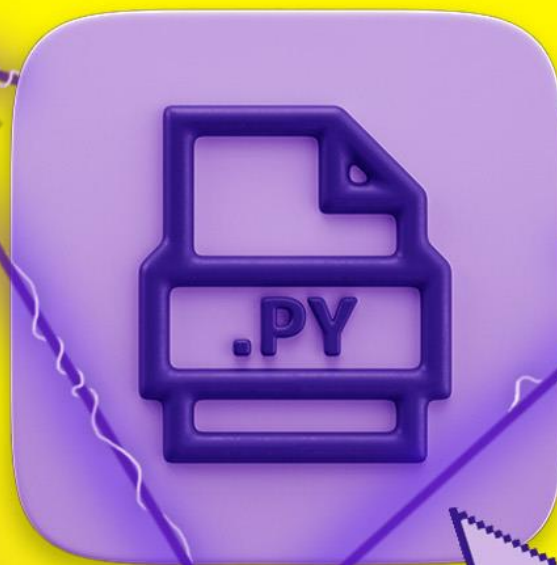


PYTHON DEVELOPMENT

INTRODUÇÃO E INSTALAÇÃO DO PYTHON



1

LISTA DE FIGURAS

Figura 1 – Segure a ansiedade, futuro dev!	5
Figura 2 – Guido van Rossum, o criador do Python	6
Figura 3 – Ranking de linguagens de programação de 2021	8
Figura 4 – Baixando o Python no Windows	10
Figura 5 – Selecione a opção “Add Python 3.x to PATH”	11
Figura 6 – Abrindo o Prompt de Comando	11
Figura 7 – Exibição da versão do Python que foi instalada	12
Figura 8 – O Xcode pode ser obtido na AppStore	13
Figura 9 – Baixando o Python no macOS	13
Figura 10 – O Ubuntu 20.04 já possui o Python 3.8 pré-instalado	14
Figura 11 – Download da versão Community do PyCharm	15
Figura 12 – Abertura do interpretador Python nos três sistemas operacionais	16
Figura 13 – Erro ao executar um comando inválido	16
Figura 14 – Execução correta da função exit()	17
Figura 15 – Erro de sintaxe na função print	18
Figura 16 – Execução do print com aspas simples e aspas duplas	18
Figura 17 – Função input solicitando dados	19
Figura 18 – Variável nome armazenando o resultado da função input	20
Figura 19 – Exibição do conteúdo da variável nome	20
Figura 20 – Documentação do Python em português	22
Figura 21 – Execução da função dir()	22
Figura 22 – Execução da função help()	23
Figura 23 – Conteúdo da documentação symbols	23

LISTA DE CÓDIGOS-FONTE

Código-fonte 1 – Programa Hello World funcionando corretamente	18
Código-fonte 2 – Utilizando a função input.....	19
Código-fonte 3 – Utilizando a função input.....	20
Código-fonte 4 – Exibindo uma mensagem com o nome do usuário de forma não recomendada	21
Código-fonte 5 – Exibindo uma mensagem com o nome do usuário de forma não recomendada	21

EMANIP

SUMÁRIO

INTRODUÇÃO E INSTALAÇÃO DO PYTHON	5
1 A COBRA VAI “CODAR”	5
1.1 Começando pelo começo!	5
1.1.1 De onde veio esse tal de Python?	6
1.1.2 Largamente adotada por quem?	7
1.2 Instalando tudo!	9
1.2.1 Instalando o Python no Windows	10
1.2.2 Instalando o Python no macOS	12
1.2.3 Instalando o Python no Linux	14
1.2.4 Instalando PyCharm no Windows, macOS ou Linux	14
2 HORA DE PROGRAMAR	15
2.1 Abrindo o interpretador	15
2.2 Exibindo mensagens e lendo dados	17
2.3 Me ajuda!	21
REFERÊNCIAS	25

INTRODUÇÃO E INSTALAÇÃO DO PYTHON

1 A COBRA VAI “CODAR”

1.1 Começando pelo começo!

Quer dizer que você decidiu que aprender a programar em Python é uma ótima ideia, certo? Mas... de onde veio essa história de que o Python é um ótimo lugar para começar e para se aprofundar no mundo do desenvolvimento de software? Dá pra virar o “Hackerman” da noite para o dia? (E por que tantas perguntas logo de cara?!)



Figura 1 – Segure a ansiedade, futuro dev!
Fonte: Pixabay (2021)

Alguém desavisado pode pensar que é uma bobeira sem tamanho entender a origem da linguagem de programação que será estudada, mas esse alguém não poderia estar mais equivocado, ainda mais se tratando do Python, que desde as suas origens tem os conceitos básicos que sustentam a linguagem, como a legibilidade, o reúso e a manutenibilidade, ou seja, a facilidade de ler e escrever um código, reutilizá-lo e mantê-lo atualizado com o passar do tempo (LUTZ, 2013).

Como este é um curso básico de desenvolvimento e a sua ansiedade de colocar a mão na massa deve estar aumentando a cada parágrafo, vamos nos ater apenas aos tópicos mais importantes da história dessa linguagem!

1.1.1 De onde veio esse tal de Python?

O Python é uma linguagem de programação e, de acordo com Sandra Puga e Gerson Rissetti, uma linguagem de programação é formada por palavras-chave que podem ser agrupadas em estruturas de programação para produzir um determinado comportamento em um computador.

Existem inúmeras linguagens de programação no mundo justamente porque existem inúmeros problemas diferentes a serem resolvidos. Sim, você não entendeu errado o início deste parágrafo: para diferentes problemas, existem diferentes linguagens de programação com diferentes abordagens.

Voltando ao Python, a linguagem foi concebida no fim dos anos 1980, por Guido van Rossum. Guido trabalhava na equipe responsável pela construção de uma linguagem chamada ABC, que tinha por objetivo ser uma linguagem de programação a ser utilizada por pessoas que sabiam fazer uso do computador, mas não eram programadores. A linguagem tinha vários trunfos, mas Guido percebeu que muitas pessoas ainda esbarravam em limitações e restrições tradicionais das linguagens de programação.



Figura 2 – Guido van Rossum, o criador do Python
Fonte: Wikimedia (2021)

Diante de tantos sucessos e fracassos do ABC, nosso estimado herói (não dá para chamar de nada a não ser herói o sujeito que criou uma das linguagens de programação mais poderosas do mundo, não é mesmo?) decidiu criar uma linguagem de programação que tivesse sintaxes simples, usasse apenas endentação (espaçamentos em um código) e um número pequeno, mas poderoso, de estruturas de dados.

Guido conta todos os detalhes sobre como construiu o Python e quais são suas semelhanças com o ABC em uma entrevista a Bill Verner, publicada em janeiro de 2003 (<https://www.artima.com/articles/the-making-of-python>). Para nós, basta saber que a linguagem continuou evoluindo mantendo-se fiel ao espírito básico do ABC: pessoas que não são programadoras (talvez seja este o seu caso) devem ter facilidade de compreender e utilizar as estruturas da linguagem, mantendo uma alta capacidade de produtividade.

A versão 2.0 do Python foi lançada no fim do ano 2000 e a versão 3.0 foi lançada no fim de 2008. Atualmente, as versões 3.10.x da linguagem já foram lançadas. Em 2021, em uma entrevista disponível no canal Microsoft Reactor (<https://www.youtube.com/watch?v=aYbNh3NS7iA>) do YouTube, Guido disse que o Python 3.x deve durar ainda muitos anos e que não veremos o Python 4 surgir tão cedo.

Juntando todos os pontinhos até aqui, dá para imaginar que se você se dedicar ao estudo do Python 3, encontrará uma linguagem de fácil aprendizado, com previsão para durar muitos anos e largamente adotada por diversos setores do mercado.

1.1.2 Largamente adotada por quem?

A edição de 2021 do ranking de linguagens de programação elaborada pelo IEEE (Institute of Electrical and Electronic Engineers), que utiliza 11 métricas diferentes de 8 fontes distintas (CareerBuilder, GitHub, Google, Hacker News, the IEEE, Reddit, Stack Overflow e Twitter), coloca o Python em primeiro lugar, considerando desenvolvimento para Web, aplicações desktop, empresariais, científicas e embarcadas. Você pode acessar o ranking e utilizar os filtros para encontrar as linguagens mais usadas para aplicações específicas: <https://spectrum.ieee.org/top-programming-languages/>.

Language Ranking: IEEE Spectrum

Rank	Language	Type	Score
1	Python	Web, Desktop, Embedded	100.0
2	Java	Web, Mobile, Desktop	95.4
3	C	Mobile, Desktop, Embedded	94.7
4	C++	Mobile, Desktop, Embedded	92.4
5	JavaScript	Web	88.1
6	C#	Web, Mobile, Desktop, Embedded	82.4
7	R	Desktop	81.7

Figura 3 – Ranking de linguagens de programação de 2021
Fonte: IEEE (2021)

Apesar de rankings serem divertidos e informativos, o Python claramente não é a única linguagem de programação poderosa disponível no mercado. E, acredite, você sempre encontrará outro desenvolvedor que tem uma linguagem de estimação. O importante é não perder o foco de que as linguagens de programação servem para resolver problemas! E o Python é muito bom para resolver problemas diferentes!

Quer desenvolver uma aplicação Web, como uma API, um CMS, um processador de e-mails ou qualquer coisa que possa ser instalada em um servidor e consumida por meio dos diferentes protocolos da internet? Você pode utilizar o Python com frameworks – como Django, Flask, Pyramid – e bibliotecas – como Requests, BeautifulSoup e diversas outras.

Sua área é computação científica? Não se preocupe, pois o Python conta com tantas bibliotecas e pacotes, como o SciPy e o Pandas, que já ultrapassou o R, uma das linguagens mais clássicas utilizadas por matemáticos, estatísticos e cientistas.

O Python pode ser utilizado até mesmo para a construção de bots que têm diferentes finalidades, ou para desenvolvimento embarcado, por meio do *micropython*! O céu é o limite para quem dominar as estruturas dessa linguagem!

E para aproveitar tudo o que o Python tem a oferecer, você precisará primeiro configurar o seu ambiente de desenvolvimento!

1.2 Instalando tudo!

Agora que você já conhece a história do Python e as áreas em que essa linguagem pode ser utilizada, é chegada a hora de sair programando, certo? Calma, jovem padawan, para programar em linguagem Python (ou na maior parte das linguagens de programação), primeiro você precisa instalar o ambiente de desenvolvimento.

Chamamos de *scripts* os códigos-fonte (texto de um programa escrito em uma linguagem de programação) elaborados em Python. Para que esses scripts possam ser interpretados e para que seu computador execute as instruções que você escrever, é necessário instalar um *interpretador*.

É fácil compreender esse cenário: você sabe (ou saberá, ao final desse curso) escrever um programa em Python, mas o seu computador precisará de um “tradutor” que seja capaz de transformar tudo o que você escrever em algo que a máquina também compreenda. Por isso a configuração do seu ambiente passará pela inevitável tarefa de instalar o Python!

“Mas não é só isso”, como diriam aqueles comerciais da TV que tentam lhe vender uma panela que pode ser utilizada para encher os pneus do carro (ou algo do gênero)! Escrever os seus scripts pode ser extremamente mais fácil se você instalar também uma IDE, um Ambiente de Desenvolvimento Integrado (*Integrated Development Environment*).

Uma IDE é um programa de computador que tem ferramentas para auxiliar no desenvolvimento de um software. As IDEs possuem recursos muito interessantes, como corrigir automaticamente uma palavra da linguagem que tenha sido escrita de forma errada, ou avisar ao desenvolvedor quando o código retornou uma exceção.

Existem inúmeras IDEs disponíveis no mercado e como a IDE é apenas uma ferramenta, é uma ótima ideia testar várias delas para ver qual tem os melhores recursos disponíveis para você.

Introdução e instalação do Python

Para o propósito deste curso básico, usaremos o PyCharm, mas à medida que você se sentir à vontade, pode testar ferramentas como o VSCode, o Jupyter e até mesmo o Google Colab.

Vamos começar?

1.2.1 Instalando o Python no Windows

O primeiro passo para instalar o Python no Windows é acessar o site oficial da linguagem (<https://www.python.org>) e clicar no menu “Downloads”. A versão mais atual do Python 3 disponível para o Windows será exibida em um botão que, ao ser clicado, iniciará o download do instalador!



Figura 4 – Baixando o Python no Windows
Fonte: Elaborado pelo autor (2021)

Uma vez que o instalador for baixado e executado, é necessário selecionar a opção “Add Python 3.x to PATH” (em que o x refere-se ao número da versão que será instalada). Essa opção é fundamental para que o Windows seja capaz de executar o interpretador do Python sem a necessidade de escrever o caminho da pasta em que ele se encontra.

Introdução e instalação do Python



Figura 5 – Selecione a opção “Add Python 3.x to PATH”
Fonte: Elaborado pelo autor (2021)

Depois de selecionar a opção indicada, basta clicar em “Install Now” e o instalador realizará todos os procedimentos necessários.

Ao final da instalação, podemos verificar se o Python foi instalado corretamente por meio do PowerShell ou do CMD. Basta clicar no botão “Iniciar”, digitar CMD ou PowerShell e abrir o terminal.

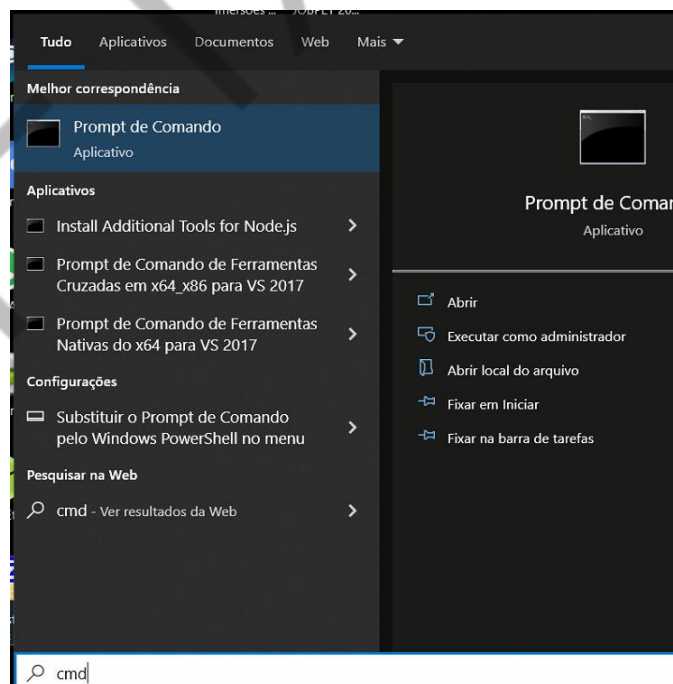
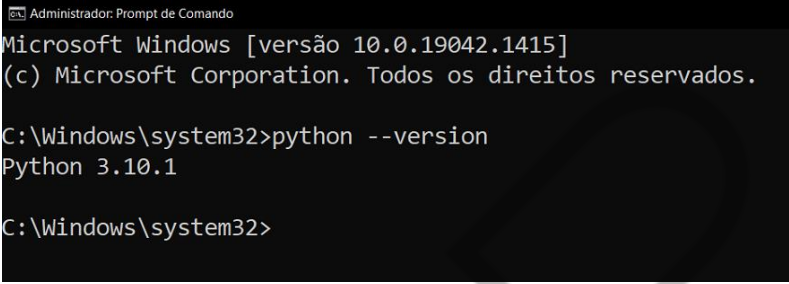


Figura 6 – Abrindo o Prompt de Comando
Fonte: Elaborado pelo autor (2021)

No prompt, basta digitar a instrução “python --version” para verificar se o Python está instalado na versão que foi obtida por meio do site oficial. Pode ser necessário reiniciar o computador para que o terminal passe a compreender corretamente a localização do interpretador do Python.



```
Administrador: Prompt de Comando
Microsoft Windows [versão 10.0.19042.1415]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Windows\system32>python --version
Python 3.10.1

C:\Windows\system32>
```

Figura 7 – Exibição da versão do Python que foi instalada
Fonte: Elaborado pelo autor (2021)

1.2.2 Instalando o Python no macOS

Se você possui um equipamento com o macOS configurado, há uma boa e uma má notícia! A boa é que o macOS já vem com o Python instalado. A má notícia é que a versão que vem configurada de fábrica é a versão 2.x da linguagem. Esta possui algumas sintaxes diferentes da versão 3.x e já foi descontinuada.

Se você planeja continuar se aventurando no mundo do desenvolvimento, pode ser interessante instalar o Xcode antes de instalar o Python. O Xcode é uma ferramenta de desenvolvimento da Apple que já instala diversas dependências e ferramentas para o desenvolvimento de software. Ele pode ser obtido na AppStore do seu macOS.

Introdução e instalação do Python

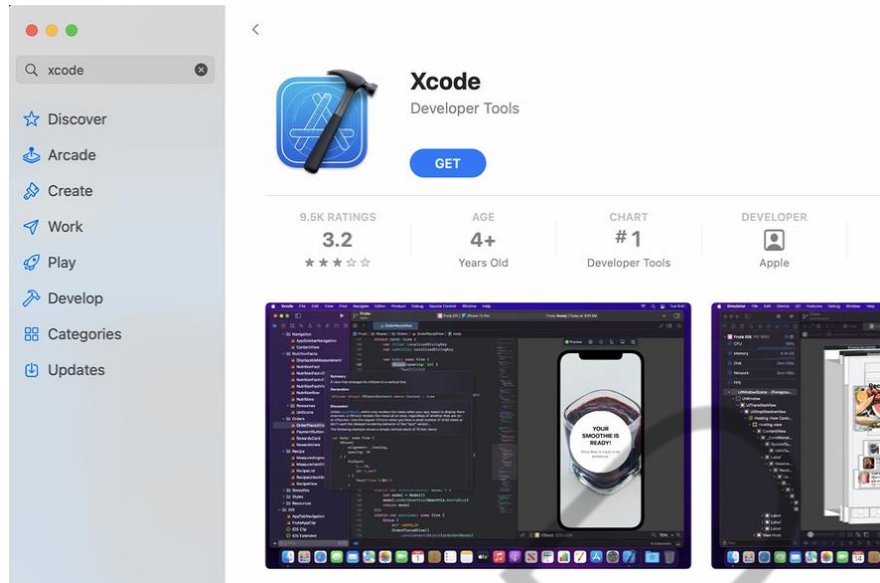


Figura 8 – O Xcode pode ser obtido na AppStore
Fonte: Elaborado pelo autor (2021)



Figura 9 – Baixando o Python no macOS
Fonte: Elaborado pelo autor (2021)

Depois de executar o instalador e aceitar os termos de licença, o Python estará instalado e configurado.

1.2.3 Instalando o Python no Linux

Então você é um fã do mundo do software livre e tem um pinguim comandando as principais funções da sua máquina? Perfeito! Talvez você nem precise instalar o Python no seu sistema operacional.

A maior parte dos sistemas Linux, como o Ubuntu na sua versão 20.04, já possui o Python 3 instalado, e podemos conferir essa informação com facilidade ao abrir o terminal do sistema operacional e executar o comando “python3 --version”.

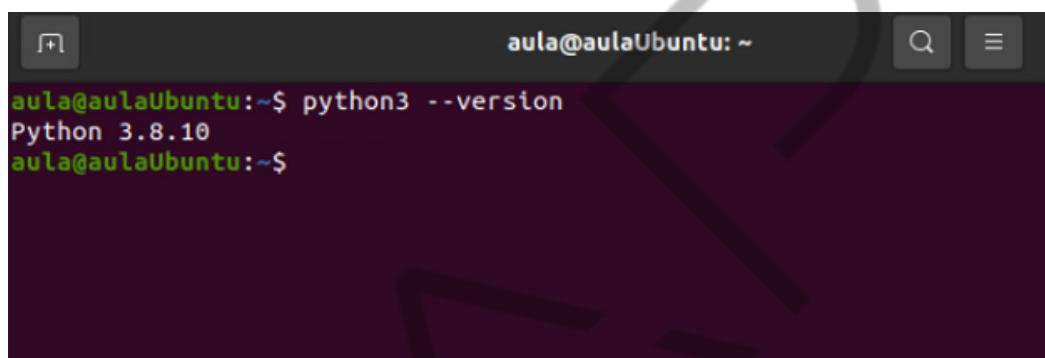
A screenshot of a terminal window on Ubuntu 20.04. The terminal has a dark purple background. The prompt is 'aula@aulaUbuntu: ~'. The command 'python3 --version' has been entered, and the output is 'Python 3.8.10'. The prompt is now 'aula@aulaUbuntu: ~\$'.

Figura 10 – O Ubuntu 20.04 já possui o Python 3.8 pré-instalado
Fonte: Elaborado pelo autor (2021)

Dada a quantidade de distribuições Linux disponíveis no mercado, a documentação oficial do Python apresenta um guia que pode ser consultado aqui: <https://docs.python.org/pt-br/3/using/unix.html>.

1.2.4 Instalando PyCharm no Windows, macOS ou Linux

Agora que o Python já está instalado e funcionando no seu sistema operacional, vamos instalar a nossa IDE. Apesar de utilizarmos o PyCharm nos exemplos que serão apresentados, para acompanhar este curso você pode instalar a IDE que desejar.

A JetBrains, fabricante do PyCharm, fornece duas versões da IDE: a versão Professional (paga) e a versão Community (gratuita). Instalaremos a versão Community, que possui todos os recursos que utilizaremos ao longo deste curso.

Introdução e instalação do Python

A instalação do PyCharm é praticamente idêntica no Windows e no macOS: acesse a página de downloads (<https://www.jetbrains.com/pt-br/pycharm/download/>), baixe e execute o instalador. Só é necessário ficar atento à versão do macOS, que é diferente para equipamentos com processadores Intel e processadores Apple Silicon.

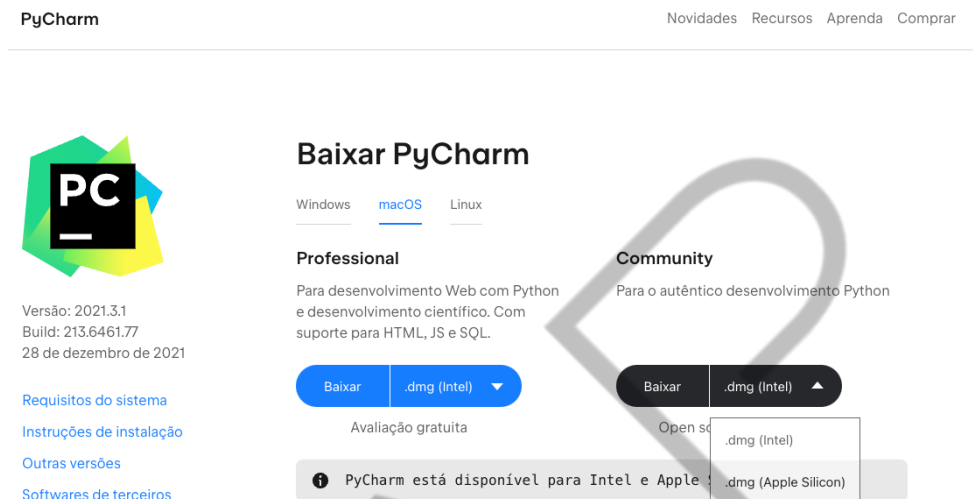


Figura 11 – Download da versão Community do PyCharm
Fonte: Elaborado pelo autor (2021)

No caso do Linux, após o download, é necessário seguir as instruções do arquivo install-Linux-tar.txt para que a IDE seja instalada.

2 HORA DE PROGRAMAR

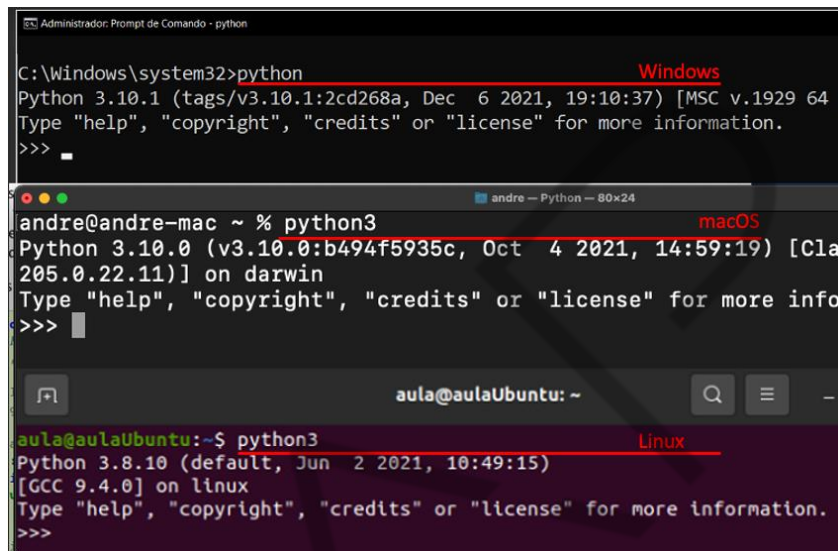
2.1 Abrindo o interpretador

Ufa! Agora que você instalou o Python e a sua IDE favorita, deve estar louco de vontade de começar a programar! É chegada a hora de colocarmos as mãos no teclado e digitarmos alguns códigos em linguagem Python!

Vamos começar a criar nossos primeiros scripts diretamente no terminal e, no Capítulo 2, utilizaremos a nossa IDE para facilitarmos a nossa vida.

Introdução e instalação do Python

O primeiro passo é abrir o terminal do seu computador (Prompt de Comando ou PowerShell no Windows, Terminal no macOS ou Linux) e abrir também o interpretador do Python. No caso do Windows, basta escrever “python” e apertar ENTER no seu teclado; no caso do macOS e do Linux, escreveremos “python3” e apertaremos ENTER.



The image displays three terminal windows side-by-side, each showing the command to start the Python interpreter. The top window is for Windows, the middle for macOS, and the bottom for Linux. Each window shows the command being entered, the version of Python installed, and the prompt for further instructions.

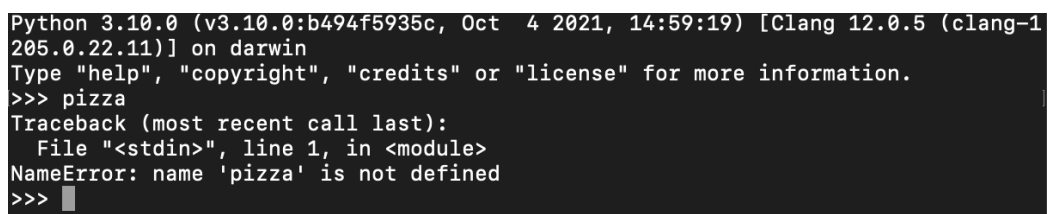
```
Administrator: Prompt de Comando - python
C:\Windows\system32>python
Python 3.10.1 (tags/v3.10.1:2cd268a, Dec 6 2021, 19:10:37) [MSC v.1929 64
Type "help", "copyright", "credits" or "license" for more information.
>>>

andre@andre-mac ~ % python3
Python 3.10.0 (v3.10.0:b494f5935c, Oct 4 2021, 14:59:19) [Clang 12.0.5 (clang-1
205.0.22.11)] on darwin
Type "help", "copyright", "credits" or "license" for more info
>>>

aula@aulaUbuntu: ~$ python3
Python 3.8.10 (default, Jun 2 2021, 10:49:15)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Figura 12 – Abertura do interpretador Python nos três sistemas operacionais
Fonte: Elaborado pelo autor (2021)

Uma vez que o interpretador está aberto, se digitarmos um comando da linguagem Python ele será imediatamente executado após pressionarmos ENTER. Caso o comando não exista, será exibido um erro. Veja este exemplo, em que escreveremos o comando “pizza”, que não existe, e depois a função “exit()”, que existe e fecha o interpretador.



The image shows a terminal window where the command 'pizza' was entered. Since 'pizza' is not a valid Python command, a NameError was raised. The error message indicates that the name 'pizza' is not defined.

```
Python 3.10.0 (v3.10.0:b494f5935c, Oct 4 2021, 14:59:19) [Clang 12.0.5 (clang-1
205.0.22.11)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> pizza
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'pizza' is not defined
>>>
```

Figura 13 – Erro ao executar um comando inválido
Fonte: Elaborado pelo autor (2021)

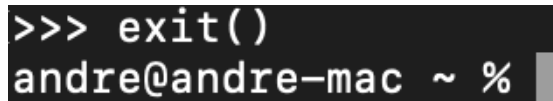
A terminal window with a black background and white text. The first line shows the command '>>> exit()' being entered. The second line shows the prompt 'andre@andre-mac ~ %' followed by a cursor, indicating the command was executed successfully and the shell prompt returned.

Figura 14 – Execução correta da função `exit()`
Fonte: Elaborado pelo autor (2021)

Podemos observar que, quando digitamos um comando inexistente, recebemos um erro como resposta do Python. Por outro lado, ao digitarmos a função “`exit()`”, a execução ocorre normalmente e o interpretador é fechado.

Para continuarmos, abra novamente o seu interpretador.

2.2 Exibindo mensagens e lendo dados

Existe uma brincadeira clássica no mundo dos desenvolvedores que diz que o seu primeiro programa em qualquer linguagem de programação deve ser um que exiba a mensagem “Olá, mundo!” para o usuário do seu programa. Essa prática tem muitas décadas e existe a teoria de que ela começou com o livro *A Tutorial Introduction to the Language B*, escrito por Brian Kernigham em 1972.

Independentemente de onde essa brincadeira começou, ela nos ensina a primeira habilidade básica em uma linguagem de programação: exibir informações para o usuário. Por isso vamos nos manter clássicos e escrever nosso primeiro “Hello World” em Python.

Para exibirmos mensagens na tela, utilizaremos a função *print*. Essa função recebe um conteúdo e exibe em formato de texto na tela. Sua sintaxe, ou seja, sua estrutura, é: *print(Conteúdo que será exibido)*.

Vamos testar? Abra seu interpretador e digite *print(Hello World!)*.

```
>>> print(Hello World)
File "<stdin>", line 1
    print(Hello World)
          ^^^^^^^^^^^^^
SyntaxError: invalid syntax. Perhaps you forgot a comma?
>>> █
```

Figura 15 – Erro de sintaxe na função print
Fonte: Elaborado pelo autor (2021)

O resultado foi frustrante, não é mesmo? A gente esperava ver um programa incrível rodando (tá bom, talvez não “incrível”, mas pelo menos rodando...) e fomos surpreendidos com uma mensagem de erro! Mas, fique calmo, porque no mundo do desenvolvimento as mensagens de erro são nossas amigas!

Se colocarmos nosso inglês para funcionar, vamos observar que o interpretador do Python está tentando nos ajudar! Ele disse: “SyntaxError: invalid syntax. Perhaps you forgot a comma?”, ou seja, “ErroSintaxe: sintaxe inválida. Talvez você tenha esquecido uma vírgula?”.

O interpretador já nos deu uma ótima pista do nosso erro: nossa sintaxe contém alguma coisa errada. A sugestão dele é que talvez tenhamos esquecido uma vírgula, o que não é bem o caso. O que aconteceu é que o comando `print()` exibe textos, e nós não avisamos ao Python que *Hello World* é um texto. Para sinalizarmos que algo é um texto em Python, devemos usar aspas simples ou aspas duplas.

Vamos tentar rodar esta versão do nosso código (que você pode copiar, se quiser):

```
print("Hello world")
```

Código-fonte 1 – Programa Hello World funcionando corretamente
Fonte: Elaborado pelo autor (2021)

```
>>> print("Hello world")
Hello world
>>> print('Hello world')
Hello world
```

Código que digitamos
Resultado do código executado

Figura 16 – Execução do print com aspas simples e aspas duplas
Fonte: Elaborado pelo autor (2021)

Podemos perceber que o código funciona normalmente quando colocamos nossa mensagem entre aspas simples ou aspas duplas. Com certeza você conhecerá alguém que tentará te convencer que o correto em Python é utilizar as aspas simples (ou duplas, dependendo do interlocutor), mas a PEP 8 (o guia de estilo oficial para códigos em Python, que estudaremos no Capítulo 6) afirma que não há regra nesse sentido, contanto que você mantenha a coerência (ou seja, use sempre uma ou outra).

Nosso *Hello World* já funciona muito bem e você pode se considerar um estudante de Python! Mas que tal sermos um pouco mais ousados e avançarmos no nosso estudo?

E se a gente quisesse perguntar para o usuário qual é o nome dele e depois exibir uma mensagem personalizada que diga “Bom dia, FULANO”?

Para isso, vamos precisar receber dados do usuário, e a função responsável por isso chama-se *input*. A sintaxe da função *input* é *input(Texto orientando o usuário)*.

Vamos executar o código abaixo em nosso interpretador:

```
input("Por favor, informe o seu nome: ")
```

Código-fonte 2 – Utilizando a função *input*
Fonte: Elaborado pelo autor (2021)

```
>>> input("Por favor, informe o seu nome: ")  
Por favor, informe o seu nome: André David  
'André David'
```

Figura 17 – Função *input* solicitando dados
Fonte: Elaborado pelo autor (2021)

Ao executarmos o código, percebemos que a função *input* funcionou muito bem! Afinal de contas, ela foi capaz de solicitar uma entrada de dados para o usuário e armazenar o que ele digitou. Mas como fazemos para usar o nome que foi digitado pelo usuário?

Para armazenar um dado que será utilizado enquanto o nosso programa funciona, precisamos utilizar uma estrutura chamada *variável*. Uma variável nada mais é do que um espaço que o nosso programa utiliza da memória RAM para gravar dados temporariamente.

Existem diversos tipos de variáveis, mas nos preocuparemos com isso no próximo capítulo. O mais importante agora é criarmos uma variável, e a sintaxe para isso é extremamente simples: *nome_da_variável = valor*. Isso quer dizer que vamos inventar um nome, escrever o sinal de igual e depois indicaremos o que será guardado nessa variável. Dessa forma, nosso programa anterior fica assim:

```
nome = input("Por favor, informe o seu nome: ")
```

Código-fonte 3 – Utilizando a função input
Fonte: Elaborado pelo autor (2021)

```
>>> nome = input("Por favor, informe o seu nome: ")
Por favor, informe o seu nome: André David
>>> █
```

Figura 18 – Variável nome armazenando o resultado da função input
Fonte: Elaborado pelo autor (2021)

Perceba que agora o nome do usuário não é mais exibido depois que o usuário digita, mas não se preocupe: ele está armazenado na variável nome! E para isso basta digitarmos *nome* e, *voilà*, o nome armazenado será exibido:

```
>>> nome = input("Por favor, informe o seu nome: ")
Por favor, informe o seu nome: André David
>>> nome
'André David'
```

Figura 19 – Exibição do conteúdo da variável nome
Fonte: Elaborado pelo autor (2021)

Legal, não é mesmo?

Para encerrarmos este programa, vamos exibir o nome do usuário com a mensagem de saudação. Porém faremos isso de uma maneira **não recomendada**, que nos entregará o resultado desejado, mas talvez traga alguns problemas que vamos explorar no próximo capítulo do curso.

Para fazer a exibição do nome do usuário com as boas-vindas, vamos modificar o nosso print:

```
print("Bom dia " + nome)
```

Código-fonte 4 – Exibindo uma mensagem com o nome do usuário de forma não recomendada
Fonte: Elaborado pelo autor (2021)

Perceba que o texto “Bom dia” foi colocado entre aspas, mas o operador + e a variável *nome* não. Fizemos isso porque queremos indicar que “Bom dia” é um texto, mas que o operador + deve ser interpretado como um operador (que nesse caso unirá os dois textos) e *nome* deverá ser interpretado como uma variável e seu conteúdo deve ser exibido.

Nosso programa completo ficou assim:

```
nome = input("Por favor, informe o seu nome: ")  
print("Bom dia " + nome)
```

Código-fonte 5 – Exibindo uma mensagem com o nome do usuário de forma não recomendada
Fonte: Elaborado pelo autor (2021)

Antes de encerrarmos, que tal conhecermos duas instruções que podem ajudá-lo em seu estudo?

2.3 Me ajuda!

Para cada pessoa que detesta ler manuais, existe uma equipe de profissionais dedicadas a escrever de forma detalhada tudo o que é necessário para operar um determinado recurso. E no caso do mundo *dev*, não podemos ter preguiça de prestigiar as verdadeiras obras de arte que são as *documentações*.

O Python possui uma documentação extremamente detalhada que pode ser acessada em português por meio do site <https://docs.python.org/pt-br/3/>.

Introdução e instalação do Python

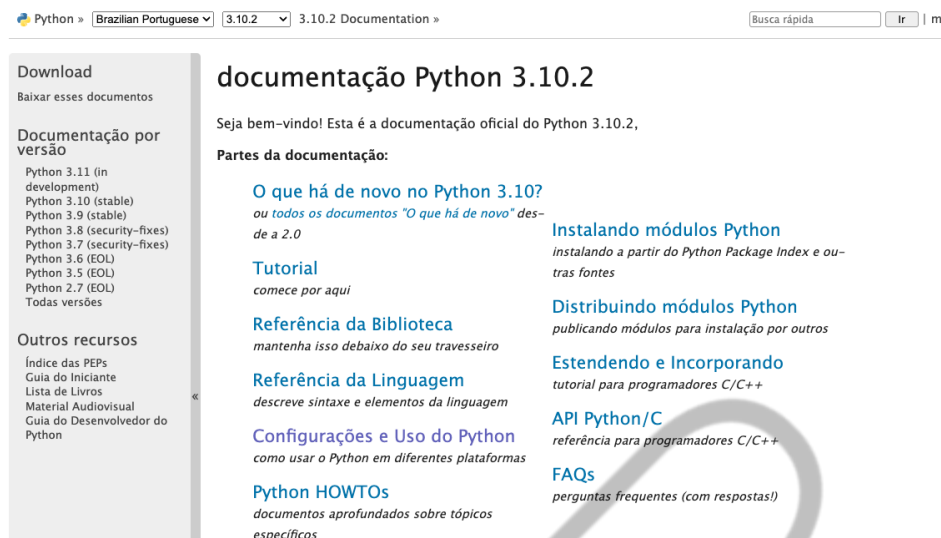


Figura 20 – Documentação do Python em português
Fonte: Elaborado pelo autor (2021)

Porém haverá momentos em que você estará “sozinho” com o seu interpretador e precisará de alguma informação importante sobre a linguagem. Por essa razão, vamos adicionar ao nosso cinto de utilidades duas funções: `dir()` e `help()`.

Suponha que você precise saber quais recursos pode utilizar em um determinado ambiente do Python, ou mesmo de um determinado objeto. Basta utilizar a função `dir()` e será retornada uma lista com esses recursos. Veja este exemplo, em que acabamos de abrir o interpretador e executamos essa função:

```
Python 3.10.0 (v3.10.0:b494f5935c, Oct 4 2021, 14:59:19) [Clang 12.0.5 (clang-1205.0.22.11)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> dir()
['__annotations__', '__builtins__', '__doc__', '__loader__', '__name__', '__package__', '__spec__']
```

Figura 21 – Execução da função `dir()`
Fonte: Elaborado pelo autor (2021)

Por ora, essa função pode não parecer ter grande utilidade, mas à medida que nos aprofundarmos você vai passar a encontrar mais valor nesse recurso.

Introdução e instalação do Python

Porém, independentemente do futuro, é preciso reconhecer: apenas listar os recursos sem saber o que eles fazem pode ser um pouco limitante. Por isso existe a função `help()`. Veja o que acontece quando executamos essa função em nosso interpretador:

```
Python 3.10.0 (v3.10.0:b494f5935c, Oct 4 2021, 14:59:19) [Clang 12.0.5 (clang-205.0.22.11)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> help()

Welcome to Python 3.10's help utility!

If this is your first time using Python, you should definitely check out
the tutorial on the internet at https://docs.python.org/3.10/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules. To quit this help utility and
return to the interpreter, just type "quit".

To get a list of available modules, keywords, symbols, or topics, type
"modules", "keywords", "symbols", or "topics". Each module also comes
with a one-line summary of what it does; to list the modules whose name
or summary contain a given string such as "spam", type "modules spam".

help> █
```

Figura 22 – Execução da função `help()`
Fonte: Elaborado pelo autor (2021)

Estamos diante de uma ferramenta interativa de ajuda! Se digitarmos *quit*, essa ferramenta será finalizada. Por outro lado, se digitarmos *modules*, *keywords*, *symbols* ou *topics*, teremos acesso a uma documentação via terminal. Vamos solicitar a documentação de símbolos, por exemplo:

```
help> symbols

Here is a list of the punctuation symbols which Python assigns special meaning
to. Enter any symbol to get more help.

!=          +          <=          ~-
"           +=          <>           b"
"""         '          ==          b'
%           _          >           f"
%=          -=          >=          f'
&           .          >>          j
&=          ...         >>=         j
'           /          @          r"
'''         //         J          r'
(           //=         [          u"
)           /=          \          u'
*           :           ]          |
**          <           ^          |=
**=         <<          ^=          ~
*=          <<=         -
```

Figura 23 – Conteúdo da documentação `symbols`
Fonte: Elaborado pelo autor (2021)

Introdução e instalação do Python

É exibida uma lista dos símbolos em Python. E se solicitarmos a explicação do símbolo de aspas, o Python nos mostrará uma explicação dos seus usos.

DICA: Caso uma documentação seja muito extensa, basta pressionar a letra Q no seu teclado para retornar ao Help. Para sair do Help, basta digitar Quit e pressionar Enter.

Agora que temos um ambiente de desenvolvimento configurado, já sabemos exibir mensagens, solicitar dados e armazenar em uma variável, acessar a documentação via web ou via terminal, estamos prontos para continuar nossa jornada no incrível mundo do Python!

REFERÊNCIAS

LUTZ, M. **Learning Python**. Sebastopol: O'Reilly Media Inc, 2013.

PUGA, S.; RISSETI, G. **Lógica de programação e estrutura de dados com aplicações em Java**. São Paulo: Pearson Prentice Hall, 2009.

THE HISTORY OF Hello World. **Software Guild**. 17 de julho de 2015. Disponível em: <<https://www.thesoftwareguild.com/blog/the-history-of-hello-world>>. Acesso em: 12 jan. 2022.

THE MAKING OF PYTHON – A Conversation with Guido van Rossum, Part I. **Artima**. 13 de janeiro de 2003. Disponível em: <<https://www.artima.com/articles/the-making-of-python>>. Acesso em: 12 jan. 2022.