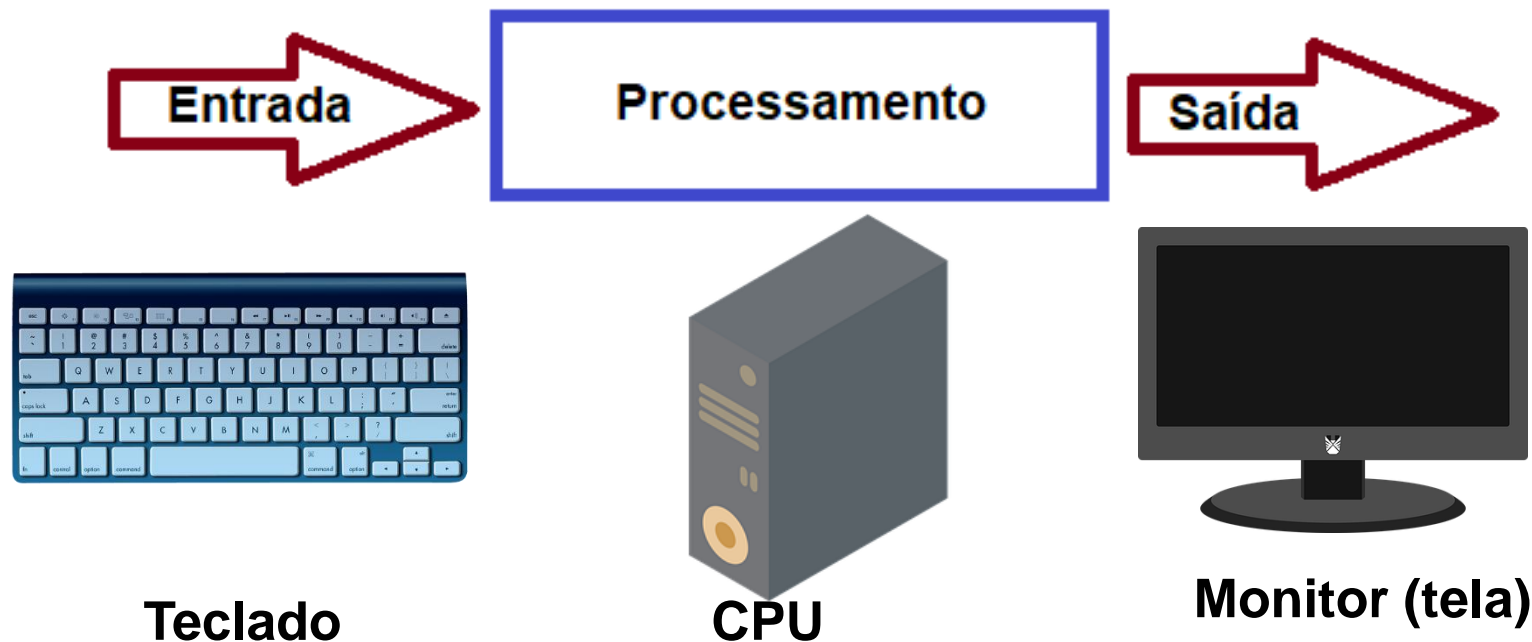
variável “e” resulta em *False* (Falso)

Estrutura sequencial

- A Estrutura Sequencial é a estrutura de controle mais básica, em que os comandos são executados na ordem em que são especificados, um após o outro (“de cima para baixo, da esquerda para a direita”).

Uma Estrutura Sequencial consiste basicamente em:

- Entrada de Dados;
- Processamento; e
- Saída de Dados.



Fonte:
<https://www.maxpixel.net/Keyboard-Characters-Pc-Letters-Input-Symbol-Keys-6111333>

Fonte: <https://www.maxpixel.net/Mid-tower-Cpu-Desktop-Tower-Server-Isometric-4971978>

Fonte: <https://www.maxpixel.net/Computing-Screen-Computer-Monitor-Technology-6012131>

Estrutura sequencial

A Entrada de Dados no programa (via digitação no teclado):

- A função ***input(...)*** é uma função que permite a entrada de dados.

Sua sintaxe é:

```
var = input("mensagem")
```

- A “mensagem” aparecerá no console, mantendo o *Prompt*, que aguardará;
- A entrada de dados é feita por digitação, e, por fim, teclando *Enter*.
- No exemplo anterior, a variável “var” receberá o valor digitado via teclado. Esse valor sempre será do tipo *String* e, portanto, para transformá-lo em número, temos que utilizar um recurso que conhecemos como “*casting*”.

Sintaxe geral:

```
var = tipo(input("mensagem"))
```

Ex.:

```
x = int(input("Entre com o 1.o número: "))
```

Estrutura sequencial

A Saída de Dados do programa (na tela do console):

Função ***print(...)*** – permite a impressão da informação na tela do console no monitor. Sua sintaxe é:

print(parâmetro)

- Para imprimir o valor de uma variável, basta indicar o nome dessa variável no parâmetro da função *print*.

```
x = 4
```

```
print(x) # imprimirá o valor 4
```

- Aspas (simples ou duplas) são utilizadas quando se quer mostrar um “texto literal” de mensagem.

```
print("Olá Mundo!") # imprime o texto "Olá Mundo!"
```

- Podemos misturar os elementos (variáveis e texto literal), fazendo uma composição de mensagem.

```
print("valor de x =", x, "cm")
```

```
# imprime:      valor de x = 4 cm
```

Interatividade

Analizando o código a seguir, o que será impresso na tela do console, se entrarmos com o valor 6 para a variável “a” e 12 para a variável “b”?

```
a = int(input("Digite o valor de a: "))
```

```
b = int(input("Digite o valor de b: "))
```

```
x = (b == 12) and (a != 6)
```

```
y = (a < b / 2) or (b >= 11)
```

```
print("x =", x, "e y =", y)
```

- a) *x = False e y = True.*
- b) *x = True e y = True.*
- c) *x = True e y = False.*
- d) *x = False e y = False.*
- e) *x = 12 e y = 11.*

Resposta

Analizando o código a seguir, o que será impresso na tela do console, se entrarmos com o valor 6 para a variável “a” e 12 para a variável “b”?

```
a = int(input("Digite o valor de a: "))
```

```
b = int(input("Digite o valor de b: "))
```

```
x = (b == 12) and (a != 6)
```

```
y = (a < b / 2) or (b >= 11)
```

```
print("x =", x, "e y =", y)
```

- a) *x = False e y = True.*
- b) *x = True e y = True.*
- c) *x = True e y = False.*
- d) *x = False e y = False.*
- e) *x = 12 e y = 11.*

ATÉ A PRÓXIMA!