2

TEXTO BASE

LINGUAGEM DE PROGRAMAÇÃO

Faculdade IMPACTA

Texto base

2

Ambiente de desenvolvimento, tipos de dados, operadores aritméticos e variáveis

Prof. Me. Lucio Nunes de Lira
Prof. MSc. Rafael Maximo Carreira Ribeiro

Resumo

Nesta aula os objetivos são: (I) preparar o ambiente computacional necessário para a disciplina; (II) acessar as duas ferramentas básicas utilizadas na disciplina para o desenvolvimento de programas em Python (interpretador interativo e editor de código fonte); (III) conceituar e utilizar tipos de dados primitivos, constantes e variáveis; (IV) conhecer os operadores aritméticos; (V) manipular o sistema online Python Tutor.

2.1. Motivação

Para criarmos nossos programas, teremos que usar um ambiente que permita escrever algoritmos em uma linguagem de programação e que possibilite a execução automática dessas instruções. Por isso, nesta aula faremos o *download*, instalação e execução do interpretador da linguagem Python e do IDLE, um software que possibilita o processo básico para criação e execução de programas em Python. Introduziremos conceitos básicos de Python para construir instruções simples e, por fim, manipularemos um sistema *online* que possibilita a visualização passo a passo do que ocorre quando nossos códigos-fonte são executados, potencialmente facilitando a aprendizagem.

VOCÊ CONHECE?

Guido van Rossum é holandês, programador, matemático e criador da linguagem Python! Contratado pelo Google entre 2005 a 2012 e, em 2013, começou a trabalhar na Dropbox. Se aposentou em 2019, mas em 2020, com 64 anos, abandonou a aposentadoria para ingressar na divisão de desenvolvimento da Microsoft.

Fonte da imagem: https://twitter.com/gvanrossum/photo

2.2. Preparação do ambiente computacional

Usaremos um computador com sistema operacional Microsoft Windows 10™, mouse, teclado e conexão com a internet, pois este é o sistema operacional mais amplamente utilizado. Mas isso não é um requisito obrigatório e é possível acompanhar as aulas a partir de outros sistemas operacionais, como Linux ou Apple macOS™, por exemplo. Nos dois sistemas mencionados, o Python está pré-instalado por padrão, mas a versão pode variar de acordo com o sistema ou distribuição, sendo comum que haja mais de uma versão do Python instalada, pois alguns programas desses sistemas usam a versão obsoleta 2.7. Portanto, é importante verificar qual a versão de seu computador e, se necessário, realizar uma atualização. Este curso não é compatível com nenhuma versão de Python 2. Geralmente recomenda-se a instalação da versão estável mais recente do Python, porém, para esta disciplina, é suficiente qualquer uma a partir da 3.6.

2.2.1. Download e instalação do Python no Windows

Iniciaremos com o *download* do interpretador do Python juntamente com o IDLE, ambos são softwares gratuitos e de código aberto e serão instalados por meio do mesmo arquivo. Para isso basta acessar o site https://www.python.org, clicar na guia "Downloads" e, em seguida, no botão "Python 3.9.1" (atualmente a última versão estável disponível), como ilustrado na Figura 2.1.

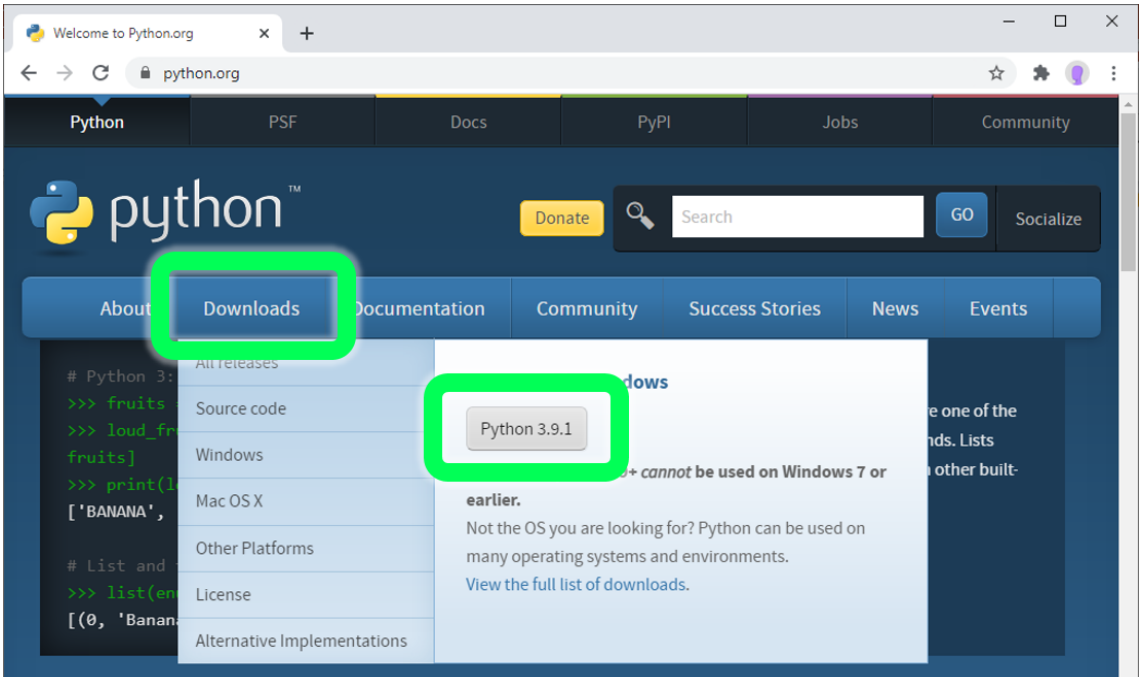

Figura 2.1: Site para *download* do Python. Fonte: Elaborado pelo autor.

Execute o arquivo baixado, preferencialmente como administrador, configure a primeira tela conforme ilustrado na Figura 2.2 e clique em "Customize installation".

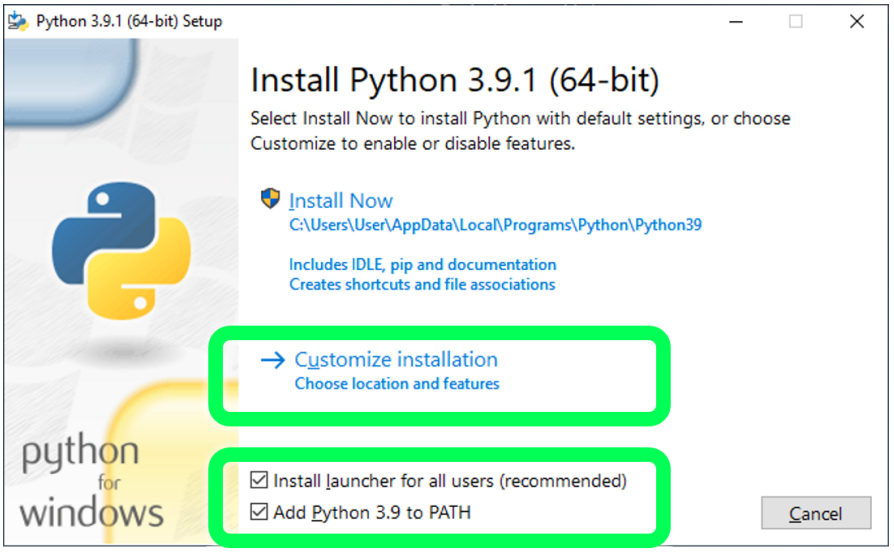

Figura 2.2: 1ª tela de instalação do Python. Fonte: Elaborado pelo autor.

Configure a segunda e a terceira tela de instalação conforme indicado pela Figura 2.3 e clique em "Next" e, em seguida, em "Install". Aguarde a conclusão da instalação e, na última tela, clique em "Close".

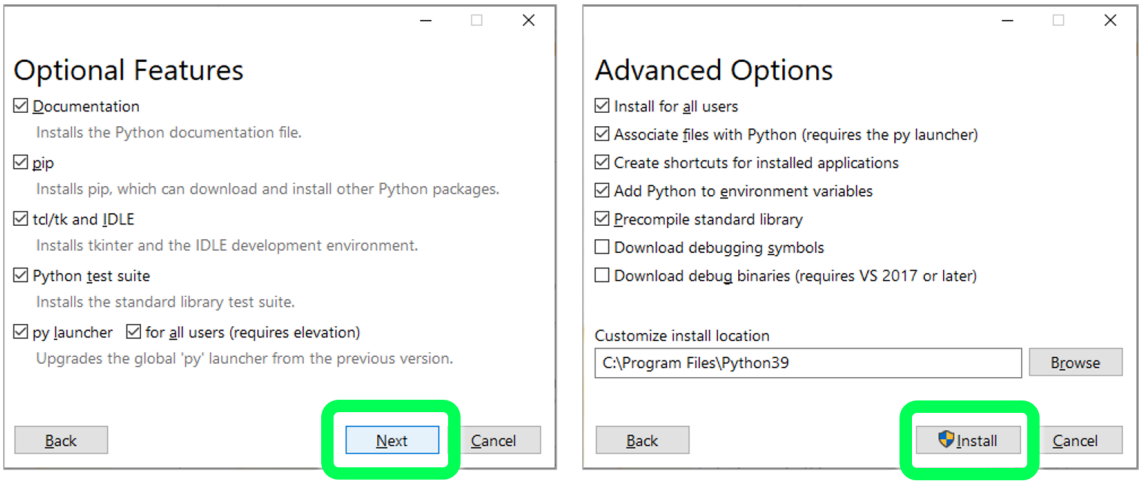

Figura 2.3: Recortes da 2ª e 3ª telas de instalação. Fonte: Elaborado pelo autor.

2.2.2. IDLE

O IDLE[1] (Ambiente Integrado de Desenvolvimento e Aprendizado, tradução nossa) é o ambiente em que programaremos e foi instalado juntamente com o interpretador do Python após no procedimento anterior. Este ambiente é programado na própria linguagem Python e funciona igualmente em outros sistemas operacionais. Para acessá-lo, procure-o no Menu Iniciar, como ilustrado na Figura 2.4.

[1] No original: *Integrated Development and Learning Environment*.

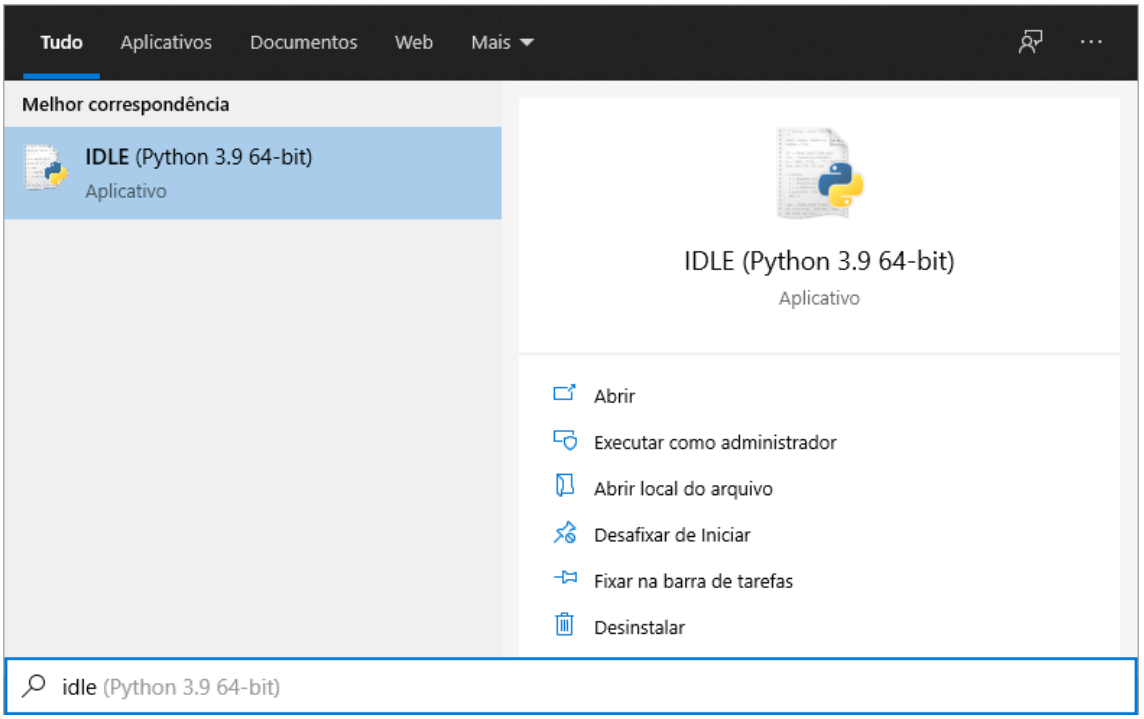

Figura 2.4: Acesso ao IDLE Python. Fonte: Elaborado pelo autor.

O IDLE possui dois módulos, o *interpretador interativo* e o *editor de código-fonte*, analisaremos as funcionalidades de cada um deles.

2.2.3. Interpretador interativo

Por padrão, ao clicar no atalho do IDLE no Menu Iniciar a tela exibida será do *interpretador interativo*, também conhecido como *Shell*, como ilustrado na Figura 2.5.

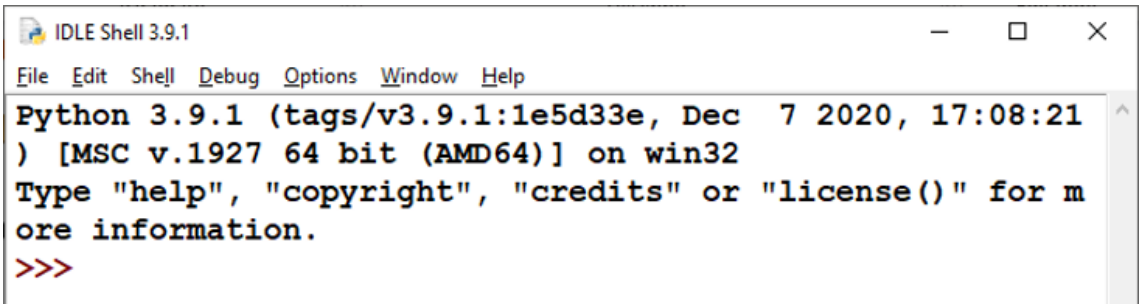

Figura 2.5: Interpretador interativo do IDLE Python (*Shell*). Fonte: Elaborado pelo autor.

Este módulo é usado principalmente para testar instruções de Python e executar programas construídos no editor. Repare na sequência de três sinais de maior (`>>>`) indicando que a *Shell* está pronta para receber novas instruções. A cada instrução, o programador deve teclar `[ENTER]` para executá-la e gerar o efeito correspondente. Na Figura 2.6 há um exemplo de uma instrução clássica em programação, sendo uma tradição que iniciantes em uma linguagem comecem seus estudos com a digitação dela.

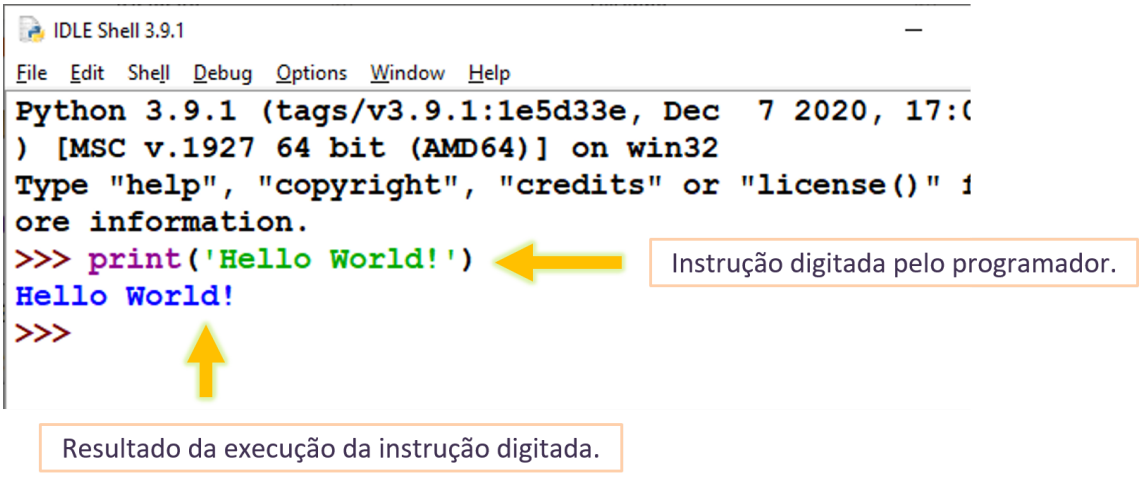

Figura 2.6: Execução de uma instrução inserida na *Shell*. Fonte: Elaborado pelo autor.

A *Shell* é bastante útil e possui atalhos e recursos interessantes, podendo, inclusive, substituir uma calculadora tradicional, veremos posteriormente como fazer isso. Alguns dos atalhos que facilitam a codificação são:

- Para regressar a última instruções inserida, posicione o cursor na linha com `>>>` e pressione as teclas `[ALT]+[P]`;
- Para avançar para a próxima instrução inserida, posicione o cursor na linha com `>>>` e pressione as teclas `[ALT]+[N]`;
- Para interromper a execução de uma instrução ou programa pressione as teclas `[CTRL]+[C]`.

VAMOS PRATICAR!

1) Execute mais instruções na *Shell*, por exemplo substituindo `'Hello World!'` pelo seu nome. Repita com conteúdos diferentes entre os apóstrofos. Utilize os dois primeiros atalhos para verificar se consegue regressar e avançar às instruções inseridas.

2) Refaça o exercício anterior, porém trocando apóstrofos por aspas. Depois elimine as aspas. Ocorreu algum erro? Você descobrirá a razão mais detalhadamente nas próximas aulas, por enquanto basta saber que todo texto deve estar entre apóstrofos ou aspas.

2.2.4. Editor de código-fonte

A *Shell* é muito útil para testar trechos de código, mas existem situações em que precisaremos construir um programa completo e guardar seu código-fonte para que possa ser editado e executado diversas vezes sem a necessidade de reescrevê-lo após encerrar o IDLE. Para isso existe o módulo *editor de código-fonte*. Para acessá-lo, clique no menu "File > New File" da janela da *Shell*. Veja a ilustração na Figura 2.7.

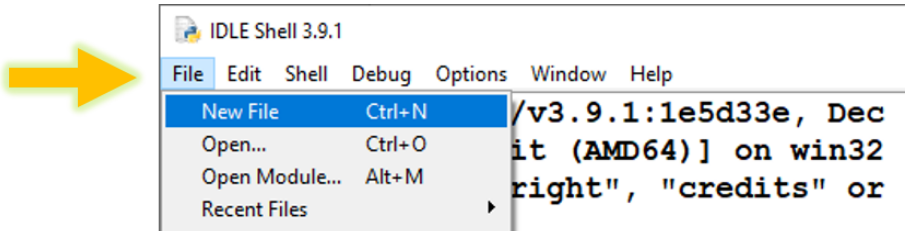

Figura 2.7: Menu para acesso ao editor de código-fonte. Fonte: Elaborado pelo autor.

Será aberta uma janela semelhante à do Bloco de Notas do Microsoft Windows em que, ao contrário da *Shell*, não há os três sinais de maior (`>>>`), como pode ser visto na Figura 2.8. É nesta janela que o programador traduzirá seu algoritmo em instruções para Python, gerando assim um código-fonte.

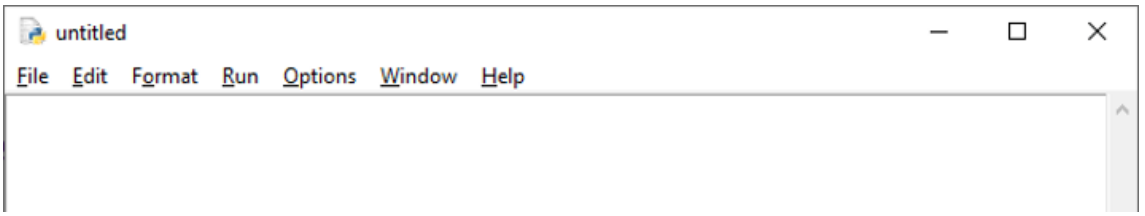

Figura 2.8: Módulo editor de código-fonte. Fonte: Elaborado pelo autor.

Insira a Codificação 2.1 no editor, de modo a deixá-lo como na Figura 2.9. Não se preocupe em entender o código completo, as instruções serão explicadas futuramente.

```
n1 = int(input('Número 1: '))
n2 = int(input('Número 2: '))
sub = n1 - n2
print('A subtração de', n1, '-', n2, 'é', sub)
```

Codificação 2.1: Programa para exibir a subtração de dois números dados pelo usuário.

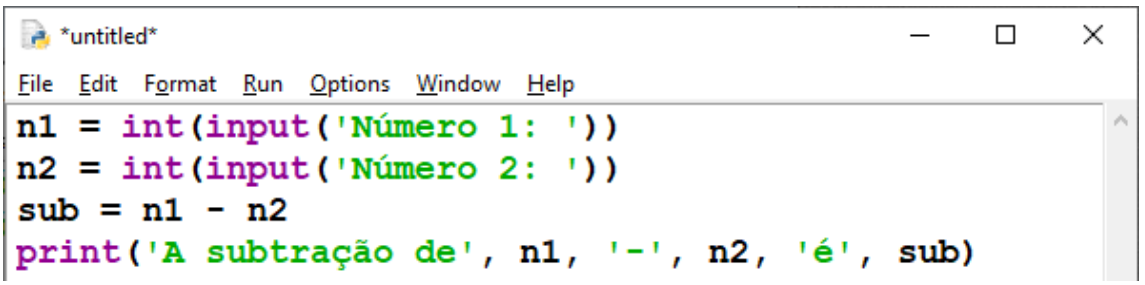

Figura 2.9: Editor preenchido com um código-fonte. Fonte: Elaborado pelo autor.

Cada caractere é importante, portanto tome cuidado para não esquecer de nenhum ou trocá-lo por acidente. Caso tenha dúvida se em Python letras maiúsculas e minúsculas são interpretadas como caracteres diferentes, a resposta é SIM! Python é uma linguagem *case-sensitive*, isso quer dizer que `n1` é completamente diferente de `N1`.

Repare que a barra de título da janela consta como “untitled”, indicando que o arquivo não foi salvo. Para executarmos o código-fonte e visualizar o programa funcionando, precisamos salvar este arquivo. Para isso, acesse o menu “File > Save”, defina o nome do arquivo e o local em que será salvo, conforme a Figura 2.10.

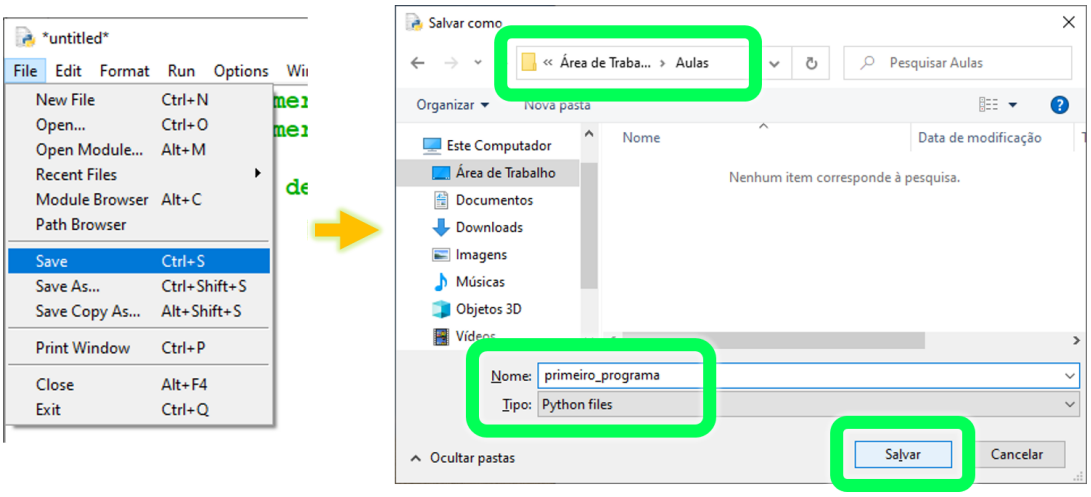

Figura 2.10: Salvamento do arquivo criado no editor. Fonte: Elaborado pelo autor.

Após o salvamento do arquivo, a barra de título será alterada para corresponder ao nome do arquivo e local em que foi salvo. Para executar o código-fonte e visualizar o programa funcionando, vá até o menu “Run > Run Module”. O procedimento de execução e o resultado, supondo as entradas `10` e `7`, estão ilustrados na Figura 2.11. Note que as saídas do programa estão em azul e as entradas do usuário em preto.

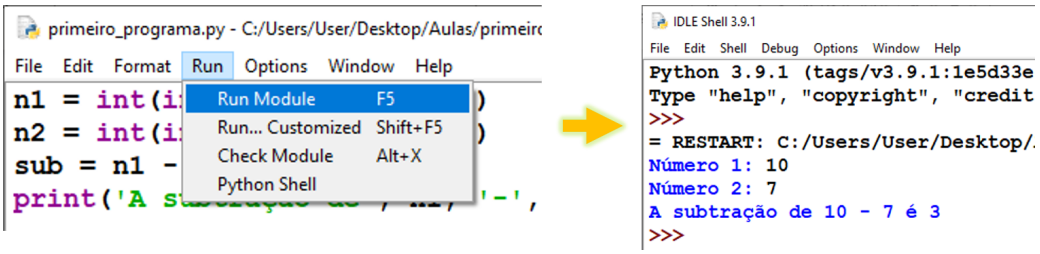

Figura 2.11: Execução do código-fonte e programa gerado. Fonte: Elaborado pelo autor.

Observe que a execução do programa, mesmo que solicitada no editor, é feita na *Shell*. Caso queira fazer alguma alteração no código, basta selecionar novamente a janela do editor, que provavelmente foi sobreposta pela da *Shell*, fazer as modificações e executá-lo novamente, o código-fonte será automaticamente salvo.

VOCÊ SABIA?

- Existe material oficial sobre o IDLE, com detalhes e outros recursos interessantes do ambiente. Interessado? Basta acessar o link: https://docs.python.org/3/library/idle.html

- Além do IDLE existem outras ferramentas que podem ser usadas para criar programas em Python, várias delas são citadas neste link: https://python.org.br/ferramentas/

2.3. Conceitos básicos de Python

Agora que temos conhecimento sobre o ambiente de programação que usaremos durante a disciplina, vamos aprender alguns conceitos básicos da linguagem Python.

2.3.1. Constantes ou Literais

Constantes são símbolos que representam valores e não podem ser alterados, são também chamadas de *literais* e geralmente são utilizadas em expressões. Exemplos de constantes em Python: `9`, `-541`, `3.1415`, `'estou aprendendo Python!'`, `"Olá Megan"`, `False` e `True`.

O Python possui alguns tipos de constantes, e cada tipo deve ser usado de acordo com o contexto do problema. Os quatro tipos que usaremos de início são:

- *Números inteiros (int):* valores numéricos que não possuem ponto decimal.

  Exemplos: `13`, `123456789`, `-65`, `0`, `0b1011`$_{\text{(base 2)}}$, `0o7`$_{\text{(base 8)}}$, `0xF`$_{\text{(base 16)}}$.

- *Números reais (float):* valores numéricos com ponto decimal ou escritos em notação científica. Note que é utilizado um ponto, não uma vírgula.

  Exemplos: `2.7345`, `.25`, `-65.0`, `0.0`, `6.02e-23`, `6.02E-23`, `2e1`.

- *Valores booleanos (bool):* também conhecidos como valores lógicos, existem apenas dois valores, um corresponde a *falso* e o outro a *verdadeiro*.

  Exemplos: `False` e `True`. Note que somente a primeira letra é maiúscula e não há aspas como nas *strings* a seguir.

- *Textos (string):* são cadeias de caracteres delimitadas por apóstrofos ou aspas. São usadas para representar textos (letras, palavras ou frases).

  Exemplos: `'Olá!'`, `'123'`, `"Python é 10"`, `"True"`, `"3+5"`, `'X'`.

2.3.2. Operadores

Em Python, assim como em outras linguagens de programação, operadores são símbolos pré-definidos que realizam uma operação sobre um ou mais operandos, produzindo um valor como resultado. O tipo do valor resultante dependerá do operador e dos operandos envolvidos. Quando um operador realiza uma operação entre 2 operandos, chamamos-o de binários, quando realiza uma operação com apenas 1 operando, chamamos-o de unários e, por fim, ternários quando envolvem 3 operandos.

2.3.2.1. Operadores aritméticos

Os operadores aritméticos, quando aplicados em operandos numéricos resultam valores numéricos, assim como na matemática. Observe esses operadores na Tabela 2.1.

Tabela 2.1: Operadores aritméticos do Python.

Operador	Descrição	Exemplos
`+` (binário)	Soma o primeiro operando com o segundo.	`7 + 4 → 11` `1 + 2.0 → 3.0`
`-` (binário)	Subtrai o segundo operando do primeiro.	`15 - 5 → 10` `5 - 15 → -10`
`+` (unário)	Mantém o sinal do operando à direita. Observação: é a função identidade.	`+ 3 → 3` `+ (-3) → -3`
`-` (unário)	Inverte o sinal do operando à direita.	`- (3) → -3` `- (-7) → 7`
`*`	Multiplica o primeiro operando pelo segundo.	`3 * 8 → 24`
`/`	Quociente da divisão real do primeiro operando pelo segundo.	`9 / 2 → 4.5` `9.0 / 2 → 4.5`
`//`	Quociente da divisão inteira do primeiro operando pelo segundo.	`9 // 2 → 4` `9.0 // 2 → 4.0`
`%`	Resto da divisão inteira do primeiro operando pelo segundo.	`9 % 2 → 1` `9.0 % 2 → 1.0`
`**`	Exponenciação, eleva o primeiro operando ao segundo.	`3 ** 2 → 9`

Note que, exceto pela divisão real, quando todos os operandos envolvidos na operação forem números inteiros o resultado será um número inteiro, porém, basta um operando real para que o resultado da operação resulte em um número real.

VAMOS PRATICAR!

Resolva as operações a seguir usando apenas lápis, papel e calculadora, em seguida confira os resultados inserindo as operações na *Shell* do Python.

a) `893 // 10`	b) `893 / 10`	c) `25.0 // 2`	d) `25.0 / 2`
e) `5678 % 1`	f) `5678 % 10`	g) `5678 % 100`	h) `5678 % 1000`
i) `5678 // 1`	j) `5678 // 10`	k) `5678 // 100`	l) `5678 // 1000`
m) `123 // 1000`	n) `123 / 1000`	o) `0 / 0`	p) `0 ** 0`
q) `1e3 + 5`	r) `0x10 - 3.5`	s) `9 ** 0.5`	t) `81 ** 0.5`

2.3.3. Variáveis

Uma variável é um espaço de memória associado a um identificador, ou seja, um nome, e serve para guardar valores que o programa poderá acessar e modificar. Toda variável possui um identificador único, de forma que possa ser referenciada pelo programador sem ambiguidade em qualquer parte do programa.

Em Python, uma variável é criada no momento em que um valor é atribuído a um identificador válido. A atribuição é feita colocando um identificador à esquerda de um sinal de igual e um valor à direita deste mesmo operador, conforme a Figura 2.12.

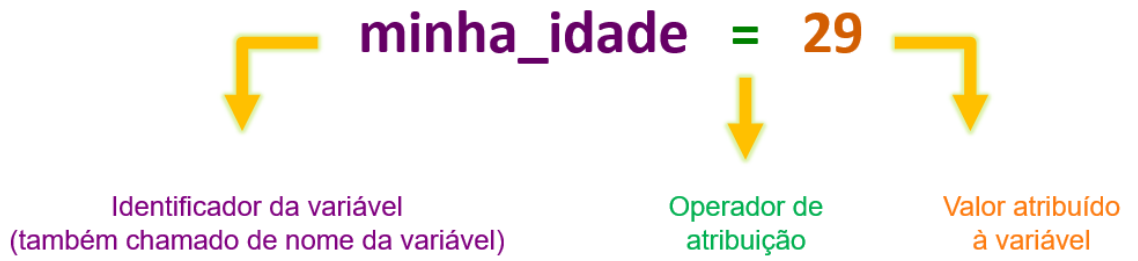

Figura 2.12: Atribuição de um valor a uma variável. Fonte: Elaborado pelo autor.

O conteúdo de uma variável pode "variar", ou seja, uma mesma variável pode guardar valores diferentes em momentos diferentes de um programa. Lembre-se: uma variável só guarda um valor por vez, portanto a cada nova atribuição o valor atual será sobrescrito pelo novo. Execute a Codificação 2.2 na *Shell* e reflita sobre os resultados.

```
>>> a = True
>>> type(a)
>>> a = 123
>>> a = 'linda casa amarela'
>>> a = 4.40
>>> type(a)
```

Codificação 2.2: Uma mesma variável recebendo valores diferentes.

Um destaque importante da linguagem Python é que o tipo do dado está relacionado ao valor atribuído e não a variável que recebeu esse valor, diferentemente de outras linguagens de programação como C, C++ e Java, dentre outras. Note também que uma mesma variável pode armazenar, inclusive, valores de tipos distintos.

2.3.4. Identificadores

Um identificador de uma variável, também referido como nome, é formado por uma sequência de um ou mais caracteres, de acordo com as seguintes regras:

- Simplificadamente, pode conter apenas combinações de letras, dígitos e sublinhados (não pode conter símbolos especiais como &, ¨, %, $, #, @, !);

- Não pode iniciar com dígito;
- Não pode ser uma palavra reservada (abordaremos a seguir).

Recomenda-se criar identificadores concisos, porém descritivos:

- `idade` é melhor que `i`;
- `tamanho_nome` é melhor que `tamanho_do_nome_da_pessoa`.

Evite abreviar exageradamente, escreva por extenso para melhorar a legibilidade:

- `sobrenome` é melhor que `sbrnome`;
- `litros` é melhor que `ltrs`;
- `data_criacao` é melhor que `dt_cri`.

Existe um guia de estilo para programação em Python, criado e mantido pela própria comunidade mundial da linguagem, com o intuito de fornecer recomendações para facilitar a leitura do código. Esse guia é conhecido como PEP8[2] e embasou a forma como foram escritos os códigos deste material.

No decorrer da disciplina, apresentaremos as partes relevantes do guia para que a sua utilização ocorra de maneira natural, não sendo necessário decorá-lo integralmente. Por enquanto, temos mais uma recomendação relativa à criação de identificadores:

- Use apenas letras minúsculas e sem acentuação, separando as palavras com um sublinhado para melhorar a legibilidade.

Vale relembrar que o Python é uma linguagem *case-sensitive*, diferenciando letras maiúsculas de minúsculas, portanto o identificador `meu_nome` não é o mesmo que `Meu_Nome` ou `MEU_NOME`.

2.3.5. Palavras reservadas

Python possui palavras reservadas, as *palavras-chave*, chamadas em inglês de *keywords*, e não podem ser usadas como identificadores, pois têm papel especial na linguagem. O Python 3.9.1 possui 36 *keywords*, porém a quantidade varia entre as versões, para saber quais são as de sua versão execute na *Shell* a Codificação 2.3.

```
>>> help('keywords')
```

Codificação 2.3: Instrução que exibe as *keywords* da versão usada do Python.

2.3.6. Comentários

Podemos inserir comentários em nossos códigos-fonte, algo útil para ajudar tanto outros programadores que lerão nossos códigos quanto a nós mesmos para recordar a razão de determinadas instruções. Comentários são ignorados na execução do

[2] PEP 8 -- Style Guide for Python Code | Python.org

programa, portanto é algo para auxiliar humanos, não computadores. Linhas iniciadas com # são interpretadas como comentários, conforme a Codificação 2.4.

```
a = 1045.00  # salário mínimo de 2020
b = 1100.00  # salário mínimo de 2021
c = b - a    # aumento do salário mínimo
print('O salário mínimo aumentou:', c, 'reais')
```

Codificação 2.4: Exemplos de comentários em um código-fonte Python.

Ainda não discutimos o que é a instrução `print()`, isso será feito em outra aula, mas nosso palpite é que você já é capaz de imaginar o que ela faz.

2.4. Python Tutor

Há um ambiente gratuito que pode ajudar a entender o que ocorre na memória e na saída de dados após a execução de cada instrução do código-fonte, é o *Python Tutor*. Esse sistema *on-line*, gera representações gráficas relacionadas ao código-fonte e são dinamicamente atualizadas a cada instrução executada.

O Python Tutor pode ser acessado no endereço http://pythontutor.com/. Para começar a usá-lo, clique em "Start visualizing your code now". Insira seu código-fonte e clique no botão "Visualize Execution", conforme a Figura 2.13. Note que você pode tanto copiar e colar o código quanto digitar diretamente na caixa de texto.

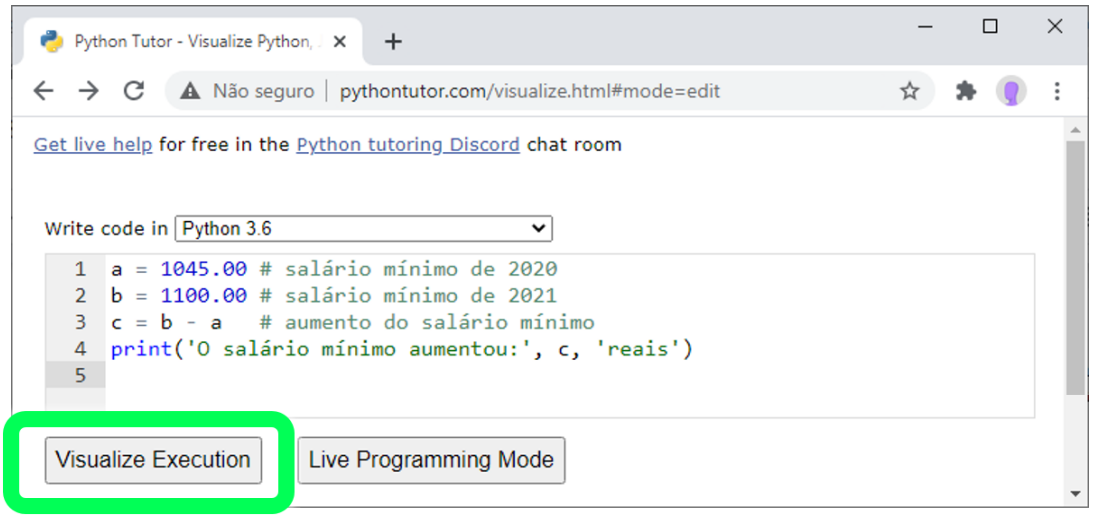

Figura 2.13: Python Tutor antes da execução do código-fonte.
Fonte: Elaborado pelo autor.

Use os botões "< Prev" e "Next >" para controlar a execução. Veja no canto inferior-direito da tela as variáveis criadas, com seus respectivos valores, e no canto superior-direito o que foi exibido na tela, conforme a Figura 2.14. A seta verde-claro indica a instrução que acabou de ser executada e a vermelha indica a próxima instrução.

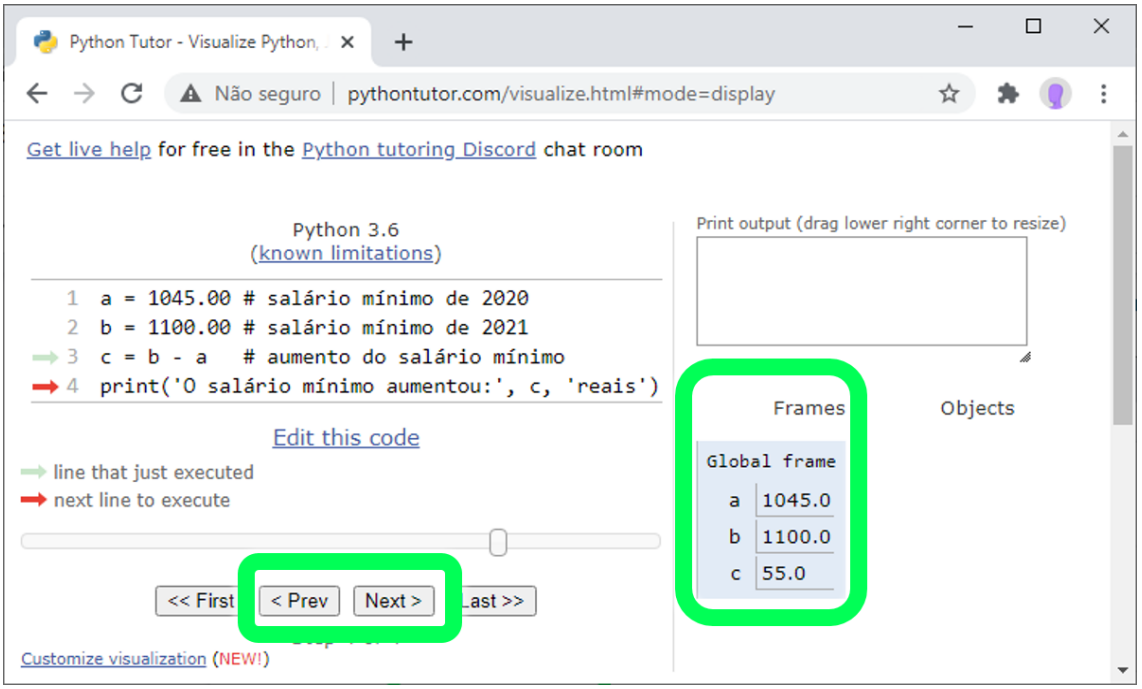

Figura 2.14: Python Tutor após a execução de algumas instruções do código-fonte.
Fonte: Elaborado pelo autor.

Bibliografia e referências

PSF. **A Referência da Linguagem Python**. 2021. Disponível em: <https://docs.python.org/pt-br/3/reference/index.html>. Acesso em: 28 fev. 2021.

PSF. **Lexical analysis**. 2020. Disponível em: <https://docs.python.org/3/reference/lexical_analysis.html>. Acesso em: 21 jan. 2021.

PSF. **Expressions**. 2020. Disponível em: <https://docs.python.org/3/reference/expressions.html>. Acesso em: 21 jan. 2021.

STURTZ, J. Operators and Expressions in Python. **Real Python**, 2018. Disponível em: <https://realpython.com/python-operators-expressions/>. Acesso em: 21 jan. 2021.