



Interativa

Introdução à Programação Estruturada

Autor: Prof. Ricardo Leandro Piantola da Silva

Colaboradoras: Profa. Vanessa Santos Lessa

Profa. Larissa Rodrigues Damiani

Professor conteudista: Ricardo Leandro Piantola da Silva

Ricardo Leandro Piantola da Silva é mestre e doutor em Engenharia de Computação pela Escola Politécnica da Universidade de São Paulo (USP), técnico em Processamento de Dados pelo Instituto Tecnológico de Barueri, bacharel em Ciência da Computação pela Unifief e especialista em Segurança na Internet pela Universidade Federal do Estado do Rio de Janeiro. Desde de 2005, ministra várias disciplinas nos cursos de Ciência da Computação e Sistemas de Informação da Universidade Paulista (UNIP). Trabalhou diretamente com programação de computadores e engenharia de software em empresas como HP e IBM.

Dados Internacionais de Catalogação na Publicação (CIP)

S586i Silva, Ricardo Leandro Piantola da.

Introdução à Programação Estruturada / Ricardo Leandro Piantola da Silva. – São Paulo: Editora Sol, 2021.

152 p., il.

Nota: este volume está publicado nos Cadernos de Estudos e Pesquisas da UNIP, Série Didática, ISSN 1517-9230.

1. Python. 2. Matriz. 3. Dados. I. Título.

CDU 681.3.01

U512.44 – 21

Prof. Dr. João Carlos Di Genio
Reitor

Prof. Fábio Romeu de Carvalho
Vice-Reitor de Planejamento, Administração e Finanças

Profa. Melânia Dalla Torre
Vice-Reitora de Unidades Universitárias

Profa. Dra. Marília Ancona-Lopez
Vice-Reitora de Pós-Graduação e Pesquisa

Profa. Dra. Marília Ancona-Lopez
Vice-Reitora de Graduação

Unip Interativa – EaD

Profa. Elisabete Brihy
Prof. Marcello Vannini
Prof. Dr. Luiz Felipe Scabar
Prof. Ivan Daliberto Frugoli

Material Didático – EaD

Comissão editorial:

Dra. Angélica L. Carlini (UNIP)
Dr. Ivan Dias da Motta (CESUMAR)
Dra. Kátia Mosorov Alonso (UFMT)

Apoio:

Profa. Cláudia Regina Baptista – EaD
Profa. Deise Alcantara Carreiro – Comissão de Qualificação e Avaliação de Cursos

Projeto gráfico:

Prof. Alexandre Ponzetto

Revisão:

Lucas Ricardi
Vera Saad

Sumário

Introdução à Programação Estruturada

| | |
|--------------------|---|
| APRESENTAÇÃO | 7 |
| INTRODUÇÃO | 8 |

Unidade I

| | |
|--|----|
| 1 APRESENTANDO O PYTHON | 9 |
| 1.1 Conceitos básicos | 9 |
| 1.2 Instalação do Python e IDE | 10 |
| 1.2.1 Instalação do Python no Windows | 11 |
| 1.2.2 Instalação do Python no MacOS X | 15 |
| 1.2.3 Instalação do Python no Linux | 16 |
| 1.2.4 Ambiente de desenvolvimento Visual Studio Code | 18 |
| 1.3 Exemplos de programas em Python | 19 |
| 2 OPERADORES, EXPRESSÕES E VARIÁVEIS | 23 |
| 2.1 Conceito de variável | 23 |
| 2.2 Constantes e números | 25 |
| 2.3 Expressões | 26 |
| 2.4 Operadores | 27 |
| 2.4.1 Operadores aritméticos | 27 |
| 2.4.2 Operadores relacionais | 27 |
| 2.4.3 Operadores lógicos | 28 |
| 2.4.4 Operadores de atribuição | 29 |
| 2.4.5 Precedência dos operadores | 30 |
| 2.5 Estrutura sequencial | 31 |
| 2.6 Exercícios sugeridos | 31 |
| 2.6.1 Respostas dos exercícios | 33 |

Unidade II

| | |
|--------------------------------|----|
| 3 ESTRUTURA CONDICIONAL | 49 |
| 3.1 If, elif e else | 49 |
| 3.2 Pass | 53 |
| 3.3 Switch-case | 53 |
| 4 ESTRUTURA DE REPETIÇÃO | 54 |
| 4.1 While | 55 |
| 4.1.1 Break/continue | 56 |

| | |
|-------------------------------------|----|
| 4.2 For..... | 57 |
| 4.3 Exercícios sugeridos | 61 |
| 4.3.1 Respostas dos exercícios..... | 63 |

Unidade III

| | |
|---|-----|
| 5 FUNÇÕES, STRINGS E ARQUIVOS..... | 77 |
| 5.1 Definindo funções..... | 77 |
| 5.1.1 Parâmetros..... | 78 |
| 5.1.2 Variáveis locais e globais | 79 |
| 5.1.3 Argumento default..... | 81 |
| 5.2 Manipulação de strings | 82 |
| 5.3 Abertura, leitura e gravação em arquivos..... | 88 |
| 6 MATRIZES E ESTRUTURA DE DADOS | 91 |
| 6.1 Manipulação básica de matrizes..... | 91 |
| 6.2 Estrutura de dados..... | 99 |
| 6.2.1 Conceito de lista..... | 100 |
| 6.2.2 Tupla..... | 101 |
| 6.2.3 Sequência..... | 102 |
| 6.2.4 Conjunto (set)..... | 104 |
| 6.3 Exercícios sugeridos | 106 |
| 6.3.1 Respostas dos exercícios..... | 107 |

Unidade IV

| | |
|---|-----|
| 7 LISTAS E DICIONÁRIOS..... | 117 |
| 7.1 Principais métodos para manipulação de listas | 117 |
| 7.2 Conceito de dicionários..... | 120 |
| 7.2.1 Principais métodos para manipulação de dicionários..... | 124 |
| 8 MÓDULOS E BANCO DE DADOS..... | 128 |
| 8.1 Introdução aos módulos..... | 128 |
| 8.1.1 SYS..... | 130 |
| 8.1.2 Instalando módulos | 131 |
| 8.1.3 From...Import | 132 |
| 8.1.4 Arquivos byte-compiled .pyc | 133 |
| 8.2 Banco de dados..... | 133 |
| 8.2.1 Conexão com banco de dados MySQL..... | 133 |
| 8.2.2 Leitura, gravação, alteração e exclusão..... | 134 |
| 8.3 Exercícios sugeridos | 139 |
| 8.3.1 Respostas dos exercícios..... | 140 |

APRESENTAÇÃO

Caro aluno,

A programação estruturada é um padrão da engenharia de software desenvolvido no final da década de 1950, com enfoque em sequência, decisão, repetição, abstração de dados e modularização, para melhor estruturação do código escrito e lido por humanos. Ela foi o padrão de escrita de software dominante até a chegada da programação orientada a objeto. O paradigma de orientação a objeto não substitui a programação estruturada como muitos pensam; ela é fundamental para abstração orientada a objeto.

Esta disciplina visa ensinar a base para a programação de computadores. Todos os conceitos aprendidos nela são fundamentais e complementares para o aprendizado de outros padrões de programação. Esses fundamentos serão levados para toda a carreira do cientista da computação nas áreas de desenvolvimento de software, ciência dos dados, inteligência artificial, entre outras.

O livro-texto está organizado da seguinte forma: a unidade I apresenta a linguagem de programação Python e os principais conceitos de programação. É descrito o passo a passo da instalação do Python nos principais sistemas operacionais e do ambiente de desenvolvimento integrado (IDE). É exposto como os computadores guardam informações em sua memória através do conceito de variáveis e constantes. Também é tratado nessa unidade como os computadores avaliam as expressões lógicas e matemáticas.

Por sua vez, a unidade II demonstra na prática os principais conceitos sobre programação estruturada: estrutura condicional e estrutura de repetição.

Já a unidade III expõe o conceito de função, que é fundamental para a organização do código e entendimento de sub-rotinas em programas estruturados. Além disso, é ensinada a manipulação de strings e arquivos. A unidade termina com o início dos estudos das estruturas de dados básicas do Python.

Por fim, a unidade VI apresenta um foco maior nas estruturas de dados lista e dicionário. Também estudam-se nessa unidade dois temas um pouco mais avançados: os módulos em Python e a conexão e manipulação de dados em um banco de dados MySQL.

INTRODUÇÃO

A programação de computadores é fundamental para a carreira do cientista da computação. É por meio dela que passamos instruções para qualquer dispositivo computacional executar tarefas. Quanto melhor o conhecimento em programação, mais eficiente será a resolução dos problemas pela máquina. A computação existe para resolver problemas do mundo real. Tais problemas são abstraídos para o mundo virtual, a computação os resolve e então tem-se o resultado na forma da solução do problema no mundo real.

Esta disciplina aborda a programação de computadores por meio do padrão de organização de código utilizando estruturas como estrutura de bloco, laços de repetição, condicionais e sub-rotinas.

A Teoria do Programa Estruturado oferece vantagens práticas de entendimento e organização de código por um humano. Ela é usada por todas as linguagens de computadores de alto nível já criadas (retirando as linguagens de baixo nível, mais próximas da máquina, como o Assembly).

Nesta disciplina, foi escolhida a linguagem Python para demonstrar os conceitos de programação estruturada. O Python é uma linguagem que vem ganhando muita importância na área de computação, por sua simplicidade e flexibilidade de uso em diversas áreas de tecnologia da informação.

A disciplina tem como objetivo não só capacitar o futuro programador e desenvolvedor de soluções, mas preparar o aluno para disciplinas que utilizam o paradigma de programação estruturada ou Python, como as de programação orientada a objeto, processamento de imagens, inteligência artificial, ciência de dados e várias outras que necessitarão de conhecimento de programação. Para alcançar esses objetivos, é necessário compreender o papel fundamental da programação estruturada no contexto da lógica e computação, o que pretendemos com este livro-texto.

Boa jornada durante a leitura!

Unidade I

1 APRESENTANDO O PYTHON

1.1 Conceitos básicos

Python é uma linguagem de programação bem popular e fácil de aprender e de lidar cujo código é simples de ler e escrever. Também é expressiva, por isso conseguimos escrever em poucas linhas o que em outras linguagens escreveríamos em muitas. É uma linguagem de alto nível, multiparadigma, interpretada, imperativa, de tipagem dinâmica e forte, conceitos abordados com detalhes em outras disciplinas. No decorrer desta seção, falaremos brevemente de cada um deles, começando com o conceito de programação imperativa; nela, são dadas ordens em forma de comandos sequenciais que a máquina executará.

Criado por Guido van Rossum em 1991, hoje o Python é desenvolvido pela comunidade e é uma linguagem aberta e livre para uso, gerenciada pela Python Software Foundation.

O Python é uma linguagem de propósito geral, ou seja, ela não foi projetada para um domínio de aplicação específico, como para criar aplicações web somente. Tanto pode criar aplicações para web como para desktop, sistemas distribuídos, smartphones, inteligência artificial, ciência de dados e outros domínios. Como é também uma linguagem multiparadigma, ela suporta programação orientada a objeto, funcional e programação estruturada.

O nome Python não vem do gênero de cobras piton, como muitos pensam. Ele é homenagem a um grupo humorístico britânico chamado Monty Python.

Como o Python é uma linguagem de programação interpretada, sua execução é geralmente feita por meio de um programa executável que leva como argumento de entrada um arquivo texto com o código do programa. O executável é o interpretador Python, que existe em versões para a maioria dos sistemas operacionais, o que inclui MacOS, Linux e Windows.

Na próxima seção será abordada a instalação do pacote padrão do Python, que nos permitirá programar nessa linguagem. Também abordaremos a instalação do Visual Studio Code, que é um editor de código-fonte multiplataforma da Microsoft.



Observação

Linguagem de programação de alto nível é a linguagem em que o foco da escrita é o entendimento humano, assim os detalhes de máquina são abstraídos.

Tipagem forte é a característica da linguagem que não permite que um dado seja tratado como outro tipo de dado.

Tipagem dinâmica é a característica da linguagem em escolher, em tempo de execução, o tipo de dado conforme o valor atribuído à variável.



Saiba mais

Para saber mais sobre a história da linguagem de programação Python, visite o site:

PYTHON BRASIL. *História do Python*. [s.d.]. Disponível em: <https://cutt.ly/An6gum0>. Acesso em: 24 jun. 2021.

1.2 Instalação do Python e IDE

Atualmente, estamos na versão 3 do Python. Existem grandes modificações entre programas escritos na versão 1 e 2, o que pode fazer com que esses programas não executem ou executem e produzam resultados incorretos na versão 3.

Na próxima seção, veremos a instalação passo a passo do Python 3 no Windows.



Saiba mais

Não é necessário instalar o Python 3; é possível utilizar um interpretador on-line. Segue o link para o Colab da Google:

GOOGLE. *O que é o Colaboratory?* [s.d.]. Disponível em: <https://cutt.ly/An6gEMF>. Acesso em: 24 jun. 2021.

1.2.1 Instalação do Python no Windows

Primeiramente, é necessário baixar o instalador. Acesse o site oficial <https://www.python.org/downloads/> e clique em download, como mostrado na figura a seguir:

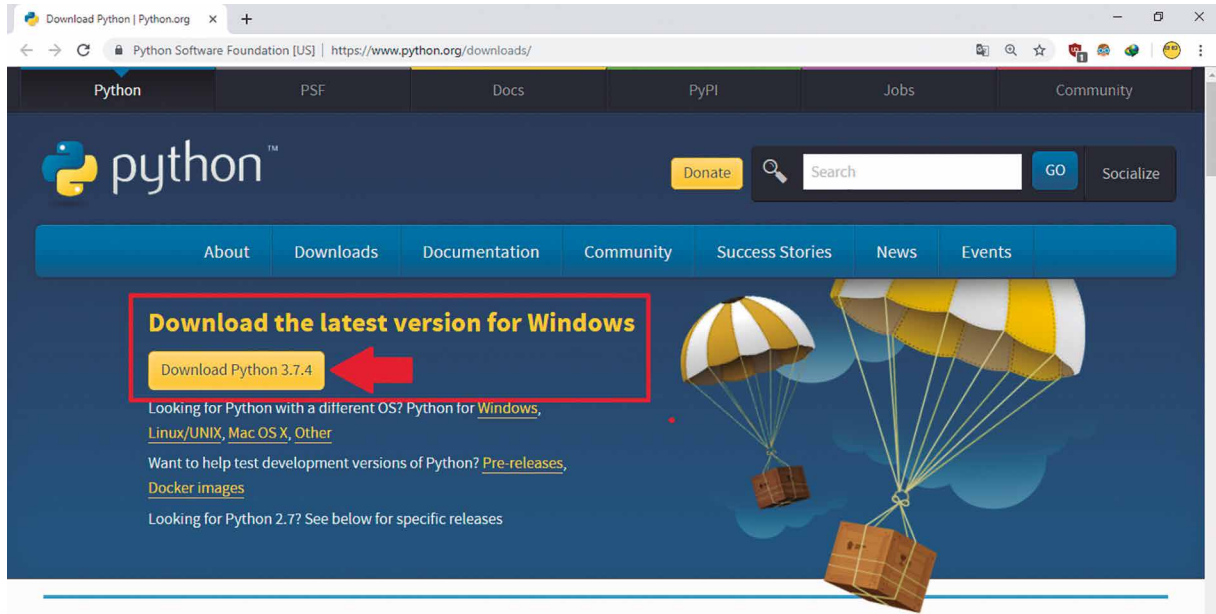


Figura 1 – Download Python 3 para Windows 32 bits

Disponível em: <https://bit.ly/36C6def>. Acesso em: 15 jul. 2021.

Isso fará o download do Python 3 para sistemas de 32 bits. Para o instalador de 64 bits, acesse e selecione o instalador de 64 bits apropriado, como mostrado na figura 2 a seguir.



Saiba mais

Para saber se o sistema Windows é de 32 ou 64 bits, acesse o suporte Microsoft oficial:

MICROSOFT. *Versões de 32 bits e 64 bits do Windows: perguntas frequentes.* [s.d.]. Disponível em: <https://cutt.ly/yn6hisx>. Acesso em: 24 jun. 2021.

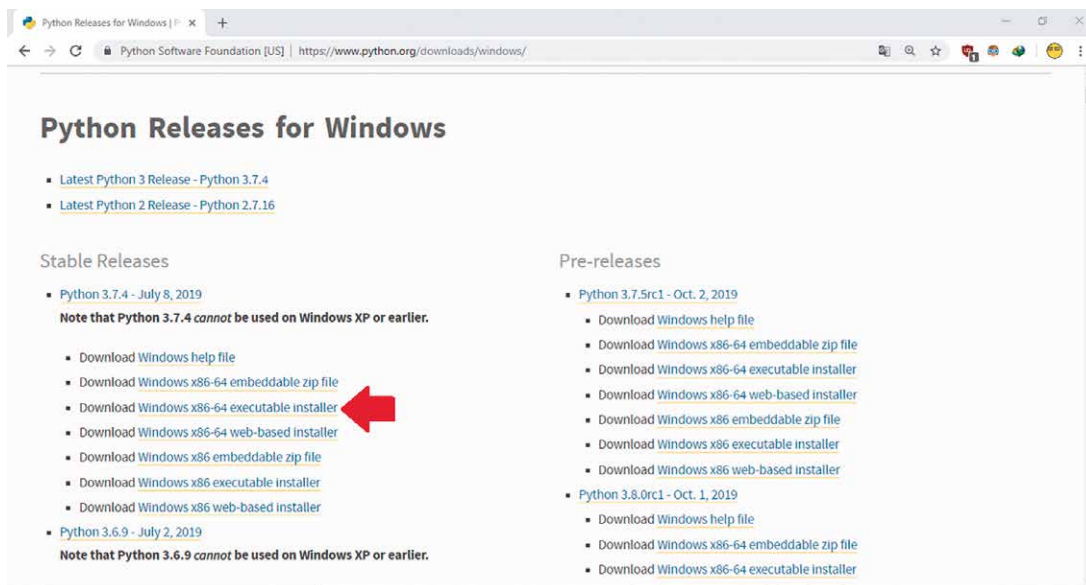


Figura 2 – Download Python 3 para Windows 64 bits

Disponível em: <https://bit.ly/3B7jc5F>. Acesso em: 15 jul. 2021.

Após fazer download do instalador do Windows (32 ou 64 bits), clique duas vezes nele para executá-lo e iniciar o assistente de instalação, como na figura a seguir:



Figura 3 – Instalador do Python 3 para Windows

Disponível em: <https://bit.ly/3xHsJOV>. Acesso em: 15 jul. 2021.

O processo de instalação é automático, conforme a figura anterior.

1. Selecione a opção "Add Python to PATH".
2. Depois clique em "Install Now".

Agora é só aguardar enquanto o instalador termina o processo de instalação. Quando a imagem da figura a seguir aparecer, click em "Close".



Figura 4 – Instalação do Python 3 para Windows concluída

Disponível em: <https://bit.ly/3kkwuWq>. Acesso em: 15 jul. 2021.

Agora, para verificar se a instalação ocorreu com sucesso, pesquise no menu iniciar por "cmd". Para abrir o prompt de comando, clique duas vezes, como mostra a figura 5 a seguir:

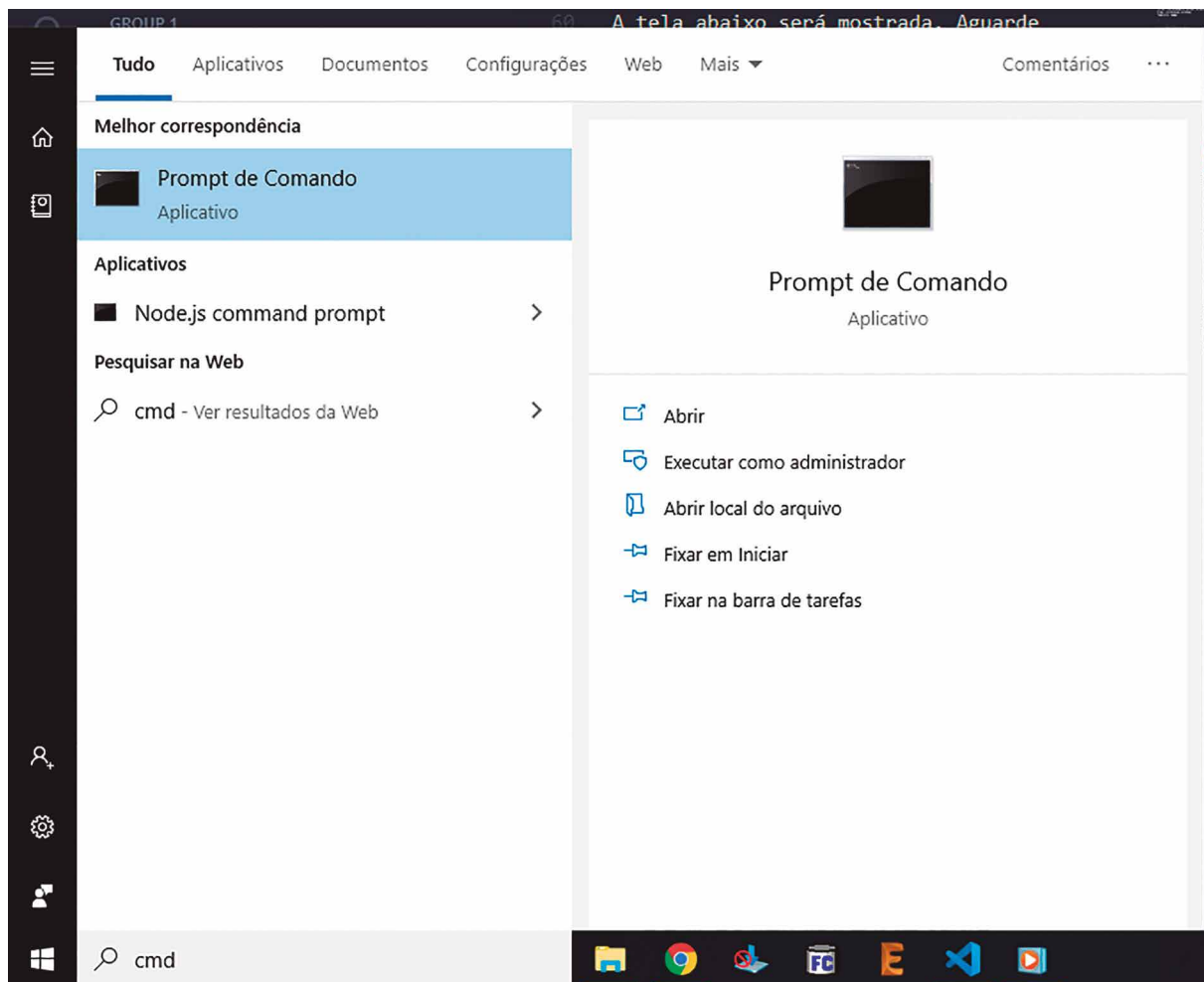


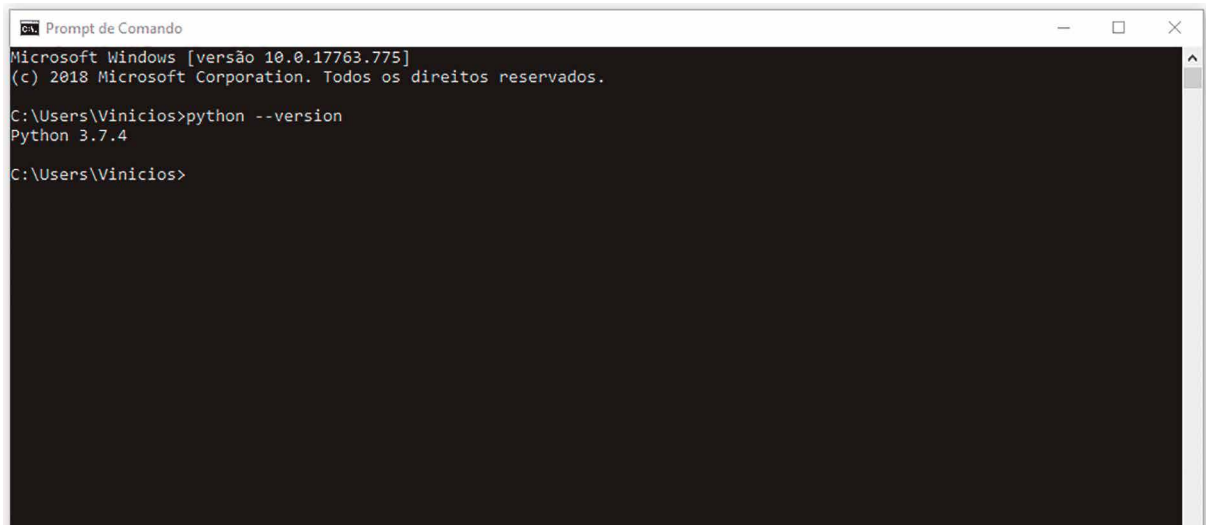
Figura 5 – Abrindo o prompt de comando do Windows

Disponível em: <https://bit.ly/3eoKwCS>. Acesso em: 15 jul. 2021.

Digite no prompt de comando:

```
python --version
```

O comando retornará qual é a versão do Python que foi instalada em sua máquina, como pode ser visto na figura 6 a seguir:



```
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>
```

Figura 6 – Verificando a instalação do Python no Windows

Disponível em: <https://bit.ly/3hE36ZF>. Acesso em: 15 jul. 2021.



Observação

No momento atual, a versão do Python é 3.9.1, porém na figura aparece 3.7.4. O importante é a versão principal, o primeiro número que aparece, que é 3.

1.2.2 Instalação do Python no MacOS X

Essa seção é dedicada à instalação do Python no sistema operacional MacOS. Fique à vontade para pular essa seção caso utilize Windows ou Linux.

Antes de começar, vamos verificar se já tem o Python instalado no sistema. Se você usa MacOS 10.2 ou superior, provavelmente já existe alguma versão do Python instalada por padrão.

Para verificar, digite em um terminal o comando:

```
$ which python
```

ou

```
$ which python3
```

Caso o comando tenha retornado algo parecido como `/usr/bin/python`, quer dizer que o Python já está instalado. Verifique a versão digitando `python -version`.

Caso não tenha o Python instalado no sistema, antes de instalá-lo é necessário executar a instalação do XCode, que pode ser baixado na App Store, do pacote para desenvolvimento em linha de comando e dos gerenciadores de pacotes Homebrew e Pip.

Para instalar as ferramentas de linha de comando, digite em um 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
```

1.2.3 Instalação do Python no Linux

Esta seção é dedicada à instalação do Python no sistema operacional Linux. Fique à vontade para pular esta seção caso utilize Windows ou MacOS X. Antes de começar, vamos verificar se já tem o Python instalado no sistema. Se você usa GNU/Linux, provavelmente já existe alguma versão do Python instalada por padrão.

Para verificar, digite em um terminal o comando:

```
$ which python
```

ou

```
$ which python3
```

Caso o comando tenha retornado algo parecido como `/usr/bin/python`, quer dizer que o Python já está instalado. Verifique a versão digitando `python --version`. Caso retorne algo como `which: no python in (/usr/local/sbin:/usr/local/bin:/usr/bin:/usr...)`, é necessário instalar o Python.

No caso do Linux, são utilizados gerenciadores de pacotes para instalação de programas; os mais comuns são o Apt-Get (Debian, Ubuntu) e o Yum (RedHat, CentOS).

Instalação por gerenciadores de pacotes

Os gerenciadores de pacotes mais comuns são Apt-Get (Debian, Ubuntu) e Yum (RedHat, CentOS). Caso sua distribuição utilize um gerenciador de pacotes diferente, acesse a página de downloads do Python.

Veja o comando para Apt-Get. No terminal, digite:

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

Para instalar o gerenciador de pacotes Python Pip:

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

Veja o comando para Yum. No terminal, digite:

```
$ sudo yum install python3
```

Para instalar o gerenciador de pacotes Python Pip:

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

Após a instalação e aberto um terminal, basta digitar: `python`

Assim aparecerá o terminal do Python com o prompt `>>>`

Para testar, digite: `print("olá mundo!")`

Parabéns pelo seu primeiro programa escrito em Python!



Saiba mais

Conheça o Ambiente de Desenvolvimento Integrado (IDE) Thonny em:

Disponível em: <https://cutt.ly/LmwxEw>. Acesso em: 13 jul. 2021.

1.2.4 Ambiente de desenvolvimento Visual Studio Code

Além do uso do próprio terminal Python para programação (basta digitar python na linha de comando do Windows ou terminal Linux/MacOS), é possível usar um IDE (Integrated Development Environment ou Ambiente de Desenvolvimento Integrado), ou até mesmo o Bloco de Notas para programar. Um dos softwares mais usados atualmente é o Visual Studio Code, pois ele é um editor de código-fonte que contém muitas facilidades para escrita dos programas e depuração (processo de procura de erros de programação). Segue link para instalação: <https://code.visualstudio.com/download>.

A figura a seguir mostra o primeiro_programa.py no Visual Studio Code. A tela é dividida em duas partes: na parte de cima há o programa com uma linha de código e na parte de baixo vemos a saída no console do depurador.

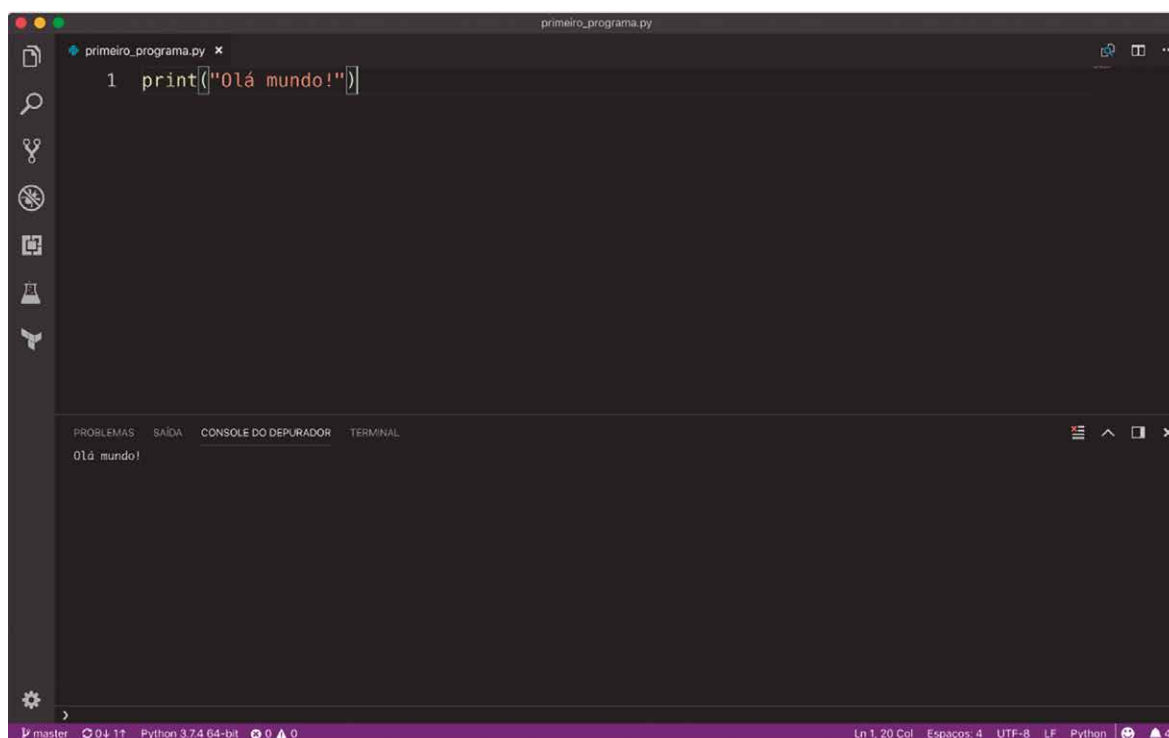


Figura 7 – Programa em Python no Visual Studio Code



Saiba mais

O Visual Studio Code foi lançado sob a licença MIT e o seu código-fonte foi postado no GitHub. Ele é um software de uso gratuito.

Para saber mais sobre a licença MIT, acesse:

Disponível em: <https://cutt.ly/imwfdgx>. Acesso em: 13 jul. 2021.

1.3 Exemplos de programas em Python

Nesta seção são apresentados três exemplos de programas simples em Python. Não é a intenção aqui entrar em detalhes sobre o funcionamento dos programas e suas instruções; as explicações e detalhes virão nos próximos tópicos. Brevemente, vale ressaltar alguns elementos dos exemplos. O primeiro elemento a ser notado são os comentários do programador, linhas que iniciam com o caractere `#`. Os comentários são ignorados pelo Python e servem de documentação. Os pontos nos números, como 1.8, são nossa vírgula em números reais, ou seja, 1,8. O último elemento é a função `print()`, que imprime caracteres na saída padrão, no caso, a tela do computador.

Exemplo 1:

```
# Este programa soma dois números

numero1 = 1.9

numero2 = 5.5

# Somando dois números

soma = numero1 + numero2

# Mostra a soma

print('A soma de {0} e {1} é {2}'.format(numero1, numero2, soma))
```

Executaremos o exemplo 1 no Visual Studio Code:

1. Abra o Visual Studio Code e selecione File -> New file.
2. Aparecerá uma mensagem escrita em azul para selecionar uma linguagem, clique e selecione Python.
3. Na primeira vez que usar o Visual Studio Code, será necessário instalar uma extensão para utilizar o Python. Nesse caso, aparecerá uma mensagem no canto direito inferior. Clique em instalar.
4. Após a instalação da extensão, outra mensagem no canto direito inferior aparecerá para recarregar a janela. Clique nela.
5. Uma mensagem no canto direito inferior aparecerá para selecionar o interpretador Python.
6. Digite o programa exemplo 1, como na figura a seguir.

7. Salve o programa como exemplo_1.py, selecionando o menu File -> Save as...
8. Para executá-lo, selecione o menu Run -> Start Debugging ou pressione a tecla F5.

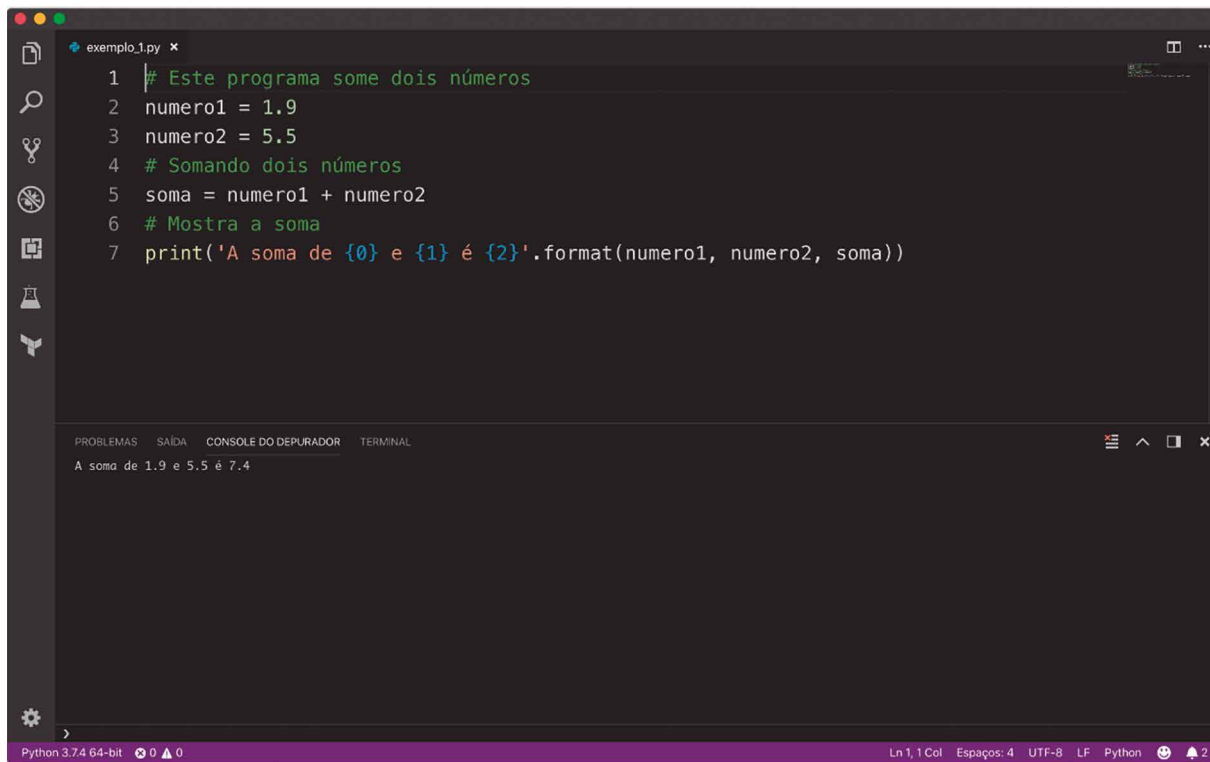


Figura 8 – Exemplo 1 no Visual Studio Code

Exemplo 2:

Guarda os números de entrada

```
numero1 = input('Digite o primeiro número: ')
```

```
numero2 = input('Digite o segundo número: ')
```

Somando dois números

```
soma = int(numero1) + int(numero2)
```

Mostra a soma

```
print('A soma de {0} e {1} é {2}'.format(numero1, numero2, soma))
```

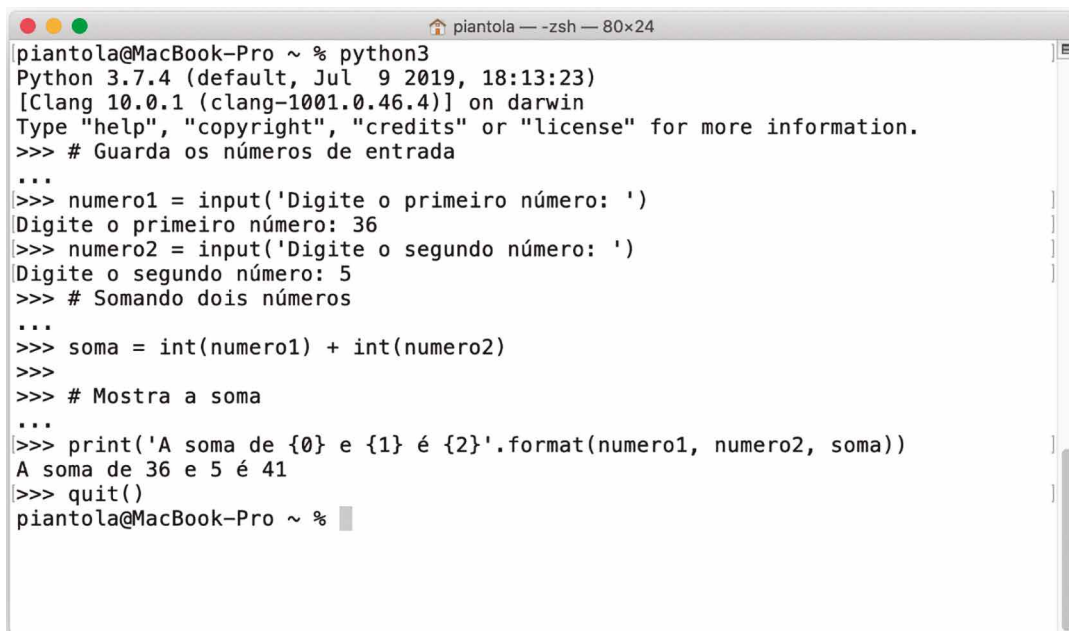
Executaremos o exemplo 2 no terminal do Windows ou MacOS:

No Windows:

1. Na barra de pesquisa no canto inferior esquerdo, ao lado do ícone símbolo do Windows, digite "cmd".
2. O prompt de comando se abrirá. Digite nele "python3" ou "python".
3. Digite o programa exemplo 2 linha a linha para a execução.

No MacOS:

1. Digite "Terminal" no campo de busca na direta superior e clique em Terminal.
2. O Terminal se abrirá (como na figura a seguir). Digite nele "python3".
3. Digite o programa exemplo 2 linha a linha para a execução, conforme a figura a seguir:



```
piantola@MacBook-Pro ~ % python3
Python 3.7.4 (default, Jul 9 2019, 18:13:23)
[Clang 10.0.1 (clang-1001.0.46.4)] on darwin
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
>>> quit()
piantola@MacBook-Pro ~ %
```

Figura 9 – Exemplo 2 executado no terminal Python linha a linha

Exemplo 3:

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)
```

Executaremos o exemplo 3 no Google Colab:

1. Abra um navegador web. Recomendamos o próprio navegador da Google, o Chrome.
2. Digite o endereço do Colab: <https://colab.research.google.com/>
3. Selecione "Novo notebook" no canto direito inferior da caixa de diálogo.
4. O aplicativo Colab mostrará uma tela parecida com a figura a seguir. Altere o nome do programa para exemplo_3.ipynb no canto esquerdo superior clicando nele.
5. Escreva o programa "exemplo 3", como na figura a seguir.
6. Execute o programa clicando na seta a esquerda do programa ou pressionando as teclas Ctrl + F9.

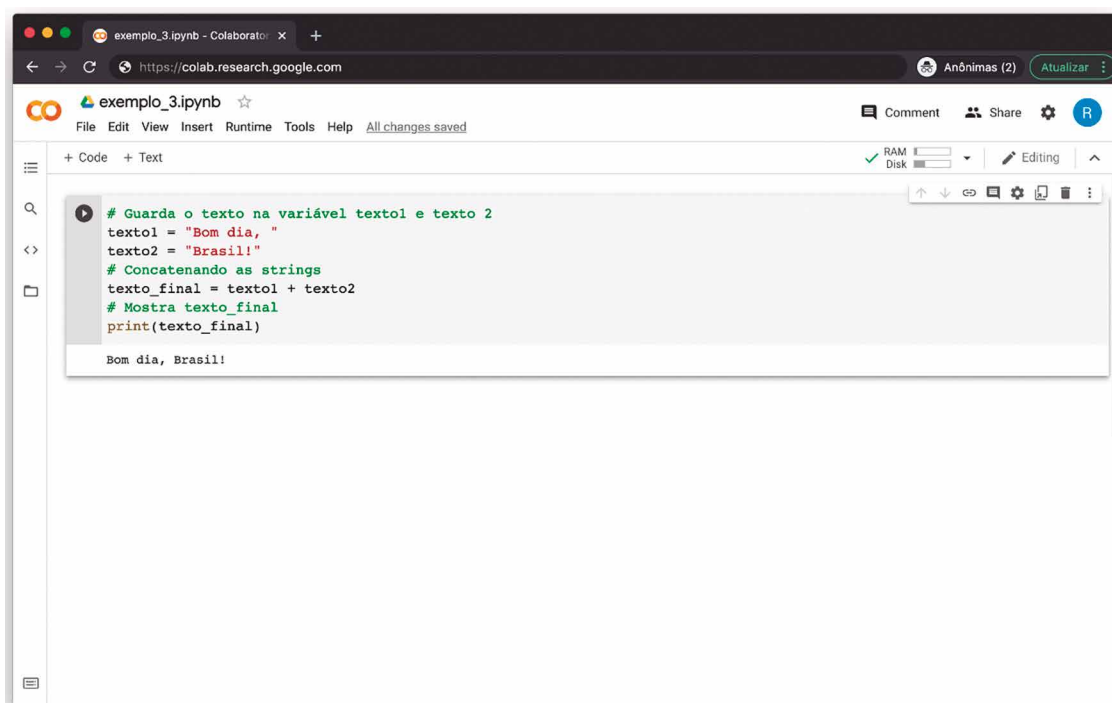


Figura 10 – Exemplo 3 on-line no Google Colab



Saiba mais

Os exemplos e exercícios de programação contidos neste livro-texto podem ser baixados a partir deste endereço:

GITHUB. *Introdução à programação estruturada*. [s.d.]. Disponível em: <https://cutt.ly/WmwhVvJ>. Acesso em: 25 jun. 2021.



Lembrete

Existem outras formas de executar um programa em Python; uma delas, já comentada neste tópico, é a criação do código em um Bloco de Notas. Depois, basta salvar e executá-lo no terminal: `python3 <nome_do_arquivo.py>`.

No próximo tópico, entraremos em detalhes sobre alguns elementos já visitados nos exemplos, como as **variáveis** `numero1`, `numero2` e `texto_final`. Abordaremos também o conceito de operadores, a **concatenação** e **soma** apresentadas nos exemplos e o conceito de **expressões**.

2 OPERADORES, EXPRESSÕES E VARIÁVEIS

Os conceitos de variáveis, operadores e expressões são essenciais para o aprendizado de qualquer linguagem de programação. Eles refletem a base para o armazenamento de dados e processamento da aplicação. O entendimento e uso correto desses recursos garantirão o bom desempenho e a corretude do programa desenvolvido. A corretude de um programa pode ser afirmada quando se diz que ele é correto com relação a determinada especificação.

2.1 Conceito de variável

Para a realização de tarefas em um computador, as aplicações (softwares desenvolvidos em uma linguagem de programação, em nosso caso, Python) necessitam, na maioria das vezes, manipular dados. Os dados ou informações a serem manipulados podem variar em tipos, como um símbolo caractere, um número real, um número inteiro, entre outros. Essas manipulações dependem também do tipo de operação realizada pela aplicação que está sendo construída. No caso de um cálculo, soma entre dois valores, podem ser armazenados valores do tipo real ou inteiro.

Essas manipulações de dados são possíveis pelo uso de variáveis, que nada mais são do que espaços reservados em memória para a alocação dos valores fornecidos pelo usuário ou por outras aplicações. Esse espaço em memória pode ser acessado por meio de seu endereço (posição da célula dentro da memória). Realizar o acesso via posição da célula de memória não é um processo tão trivial, o que dificultaria e muito o desenvolvimento de software.

Na criação de uma variável, é especificado um rótulo, nome dado pelo programador, que possibilita identificar a posição de memória local de armazenamento do conteúdo no computador. Esse nome fornecido pelo programador deve ser coerente com o conteúdo que será armazenado, por exemplo: `nota_p1`, `nota_p2`, `nome_usuario`, `media_aluno`.

Ao se declarar uma variável em uma aplicação, deve-se especificar o tipo de valor que esta poderá armazenar de acordo com a sua necessidade. Uma definição correta do tipo de variável é aquela que estabelece o espaço adequado para seu armazenamento, evitando desperdício e, no caso de manipulações futuras, possíveis transformações adicionais ao longo do programa.

O Python possui tipagem dinâmica. Assim, o próprio interpretador infere o tipo dos dados que uma variável recebe. Dessa forma, não existe a necessidade de especificar qual será o tipo.

Existem quatro tipos de **variáveis numéricas** em Python:

- Inteiro (int)
`n = 1`
- Ponto flutuante (float)
`n = 1.2`
- Complexo (complex)
`n = 4+3j`
- Booleano (bool)
`n = True`

É interessante notar que o inteiro em Python, na Teoria dos Conjuntos em matemática, é o conjunto dos números inteiros. Os números em ponto flutuante representam os números reais, assim como os complexos (complex) são os números complexos na matemática. Já os booleanos, representados por `True` (verdadeiro) e `False` (falso), são usados nas avaliações lógicas dos programas.

O "`n`" é o nome ou rótulo da variável, e sempre quando no programa aparecer esse rótulo, o conteúdo dela é que será trabalhado. O sinal de igual "`=`" é o comando de atribuição, ou seja, o que está do lado direito do "`=`" é atribuído como conteúdo do rótulo à esquerda do "`=`".

A definição do nome de uma variável pode ser realizada segundo as regras:

- Python possui um conjunto de palavras reservadas, e elas não poderão ser utilizadas na composição de nomes. As palavras reservadas são: '`False`', '`None`', '`True`', '`and`', '`as`', '`assert`', '`async`', '`await`', '`break`', '`class`', '`continue`', '`def`', '`del`', '`elif`', '`else`', '`except`', '`finally`', '`for`', '`from`', '`global`', '`if`', '`import`', '`in`', '`is`', '`lambda`', '`nonlocal`', '`not`', '`or`', '`pass`', '`raise`', '`return`', '`try`', '`while`', '`with`' e '`yield`'.

- Caracteres especiais, por exemplo, os caracteres latinos, como o cê-cedilha (ç), o til (~) etc., não são permitidos na composição de nomes de variáveis. O único caractere tido como especial e que pode ser utilizado é o underline (_).
- O primeiro caractere precisa ser uma letra ou o caractere _.
- A composição pode ser de A a Z, de a a z e de 0 a 9, seguindo as regras anteriores.

Como já visto, para atribuir um valor a uma variável basta escrever o nome dela seguido do símbolo de igualdade e por fim seu valor.

Outro tipo de variável bastante usada nas linguagens de programação é a do tipo strings ou texto. Strings são um tipo de variável especial que se constitui de uma sequência justaposta de caracteres quaisquer:

```
nome = "Claudio Nunes"
```

ou

```
nome = 'Claudio Nunes'
```

Podem-se utilizar aspas ou aspas simples para definir uma string. Um tópico posterior será dedicado somente à manipulação de textos (strings) por sua importância na programação.



Observação

O tipo booleano em Python, diferente de outras linguagens que utilizam 0 para falso e 1 para verdadeiro, pode ser considerado como tipo não numérico por utilizar os valores True e False.

2.2 Constantes e números

Uma constante é um tipo de "variável" cujo valor não pode ser modificado. Os números em si são constantes numéricas, não podendo ser atribuídos outros valores, por exemplo, ao número 2. Internamente, os números em Python são implementados como objetos.

Uma forma de criar constantes em Python é criar variáveis em um arquivo externo ao arquivo do programa (esse arquivo externo é chamado de módulo).

Por convenção, as constantes são escritas em letra maiúscula, podendo utilizar o caractere _.

Exemplo:

Criar arquivo constante.py com:

```
PI = 3.14
```

```
GRAVIDADE = 9.8
```

Criar o programa teste.py com:

```
import constante  
print(constante.PI)  
print(constante.GRAVIDADE)
```

Um bom motivo para se criar, constantes são valores utilizados no programa que sempre serão o mesmo, como no exemplo, em que os valores de pi e a gravidade na Terra nunca mudam.

2.3 Expressões

O tipo de dado booleano assume somente um valor entre dois valores constantes: True (verdadeiro) ou False (falso). As linguagens de programação usam expressões lógicas para avaliar desvios no fluxo de comandos, por exemplo, e a lógica booleana é utilizada para isso. Os desvios são importantes na programação estruturada e serão detalhados nas próximas unidades.

Os conectivos lógicos utilizados em Python são OR (ou), AND (e) e NOT (não):

Exemplo:

```
x = 3  
y = 2  
(x == y) or (y > x)
```

Essa expressão retornará falso.

```
(x > 2) and (y > 1)
```

Retorna verdadeiro.

```
not (x == x)
```

É falso.

Além dos conectivos lógicos das expressões, aparecem nos exemplos os operadores lógicos "==" (igualdade), "<" (menor) e ">" (maior). Os operadores serão descritos no próximo tópico.

2.4 Operadores

Os operadores podem ser classificados de várias formas. Uma delas diz respeito à quantidade de operandos necessários para a operação (operador unário, binário e ternário). Por exemplo, o operador de atribuição é binário, pois funciona com dois operandos: à esquerda, o nome da variável, e à direita, o valor a ser atribuído à memória. Outra forma de classificar os operadores é pela sua função: atribuição, matemáticos ou aritméticos, lógicos, relacionais e de strings.

A forma mais fácil de compreendermos as operações é pelo fato de toda operação receber operandos como entrada e prover um resultado dependente de sua função. Como exemplo, temos o operador aritmético +, que recebe dois operandos e como resultado devolve a soma entre eles. Nas próximas seções, entraremos em mais detalhes sobre os operadores e suas funções.

2.4.1 Operadores aritméticos

Existem alguns tipos básicos de operadores aritméticos que são utilizados na linguagem Python e que podem ser combinados de formas diferentes. A forma de associarmos esses operadores ao operador de atribuição pode gerar resultados diferentes.

Quadro 1

| Operação | Operador |
|---------------|----------|
| Adição | + |
| Subtração | - |
| Multiplicação | * |
| Divisão | / |
| Exponenciação | ** |
| Parte inteira | // |
| Módulo | % |

Os operadores matemáticos ou aritméticos mais usados são: adição, subtração, multiplicação, divisão e exponenciação. Além desses, temos o "//", que é a parte inteira da divisão, e o "%", que retorna o resto da divisão.

2.4.2 Operadores relacionais

Os operadores relacionais são bastante usados em expressões lógicas para a definição de laços e decisões (abordados nas próximas unidades).

Quadro 2

| Operação Relacional | Operador |
|---------------------|----------|
| Igualdade | == |
| Maior | > |
| Menor | < |
| Maior ou igual | >= |
| Menor ou igual | <= |
| Diferente | != |
| Não | ! |

2.4.3 Operadores lógicos

Os operadores lógicos utilizados em Python são OR (ou), AND (e) e NOT (não). Eles são utilizados para avaliar expressões lógicas, como discutido na seção 2.3.

Quadro 3

| Operação lógica | Função |
|-----------------|-------------------|
| and | Lógico E |
| or | Lógico OU |
| not | Lógico de negação |

Segue a tabelas-verdade e sua resposta usando o booleano True e False:

Quadro 4

| Expressão 1 | Expressão 2 | AND |
|-------------|-------------|-------|
| False | False | False |
| False | True | False |
| True | False | False |
| True | True | True |

No caso do E lógico (AND), o resultado só será verdadeiro (True) caso as duas expressões sejam verdadeiras.

Quadro 5

| Expressão 1 | Expressão 2 | OR |
|-------------|-------------|-------|
| False | False | False |
| False | True | True |
| True | False | True |
| True | True | True |

Já OU lógico (OR), o resultado só será falso (False) caso as duas expressões sejam falsas.

Quadro 6

| Expressão 1 | NOT |
|-------------|-------|
| False | True |
| True | False |

O NÃO lógico é um inversor. Caso a expressão seja verdadeira, ele retorna falso e vice-versa.

2.4.4 Operadores de atribuição

Quando criamos uma variável, atribuímos um valor para ela através do operador de atribuição '='. Nesse momento, o Python verifica qual é o tipo da variável e aloca um espaço na memória do computador para guardá-la. Existem outros operadores de atribuição que facilitam nosso trabalho de programação fazendo uma integração com os operadores aritméticos/matemáticos.

Exemplos:

```
x = 3
y = x
x = y = 10
x += 1
```

Quando usamos o operador +=, é o mesmo que usar a expressão $x = x + 1$, ou seja, somar 1 em x. Se o valor de x era 10, após a execução $x += 1$ o valor de x se torna 11. Assim, $x -= 2$ subtrai 2 da variável x.

Tabela 1

| Operador | Exemplo | Equivalente |
|----------|------------|--------------|
| = | $x = 1$ | $x = 1$ |
| += | $x += 1$ | $x = x + 1$ |
| -= | $x -= 1$ | $x = x - 1$ |
| /= | $x /= 1$ | $x = x / 1$ |
| //= | $x //= 1$ | $x = x // 1$ |
| *= | $x *= 1$ | $x = x * 1$ |
| %= | $x \% = 1$ | $x = x \% 1$ |
| **= | $x ** = 1$ | $x = x ** 1$ |

Algumas dessas operações são comuns na computação, por exemplo, um contador, que adiciona 01 ao total contido em uma variável. Para não repetir o nome da variável do lado direito da operação e deixar o código mais limpo, usamos o += nesse caso.



Observação

O operador de atribuição pode ser facilmente confundido com o operador relacional de igualdade. A atribuição é um símbolo de igual (=) e a igualdade são dois símbolos (==).

2.4.5 Precedência dos operadores

Quando se cria uma expressão em Python, existe uma ordem em que as subexpressões são avaliadas. Essa ordem é chamada de precedência de operadores. Segue a ordem dos principais operadores em Python:

Quadro 7 – Ordem de precedência do operador

| |
|---------------------------------|
| ** |
| *, /, %, // |
| <=, <, >, >= |
| ==, != |
| =, %=, /=, //=, -=, +=, *=, **= |
| not, or, and |

Exemplo:

Assumindo $x = 3$ e $y = 4$

$x ** 1 + y * 2 \% 1$

x e y são substituídos por seus valores: $3 ** 1 + 4 * 2 \% 1$

$**$ tem precedência, $3 ** 1$ é avaliado primeiro: $3 + 4 * 2 \% 1$

$4 * 2$ é avaliado, pois apareceu primeiro que $\%$ e tem a mesma importância de ordem: $3 + 8 \% 1$

É avaliada a expressão $8 \% 1$; assim: $3 + 0$

$3 + 0 = 3$



Lembrete

A mesma regra da matemática vale para os parênteses, que ditam qual é a expressão a ser avaliada primeiro. Em $(9 + 4) * 8$, a expressão a ser avaliada primeiro é $9 + 4$.

2.5 Estrutura sequencial

Com relação ao modelo de programação estruturada, estudamos nesta unidade a estrutura sequencial. Ela é a estrutura de controle mais básica, em que os comandos em uma aplicação são executados na ordem em que são especificados, um após o outro.

A estrutura sequencial consiste em:

- entrada de dados;
- processamento;
- saída.

A entrada de dados pode ser feita pelo usuário com o comando `input()` ou através da atribuição de valor a uma variável, por exemplo. O processamento da entrada acontece utilizando operadores matemáticos, funções de manipulação de textos e outros comandos que veremos no decorrer dos tópicos. A saída pode ser na forma do comando `print()`, na tela, e também em arquivo, impressora, rede etc.

Exemplo:

```
# Guarda os números de entrada (Entrada de dados)
numero1 = input('Digite o primeiro número: ')
numero2 = input('Digite o segundo número: ')
# Somando dois números (Processamento)
soma = int(numero1) + int(numero2)
# Mostra a soma (Saída)
print('A soma de {0} e {1} é {2}'.format(numero1, numero2, soma))
```

2.6 Exercícios sugeridos

1. Construa um programa que mostre o texto "Ola, mundo!" na tela. Dica: utilize uma função `print()` em um arquivo com nome `ola.py` e execute-o na linha de comando `python ola.py`
2. Crie um programa que imprima na tela a palavra Python, centralizada na tela. Dica: o terminal geralmente tem 80 caracteres por linha.
3. Construa um programa que solicite três números inteiros e imprima a soma desses três números.
4. Construa um programa que solicite ao usuário um nome de fruta e então mostre a seguinte mensagem na tela: "A fruta digitada foi: <nome da fruta digitada>"
5. Crie um programa que receba a distância de uma cidade em quilômetros e a transforme em metros, mostrando o resultado na tela.

6. Crie um programa que solicite ao usuário duas notas semestrais e retorne na tela a média final da disciplina.

7. Faça um programa que receba o tamanho de um lado de um quadrado e calcule sua área, mostrando na tela a área desse quadrado.

8. Crie um programa que solicite ao usuário o tamanho do raio de um círculo, calcule e imprima na tela sua área. Dica: $\pi = 3.141592653589793$

9. Faça um programa que pergunte ao usuário o valor que ele tem investido e a taxa percentual que ele ganha ao mês (juros simples). Calcule e imprima o total ganho com o investimento em 3 meses.

10. Construa um programa que solicite a temperatura em graus Fahrenheit ao usuário e transforme a temperatura em graus Celsius. Imprima na tela. Dica:

$$\text{Celsius} = 5 * ((\text{Fahrenheit} - 32) / 9)$$

11. Agora, construa um programa que solicite ao usuário a temperatura em graus Celsius e a transforme em Fahrenheit. Mostre o resultado na tela.

12. Construa um programa que receba a altura de três pessoas e calcule a média das três alturas. Mostre o resultado na tela.

13. Crie um programa que solicite ao usuário três números inteiros. Compute e exiba na tela:

a) o resto da divisão do primeiro com o segundo;

b) o segundo exponencial com o terceiro;

c) o primeiro dividido pelo terceiro, somado com o segundo.

14. Crie um programa que receba como a entrada a altura da pessoa e calcule o peso ideal. Utilize as fórmulas: para os homens, $(72.7 * \text{altura}) - 58$; para as mulheres, $(62.1 * \text{altura}) - 44.7$

15. Construa um programa que receba o tamanho dos catetos de um triângulo retângulo e compute sua hipotenusa. Imprima o valor da hipotenusa na tela.

16. Crie um programa que receba como entrada dois números inteiros, divida o primeiro número pelo segundo e imprima na tela, separadamente, a parte inteira da divisão e o resto.

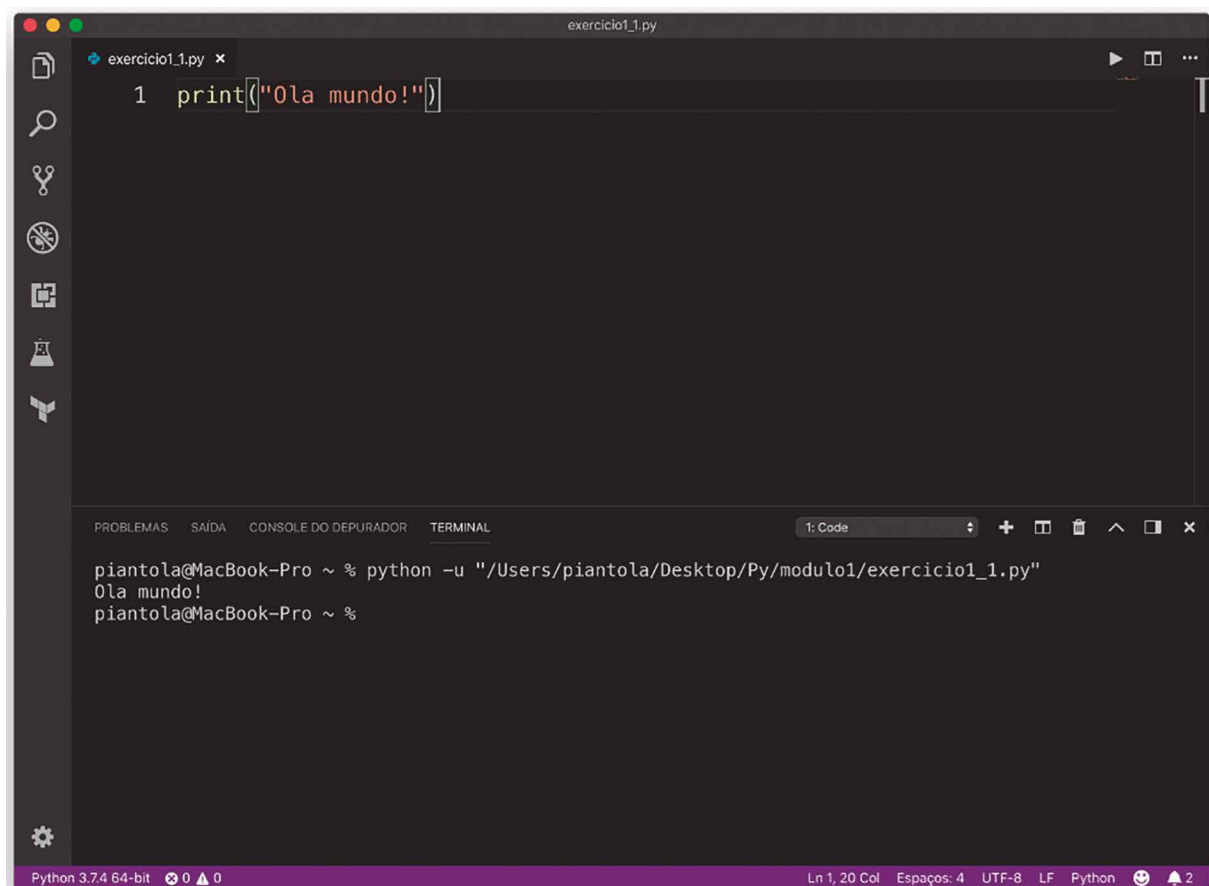
17. Construa um aplicativo para uma loja de pisos. O aplicativo solicita o tamanho em metros quadrados da área a ser colocado o piso. Cada piso tem 60 × 60 centímetros; eles são vendidos em caixas com 10 pisos, que custam R\$ 70,00. Mostre na tela para o cliente a quantidade de caixas que ele deve comprar e o preço total do orçamento.

18. Faça um programa que receba a altura e a base de um retângulo e calcule sua área. Mostre na tela o resultado.

19. Construa um programa que solicite o tamanho em megabytes de um arquivo para transferência, e a velocidade do link de internet em megabits por segundo. Compute o tempo de transferência do arquivo, passando por esse link em minutos. Imprima o resultado na tela. Dica: 1 MB são 1000000 de bytes e 1 byte são 8 bits.

20. Crie um programa que receba três palavra e as imprima na tela na ordem inversa que as recebeu.

2.6.1 Respostas dos exercícios

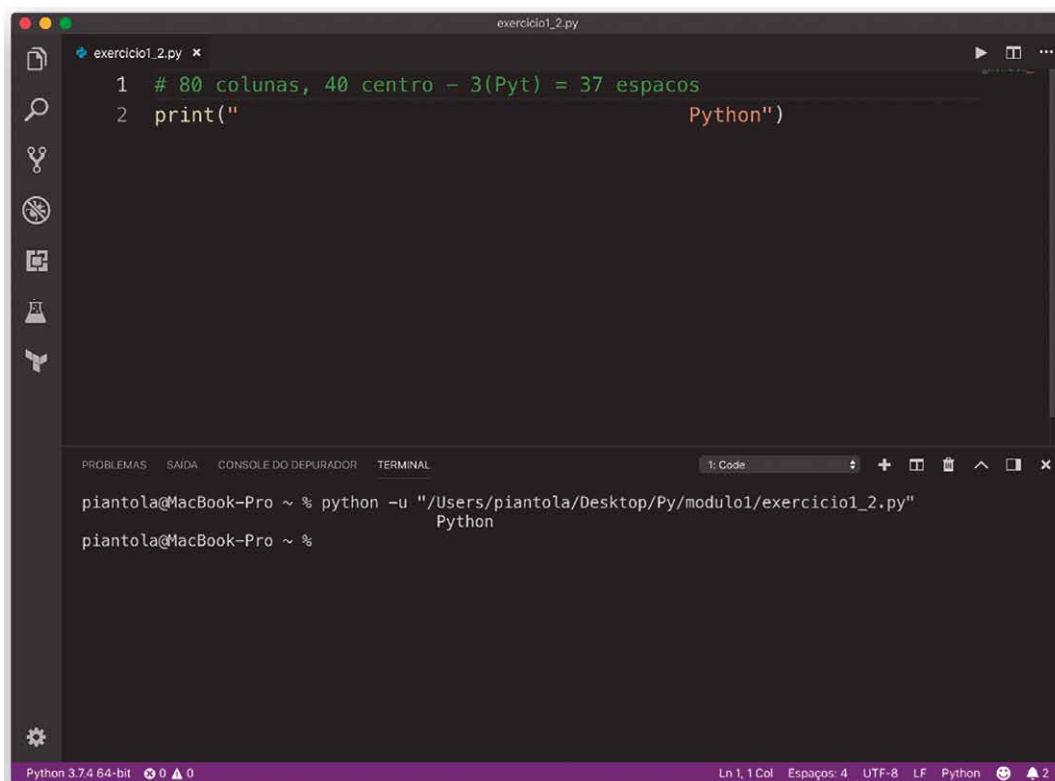


The screenshot shows a code editor window titled "exercicio1_1.py". The code contains a single line: `1 print("Ola mundo!")`. Below the code editor, there is a terminal window with the following output:

```
piantola@MacBook-Pro ~ % python -u "/Users/piantola/Desktop/Py/modulo1/exercicio1_1.py"
Ola mundo!
piantola@MacBook-Pro ~ %
```

The status bar at the bottom indicates "Python 3.7.4 64-bit", "Ln 1, 20 Col", "Espaços: 4", "UTF-8", "LF", "Python", and "2" notifications.

Figura 11 – Resolução do exercício 1

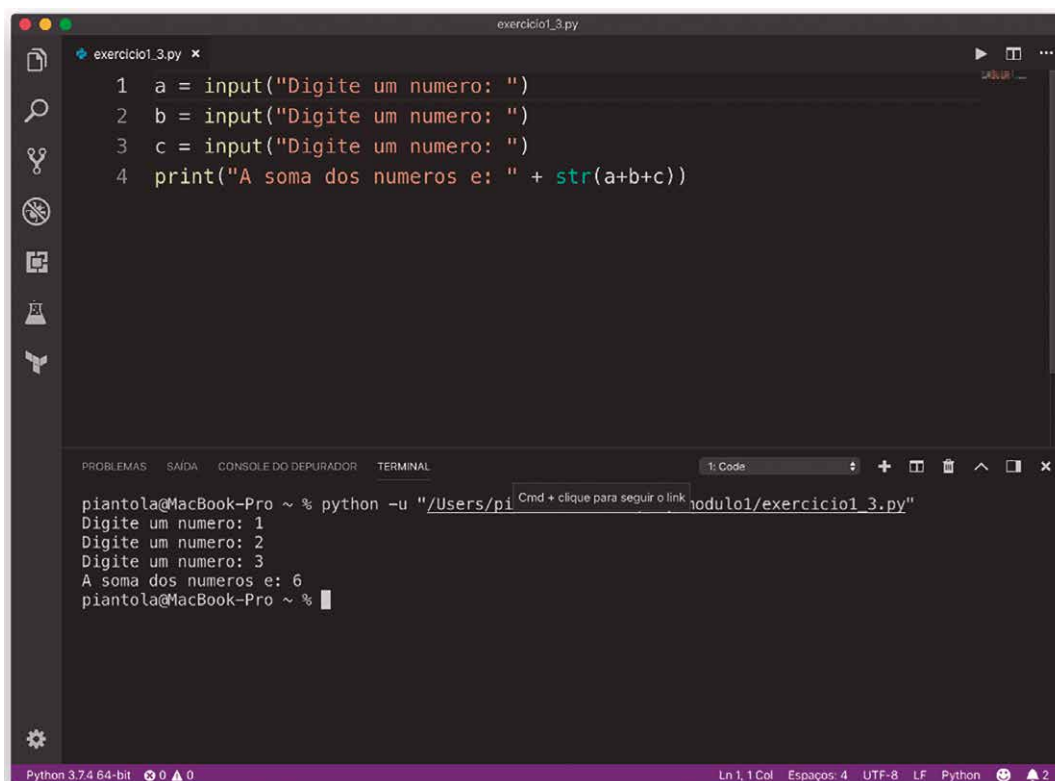


```
exercicio1_2.py
1 # 80 colunas, 40 centro - 3(Pyt) = 37 espaços
2 print("                                Python")

PROBLEMAS SAÍDA CONSOLE DO DEPURADOR TERMINAL 1: Code
piantola@MacBook-Pro ~ % python -u "/Users/piantola/Desktop/Py/modulo1/exercicio1_2.py"
Python
piantola@MacBook-Pro ~ %

Python 3.7.4 64-bit 0 0 Ln 1, 1 Col Espaços: 4 UTF-8 LF Python 2
```

Figura 12 – Resolução do exercício 2

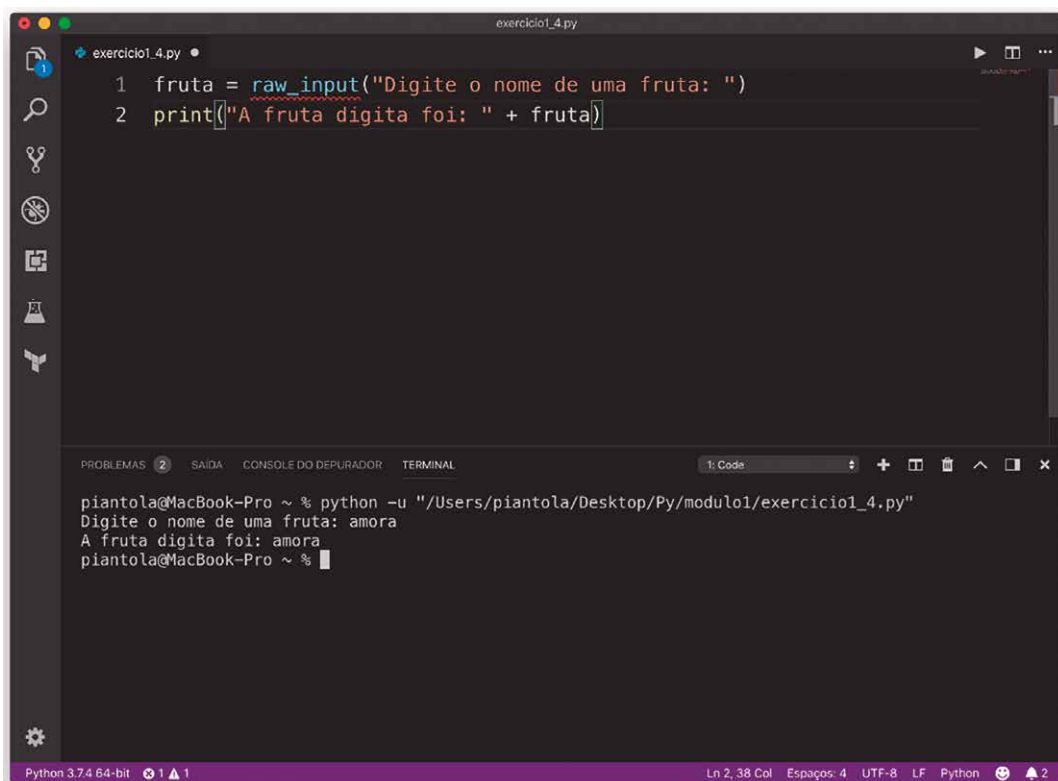


```
exercicio1_3.py
1 a = input("Digite um numero: ")
2 b = input("Digite um numero: ")
3 c = input("Digite um numero: ")
4 print("A soma dos numeros e: " + str(a+b+c))

PROBLEMAS SAÍDA CONSOLE DO DEPURADOR TERMINAL 1: Code
piantola@MacBook-Pro ~ % python -u "/Users/piantola/Desktop/Py/modulo1/exercicio1_3.py"
Digite um numero: 1
Digite um numero: 2
Digite um numero: 3
A soma dos numeros e: 6
piantola@MacBook-Pro ~ %

Python 3.7.4 64-bit 0 0 Ln 1, 1 Col Espaços: 4 UTF-8 LF Python 2
```

Figura 13 – Resolução do exercício 3



```
exercico1_4.py
1 fruta = raw_input("Digite o nome de uma fruta: ")
2 print("A fruta digita foi: " + fruta)
```

piantola@MacBook-Pro ~ % python -u "/Users/piantola/Desktop/Py/modulo1/exercico1_4.py"

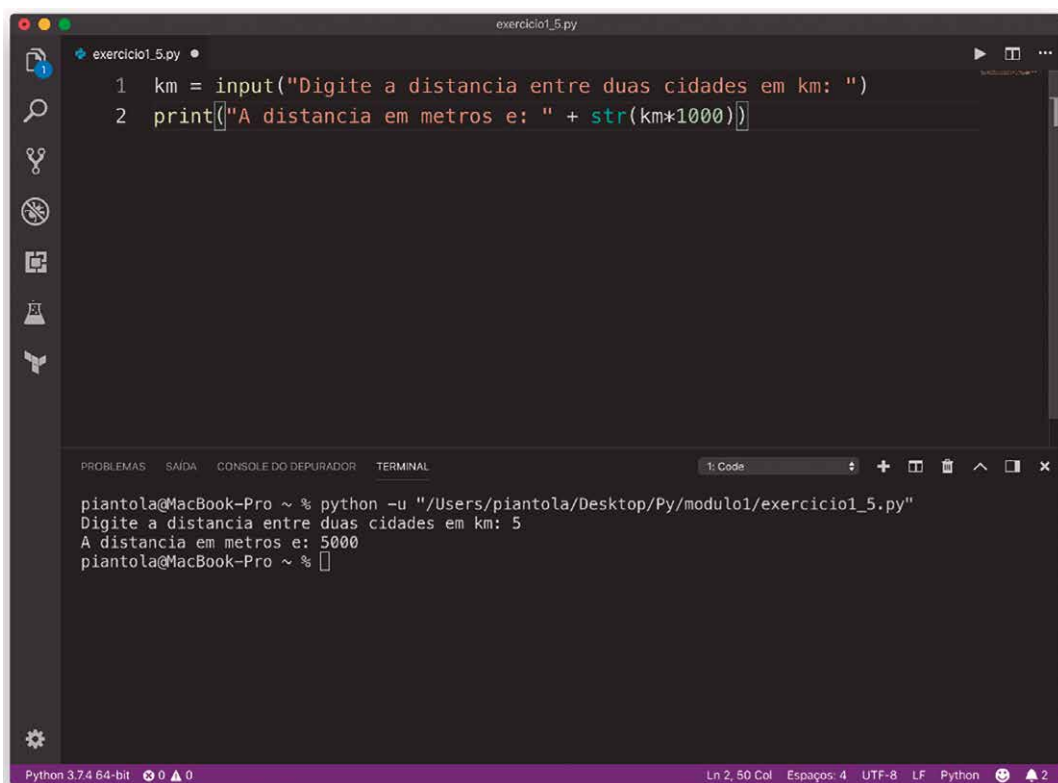
Digite o nome de uma fruta: amora

A fruta digita foi: amora

piantola@MacBook-Pro ~ %

Python 3.7.4 64-bit | 1 | Ln 2, 38 Col | Espaços: 4 | UTF-8 | LF | Python

Figura 14 – Resolução do exercício 4



```
exercico1_5.py
1 km = input("Digite a distancia entre duas cidades em km: ")
2 print("A distancia em metros e: " + str(km*1000))
```

piantola@MacBook-Pro ~ % python -u "/Users/piantola/Desktop/Py/modulo1/exercico1_5.py"

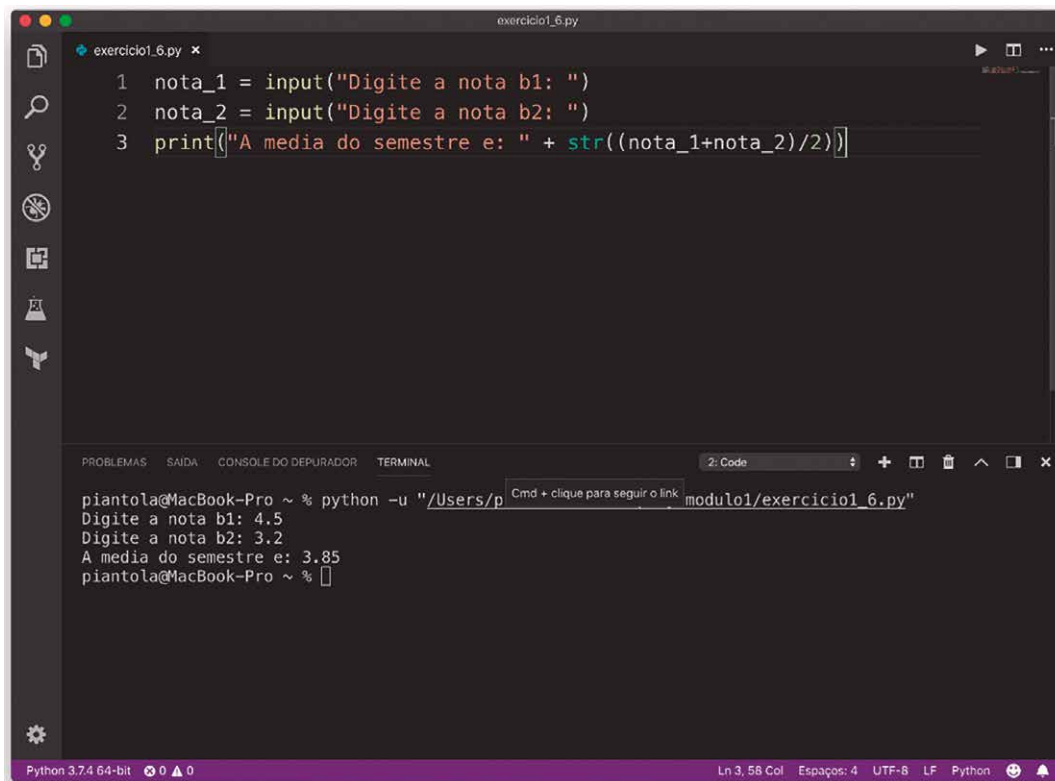
Digite a distancia entre duas cidades em km: 5

A distancia em metros e: 5000

piantola@MacBook-Pro ~ %

Python 3.7.4 64-bit | 0 | Ln 2, 50 Col | Espaços: 4 | UTF-8 | LF | Python

Figura 15 – Resolução do exercício 5



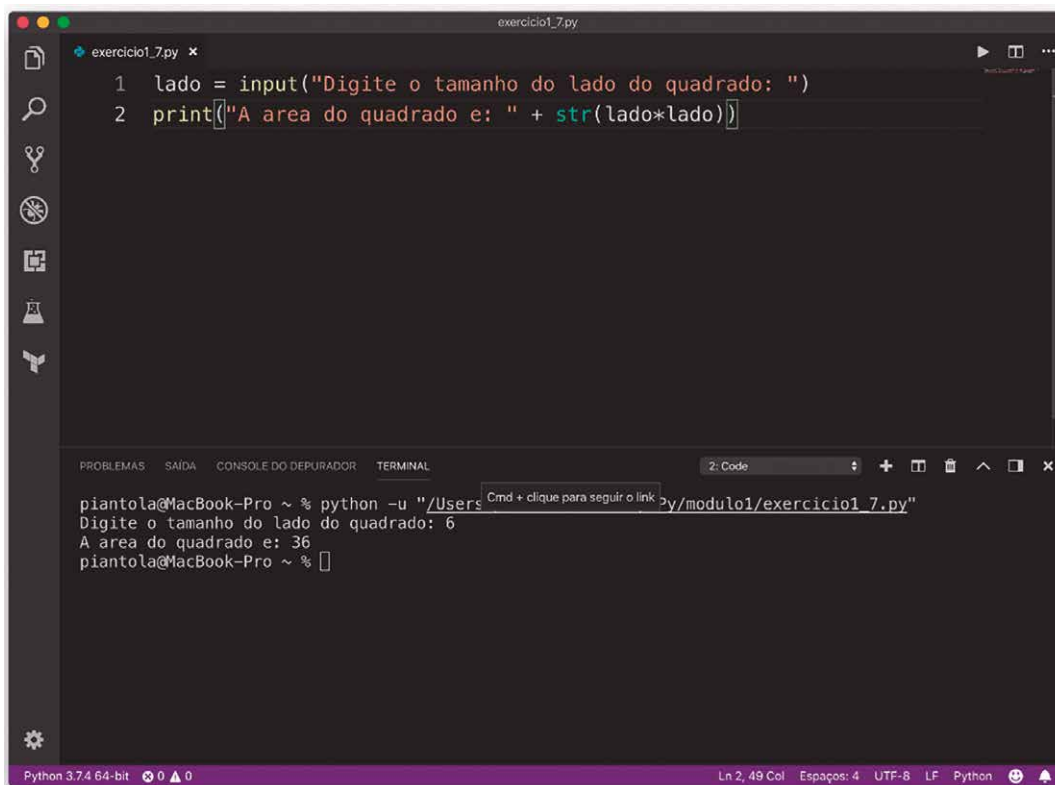
```
exercicio1_6.py
1 nota_1 = input("Digite a nota b1: ")
2 nota_2 = input("Digite a nota b2: ")
3 print("A media do semestre e: " + str((nota_1+nota_2)/2))
```

PROBLEMAS SAÍDA CONSOLE DO DEPURADOR TERMINAL 2: Code

```
piantola@MacBook-Pro ~ % python -u "/Users/piantola/PycharmProjects/modulo1/exercicio1_6.py"
Digite a nota b1: 4.5
Digite a nota b2: 3.2
A media do semestre e: 3.85
piantola@MacBook-Pro ~ %
```

Python 3.7.4 64-bit 0 0

Figura 16 – Resolução do exercício 6



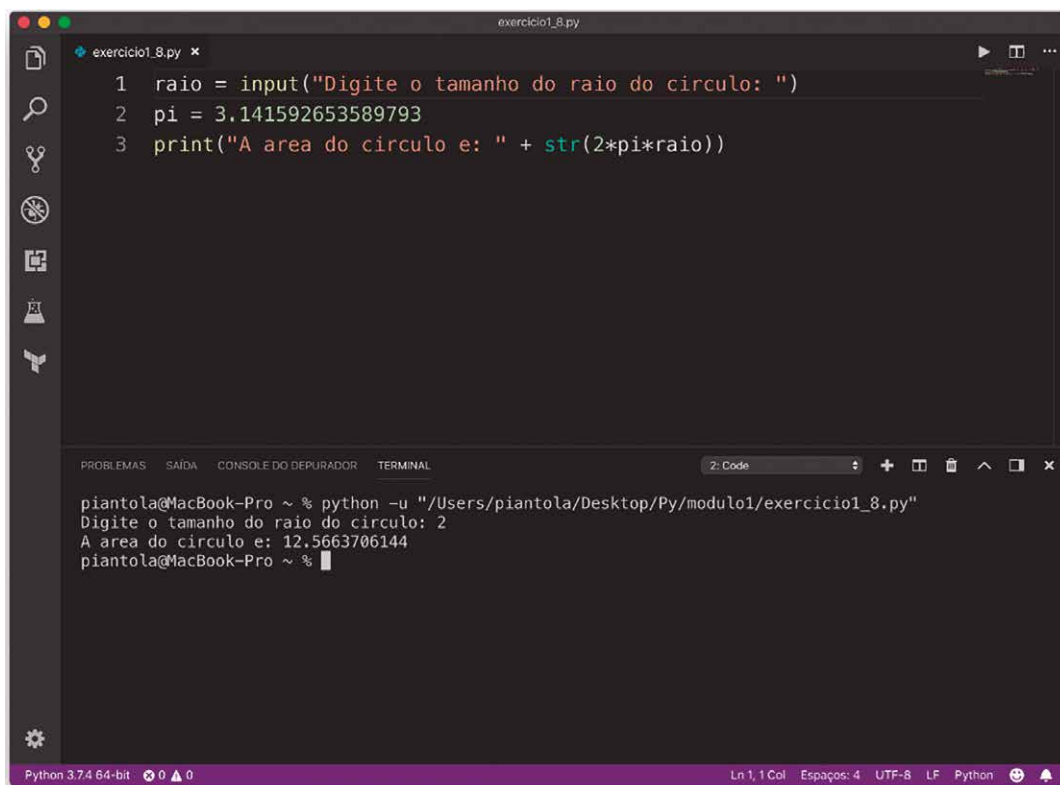
```
exercicio1_7.py
1 lado = input("Digite o tamanho do lado do quadrado: ")
2 print("A area do quadrado e: " + str(lado*lado))
```

PROBLEMAS SAÍDA CONSOLE DO DEPURADOR TERMINAL 2: Code

```
piantola@MacBook-Pro ~ % python -u "/Users/piantola/PycharmProjects/modulo1/exercicio1_7.py"
Digite o tamanho do lado do quadrado: 6
A area do quadrado e: 36
piantola@MacBook-Pro ~ %
```

Python 3.7.4 64-bit 0 0

Figura 17 – Resolução do exercício 7

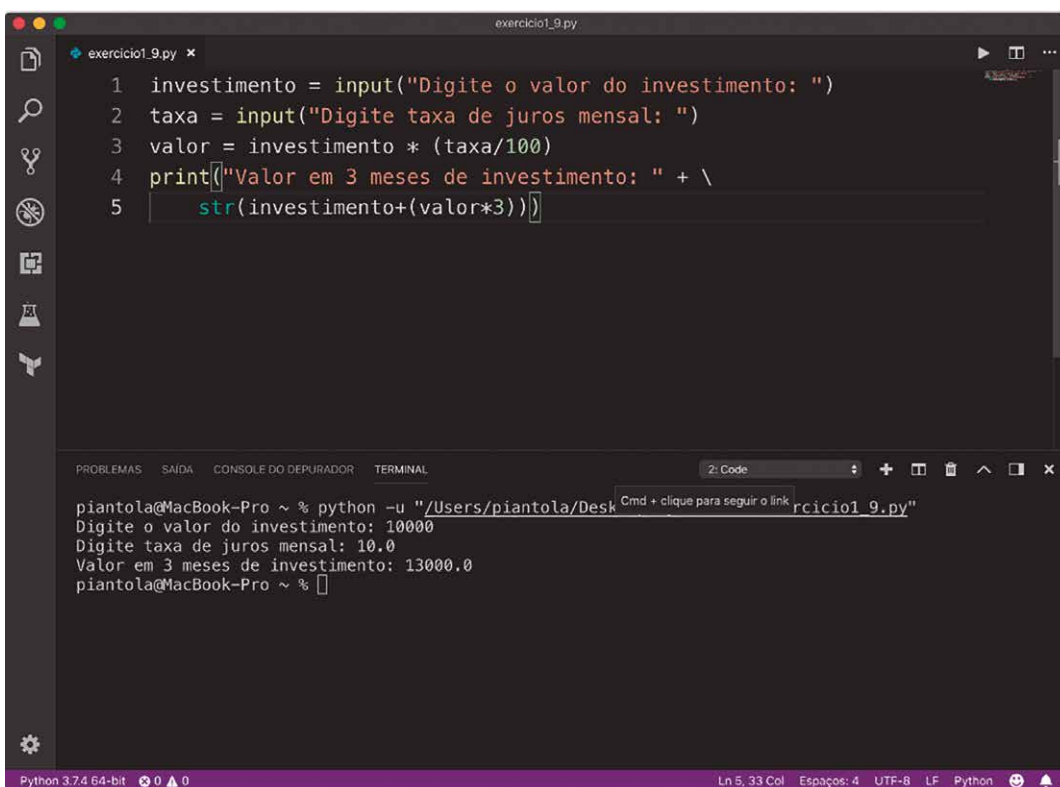


```
exercício1_8.py
1 raio = input("Digite o tamanho do raio do círculo: ")
2 pi = 3.141592653589793
3 print("A área do círculo é: " + str(2*pi*raio))

PROBLEMAS SAÍDA CONSOLE DO DEPURADOR TERMINAL
2: Code
piantola@MacBook-Pro ~ % python -u "/Users/piantola/Desktop/Py/modulo1/exercicio1_8.py"
Digite o tamanho do raio do círculo: 2
A área do círculo é: 12.5663706144
piantola@MacBook-Pro ~ %
```

Python 3.7.4 64-bit 0 0 0 Ln 1, 1 Col Espaços: 4 UTF-8 LF Python

Figura 18 – Resolução do exercício 8

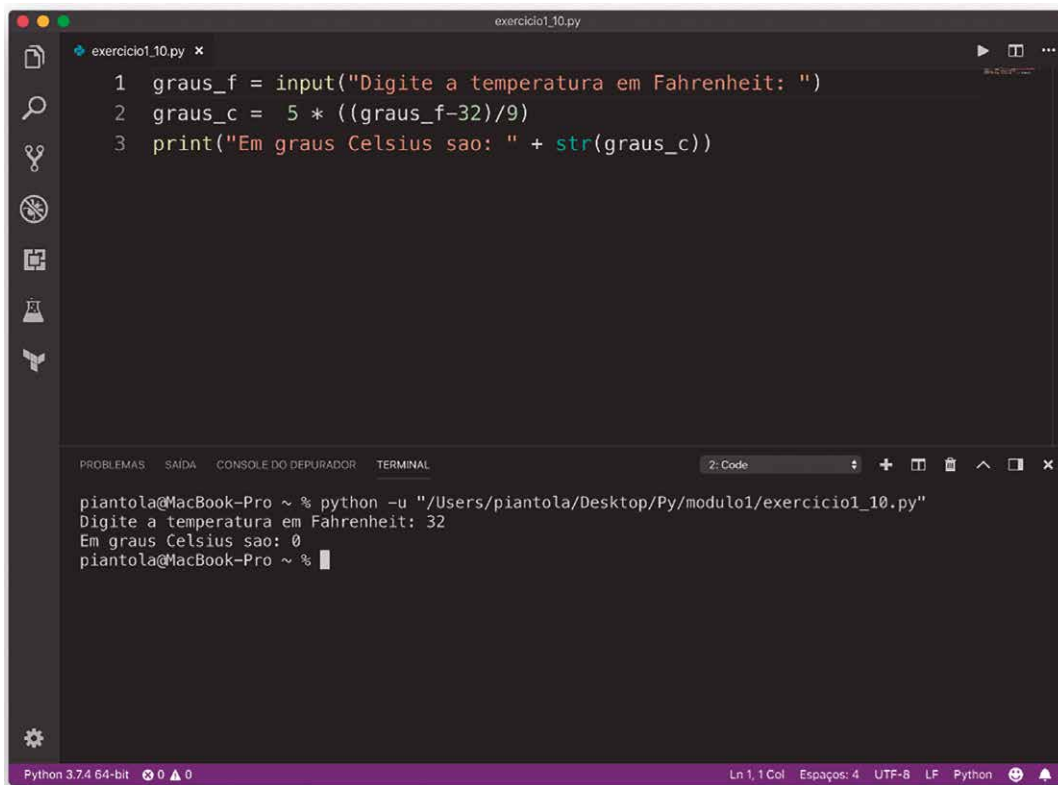


```
exercício1_9.py
1 investimento = input("Digite o valor do investimento: ")
2 taxa = input("Digite taxa de juros mensal: ")
3 valor = investimento * (taxa/100)
4 print("Valor em 3 meses de investimento: " + \
5       str(investimento+(valor*3)))

PROBLEMAS SAÍDA CONSOLE DO DEPURADOR TERMINAL
2: Code
piantola@MacBook-Pro ~ % python -u "/Users/piantola/Desktop/Py/modulo1/exercicio1_9.py"
Digite o valor do investimento: 10000
Digite taxa de juros mensal: 10.0
Valor em 3 meses de investimento: 13000.0
piantola@MacBook-Pro ~ %
```

Python 3.7.4 64-bit 0 0 0 Ln 5, 33 Col Espaços: 4 UTF-8 LF Python

Figura 19 – Resolução do exercício 9

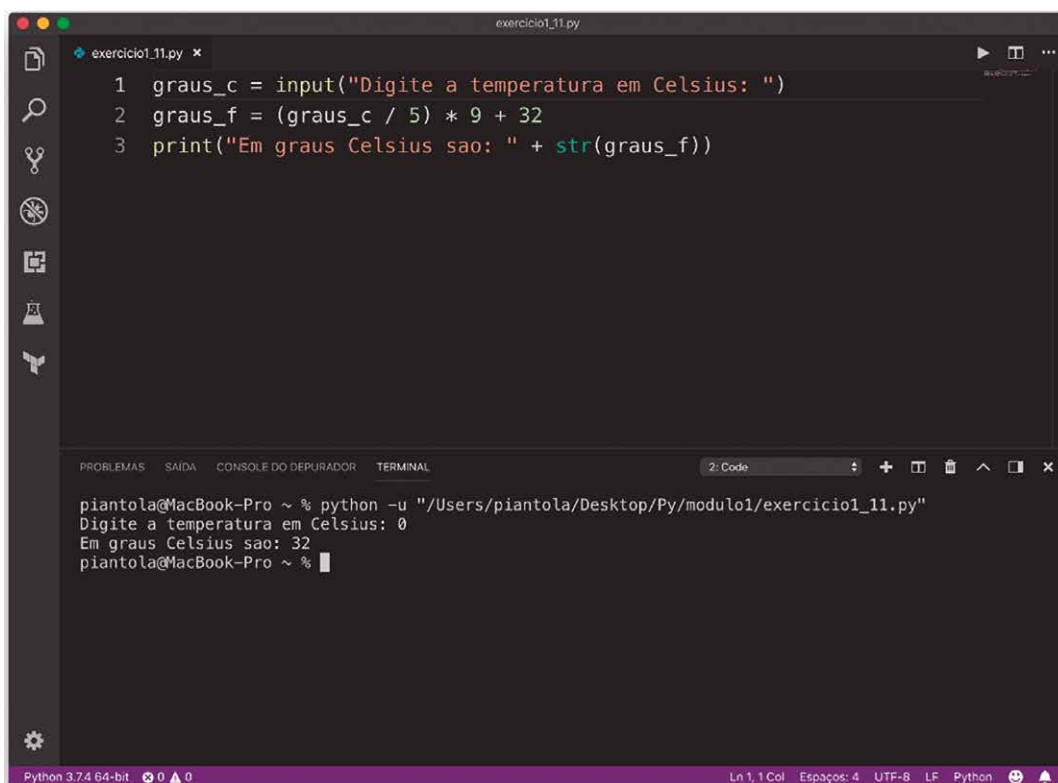


```
exercicio1_10.py
1 graus_f = input("Digite a temperatura em Fahrenheit: ")
2 graus_c = 5 * ((graus_f-32)/9)
3 print("Em graus Celsius sao: " + str(graus_c))

PROBLEMAS SAIDA CONSOLE DO DEPURADOR TERMINAL
2: Code
piantola@MacBook-Pro ~ % python -u "/Users/piantola/Desktop/Py/modulo1/exercicio1_10.py"
Digite a temperatura em Fahrenheit: 32
Em graus Celsius sao: 0
piantola@MacBook-Pro ~ %
```

Python 3.7.4 64-bit 0 0 Ln 1, 1 Col Espaços: 4 UTF-8 LF Python

Figura 20 – Resolução do exercício 10

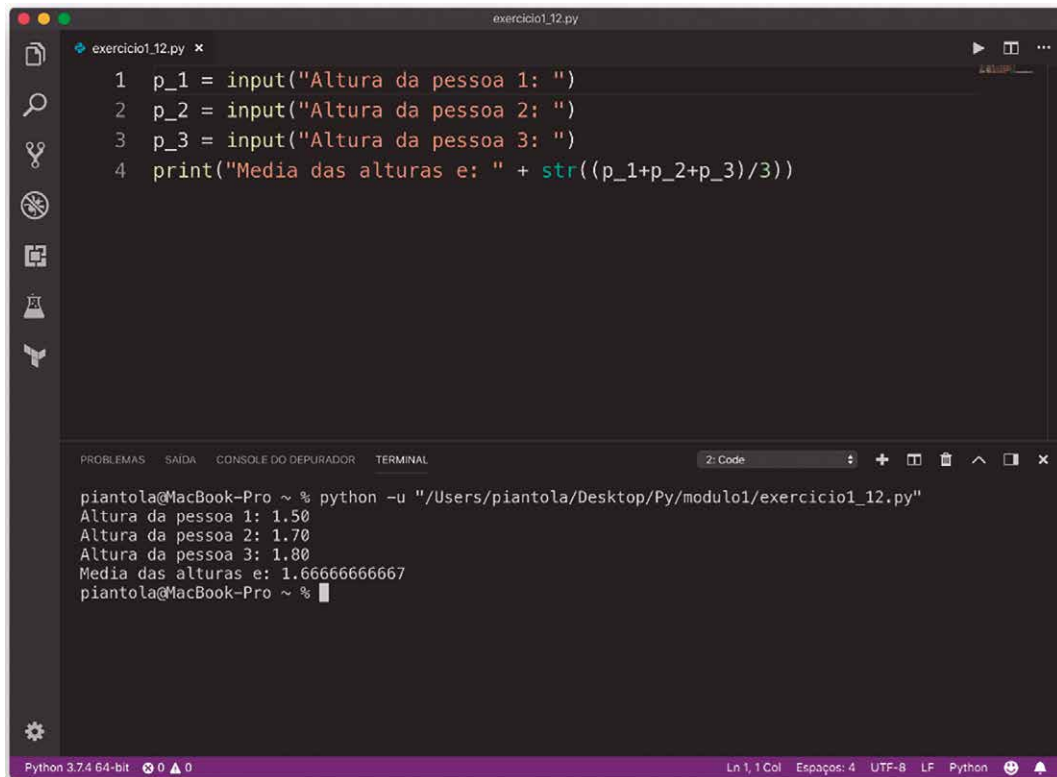


```
exercicio1_11.py
1 graus_c = input("Digite a temperatura em Celsius: ")
2 graus_f = (graus_c / 5) * 9 + 32
3 print("Em graus Celsius sao: " + str(graus_f))

PROBLEMAS SAIDA CONSOLE DO DEPURADOR TERMINAL
2: Code
piantola@MacBook-Pro ~ % python -u "/Users/piantola/Desktop/Py/modulo1/exercicio1_11.py"
Digite a temperatura em Celsius: 0
Em graus Celsius sao: 32
piantola@MacBook-Pro ~ %
```

Python 3.7.4 64-bit 0 0 Ln 1, 1 Col Espaços: 4 UTF-8 LF Python

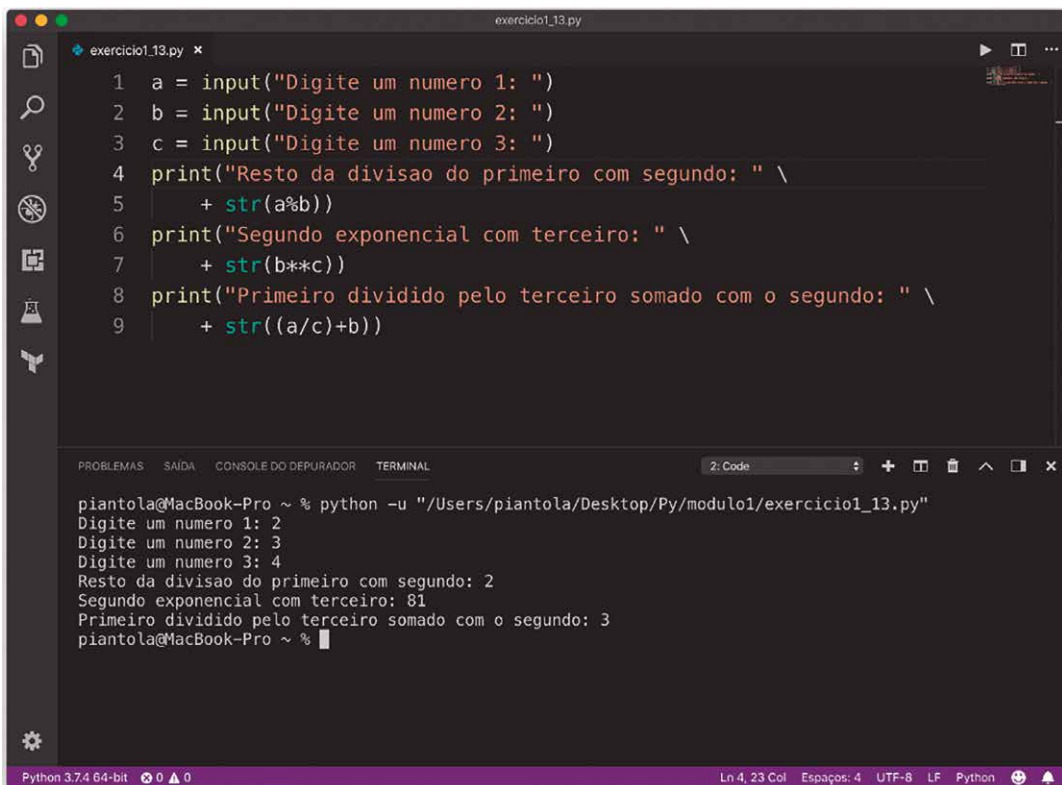
Figura 21 – Resolução do exercício 11



```
exercício1_12.py
1 p_1 = input("Altura da pessoa 1: ")
2 p_2 = input("Altura da pessoa 2: ")
3 p_3 = input("Altura da pessoa 3: ")
4 print("Media das alturas e: " + str((p_1+p_2+p_3)/3))

PROBLEMAS SAÍDA CONSOLA DO DEPURADOR TERMINAL
2: Code
piantola@MacBook-Pro ~ % python -u "/Users/piantola/Desktop/Py/modulo1/exercicio1_12.py"
Altura da pessoa 1: 1.50
Altura da pessoa 2: 1.70
Altura da pessoa 3: 1.80
Media das alturas e: 1.6666666666666667
piantola@MacBook-Pro ~ %
```

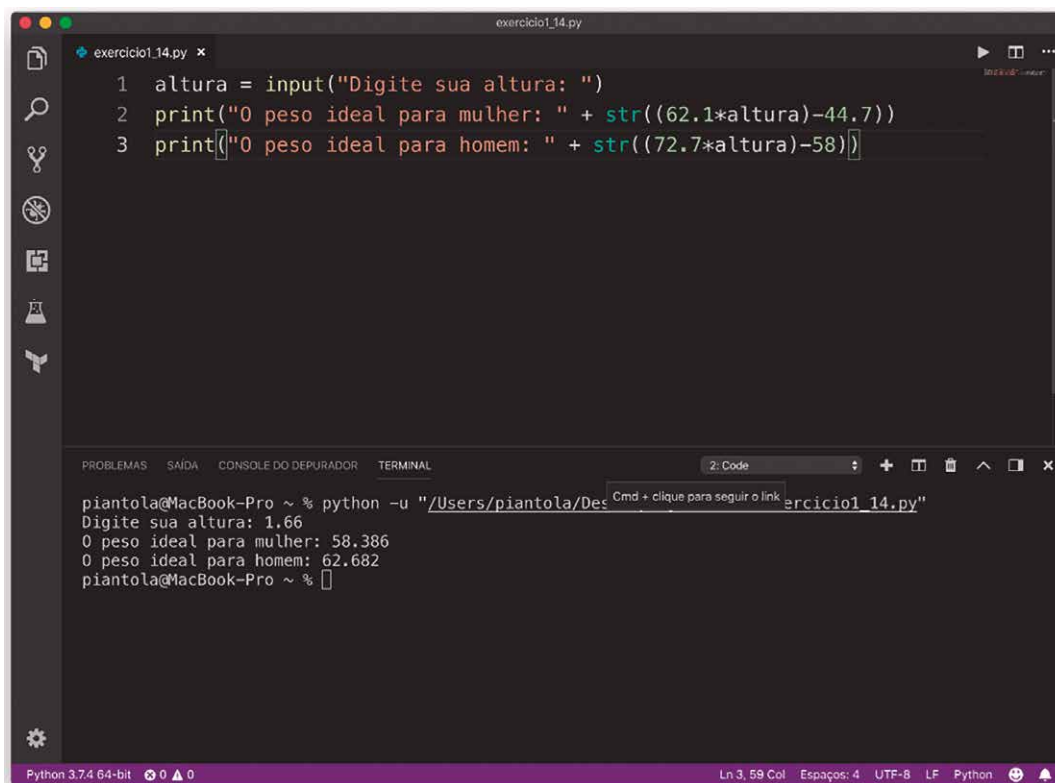
Figura 22 – Resolução do exercício 12



```
exercício1_13.py
1 a = input("Digite um numero 1: ")
2 b = input("Digite um numero 2: ")
3 c = input("Digite um numero 3: ")
4 print("Resto da divisao do primeiro com segundo: " \
5       + str(a%b))
6 print("Segundo exponencial com terceiro: " \
7       + str(b**c))
8 print("Primeiro dividido pelo terceiro somado com o segundo: " \
9       + str((a/c)+b))

PROBLEMAS SAÍDA CONSOLA DO DEPURADOR TERMINAL
2: Code
piantola@MacBook-Pro ~ % python -u "/Users/piantola/Desktop/Py/modulo1/exercicio1_13.py"
Digite um numero 1: 2
Digite um numero 2: 3
Digite um numero 3: 4
Resto da divisao do primeiro com segundo: 2
Segundo exponencial com terceiro: 81
Primeiro dividido pelo terceiro somado com o segundo: 3
piantola@MacBook-Pro ~ %
```

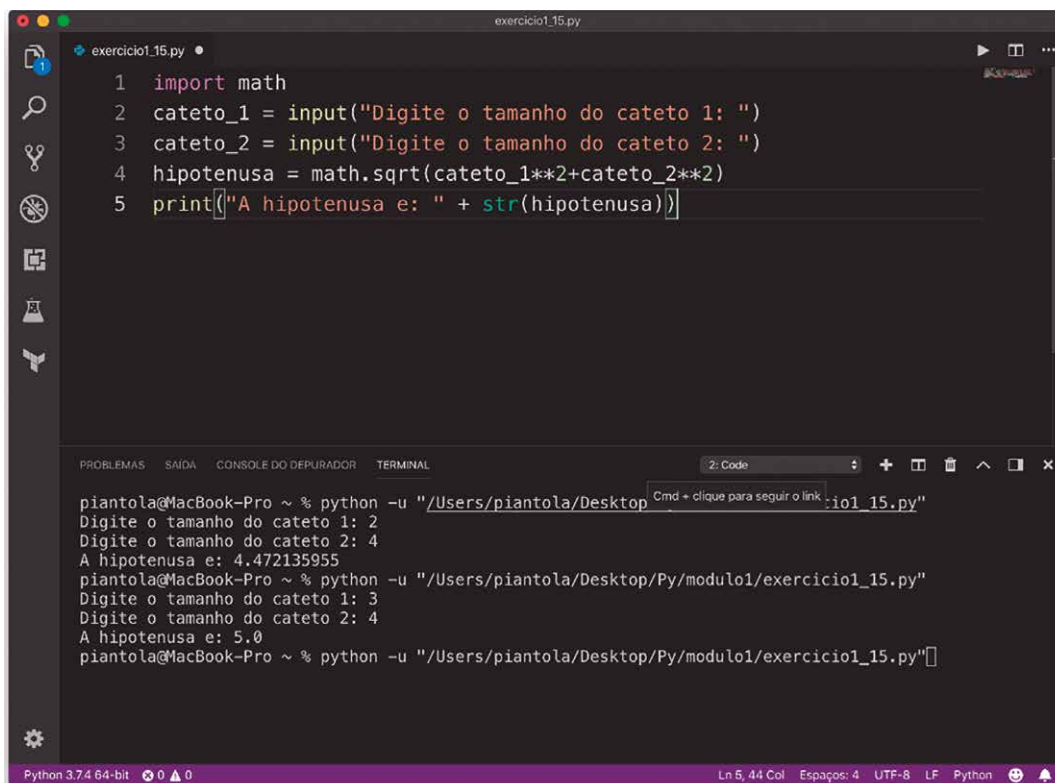
Figura 23 – Resolução do exercício 13



```
exercicio1_14.py
1 altura = input("Digite sua altura: ")
2 print("0 peso ideal para mulher: " + str((62.1*altura)-44.7))
3 print("0 peso ideal para homem: " + str((72.7*altura)-58))
```

```
piantola@MacBook-Pro ~ % python -u "/Users/piantola/Desktop/exercicio1_14.py"
Digite sua altura: 1.66
0 peso ideal para mulher: 58.386
0 peso ideal para homem: 62.682
piantola@MacBook-Pro ~ %
```

Figura 24 – Resolução do exercício 14

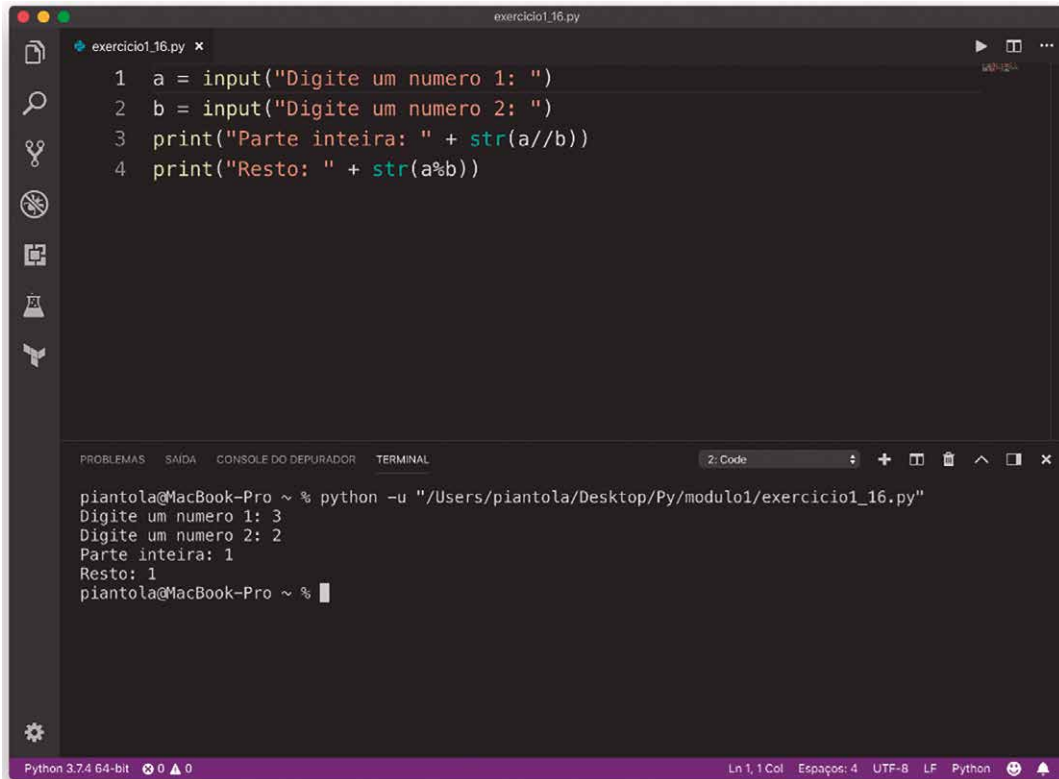


```
exercicio1_15.py
1 import math
2 cateto_1 = input("Digite o tamanho do cateto 1: ")
3 cateto_2 = input("Digite o tamanho do cateto 2: ")
4 hipotenusa = math.sqrt(cateto_1**2+cateto_2**2)
5 print("A hipotenusa e: " + str(hipotenusa))
```

```
piantola@MacBook-Pro ~ % python -u "/Users/piantola/Desktop/exercicio1_15.py"
Digite o tamanho do cateto 1: 2
Digite o tamanho do cateto 2: 4
A hipotenusa e: 4.472135955
piantola@MacBook-Pro ~ % python -u "/Users/piantola/Desktop/Py/modulo1/exercicio1_15.py"
Digite o tamanho do cateto 1: 3
Digite o tamanho do cateto 2: 4
A hipotenusa e: 5.0
piantola@MacBook-Pro ~ % python -u "/Users/piantola/Desktop/Py/modulo1/exercicio1_15.py"
```

Figura 25 – Resolução do exercício 15

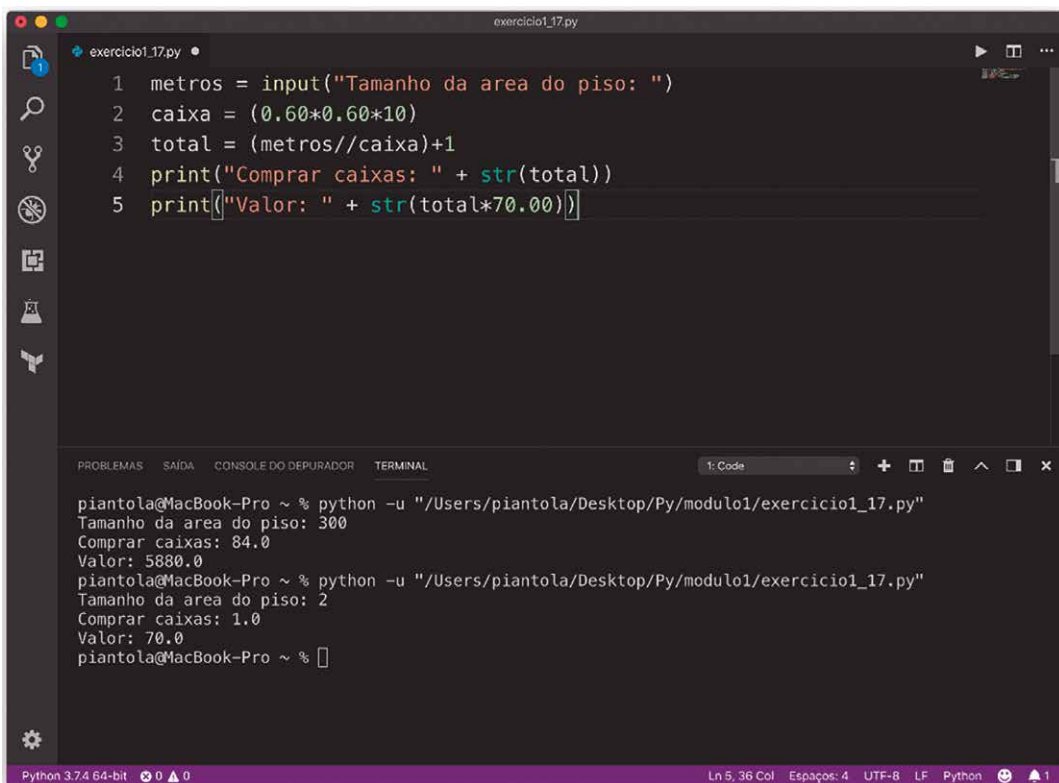
INTRODUÇÃO À PROGRAMAÇÃO ESTRUTURADA



```
exercício1_16.py
1 a = input("Digite um numero 1: ")
2 b = input("Digite um numero 2: ")
3 print("Parte inteira: " + str(a//b))
4 print("Resto: " + str(a%b))

PROBLEMAS SAÍDA CONSOLE DO DEPURADOR TERMINAL
2: Code
piantola@MacBook-Pro ~ % python -u "/Users/piantola/Desktop/Py/modulo1/exercicio1_16.py"
Digite um numero 1: 3
Digite um numero 2: 2
Parte inteira: 1
Resto: 1
piantola@MacBook-Pro ~ %
```

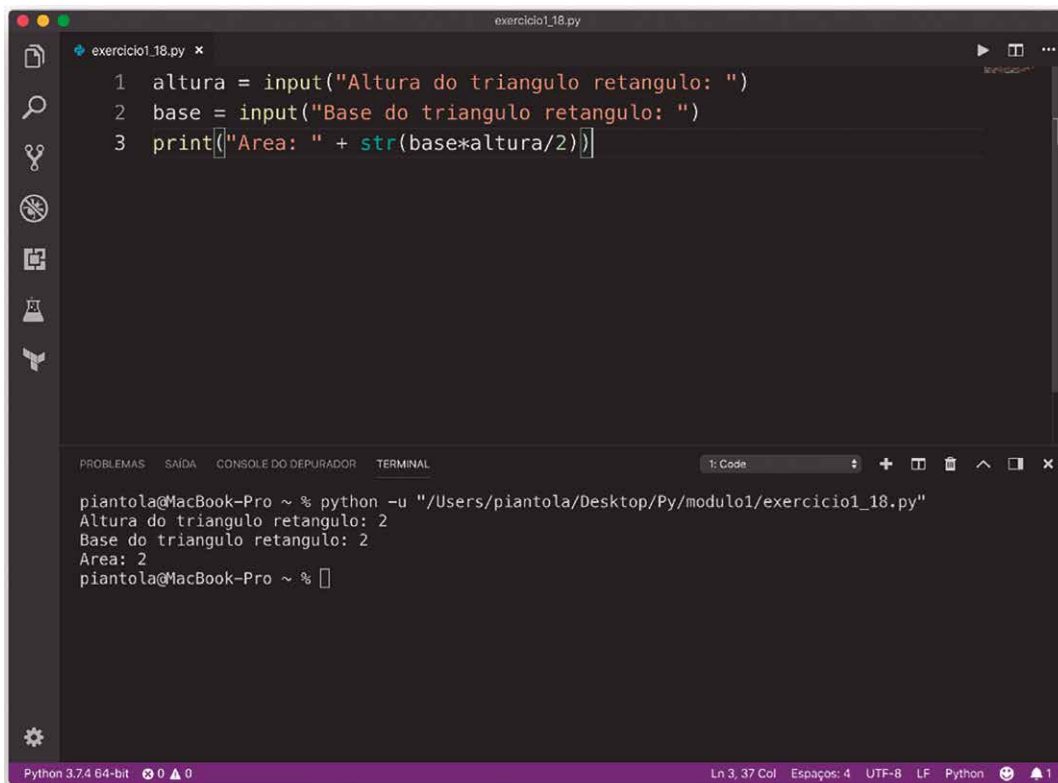
Figura 26 – Resolução do exercício 16



```
exercício1_17.py
1 metros = input("Tamanho da area do piso: ")
2 caixa = (0.60*0.60*10)
3 total = (metros//caixa)+1
4 print("Comprar caixas: " + str(total))
5 print("Valor: " + str(total*70.00))

PROBLEMAS SAÍDA CONSOLE DO DEPURADOR TERMINAL
1: Code
piantola@MacBook-Pro ~ % python -u "/Users/piantola/Desktop/Py/modulo1/exercicio1_17.py"
Tamanho da area do piso: 300
Comprar caixas: 84.0
Valor: 5880.0
piantola@MacBook-Pro ~ % python -u "/Users/piantola/Desktop/Py/modulo1/exercicio1_17.py"
Tamanho da area do piso: 2
Comprar caixas: 1.0
Valor: 70.0
piantola@MacBook-Pro ~ %
```

Figura 27 – Resolução do exercício 17



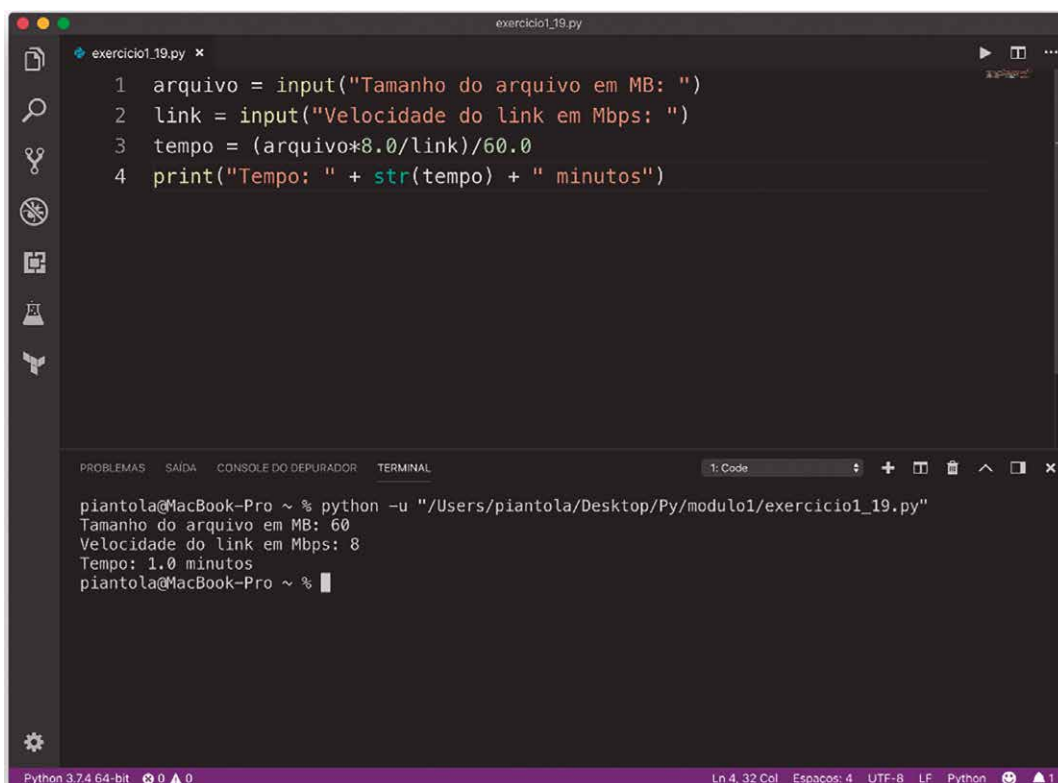
```
exercicio1_18.py
1 altura = input("Altura do triangulo retangulo: ")
2 base = input("Base do triangulo retangulo: ")
3 print("Area: " + str(base*altura/2))
```

PROBLEMAS SAÍDA CONSOLE DO DEPURADOR TERMINAL 1: Code

```
piantola@MacBook-Pro ~ % python -u "/Users/piantola/Desktop/Py/modulo1/exercicio1_18.py"
Altura do triangulo retangulo: 2
Base do triangulo retangulo: 2
Area: 2
piantola@MacBook-Pro ~ %
```

Python 3.7.4 64-bit 0 0 Python Ln 3, 37 Col Espaços: 4 UTF-8 LF Python 1

Figura 28 – Resolução do exercício 18



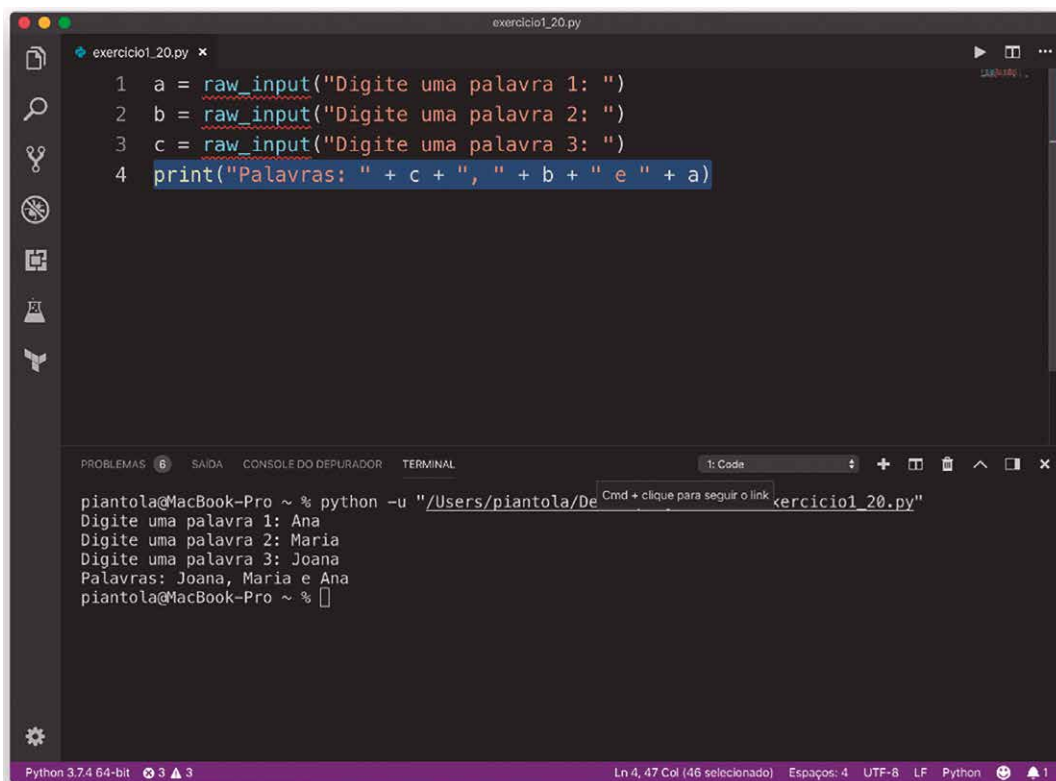
```
exercicio1_19.py
1 arquivo = input("Tamanho do arquivo em MB: ")
2 link = input("Velocidade do link em Mbps: ")
3 tempo = (arquivo*8.0/link)/60.0
4 print("Tempo: " + str(tempo) + " minutos")
```

PROBLEMAS SAÍDA CONSOLE DO DEPURADOR TERMINAL 1: Code

```
piantola@MacBook-Pro ~ % python -u "/Users/piantola/Desktop/Py/modulo1/exercicio1_19.py"
Tamanho do arquivo em MB: 60
Velocidade do link em Mbps: 8
Tempo: 1.0 minutos
piantola@MacBook-Pro ~ %
```

Python 3.7.4 64-bit 0 0 Python Ln 4, 32 Col Espaços: 4 UTF-8 LF Python 1

Figura 29 – Resolução do exercício 19



```
exercício1_20.py
1 a = raw_input("Digite uma palavra 1: ")
2 b = raw_input("Digite uma palavra 2: ")
3 c = raw_input("Digite uma palavra 3: ")
4 print("Palavras: " + c + ", " + b + " e " + a)

PROBLEMAS 6 SAÍDA CONSOLE DO DEPURADOR TERMINAL 1: Code
piantola@MacBook-Pro ~ % python -u "/Users/piantola/De Cmd + clique para seguir o link exercício1_20.py"
Digite uma palavra 1: Ana
Digite uma palavra 2: Maria
Digite uma palavra 3: Joana
Palavras: Joana, Maria e Ana
piantola@MacBook-Pro ~ %
```

Figura 30 – Resolução do exercício 20



Resumo

Na unidade I, vimos que a programação estruturada surgiu como um padrão de desenvolvimento de software no final da década de 1950 e até hoje é muito utilizada. Outros paradigmas de programação, como a programação orientada a objeto, incorporaram como base as técnicas de programação estruturada, comprovando sua importância.

Foi apresentada a linguagem Python, como linguagem de alto nível, multiparadigma, interpretada, imperativa, de tipagem dinâmica, popular e fácil de aprender, sendo a linguagem objeto dos estudos da programação estruturada deste livro-texto.

O Python está atualmente em sua versão 3 e é distribuído para sistemas de 32 e 64 bits.

Foi descrito o passo a passo das instalações em Windows, Linux e MacOS. Não é necessário instalar o Python, existe a opção de trabalhar on-line, no browser, através do Colab da Google.

Ambientes de desenvolvimento integrados (IDE) ajudam o programador a ser mais produtivo ao desenvolver os softwares. O Visual Studio Code, que foi desenvolvido pela Microsoft, é um IDE com muitas facilidades para a escrita de programas e depuração. Os IDE não são necessários ao desenvolvimento; é possível usar editores comuns de texto ou o próprio console do Python.

Os exemplos de código do primeiro tópico mostraram uma das bases da programação estruturada, as sequências. Elas são comandos executados linha a linha, em ordem de cima para baixo, como uma receita de bolo.

As variáveis são rótulos para referenciar espaços na memória do computador. Elas armazenam os dados que serão processados pelo programa e possuem um tipo dependente de seu conteúdo. No caso dos tipos numéricos: inteiro (int), ponto flutuante (float), complexo (complex) e booleano (bool). O Python possui um conjunto de palavras reservadas que não podem ser usadas como nome das variáveis. Além disso, os nomes de variáveis devem seguir algumas regras. Outro tipo muito usado de variáveis são as string ou textos, que têm uma seção exclusiva em outra unidade deste livro-texto.

Uma constante é um tipo de "variável" cujo valor não pode ser modificado. Os números são constantes numéricas. Outros exemplos são o número pi e a gravidade na Terra. Os programadores podem criar constantes em um módulo externo ao programa principal. Como as strings, os módulos têm uma seção própria em outra unidade deste material.

As expressões lógicas e aritméticas são importantes ferramentas de processamento de dados. As expressões lógicas estudadas foram: igualdade, maior, menor, maior ou igual, menor ou igual, não, diferente e seus conectivos OR (ou) e AND (e). Os operadores aritméticos ou matemáticos estudados foram: adição, subtração, multiplicação, divisão, exponenciação, parte inteira e módulo. É importante notar que os operadores têm uma ordem de precedência para resolução das expressões.

A estrutura sequencial é o controle mais básico da programação estruturada, em que os comandos são executados na ordem em que são especificados, um após o outro. Ela consiste em: entrada de dados, processamento e saída.



Exercícios

Questão 1. No contexto de linguagens de programação, podemos definir uma variável como um espaço na memória do computador destinado a um dado. Uma variável possui, além de um nome de identificação, um tipo, que deve ser compatível com a natureza do dado que se pretende armazenar. A linguagem Python trabalha com tipagem dinâmica. Isso significa que o programador, ao declarar uma variável, não necessita especificar qual é o seu tipo, pois o interpretador é capaz de inferi-lo de acordo com o valor atribuído à variável em questão. Portanto, para criar uma variável em Python, o programador precisa apenas especificar um nome e, em seguida, atribuir a ela um valor.

Considerando esse contexto, avalie as afirmativas a seguir e a relação proposta entre elas.

I – Ao digitar a linha `nota = 8.0` no código-fonte, o programador atribuiu o valor 8.0 à variável cujo nome é `nota`.

porque

II – O interpretador Python atribuirá o tipo inteiro (`int`) à variável `nota`.

A respeito dessas afirmativas, assinale a opção correta.

- A) As afirmativas I e II são proposições verdadeiras, e a II é uma justificativa correta da I.
- B) As afirmativas I e II são proposições verdadeiras, e a II não é uma justificativa correta da I.
- C) A afirmativa I é uma proposição verdadeira, e a II é uma proposição falsa.
- D) A afirmativa I é uma proposição falsa, e a II é uma proposição verdadeira.
- E) As afirmativas I e II são proposições falsas.

Resposta correta: alternativa C.

Análise da questão

Ao escrever a linha `nota = 8.0` no código-fonte, o programador escolheu o nome `nota` e atribuiu o valor inicial 8.0 à variável. O sinal de igualdade é o próprio operador de atribuição. Dessa forma, temos a seguinte sintaxe:

```
nome_da_variavel = valor_da_variavel
```

Logo, a afirmativa I é uma proposição verdadeira. O valor atribuído à variável será responsável pela definição do seu tipo. No caso, o valor 8.0 levará o interpretador a atribuir o tipo float à variável nota, mesmo que o algarismo após o ponto seja zero. Lembre-se de que, em Python, o ponto é o separador entre a parte inteira e a parte decimal dos numerais, e não a vírgula. Para que o interpretador atribuísse o tipo inteiro (int) à variável nota, a linha do código-fonte deveria ser modificada para `nota = 8`, pois, dessa forma, o interpretador infere que apenas valores inteiros serão atribuídos a ela. Portanto, a afirmativa II é uma proposição falsa.

Pode-se receber o tipo de uma variável por meio da função `type()`. Com ela, demonstramos o que foi discutido nesta questão. Observe o código a seguir:

```
a = 8
b = 8.0
c = True
d = "Ana"
e = 2 + 3.0

print(type(a))
print(type(b))
print(type(c))
print(type(d))
print(type(e))
```

A execução desse código resultará na impressão dos tipos das variáveis, que foram dinamicamente atribuídos. Observamos como resultado o que se mostra a seguir:

```
<class 'int'>
<class 'float'>
<class 'bool'>
<class 'str'>
<class 'float'>
```

Questão 2. Leia o texto a seguir, a respeito dos operadores lógicos utilizados na linguagem Python:

George Boole (1815–1864) desenvolveu a álgebra booleana, a base sobre a qual foram criadas a lógica digital do hardware de computador e a especificação formal das linguagens de programação. A álgebra booleana é a álgebra dos valores verdadeiro e falso. Inclui os operadores `and`, `or` e `not`, que podem ser usados para criar expressões booleanas, ou seja, expressões que são avaliadas como verdadeira ou falsa. As tabelas-verdade definem como essas operações são avaliadas.

Quadro 8

| p | q | p e q | p | q | p ou q | p | não p |
|------------|------------|------------|------------|------------|------------|------------|------------|
| verdadeiro | verdadeiro | verdadeiro | verdadeiro | verdadeiro | verdadeiro | verdadeiro | falso |
| verdadeiro | falso | falso | verdadeiro | falso | verdadeiro | falso | verdadeiro |
| falso | verdadeiro | falso | falso | verdadeiro | verdadeiro | | |
| falso | falso | falso | falso | falso | falso | | |

Adaptado de: PERKOVIC, L. *Introdução à computação usando Python*: um foco no desenvolvimento de aplicações. Rio de Janeiro: LTC, 2016. p. 20.

Com base nas informações do texto e nos seus conhecimentos, avalie o código Python a seguir:

```
x = True
y = False
z = True

a = x and y
b = x and (y or z)
c = x and (not z)

print('Valor lógico de a:', a)
print('Valor lógico de b:', b)
print('Valor lógico de c:', c)
```

Qual é a saída esperada após a execução do programa?

- A) Valor lógico de a: True
Valor lógico de b: True
Valor lógico de c: True
- B) Valor lógico de a: True
Valor lógico de b: True
Valor lógico de c: False
- C) Valor lógico de a: False
Valor lógico de b: False
Valor lógico de c: False
- D) Valor lógico de a: False
Valor lógico de b: True
Valor lógico de c: True
- E) Valor lógico de a: False
Valor lógico de b: True
Valor lógico de c: False

Resposta correta: alternativa E.

Análise da questão

O programa trabalha com seis variáveis distintas, todas elas do tipo booleano. As três primeiras (x, y e z) têm seus valores lógicos diretamente definidos no código. As outras três (a, b e c) têm seus valores lógicos atribuídos de acordo com expressões que utilizam os operadores lógicos and, or e not. As impressões de saída demonstram, justamente, os valores lógicos assumidos pelo último grupo de variáveis. Devemos, portanto, encontrar o valor atribuído a cada uma delas, observando as saídas dos operadores lógicos em questão.

Para encontrar valor lógico da variável a, basta observarmos a segunda linha da tabela-verdade do operador and.

```
a = x and y  
a = True and False  
a = False
```

No caso da variável b, precisamos primeiro definir o resultado da expressão entre parênteses, que utiliza o operador or, para, em seguida, encontrar seu nível lógico.

```
b = x and (y or z)  
b = True and (False or True)  
b = True and True  
b = True
```

Finalmente, devemos encontrar o valor da variável c, também respeitando os parênteses adotados.

```
c = x and (not z)  
c = True and (not True)  
c = True and False  
c = False
```

Portanto, as impressões em tela indicarão o que segue.

```
Valor lógico de a: False  
Valor lógico de b: True  
Valor lógico de c: False
```