

PROGRAMAÇÃO PARA BIG DATA

Prof. Rodrigo Ramos Nogueira



Indaiá - 2020

1ª Edição



Copyright © UNIASSELVI 2020

Elaboração:

Prof. Rodrigo Ramos Nogueira

Revisão, Diagramação e Produção:

Centro Universitário Leonardo da Vinci – UNIASSELVI

Ficha catalográfica elaborada na fonte pela Biblioteca Dante Alighieri

UNIASSELVI – Indaial.

N778p

Nogueira, Rodrigo Ramos

Programação para big data. / Rodrigo Ramos Nogueira. – Indaial:
UNIASSELVI, 2020.

175 p.; il.

ISBN 978-85-515-0438-3

1. Programação em big data. - Brasil. Centro Universitário Leonardo Da Vinci.

CDD 004

APRESENTAÇÃO

Caro acadêmico!

Estamos iniciando o estudo da disciplina Programação para Big Data. Esta disciplina objetiva proporcionar uma imersão de conceitos para programação em Big Data. Iremos aprender desde os mais simples conceitos para que você se habitue com a programação, até nos aprofundarmos nas estruturas de dados mais utilizados por engenheiros de dados em todo o mundo.

Este livro conta com diversos recursos didáticos externos, recomendamos fortemente que você realize todos os exemplos e exercícios resolvidos para um aproveitamento excepcional da disciplina.

Neste contexto, o Livro Didático de Programação para Big Data está dividido em três unidades de estudo: Unidade 1 - Introdução à Programação para Big Data; Unidade 2 - Principais Estruturas de Dados do Python e Unidade 3 - Ferramentas Avançadas para Desenvolver um projeto.

Aproveitamos a oportunidade para destacar a importância das autoatividades, lembrando que essas atividades NÃO SÃO OPCIONAIS. Elas objetivam a fixação dos conceitos apresentados. Em caso de dúvida na realização das atividades, sugerimos que você entre em contato com seu tutor externo ou com a tutoria da UNIASSELVI, não prosseguindo as atividades sem ter sanado todas as dúvidas que surgiram.

Bom estudo! Sucesso na sua trajetória acadêmica e profissional!

Prof. Rodrigo Ramos Nogueira



Você já me conhece das outras disciplinas? Não? É calouro? Enfim, tanto para você que está chegando agora à UNIASSELVI quanto para você que já é veterano, há novidades em nosso material.

Na Educação a Distância, o livro impresso, entregue a todos os acadêmicos desde 2005, é o material base da disciplina. A partir de 2017, nossos livros estão de visual novo, com um formato mais prático, que cabe na bolsa e facilita a leitura.

O conteúdo continua na íntegra, mas a estrutura interna foi aperfeiçoadas com nova diagramação no texto, aproveitando ao máximo o espaço da página, o que também contribui para diminuir a extração de árvores para produção de folhas de papel, por exemplo.

Assim, a UNIASSELVI, preocupando-se com o impacto de nossas ações sobre o ambiente, apresenta também este livro no formato digital. Assim, você, acadêmico, tem a possibilidade de estudá-lo com versatilidade nas telas do celular, tablet ou computador.

Eu mesmo, UNI, ganhei um novo layout, você me verá frequentemente e surgirei para apresentar dicas de vídeos e outras fontes de conhecimento que complementam o assunto em questão.

Todos esses ajustes foram pensados a partir de relatos que recebemos nas pesquisas institucionais sobre os materiais impressos, para que você, nossa maior prioridade, possa continuar seus estudos com um material de qualidade.

Aproveito o momento para convidá-lo para um bate-papo sobre o Exame Nacional de Desempenho de Estudantes – ENADE.

Bons estudos!

BATE SOBRE O PAPO ENADE!



Olá, acadêmico!



Você já ouviu falar sobre o ENADE?

Se ainda não ouviu falar nada sobre o ENADE, agora você receberá algumas informações sobre o tema.

Ouviu falar? Ótimo, este informativo reforçará o que você já sabe e poderá lhe trazer novidades.



Vamos lá!



Qual é o significado da expressão ENADE?

EXAME NACIONAL DE DESEMPENHO DOS ESTUDANTES

Em algum momento de sua vida acadêmica você precisará fazer a prova ENADE.



Que prova é essa?



É **obrigatória**, organizada pelo INEP – Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira.

Quem determina que esta prova é obrigatória... O **MEC – Ministério da Educação**.



O objetivo do MEC com esta prova é o de avaliar seu desempenho acadêmico assim como a qualidade do seu curso.

Fique atento! Quem não participa da prova fica impedido de se formar e não pode retirar o diploma de conclusão do curso até regularizar sua situação junto ao MEC.



Não se preocupe porque a partir de hoje nós estaremos auxiliando você nesta caminhada.



Você receberá outros informativos como este, complementando as orientações e esclarecendo suas dúvidas.



Você tem uma trilha de aprendizagem do ENADE, receberá e-mails, SMS, seu tutor e os profissionais do polo também estarão orientados.

Participará de webconferências entre outras tantas atividades para que esteja preparado para #mandar bem na prova ENADE.

Nós aqui no NEAD e também a equipe no polo estamos com você para vencermos este desafio.

Conte sempre com a gente, para juntos mandarmos bem no ENADE!



UNIASSELVI





Olá, acadêmico! Iniciamos agora mais uma disciplina e com ela um novo conhecimento.



Com o objetivo de enriquecer seu conhecimento, construímos, além do livro que está em suas mãos, uma rica trilha de aprendizagem, por meio dela você terá contato com o vídeo da disciplina, o objeto de aprendizagem, materiais complementares, entre outros, todos pensados e construídos na intenção de auxiliar seu crescimento.

Acesse o QR Code, que levará ao AVA, e veja as novidades que preparamos para seu estudo.

Conte conosco, estaremos juntos nesta caminhada!

Sumário

UNIDADE 1 – INTRODUÇÃO À PROGRAMAÇÃO PARA BIG DATA	1
TÓPICO 1 – BIG DATA: MOTIVAÇÃO PARA PROGRAMAÇÃO EM UM CENÁRIO DE GRANDES VOLUMES DE DADOS	3
1 INTRODUÇÃO	3
2 O QUE É UM CIENTISTA DE DADOS?	6
3 PRINCIPAIS LINGUAGENS DE PROGRAMAÇÃO PARA BIG DATA.....	7
3.1 MATLAB	8
3.2 LINGUAGEM R	11
3.3 SCALA.....	13
4 PYTHON – A LINGUAGEM QUE GANHOU O CORAÇÃO DO BIG DATA	14
RESUMO DO TÓPICO 1.....	20
AUTOATIVIDADE	21
TÓPICO 2 – INSTALAÇÃO E UTILIZAÇÃO DO PYTHON	23
1 INTRODUÇÃO	23
2 INSTALAÇÃO EM AMBIENTE WINDOWS	23
3 INSTALAÇÃO EM AMBIENTE LINUX	25
4 INSTALAÇÃO EM AMBIENTE MACOS.....	26
5 JUPYTER NOTEBOOK – COLAB UMA MANEIRA DE CODIFICAR EM..... PYTHON SEM PREOCUPAÇÃO DE INSTALAÇÃO.....	27
6 HORA DA PRÁTICA.....	30
RESUMO DO TÓPICO 2.....	33
AUTOATIVIDADE	34
TÓPICO 3 – OS PRIMEIROS PASSOS COM PYTHON	35
1 INTRODUÇÃO	35
2 SINTAXE PYTHON.....	35
2.1 OPERADORES NUMÉRICOS	35
3 TIPOS DE DADOS.....	39
4 COMENTÁRIOS.....	40
5 VARIÁVEIS NO PYTHON	41
6 OPERADORES DE COMPARAÇÃO	44
7 OPERADORES LÓGICOS	46
7.1 RESOLVENDO PROBLEMAS COM PYTHON	47
LEITURA COMPLEMENTAR.....	50
RESUMO DO TÓPICO 3.....	56
AUTOATIVIDADE	57
UNIDADE 2 – CONHECENDO AS ESTRUTURAS DE DADOS DO PYTHON	59
TÓPICO 1 – LISTAS	61
1 INTRODUÇÃO	61
2 LISTAS	62

2.1 TIPOS DE DADOS DAS LISTAS	64
2.2 MANIPULAÇÃO DE LISTAS.....	67
2.3 PERCORRENDO UMA LISTA	69
3 LENDO ARQUIVO DE TEXTO EM PYTHON.....	71
4 RESOLVENDO PROBLEMAS COM LISTAS	74
RESUMO DO TÓPICO 1.....	81
AUTOATIVIDADE	82
 TÓPICO 2 – NUMPY.....	83
1 INTRODUÇÃO.....	83
2 NUMPY A BIBLIOTECA MAIS FAMOSA.....	83
3 INSTALANDO E UTILIZANDO NUMPY	85
4 NUMPY ARRAYS.....	85
5 ARRAYS MULTIDIMENSIONAIS	86
6 OPERAÇÕES COM ARRAYS MULTIDIMENSIONAIS.....	89
7 MANIPULANDO ALGEBRICAMENTE UM ARRAY	95
8 NUMPY NA PRÁTICA	99
RESUMO DO TÓPICO 2.....	101
AUTOATIVIDADE	102
 TÓPICO 3 – PANDAS.....	103
1 INTRODUÇÃO.....	103
2 CONHECENDO O PANDAS	103
3 PANDAS SÉRIES.....	105
4 PANDAS DATAFRAME.....	109
5 ACESSANDO E MANIPULANDO DADOS COM DATAFRAMES	110
LEITURA COMPLEMENTAR.....	114
RESUMO DO TÓPICO 3.....	118
AUTOATIVIDADE	119
 UNIDADE 3 – CONCEITOS AVANÇADOS DE PROGRAMAÇÃO EM PYTHON	121
 TÓPICO 1 – TÉCNICAS DE PROGRAMAÇÃO EM PYTHON.....	123
1 INTRODUÇÃO	123
2 FUNÇÕES EM PYTHON	123
3 ORIENTAÇÃO A OBJETOS EM PYTHON	127
RESUMO DO TÓPICO 1.....	139
AUTOATIVIDADE	140
 TÓPICO 2 – CONEXÃO COM BANCO DE DADOS	141
1 INTRODUÇÃO	141
2 UTILIZANDO POSTGRESQL COM PYTHON.....	142
RESUMO DO TÓPICO 2.....	151
AUTOATIVIDADE	152
 TÓPICO 3 – BIBLIOTECAS COMPLEMENTARES	153
1 INTRODUÇÃO	153
2 MATH.....	153
3 MATPLOTLIB.....	155
4 BIBLIOTECAS PARA WEB.....	160

LEITURA COMPLEMENTAR.....	161
RESUMO DO TÓPICO 3.....	171
AUTOATIVIDADE	172
REFERÊNCIAS	173

UNIDADE 1

INTRODUÇÃO À PROGRAMAÇÃO PARA BIG DATA

OBJETIVOS DE APRENDIZAGEM

A partir do estudo desta unidade, você deverá ser capaz de:

- contextualizar sobre Big Data e o futuro da programação;
- conhecer as principais linguagens de programação para Big Data;
- ser apresentado ao Python como linguagem de programação;
- instalar e utilizar o Python;
- aprender conceitos básicos sobre programação em Python;
- criar seus primeiros programas em Python.

PLANO DE ESTUDOS

Esta unidade está dividida em três tópicos. No decorrer da unidade você encontrará autoatividades com o objetivo de reforçar o conteúdo apresentado.

TÓPICO 1 – BIG DATA: MOTIVAÇÃO PARA PROGRAMAÇÃO EM UM CENÁRIO DE GRANDES VOLUMES DE DADOS

TÓPICO 2 – INSTALAÇÃO E UTILIZAÇÃO DO PYTHON

TÓPICO 3 – OS PRIMEIROS PASSOS COM PYTHON



Preparado para ampliar seus conhecimentos? Respire e vamos em frente! Procure um ambiente que facilite a concentração, assim absorverá melhor as informações.

BIG DATA: MOTIVAÇÃO PARA PROGRAMAÇÃO EM UM CENÁRIO DE GRANDES VOLUMES DE DADOS

1 INTRODUÇÃO

Durante muitos anos os cursos de computação tiveram uma grade muito similar, primeiro aprendiam-se lógica proposicional, lógica de programação, estrutura de dados, programação orientada a objetos. Assim, se formam programadores de computadores.

Em áreas do estudo ligadas à otimização de código fonte, como estrutura de dados e análise de algoritmos, muitas vezes, foi se questionado porque tal preocupação com otimização de código sendo que os computadores atuais são capazes de executá-los de qualquer maneira. Tais técnicas foram desenvolvidas para que pudessem se economizar os poucos recursos dos computadores nas décadas passadas.

Uma simples repetição em um 286 com seus 8MB de memória RAM e processador de 16 MHz poderia travá-lo. O mesmo com certeza não aconteceria com o computador ASUS e seus 4.6GHz de processador e 128 GB de memória RAM.

FIGURA 1 - 286 VERSUS COMPUTADORES ATUAIS



FONTE: O autor.

Mas afinal, se atualmente temos computadores tão poderosos por que se preocupar com qualidade de código e em como eles irão executar? Resposta: os dados. Se até algumas décadas atrás a preocupação era ter memória suficiente, agora o jogo virou. Existem muitos dados e nem o supercomputador mais poderoso do mundo dará conta de processá-los.

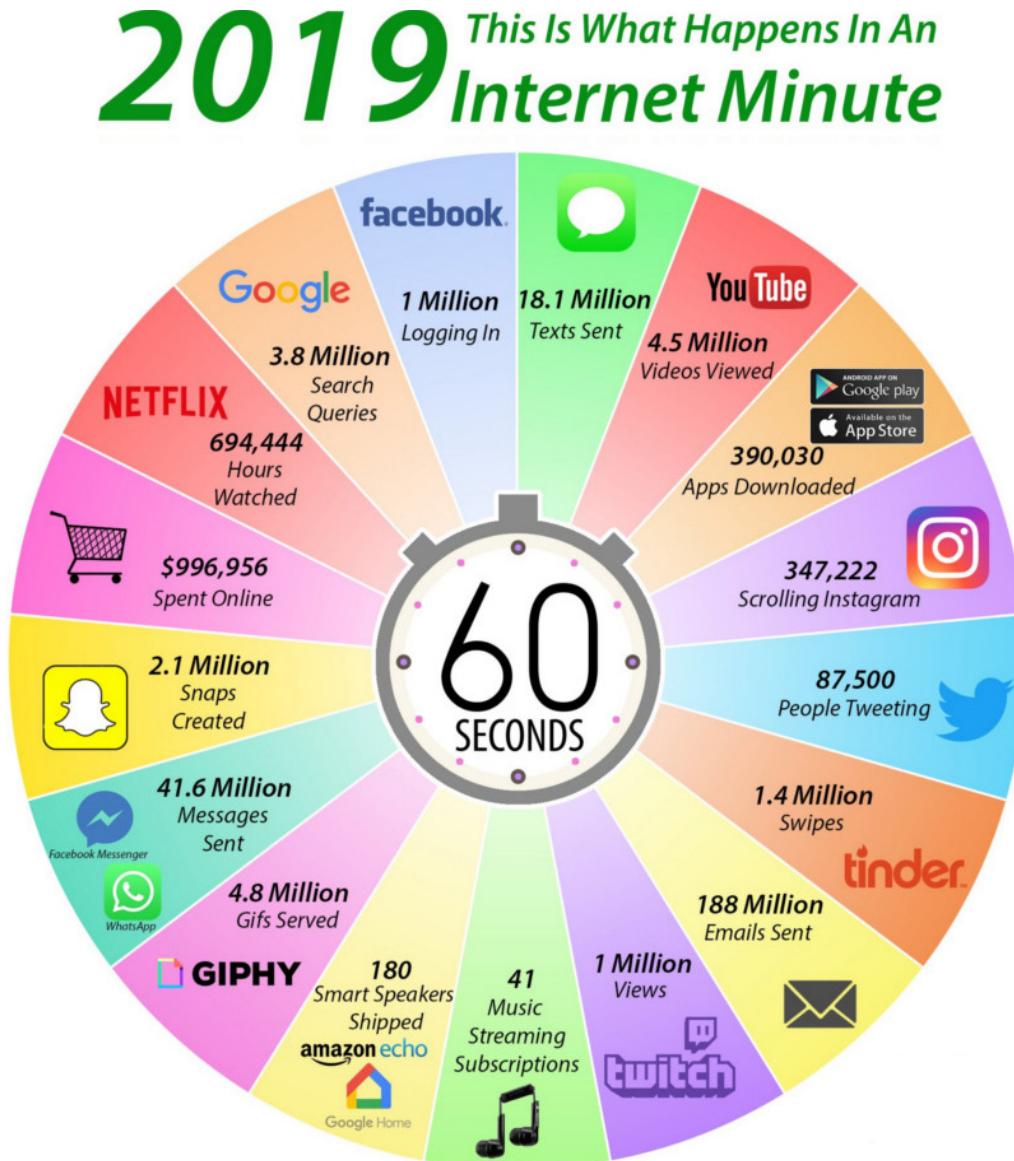
Para você ter uma noção de quanta informação o mundo está gerando, vamos a um valor simples: número de sites na Web. O *Internet Live Stats* é um projeto que mensura o número de sites na Web, ao acessá-lo em outubro de 2019 pode-se notar que o número se aproxima de dois bilhões de sites em todo o mundo.



Conheça o Internet Live Stats, acessando o endereço: <https://www.internetlivestats.com/>.

Em outra perspectiva atual, em um tradicional infográfico denominado O mundo em 60 segundos, mostrado pela Figura 2, é possível ver o volume de informações geradas a cada minuto, em todo mundo, nos principais veículos. Note que apenas buscas no Google são 3.8 milhões, no Youtube são 400 horas de vídeos enviados e cerca de 700 horas de vídeos assistidos, tudo isso desde que você começou a leitura desta seção.

FIGURA 2 - O MUNDO EM 60 SEGUNDOS



FONTE: <<http://bit.ly/3L7D4Q>>. Acesso em: 15 out. 2018.

O *Institute Data Corporation* (REINSEL; GANTZ; RYDNING, 2017) prevê que até 2025 a esfera de dados global aumentará para 163 zettabytes (ou seja, um trilhão de gigabytes). São dez vezes mais que os 16,1ZB de dados gerados em 2016. Todos esses dados desbloquearão experiências únicas do usuário e um novo mundo de oportunidades de negócios.

Complementando essas perspectivas sobre o volume de dados, seguem as perspectivas sobre novas tecnologias e demanda de profissionais para desenvolver tecnologias para Big Data e Inteligência Artificial.

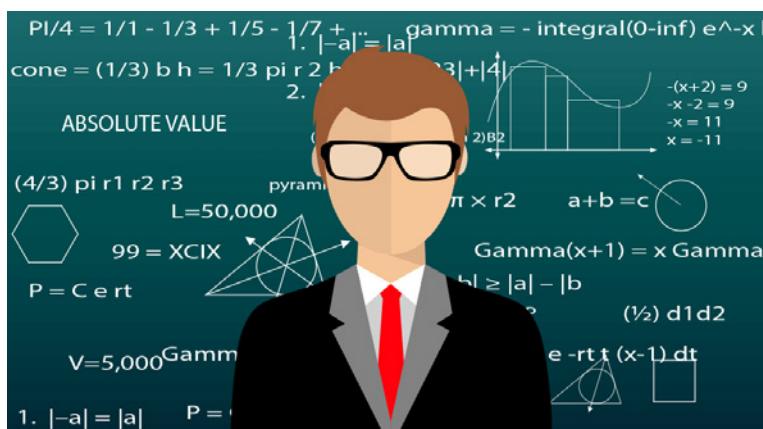
Nosso objetivo, a partir daqui, é trazer-lhe o conhecimento necessário para enfrentar o desafio do volume de dados em frente a um mercado de trabalho que está convergindo para uma mudança não somente de perfil.

Por isso, a partir de agora iremos trazer conceitos fundamentais que farão parte não somente para sua carreira de programador, mas também de cientista de dados.

2 O QUE É UM CIENTISTA DE DADOS?

Como aluno do curso de Big Data e Inteligência Analítica, você está sendo preparado para enfrentar os mais diversos desafios que as grandes massas de dados podem lhe proporcionar. Dentre suas mais diversas atribuições o cientista de dados é o profissional capacitado para reunir, interpretar e comunicar toda informação relevante contida em Gigabytes de dados que, diariamente, as empresas armazenam sobre os mais diversos segmentos.

FIGURA 3 - CIENTISTA DE DADOS



FONTE: <<http://bit.ly/36JBvOA>>. Acesso em: 4 dez. 2019.

Segundo a revista Exame (CIENTISTA, 2019, s.p.), o cientista de dados tem sido “um dos profissionais mais relevantes dos últimos anos permanece forte no mercado” e com uma forte tendência para o futuro.

O cientista de dados é capaz de analisar grandes volumes de dados gerados diariamente e transformá-los em informações relevantes. E isso é cada vez mais valorizado pelas empresas. Para se ter uma ideia, a média de salário mensal para o cargo no Brasil, segundo o site *Love Mondays*, é de R\$ 9.000,00, podendo chegar a mais de R\$ 20.000,00 reais (CIENTISTA, 2019, s.p.).

Complementarmente, ainda há empresas que se referem aos mais diversos profissionais para atuarem com dados. Dentre estes profissionais, o engenheiro de dados tem, cada vez mais, tido demanda nas organizações.

Para a *Data Science Academy* (EQUIPE DSA, 2019, s.p.):

Um Engenheiro de Dados é o profissional dedicado ao desenvolvimento, construção, teste e manutenção de arquiteturas, como um sistema de processamento em grande escala. A principal diferença entre um Engenheiro de Dados e um Cientista de Dados é que o segundo é alguém que limpa, organiza e examina Big Data. O Engenheiro de Dados é responsável por criar o pipeline dos dados, desde a coleta, até a entrega para análise ou para alimentar um produto ou serviço baseado em análise preditiva já em produção (produto ou serviço que pode ter sido desenvolvido com a ajuda de um Engenheiro de Software).

FIGURA 4 - CIENTISTA DE DADOS VERSUS ENGENHEIRO DE DADOS

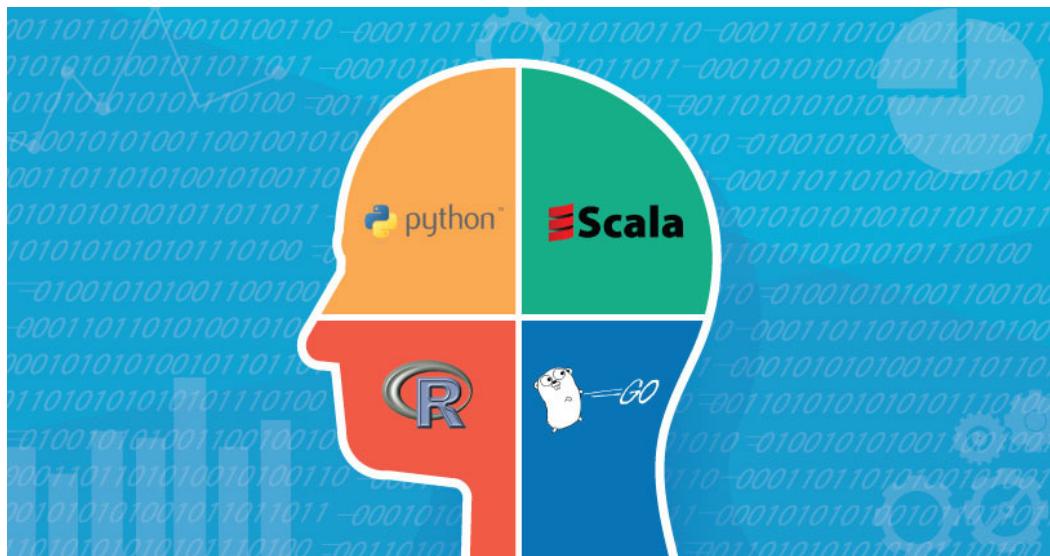


FONTE: <<http://bit.ly/35sn1kN>>. Acesso em: 20 nov. 2019.

3 PRINCIPAIS LINGUAGENS DE PROGRAMAÇÃO PARA BIG DATA

Este Livro Didático tem como ênfase preparar nosso leitor para um mercado específico: programação para Big Data. Ainda que no decorrer do livro iremos abordar uma linguagem específica é importante tomar conhecimento sobre todo farramental disponível, bem como saber da possibilidade de sua integração.

FIGURA 5 - PROGRAMAÇÃO PARA BIG DATA



FONTE: <<http://bit.ly/2N5KF0l>>. Acesso em: 29 nov. 2019.

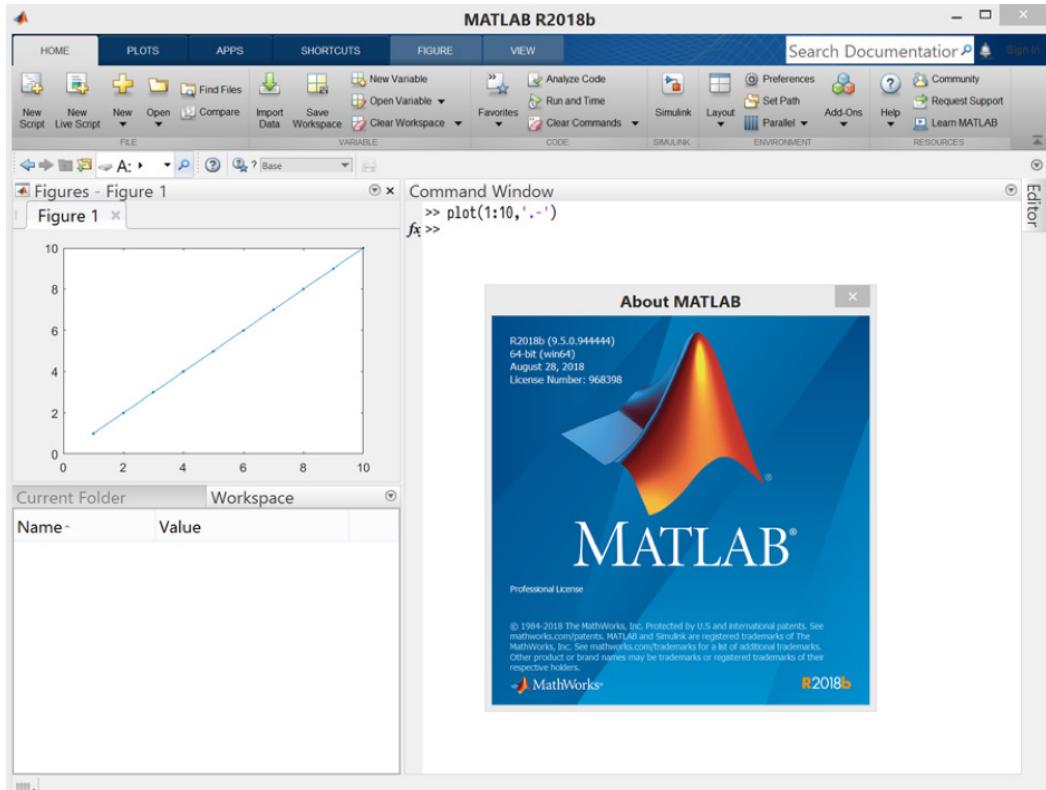
Assim, este tópico apresenta as principais linguagens e suas suítes para programação em Big Data.

3.1 MATLAB

Matlab é um poderoso software matemático utilizado pelas mais diversas áreas do conhecimento, desde matemáticos, passando por engenheiros que atuam com otimização.

O software Matlab é uma plataforma de programação projetada especificamente para engenheiros e cientistas. O coração do software Matlab é a linguagem Matlab, “uma linguagem baseada em matriz que permite a expressão mais natural da matemática computacional” (MATHWORKS, 2018 *apud* DONADEL, 2018, p. 39). Este tipo de programação matricial será bastante discutido ao decorrer deste livro, pois também faz parte das demais linguagens voltadas para Big Data.

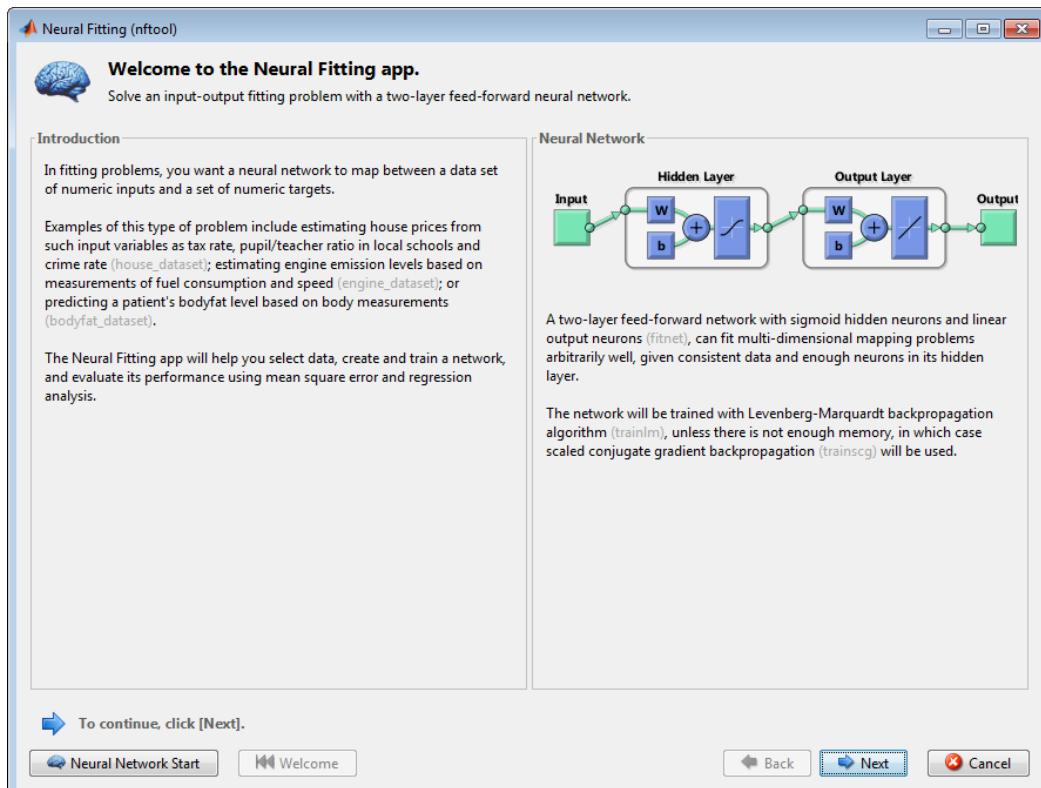
FIGURA 6 - TELA INICIAL DO MATLAB



FONTE: O autor

No que se refere à Big Data, o Matlab contém recursos poderosos, inclusive uma suíte completa para implementação de redes neurais artificiais. Tal suite permite apenas fornecer os dados e sem muito esforço obter um modelo computacional completo.

FIGURA 7 - REDES NEURAIS NO MATLAB



FONTE: O autor

Apesar de poderosos recursos, é dito que o Matlab é utilizado para prototipar métodos de Big Data. Isto significa que os métodos são desenvolvidos, testados, os parâmetros obtidos e após avaliados são implementados em outro ambiente. Isto acontece principalmente pois é uma suíte científica e não uma suíte de desenvolvimento de sistemas, ainda que possa ser integrada a um software, não tem esta finalidade.

Um exemplo de como a programação vetorial se comporta no Matlab é mostrado pelo quadro a seguir:

QUADRO 1 - EXEMPLO DE CÓDIGO MATLAB

```
>> a = [1 2 3]
```

```
a =
```

```
1 2 3
```

```
>> b = [2 2 2]
```

```
b =
```

```
2 2 2
```

```
>> a+b
```

```
ans =
```

```
3 4 5
```

FONTE: O autor



O Matlab pode ser obtido através do link: <https://www.mathworks.com/downloads/>. Vale ressaltar que o Matlab possui uma licença paga e você poderá utilizá-lo gratuitamente por 30 dias.

Uma opção para não se utilizar a IDE paga é a utilização do Octave. O Octave é uma IDE GNU que utiliza a linguagem Matlab. O Octave pode ser obtido no endereço: <https://www.gnu.org/software/octave/download.html>.

3.2 LINGUAGEM R

O R, além de ser uma linguagem de programação, é uma IDE completa para estatística. É uma linguagem popular entre os cientistas de dados e entre quem atua com Big Data, uma vez que grande parte dos métodos utilizados terem base estatística, complementarmente a isso, pode-se utilizar estatística para validar a exatidão dos métodos empregados.

FIGURA 8 - INTERFACE DO R EM SISTEMA OPERACIONAL LINUX

```

rodrigo@rodrigo-275E4E-275E5E: ~
Arquivo Editar Ver Pesquisar Terminal Ajuda
rodrigo@rodrigo-275E4E-275E5E:~$ R

R version 3.6.1 (2019-07-05) -- "Action of the Toes"
Copyright (C) 2019 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R é um software livre e vem sem GARANTIA ALGUMA.
Você pode redistribuí-lo sob certas circunstâncias.
Digite 'license()' ou 'licence()' para detalhes de distribuição.

R é um projeto colaborativo com muitos contribuidores.
Digite 'contributors()' para obter mais informações e
'citation()' para saber como citar o R ou pacotes do R em publicações.

Digite 'demo()' para demonstrações, 'help()' para o sistema on-line de ajuda,
ou 'help.start()' para abrir o sistema de ajuda em HTML no seu navegador.
Digite 'q()' para sair do R.

> []

```

FONTE: O autor

O R é uma linguagem também muito apreciada por matemáticos e estatísticos, já que possui suporte para cálculos e análises complexas. E, claro, porque ela também foi criada por estatísticos.

Segundo a IMASTERS (VICTÓRIA, 2019, s.p.), “se trata de uma linguagem para modelagem linear e não linear, análises temporais, agrupamento etc. E há, algo muito importante: se você quiser trabalhar com R, você vai precisar de um computador com memória RAM suficiente”.

Em relação à programação vetorial, o R segue como o Matlab, embora a sintaxe seja diferente. O quadro a seguir mostra o mesmo exemplo de manipulação matricial.

QUADRO 2 - EXEMPLO DE CÓDIGO R

```

>> a <- c(1,2,3)
>> b <- c(2,2,2)
>> a+b
 3 4 5

```

FONTE: O autor

No que se refere aos pontos contra, um dos maiores ‘poréns’ de seu uso, é a maior curva no aprendizado, uma vez que grande parte dos métodos e funções são baseados em estatística é necessária uma base nessa.

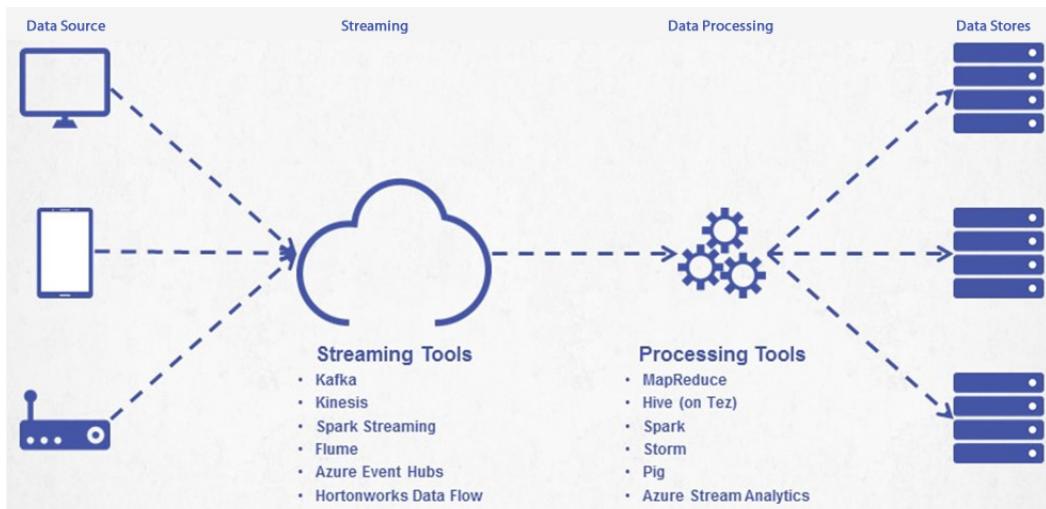


Para obter o R basta acessar ao link: <https://cran.r-project.org/bin/windows/base/>.

3.3 SCALA

Quando falamos de Big Data, ainda que o maior interesse das organizações seja obter conhecimento e previsões através destes dados. A linguagem Scala tem em sua função principal a etapa anterior, a coleta e armazenamento de dados em tempo real: streaming.

FIGURA 9 - BIG DATA STREAMING



FONTE: <<http://insight360.com/big-data/streaming/>>. Acesso em: 19 nov. 2019.

A combinação Scala e Apache Spark oferece a oportunidade de aproveitar ao máximo a computação distribuída em cluster de computadores. Portanto, a linguagem possui muitas ótimas bibliotecas para aprendizado de máquina e engenharia; no entanto, falta possibilidades de análise e visualização de dados em comparação com as linguagens anteriores. Se você não estiver trabalhando com Big Data, o Python e R podem mostrar um desempenho melhor que Scala. Mas se estiver trabalhando com Big Data Streaming, Scala pode ser a melhor opção (MATOS, 2019).



O Scala está disponível em: <https://www.scala-lang.org/download/>.

Você aprendeu que o Matlab é muito bom e tem muitos recursos para prototipar métodos de aprendizado de máquina, que o R será muito útil para realizar análises estatísticas e que o Scala é muito importante durante o processo de Hadoop streaming.

Além destas ainda existem diversas outras: Julia, SAS, GO, e a Python, a qual iremos aprofundar neste Livro Didático. Mas devido a tantas opções qual é a melhor linguagem de programação para Big Data? Com auxílio do estudo "*Comparative study of data mining tools*", podemos afirmar que **não há**.

Cada linguagem possui seus pontos positivos e pontos negativos, tais pontos podem ser contornados pelo emprego de outras tecnologias. Por isso, é importante ter em mente que é um conceito ferramental, é saber explorar o máximo de cada tecnologia e que cada linguagem pode ser adequada a um cenário.

FIGURA 10 - FERRAMENTAS DE PROGRAMAÇÃO



FONTE: <<http://bit.ly/2SYD8E6>>. Acesso em: 19 nov. 2019.

4 PYTHON – A LINGUAGEM QUE GANHOU O CORAÇÃO DO BIG DATA

Formado pela Universidade de Amsterdã, Holanda, Guido van Rossum foi o criador da linguagem Python. Foi no ano de 1991 que o matemático criou a linguagem de scripts para um sistema operacional distribuído denominado Amoeba.

FIGURA 11 - GUIDO VAN ROSSUM



FONTE: <<http://bit.ly/39GIESW>>. Acesso em: 19 nov. 2019.

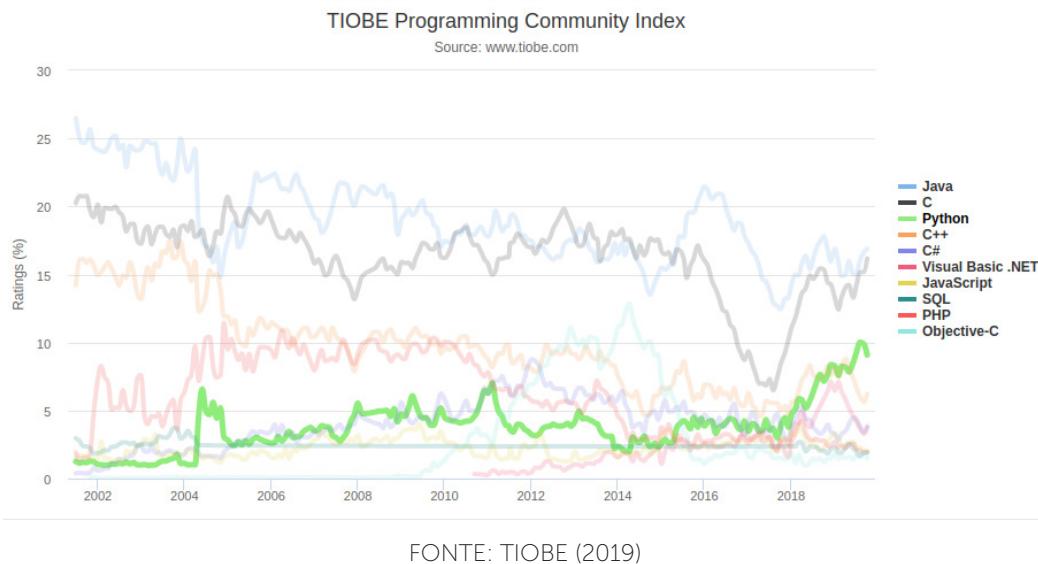
Percebi que o desenvolvimento de utilitários para administração de sistema em C (do Amoeba) estava tomando muito tempo. Além disso, fazê-los em shell Bourne não funcionaria por diversas razões. O motivo mais importante foi que, sendo um sistema distribuído de microkernel com um design novo e radical, as operações primitivas do Amoeba eram diferiam muito (além de serem mais refinadas) das operações primitivas disponíveis no shell Bourne. Portanto, havia necessidade de uma linguagem que "preencheria o vazio entre C e o shell". Por um tempo longo, esse foi o principal objetivo do Python. – Guido van Rossum (LENO, 2014, s.p.).

Nos últimos 10 anos, o Python tem se tornado uma ferramenta essencial na análise de dados. O Python pode ser definido como uma linguagem de programação interpretada, orientada a objetos e de alto nível com semântica dinâmica. Suas estruturas de dados incorporadas, de alto nível, combinadas com a digitação dinâmica e a vinculação dinâmica, o tornam muito atraente para o desenvolvimento de aplicações rápidas, bem como para ser usado como uma linguagem de script ou cola para conectar componentes existentes.

O Python é de uso geral da linguagem de programação, o que significa que pode ser usada no desenvolvimento de aplicativos da Web e de desktop. Também é útil no desenvolvimento de aplicações numéricas e científicas complexas. Com esse tipo de versatilidade, não é surpresa que o Python seja uma das linguagens de programação que mais crescem no mundo.

Segundo o índice TIOBE (2019), que avalia as linguagens de programação utilizadas no mundo, perde apenas para C e para Java. O índice também mostra que o Python é a linguagem que mais cresceu nos últimos anos, uma vez que em 1999 era a 20^a linguagem mais utilizada, passando pela 7^a posição em 2014 e hoje ocupa a 3^a posição. Outro ponto em destaque, que pode ser justificado pelos pontos que discutimos sobre a ascensão e perspectivas de Big Data, que é o pico da linguagem de 2018 para 2019.

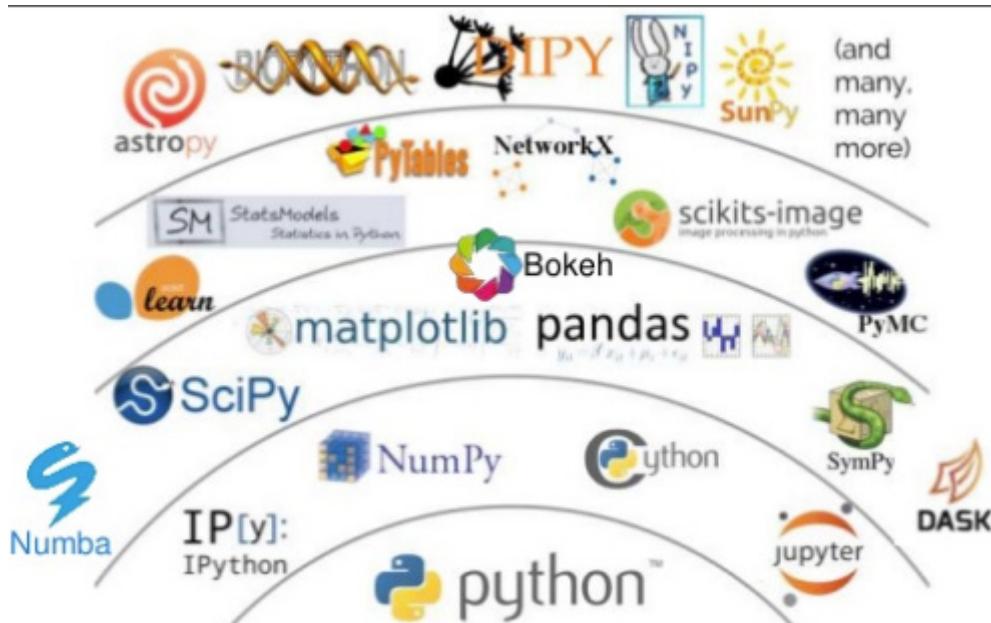
FIGURA 12 - GRÁFICO TIOBE LINGUAGENS DE PROGRAMAÇÃO



FONTE: TIOBE (2019)

Um destaque importante se dá ao crescente número de bibliotecas, que faz com que aumente o seu emprego nas mais diversas áreas do conhecimento. Dentro da área da computação há o uso em segurança da informação, redes de computadores, banco de dados, desenvolvimento de jogos, mas o que chama atenção nessa linguagem é o emprego por profissionais de outra área, como na biologia, por intermédio do projeto BioPython, que é uma associação internacional de desenvolvedores de ferramentas computacionais para biologia molecular em Python.

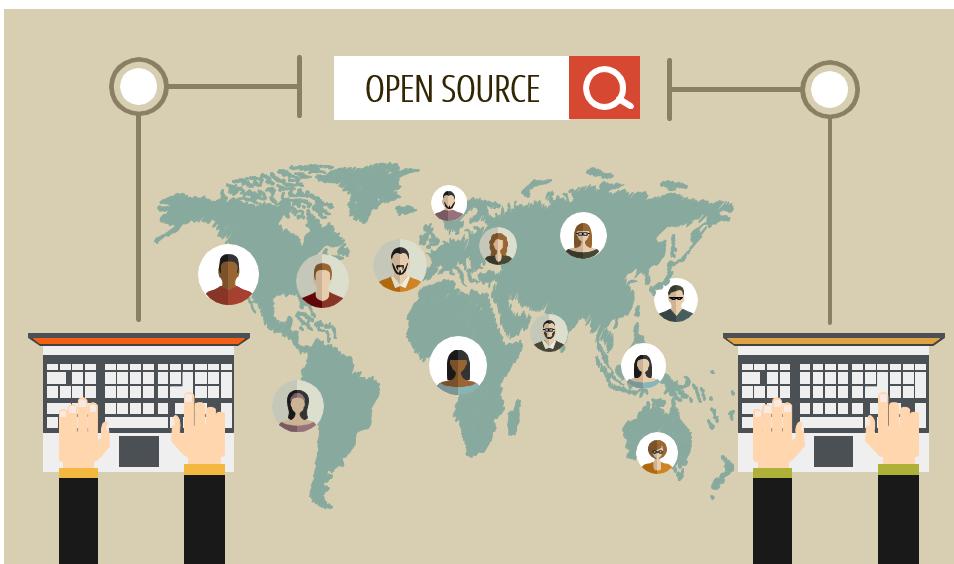
FIGURA 13 - UNIVERSO PYTHON

FONTE: <<http://bit.ly/35xCDDM>>. Acesso em: 19 nov. 2019.

Quando falamos especificamente de Big Data é notório um domínio de Python nesse segmento, para corroborar com essa informação faça uma busca em sites de emprego e verá que todas as vagas terão como requisito, senão principal, secundário, o conhecimento em linguagem Python.

Outro fator de impacto é que o Python é uma linguagem de programação de código aberto desenvolvida usando um modelo baseado na comunidade. Pode ser executado em ambientes Windows, Linux e MacOS.

FIGURA 14 - COMUNIDADES DE SOFTWARE



FONTE: <<http://bit.ly/37FEhEE>>. Acesso em: 20 nov. 2019.

O fator **comunidade** é o principal diferencial quando falamos da linguagem Python. Além das comunidades oficiais espalhadas por todo o mundo, contando com centenas de milhares de usuários pelo mundo, ainda existem grupos informais espalhados pelo mundo.

Somente a Python Brasil conta com uma grande rede de colaboradores em todo o país, atuando com comunidades locais, lista de discussão, organização de eventos, PyLadies (uma comunidade mundial com propósito de ajudar as mulheres a se tornarem participantes ativas e líderes na comunidade *open source* Python), entre as mais diversas atividades.

O fator comunidade forte faz com que, mesmo sendo uma linguagem de curva de aprendizagem curta, existam um grande número de usuários pelo mundo dispostos a auxiliar na resolução de problemas. É muito certo que alguma dúvida ou problema que você tenha durante o desenvolvimento, alguém já o tenha resolvido. Tanto para problemas resolvidos, quanto para buscar soluções na comunidade é comum o uso de sites como o *stack overflow*.



Stack Overflow: é um site americano gratuito de perguntas e respostas sobre desenvolvimento de software de grande força no exterior. No Brasil há uma crescente e uma versão em nosso idioma, no entanto a maioria das dúvidas ainda são postadas em idioma inglês.

Às perguntas atribuem-se tags, que juntamente com o título serão o meio pelo qual os experts na tecnologia em questão irão encontrar sua pergunta e respondê-la. Pode-se fazer comentários em perguntas e respostas pedindo mais detalhes, indicando se é o melhor jeito de fazê-lo, ou até questionar a necessidade de fazer o que a pergunta pede.

Pontos de reputação indicam sua dedicação ao website, e reputações maiores dão poder de tomar ações de maior responsabilidade como editar perguntas de outras pessoas, alterar tags, fechar ou deletar perguntas, entre outros. Esse sistema é interessante, pois usuários mais experientes guiam e alertam usuários novos, mantendo assim sempre a qualidade nas perguntas e respostas.

FONTE: <<https://glo.bo/3032Qc0>>. Acesso em: 20 nov. 2019.

Fazendo um apanhado geral sobre os pontos fortes do Python como linguagem de programação, podemos elencá-los em:

- Um grande número de bibliotecas;
- Suporte ao pré-processamento de dados;
- Um poderoso pacote de bibliotecas científicas;
- Compatível com frameworks de Big Data (Spark e Hadoop);
- Permite a escalabilidade das aplicações;
- Pequena curva de aprendizagem;
- Grande comunidade e bastante conteúdo on-line;
- Demanda de mão de obra no mercado de trabalho;
- Muito utilizado no meio acadêmico;
- Utilizado por grandes organizações;

FIGURA 15 - TOP ORGANIZAÇÕES QUE UTILIZAM PYTHON



FONTE: <<http://bit.ly/36Bt5sP>>. Acesso em: 20 nov. 2019.

Até aqui falamos bastante sobre o Python e temos certeza que agora você sabe por que trataremos desta poderosa linguagem de programação. Vamos, agora, começar a colocar a mão no código!

RESUMO DO TÓPICO 1

Neste tópico, você aprendeu que:

- Big Data trata de grandes volumes de dados.
- Os conceitos de programação para Big Data são diferentes dos tradicionais.
- Existem as principais Linguagens de Programação para Big Data.
- O Python é linguagem de programação muito utilizada em Big Data.

AUTOATIVIDADE



- 1 Estrutura de dados é o ramo da computação que estuda os diversos mecanismos de organização de dados para atender aos diferentes requisitos de processamento. Durante muitos anos o estudo de estruturas de dados esteve em evidência. Com relação ao estudo de estruturas de dados e Big Data selecione a alternativa CORRETA:
 - a) () Com o aumento da potência dos computadores, não se torna necessário estudar tais estruturas.
 - b) () Com o aumento da potência dos computadores, as estruturas de dados se tornaram obsoletas.
 - c) () Com o aumento da potência dos computadores, o volume de dados também cresceu, se tornando importante o emprego das estruturas de dados para que se consiga processar os dados em tempo viável.
 - d) () Com o aumento da potência dos computadores, o volume de dados também cresceu, se tornando importante o emprego destas estruturas para economizar banda larga.
- 2 Big Data é o termo em Tecnologia da Informação que trata de grandes conjuntos de dados que precisam ser processados e armazenados. O conceito do Big Data se iniciou com 3 Vs: Velocidade, Volume e Variedade. Com relação ao estudo de programação vetorial em Big Data selecione a alternativa CORRETA:
 - a) () A programação vetorial permite resolver apenas problemas matemáticos.
 - b) () A programação vetorial permite fazer operações on-line.
 - c) () A programação vetorial funciona apenas para problemas matemáticos.
 - d) () A programação vetorial permite com que grandes volumes de dados possam ser processados com pequenas linhas.
- 3 A linguagem de programação é um método padronizado para comunicar instruções para um computador. É um conjunto de regras sintáticas e semânticas usadas para definir um programa de computador. Sobre o estudo sobre linguagens de programação para Big Data selecione a alternativa CORRETA:
 - a) () Não se pode afirmar que existe uma melhor linguagem de programação, pois cada uma tem seus pontos fortes e fracos.
 - b) () O Python é a melhor linguagem de programação para Big Data.
 - c) () O R é a melhor linguagem de programação para Big Data.
 - d) () O Matlab é a melhor linguagem de programação para Big Data.
- 4 Em ciências da computação; programador, desenvolvedor, codificador ou engenheiro de software é alguém que escreve, desenvolve ou faz manutenção de software em um grande sistema ou alguém que desenvolve softwares para uso em computadores pessoais. O cientista de dados é uma nova profissão, que descende da de programador. Sobre os cientistas de dados selecione a alternativa CORRETA:

- a) () Cientista de dados são todos os programadores.
 - b) () Cientista de dados são todos os programadores Python.
 - c) () Cientista de dados é o profissional capaz de analisar grandes volumes de dados gerados diariamente e transformá-los em informações relevantes.
 - d) () Cientista de dados são todos os programadores Scala.
- 5 Python é uma linguagem de programação de alto nível, interpretada, de script, imperativa, orientada a objetos, funcional, de tipagem dinâmica e forte. Foi lançada por Guido van Rossum em 1991. Sobre as características do Python selecione a alternativa CORRETA:
- a) () Python não trabalha de maneira vetorial.
 - b) () Python foi escrito em Java.
 - c) () Python é linguagem de programação interpretada, orientada a objetos e de alto nível com semântica dinâmica.
 - d) () Python foi escrito para ser utilizado com Big Data em 1991.

INSTALAÇÃO E UTILIZAÇÃO DO PYTHON

1 INTRODUÇÃO

Aprendemos que o Python é uma poderosa linguagem de programação que pode ser executada em diversas plataformas. Existem diversas maneiras de se utilizar a linguagem de programação Python.

O Python é uma linguagem de programação que pode ser executada em diversos sistemas operacionais, bem como em diversos ambientes, na nuvem por exemplo.

Este tópico será dedicado às diversas maneiras de se obter, instalar e utilizar o Python.



Durante a confecção deste livro foi utilizado o Python versão 3.6.8.

2 INSTALAÇÃO EM AMBIENTE WINDOWS

Assim como grande parte dos recursos dos sistemas operacionais Microsoft, a instalação do Python é simples quando se tem o instalador em mãos.



O instalador pode ser obtido através da página oficial do Python em: <https://www.python.org/downloads/>. Recomendamos fazer o download de versões estáveis, ou seja, que não sejam beta.

Uma vez obtendo o instalador no arquivo executável, basta executá-lo. Na tela de instalação, uma opção importante é selecionar a opção “Add Python 3.X to PATH” ou “Adicionar Python 3.X ao PATH”. Esta opção irá adicionar as configurações do Python às variáveis de ambiente do Windows.

FIGURA 16 - TELA DE INSTALAÇÃO DO PYTHON EM AMBIENTE WINDOWS



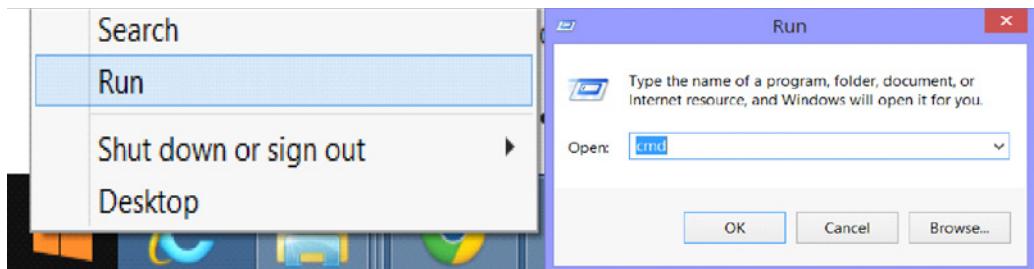
FONTE: O autor

Uma vez setando a configuração de variável de ambiente, basta seguir o roteiro pela própria instalação. Como já foi dito anteriormente, o Python conta com uma gama de pacotes e bibliotecas. Para realizar a instalação destes recursos será utilizado o PIP. O PIP é a principal maneira de a instalação de bibliotecas e pacotes no Python. Este termo é um acrônimo de Índice de Pacotes Python. Essa é uma forma de acessar o repositório de código oficial através da linha de comando e somos capazes de instalar, atualizar e remover os pacotes desejados.

A grande questão é que o PIP não vem em conjunto com o instalador em ambientes Windows, havendo a necessidade de ser instalado e configurado manualmente.

Para instalar o PIP, primeiramente abra o prompt de comando no Windows. Preferencialmente execute como administrador.

FIGURA 17 - CONFIGURANDO PYTHON NO WINDOWS



FONTE: O autor

Uma vez estando com o prompt aberto digite na linha de comando o seguinte: Python -m ensurepip. Isto irá instalar o pip.

Caso a linha de comando retornar que os requisitos já foram satisfeitos, para atualizar o Pip execute: Python -m ensurepip –upgrade.

3 INSTALAÇÃO EM AMBIENTE LINUX

“O mundo Linux é repleto de opções, que agradam diversos tipos e níveis de usuários, e sempre existe aquele sistema que por algum motivo temos uma maior afinidade. Às vezes por uma paixão acabamos por ficar presos em uma bolha e não enxergar sua real popularidade” (AD, 2019, s.p.).

Como a distribuição Ubuntu tem sido uma das mais populares, e a porta de entrada para os novos usuários do linux, vamos utilizá-lo como referência. De todo modo, na página oficial do Python é possível obter os arquivos binários em: <https://www.Python.org/downloads/source/>.

As versões atuais do Ubuntu já estão vindo com uma versão do Python, no entanto, a grande maioria é o Python 2.7. Para validar qual é a versão a ser instalada digite o comando Python –version.

Por questões de estabilidade, bem como do emprego de bibliotecas atuais, iremos instalar a versão 3 do Python. Para isso, utilizando o apt-get execute: sudo apt-get install Python3.

FIGURA 18 - INSTALANDO PYTHON NO UBUNTU

```
Arquivo Editar Ver Pesquisar Terminal Ajuda
rodrigo@rodrigo-275E4E-275E5E:~$ sudo apt-get install python3
[sudo] senha para rodrigo:
Lendo listas de pacotes... Pronto
Construindo árvore de dependências
Lendo informação de estado... Pronto
python3 is already the newest version (3.6.7-1~18.04).
^C
rodrigo@rodrigo-275E4E-275E5E:~$ 
```

FONTE: O autor

FIGURA 19 - INSTALANDO PIP NO UBUNTU

```
Arquivo Editar Ver Pesquisar Terminal Ajuda
rodrigo@rodrigo-275E4E-275E5E:~$ sudo apt-get install python3-pip
[sudo] senha para rodrigo:
Lendo listas de pacotes... Pronto
Construindo árvore de dependências
Lendo informação de estado... Pronto
O seguinte pacote foi instalado automaticamente e já não é necessário:
  linux-modules-4.15.0-50-generic
Utilize 'sudo apt autoremove' para o remover.
The following additional packages will be installed:
  dh-python libpython3-dev libpython3.6-dev python3-dev python3-setuptools
  python3-wheel python3.6-dev
Pacotes sugeridos:
  python-setuptools-doc
Os NOVOS pacotes a seguir serão instalados:
  dh-python libpython3-dev libpython3.6-dev python3-dev python3-pip
  python3-setuptools python3-wheel python3.6-dev
0 pacotes atualizados, 8 pacotes novos instalados, 0 a serem removidos e 212 não
atualizados.
É preciso baixar 45,8 MB de arquivos.
Depois desta operação, 79,2 MB adicionais de espaço em disco serão usados.
Você quer continuar? [S/n] 
```

FONTE: O autor

4 INSTALAÇÃO EM AMBIENTE MACOS

Contando com quatro vezes menos usuários do que a Microsoft. A Apple afirma que o número de usuários de todos os computadores da companhia é de 100 milhões. Como uma linguagem poderosa, o Python também funciona bem nesse sistema operacional.



Verifique se já tem o Python instalado, se você usa macOS 10.2 ou superior, provavelmente já possui alguma versão do Python instalada por padrão. Para conferir, digite em um terminal: \$ which Python ou \$ which Python3.

Que deve retornar algo como /usr/bin/Python. Isso significa que o Python está instalado nesse endereço. Antes de fazer a instalação do Python, é preciso fazer a instalação do XCode, que pode ser baixado na App Store, do pacote para desenvolvimento em linha de comando no macOS, command line tools e dos gerenciadores de pacotes pip e homebrew.

Para instalar o command line tools, digite em um terminal: \$ xcode-select --install.

Para instalar o pip, digite em um terminal: \$ sudo easy_install pip.

Para atualizar o pip, digite em um terminal: \$ sudo pip install --upgrade pip.

Para instalar o Python 3, digite em um terminal: \$ brew install Python3.

Instalação do Python em MacOs.

FONTE: <https://Python.org.br/installacao-mac/>. Acesso em: 20 nov. 2019.

5 JUPYTER NOTEBOOK - COLAB UMA MANEIRA DE CODIFICAR EM PYTHON SEM PREOCUPAÇÃO DE INSTALAÇÃO

O que teve início com o IPython, um interpretador interativo para várias linguagens de programação, mas especialmente focado em Python. O IPython oferece "type introspection", "rich media", syntax shell, completação por tab e edição auxiliada por histórico de comando, agora é conhecido como o poderoso editor on-line de nome Jupyter Notebook.

FIGURA 20 - TELA INICIAL DO JUPYTER NOTEBOOK

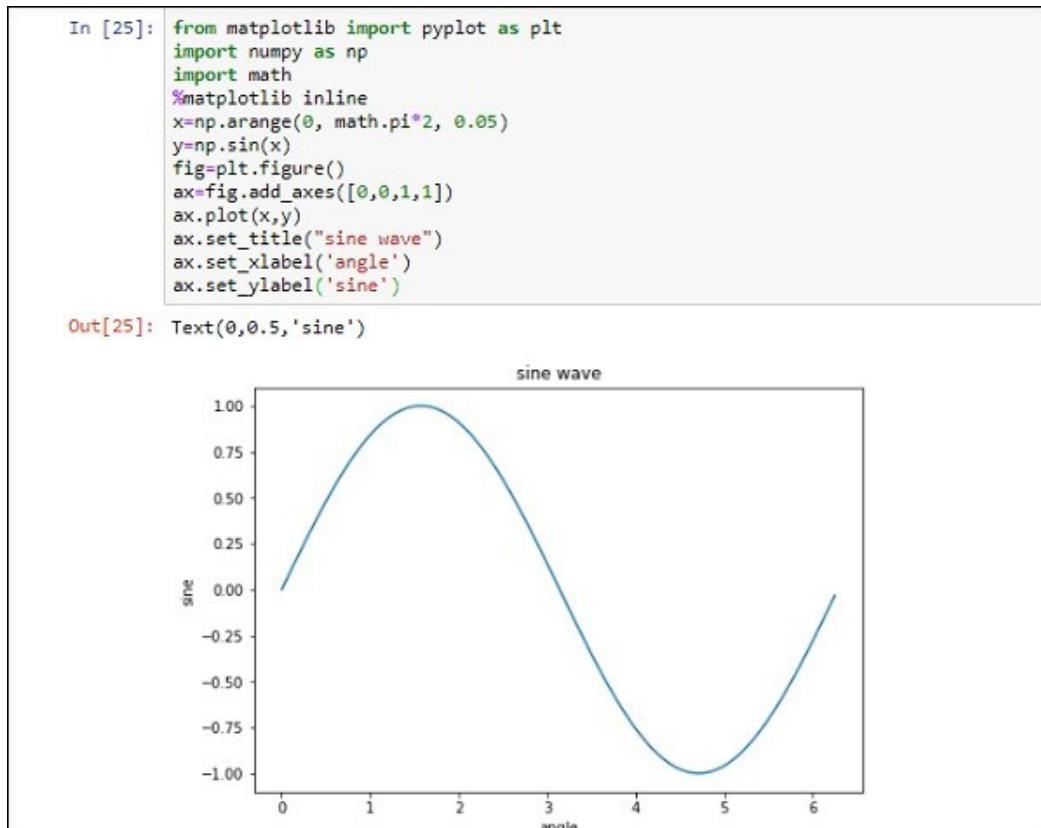


FONTE: O autor

Atualmente, os pesquisadores de todas as disciplinas acadêmicas precisam escrever código de computador para coletar e processar dados, realizar testes estatísticos, executar simulações ou desenhar figuras. As bibliotecas e ferramentas amplamente aplicáveis para isso são frequentemente desenvolvidas como projetos de código aberto (como NumPy, Julia ou FEniCS), mas o código específico que os pesquisadores escrevem para uma determinada peça de trabalho geralmente é deixado inédito, dificultando a reproduzibilidade (KLUYVER, 2016).

Os Jupyter Notebooks são documentos na forma de uma página Web que integram desenvolvimento, código e resultados. Deste modo, oferecem uma maneira de publicar os métodos computacionais que podem ser facilmente lidos e replicados.

FIGURA 21 - EXEMPLO DO JUPYTER NOTEBOOK



FONTE: O autor

Do ponto de vista do nosso livro, bem como do curso de Big Data e Inteligência Analítica, o Jupyter se destaca como uma ferramenta de apoio didático, pois permitirá aos alunos executarem códigos sem o esforço de instalação, bem como os docentes na explanação dos mesmos.

Para Cardoso, Leitão e Teixeira (2018), os processos de ensino e aprendizagem podem se beneficiar do uso de recursos on-line, permitindo a melhoria da produtivi-

dade de professores e alunos e dando-lhes flexibilidade e apoio ao trabalho colaborativo. Particularmente, nos cursos de engenharia, as ferramentas de código aberto, como o Jupyter Notebook, fornecem um ambiente de programação para o desenvolvimento e o compartilhamento de materiais educacionais, combinando diferentes tipos de recursos, como texto, imagens e código, em várias linguagens de programação em um único documento, acessível pela Web navegador. Esse ambiente também é adequado para fornecer acesso a experimentos on-line e explicar como usá-los.

Existem diversas maneiras de instalar e utilizar o Jupyter Notebook, no entanto, partiremos do princípio da simplicidade e colaboração do código, bem como tendo em vista que muitos dos leitores nunca tiveram contato com o ambiente.

Para a utilização do Jupyter Notebook vamos considerar o Google Colab. Segundo Machina Sapiens (SIGRID, 2019, s.p.), “o Google Colab ou ‘Colaboratório’ é um serviço de nuvem gratuito hospedado pelo Google para incentivar a pesquisa de Aprendizado de Máquina e Inteligência Artificial, em que muitas vezes a barreira para o aprendizado e o sucesso é a exigência de um tremendo poder computacional”. São elencados os seguintes benefícios do uso do Google Colab:

- Suporte para Python 2.7 e Python 3.6;
- Aceleração de GPU grátis;
- Bibliotecas pré-instaladas: todas as principais bibliotecas Python, como o TensorFlow, o Scikit-learn, o Matplotlib, entre muitas outras, estão pré-instaladas e prontas para serem importadas;
- Construído com base no Jupyter Notebook;
- Recurso de colaboração (funciona com uma equipe igual ao Google Docs): o Google Colab permite que os desenvolvedores usem e compartilhem o Jupyter Notebook entre si sem precisar baixar, instalar ou executar qualquer coisa que não seja um navegador;
- Suporta comandos bash;

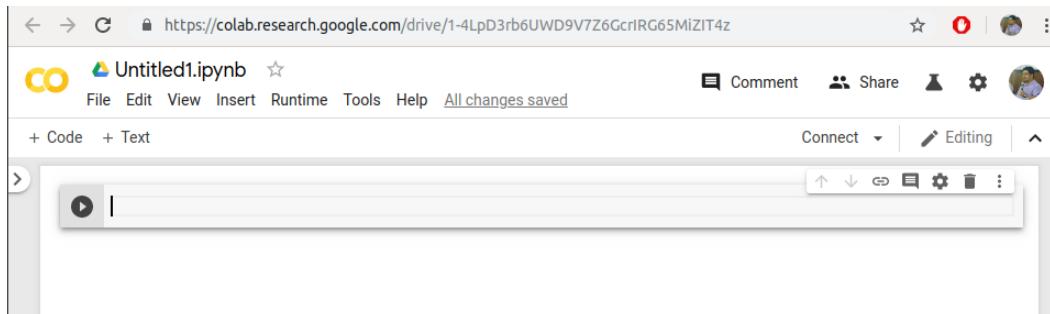
Para dar início ao seu primeiro Jupyter, basta ter uma conta Google e acessar: <https://colab.research.google.com/>.



Como o foco deste Livro Didático será na codificação em linguagem Python e não na ferramenta em si. Recomendamos a leitura da documentação e do *get started* do Jupyter Colab. Disponível em: <https://colab.research.google.com/notebooks/welcome.ipynb>.

Para criar seu primeiro documento Jupyter Notebook, utilizando o Colab, basta clicar em File >> New Python 3 Notebook. E terá um documento em branco conforme mostra a figura a seguir.

FIGURA 22 - CRIANDO UM JUPYTER NOTEBOOK



FONTE: O autor

6 HORA DA PRÁTICA

Seguindo um ou vários dos passos anteriores você estará apto a criar o seu primeiro código em Python. Há uma antiga história no mundo dos programadores que quando se aprende uma nova linguagem deve-se dizer “Olá Mundo” e com ela, o tradicional “Hello World”.

FIGURA 23- MEME COMPARANDO LINGUAGENS

C++ “Hello World”

```
#include <iostream.h>
main ( )
{
    cout << "Hello World! " ;
}
return ()
```

Java “Hello World”

```
class HelloWorldApp
{
    public static void main (String [] args)
    {
        System.out.println ("Hello World!");
    }
}
```

Python

```
print "Hello world"
```



FONTE: <<http://bit.ly/2R0t7Uf>>. Acesso em: 19 nov. 2019.

Uma vez que você já deve ter acesso ao Jupyter pelo Colab, vamos fazer nosso primeiro exemplo por meio dele. Para isso basta digitar print ("Hello World!") e em seguida apertar o play ou o atalho (SHIFT+ENTER).

FIGURA 24 - MEME COMPARANDO LINGUAGENS (2)

The screenshot shows a browser window for Google Colab. The URL is https://colab.research.google.com/drive/1-4LpD3rb6UWD9V7Z... The notebook title is 'exemplo1.ipynb'. The code cell contains the Python command 'print("Hello World!")'. The output cell shows the result 'Hello World!'. The interface includes standard Colab controls like 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a toolbar with icons for code, text, comment, share, and settings.

FONTE: O autor

FIGURA 25 - HELLO WOLRD NO TERMINAL

The screenshot shows a terminal window with a dark background. The menu bar includes 'Arquivo', 'Editar', 'Ver', 'Pesquisar', 'Terminal', and 'Ajuda'. The terminal prompt is 'rodrigo@rodrigo-275E4E-275E5E:~/Documentos/cursobigdata\$'. The user runs the command 'gedit' followed by 'python3 exemplo1.py'. The output is 'Hello World!'. The terminal has a standard dark-themed look with white text.

FONTE: O autor



Dependendo do instalador você pode ter instalado uma versão diferente do interpretador utilizado, python 2 por exemplo. A principal alternativa é que o comando seja somente print "Hello World!", mas é de muita importância que você execute esse exemplo e verifique qual versão do Python está sendo utilizada. Não deixe de seguir sem corrigir essa etapa.

Em uma terceira opção, que também é muito útil para quem está aprendendo Python, é executar o interpretador direto pelo terminal. Para isso, primeiro basta abrir o Python no terminal pelo comando `python3`. A partir daí estará no interpretador, e quando desejar sair basta digitar `exit()`. A figura a seguir mostra o exemplo completo:

FIGURA 26 - EXECUTANDO SCRIPTS PELO PROMPT DE COMANDOS

```
rodrigo@rodrigo-275E4E-275E5E:~/Documentos/cursobigdata$ python3
Python 3.6.8 (default, Oct  7 2019, 12:59:55)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello World!")
Hello World!
>>> exit()
rodrigo@rodrigo-275E4E-275E5E:~/Documentos/cursobigdata$
```

FONTE: O autor

RESUMO DO TÓPICO 2

Neste tópico, você aprendeu que:

- É possível instalar Python em Ambiente Windows.
- A instalação do Python em Ambiente Linux é feita por linha de comando.
- Existem comandos para fazer a Instalação do Python em Ambiente MacOs.
- Jupyter Notebook é uma ferramenta prática e didática para utilizar Python.
- É possível criar acesso ao Google Colab nossa importante ferramenta de suporte didático para esse curso.
- É simples criar seu primeiro Hello World utilizando Python.

AUTOATIVIDADE



- 1 Em muitos sistemas operacionais Linux é comum que nativamente exista uma versão pré-instalada. Selecione a alternativa CORRETA com comando para instalar o Python3 em sistema operacional Ubuntu.
 - a) () Pip install Python3.
 - b) () Ubuntu install Python3.
 - c) () Sudo apt-get install Python3.
 - d) () Python3.
- 2 Durante as instalações do Python nos sistemas operacionais você aprendeu que deve também instalar o PIP. Para cada sistema operacional existe um procedimento para instalação desta ferramenta. Selecione a alternativa CORRETA sobre o que é o PIP.
 - a) () O PIP é o servidor onde roda o Python.
 - b) () O PIP é o gerenciador de pacotes do Python.
 - c) () O PIP é o sistema operacional do Python.
 - d) () O PIP é uma biblioteca do Python.

OS PRIMEIROS PASSOS COM PYTHON

1 INTRODUÇÃO

Até o momento aprendemos que existem diversas linguagens para Big Data e que o Python tem sido a mais utilizada. Por isso realizamos exemplos de instalação em diversos ambientes.

A partir de agora iremos entrar em maiores detalhes sobre a linguagem de programação em si, como funciona sua sintaxe, manipulação de dados, entre outros. Para isso você deve ter o Python instalado em sua máquina ou utilizar o Google Colab.

Para dar início ao estudo vamos começar com a sintaxe do Python e as principais operações.

2 SINTAXE PYTHON

A sintaxe do Python é muito próxima do que conhecemos como linguagem natural e até mesmo de um algoritmo em pseudocódigo. Deve-se ter cuidado com a identação, pois a cada tabulação significa um bloco de código.

2.1 OPERADORES NUMÉRICOS

Em nosso primeiro contato com a programação Python vamos conhecer as operações matemáticas simples. A partir de agora todo conteúdo também será disponibilizado em um Jupyter Notebook.

- Soma: Operador +

QUADRO 3 - EXEMPLOS DE SOMA

```
>> 20+50  
70  
>>7.5+1.7  
9.2  
>>7.5+50  
57.5
```

FONTE: O autor

- Subtração: Operador -

QUADRO 4 - EXEMPLOS DE SOMA

```
>> 50-20  
70  
>>7.5-1.7  
9.2  
>>50-7.5  
57.5
```

FONTE: O autor

- Multiplicação: Operador *

QUADRO 5 - EXEMPLOS DE MULTIPLICAÇÃO

```
>> 10*5  
50  
>>10*0.5  
5  
>>1.1*8.7  
9.57
```

FONTE: O autor

- Divisão: Operador /

QUADRO 6 - EXEMPLOS DE DIVISÃO

```
>> 50/10  
5  
>>1500/50  
30  
>>1000/30  
3.333333333333
```

FONTE: O autor

- Divisão Inteira: Operador //

A divisão inteira é a divisão que considera apenas a parte inteira de uma divisão.

QUADRO 7 - EXEMPLOS DE DIVISÃO INTEIRA

```
>> 5//3
1
>> 7//3
2
>> 26//3
8
```

FONTE: O autor

- Resto da divisão: Operador %

Considerado o oposto de uma divisão inteira, o resto considera o que sobrou da mesma.

QUADRO 8 - EXEMPLOS DE DIVISÃO INTEIRA

```
>> 5//3
1
>> 7//3
2
>> 26//3
8
```

FONTE: O autor

- Potência: Base**expoente. $2^3 = 2**3$

QUADRO 9 - EXEMPLOS DE POTENCIACÃO

```
>> 2**2
4
>> 2**3
8
>> 2**4
16
```

FONTE: O autor

Raiz quadrada: a raiz quadrada não tem um operador específico, por isso existem duas maneiras de fazer o seu cálculo.

A primeira é utilizando o raciocínio matemático sendo que a potência da raiz é a raiz da potência. Para isso é necessário seguir os seguintes passos:

I- Passo 1: escrever o radicando.

II-Passo 2: elevar o radicando à unidade dividida pelo índice da raiz.

São exemplos de tais equivalências:

$$a^{\frac{1}{2}} = \sqrt[2]{a}$$

QUADRO 10 - EXEMPLOS DE RAIZ QUADRADA PELO MÉTODO MATEMÁTICO

```
>> 4**(1/2) #raiz quadrada de 4  
2  
>>8**(1/3) #raiz cúbica de 8  
2  
>>27**(1/3) #raiz cúbica de 27  
3
```

FONTE: O autor

Raiz quadrada: uma segunda opção é com a utilização do pacote matemático do Python através da importação da biblioteca math.

QUADRO 11 - EXEMPLOS DE RAIZ QUADRADA PELO MÉTODO MATEMÁTICO

```
>> 4**(1/2) #raiz quadrada de 4  
2  
>>8**(1/3) #raiz cúbica de 8  
2  
>>27**(1/3) #raiz cúbica de 27  
3
```

FONTE: O autor



Os exemplos utilizados neste subtópico sobre operadores numéricos estão disponíveis em: <http://bit.ly/2N8viEd>.

3 TIPOS DE DADOS

Tipos de dados são rótulos ou categorização de itens de dados. Estes tipos representam um valor que determina quais operações podem ser executadas nesses dados. Dados numéricos, não numéricos e booleanos (verdadeiro / falso) são os tipos de dados mais usados.

No entanto, cada linguagem de programação tem sua própria classificação, refletindo amplamente sua filosofia e paradigmas de programação. Os tipos de dados do Python são:

- Integer: números inteiros positivos ou negativos (sem uma parte fracionária).
 - Exemplo de variáveis: idade, número de casa.
 - Exemplo de valores: 25, 23, 1250, 52.
- Float: qualquer número real com uma representação de ponto flutuante na qual um componente fracionário é indicado por um símbolo decimal ou notação científica.
 - Exemplo de variáveis: peso, altura, medidas.
 - Exemplo de valores: 10.7 , 100.5.
- Número complexo: um número com um componente real e imaginário representado como $x + yi$. x e y são flutuadores e i é $\sqrt{-1}$ (a raiz quadrada de -1 é denominada número imaginário).
 - Exemplo de variáveis: x^2+1 , $x^3= -3$.
 - Exemplo de valores: $i^2=-1$, $i^3= 3$.
- Boolean: dados com um dos dois valores internos True ou False. Observe que 'T' e 'F' são maiúsculos, true e false não são booleanos válidos e o Python lançará um erro para eles.
 - Exemplo de variáveis: status, presença.
 - Exemplo de valores: Verdadeiro ou Falso (True or False).
- String: um valor de string é uma coleção de um ou mais caracteres colocados entre aspas simples, duplas ou triplas.
 - Exemplo de variáveis: nome, endereço, cidade.
 - Exemplo de valores: "João", "Rua Luiz Peinado", "Balneário Camboriú".
- Tipos Especiais de Dados:
 - Lista: um objeto de lista é uma coleção ordenada de um ou mais itens de dados, não necessariamente do mesmo tipo, entre colchetes.
 - Lista de Inteiro: [3, 8, 1, 6, 0, 8, 4].
 - Lista de Strings: ["João", "Jorge", "José"].

- Tupla: um objeto Tupla é uma coleção ordenada de um ou mais itens de dados, não necessariamente do mesmo tipo, entre parênteses.
 - Tupla de Inteiro: (3, 8, 1, 6, 0, 8, 4).
 - Tupla de Strings: ("João", "Jorge", "José").



Diferença entre listas e tuplas: é bem provável que você tenha analisado que as estruturas lista e tupla são similares. No entanto, a principal diferença entre elas não é o fator de uma utilizar chaves e outra parêntese. Mas sim o fato de que uma tupla é imutável e uma lista é mutável, ou seja, que os dados não são alterados. Durante o decorrer deste livro essa diferença ficará mais clara.

4 COMENTÁRIOS

Em programação de programadores, os comentários geralmente são úteis para alguém que mantém ou aprimora seu código quando você não está mais por perto para responder a perguntas sobre ele.

Um comentário é um trecho de um código fonte que não será lido pelo interpretador ou compilador. Ou seja, um trecho que não será executado. Em Python os comentários são feitos utilizando #, ou conforme o exemplo no quadro explicativo.

QUADRO 12 - EXEMPLOS DE COMENTÁRIO

```
"""
Exemplo de código utilizando três aspas

"""

x = 10 #Exemplo de comentário de uma linha
```

FONTE: O autor

5 VARIÁVEIS NO PYTHON

Agora que já aprendemos os tipos de dados em Python, vamos imergir em como utilizá-los, bem como em como melhor utilizar os operadores aritméticos que aprendemos.

Quando falamos de variável em programação, estamos falando de um espaço na memória do computador destinado a um dado que é alterado durante a execução de um programa.

Como o Python é uma linguagem orientada a objetos e tipagem dinâmica. Isto significa que cada variável será um objeto e não é necessário declarar qual é o tipo de dados.

O processo de atribuição e declaração de variável em Python é com a utilização do operador `=`. Ou seja, `x=10`, significa que a variável `x` será do tipo inteiro e recebe o valor 10. Vejamos mais alguns exemplos no quadro, com a explicação seguida de comentário.



A partir dessa etapa é muito importante que você realize os quadros práticos e seus próprios testes para compreender os códigos executados.

QUADRO 13 - EXEMPLOS DE COMENTÁRIO

```
"""

```

Exemplo de Atribuição de variável

```
"""

```

```
x = 10 #Inicializando variável do tipo inteiro com valor 10
x = 20 #Atribuindo o valor 20
x = 17.5 #Atribuindo o valor 17.5
print(x) #impressao do valor de X
```

FONTE: O autor

No exemplo acima qual é o valor de x? Se você já executou o programa, notou que o valor de x foi a última atribuição, ou seja, x é 17.5, podendo perceber a atribuição dinâmica de uma variável. Agora, considere o seguinte trecho de código fonte:

QUADRO 14 - EXEMPLOS DE VARIÁVEIS

```
nome    = "João Francisco"
sexo    = "M"
escalado = True
altura  = 1.79
peso    = 80.0
```

FONTE: O autor

No quadro anterior, podemos perceber que nome é do tipo de dados string, sexo é uma string de um caractere, escalado é um tipo de dado booleano e altura é um flutuante. Mas e o peso, é um valor inteiro ou um tipo float? Como responder a esta questão com o código?

O comando **type** é um comando de depuração que lhe permite descobrir qual é o tipo de dados no qual a variável está armazenando. Para isso vamos fazer teste utilizando para cada uma das variáveis anteriores e criando uma segunda variável peso, sem o ponto flutuante.

QUADRO 15 - TIPOS DE DADOS COM TYPE

```
print(type(nome))
print(type(sexo))
print(type(escalado))
print(type(altura))
print(type(peso))
peso2 = 80
print(type(peso2))
```

FONTE: O autor



Debug: o debug nada mais é do que um encontrador de erros que irão impedir o código de funcionar normalmente, com ele é possível saber exatamente o que está acontecendo dentro do código fonte, muitas vezes ele mesmo sugere ações para que você consiga solucioná-lo.

FIGURA - PROGRAMADOR DEBUGANDO



FONTE: <<http://bit.ly/39PYRUX>>. Acesso em: 20 nov. 2019.

Cada ferramenta de desenvolvimento tem sua própria forma de debugar o código fonte implementado e é graças a esse recurso que muito tempo é economizado. Não há necessidade de ler o código inteiro para identificar onde está o erro, o debug vai direto ao ponto. Outra função do debug é ajudar o programador a entender melhor o que acontece com o seu código em tempo de execução, muitos erros de lógica são identificados por causa disso.

FONTE: <<https://www.scriptcaseblog.com.br/scriptcase/o-que-e-debug/>>. Acesso em: 20 nov. 2019.

6 OPERADORES DE COMPARAÇÃO

Os operadores de comparação são aqueles que comparam dois valores. Operadores de comparação são usados para comparar valores. Ele retorna True ou False de acordo com a condição.

- Operador Maior: operador $a > b$.
- Significado: o valor a é maior a que o valor b?

QUADRO 16 - EXEMPLOS DO OPERADOR DE COMPARAÇÃO MAIOR

```
>> 20 > 50  
False  
>> 50 > 20  
True  
>> 20 > 20  
False
```

FONTE: O autor

- Operador Menor: operador $a < b$.
- Significado: o valor a é menor que o valor b?

QUADRO 17 - EXEMPLOS DO OPERADOR DE COMPARAÇÃO MENOR

```
>> 20 < 50  
True  
>> 50 < 20  
False  
>> 20 > 20  
False
```

FONTE: O autor

- Operador Igual: operador $a == b$.
- Significado: o valor a é igual ao valor b?

QUADRO 18 - EXEMPLOS DO OPERADOR DE COMPARAÇÃO IGUAL

```
>> 20 == 50  
False  
>> 50 == 20  
False  
>> 20 == 20  
True
```

FONTE: O autor

- Operador Maior Igual: operador $a \geq b$.
- Significado: o valor a é maior ou igual ao valor b?

QUADRO 19 - EXEMPLOS DO OPERADOR DE COMPARAÇÃO MAIOR IGUAL

```
>> 20 >= 50
False
>> 50 >= 20
True
>> 20 >= 20
True
```

FONTE: O autor

- Operador Menor Igual: operador $a \leq b$.
- Significado: o valor a é menor ou igual ao valor b?

QUADRO 20 - EXEMPLOS DO OPERADOR DE COMPARAÇÃO MENOR IGUAL

```
>> 20 >= 50
False
>> 50 >= 20
True
>> 20 >= 20
True
```

FONTE: O autor

Agora vamos combinar o conceito de operadores de comparação com a criação de variáveis.

QUADRO 21 - EXEMPLOS DE OPERADORES DE COMPARAÇÃO

```
x = 100
y = 120
# Saída: x > y is False
print('x > y is', x > y)
# Saída: x < y is True
print('x < y is', x < y)
# Saída: x == y is False
print('x == y is', x == y)
# Saída: x != y is True
print('x != y is', x != y)
# Saída: x >= y is False
print('x >= y is', x >= y)
# Saída: x <= y is True
print('x <= y is', x <= y)
```

FONTE: O autor

7 OPERADORES LÓGICOS

Operadores lógicos comparam valores booleanos, sejam armazenados em variáveis ou resultantes de operadores condicionais. Segundo Cechinel (2019), os operadores lógicos são usados para representar situações lógicas que não podem ser representadas por operadores aritméticos. Também são chamados conectivos lógicos por unirem duas expressões simples numa composta. Podem ser operadores binários, que operam em duas sentenças ou expressões, ou unário que opera numa sentença só.



Operadores Lógicos: o primeiro deles é o operador binário de conjunção

ou e lógico, representado por Δ ou AND. Quando duas expressões são unidas por este operador, a expressão resultante só é verdadeira se ambas expressões constituintes também são. Por exemplo "chove e venta" só é verdadeiro se as duas coisas forem verdadeiras, "chove" e "venta". Se uma das sentenças não ocorrer, a sentença como um todo é falsa.

O segundo operador é o operador binário de disjunção ou ou lógico, representado por \vee ou OR. Neste caso, se qualquer uma das expressões constituintes for verdadeira, a expressão completa também será. Por exemplo, "vou à praia ou vou ao campo" é uma sentença verdadeira caso qualquer uma das duas ações acontecer, ou ambas. É verdadeira, se eu for a praia e não ao campo, se eu for ao campo e não a praia e se eu for a ambos.

Para o caso em que se deve garantir que somente uma das sentenças aconteça, define-se

o operador ou-exclusivo, cujo símbolo é \oplus ou XOR. Como o nome diz, é semelhante ao operador ou com exclusividade na veracidade dos operandos, isto é, somente um dos operandos pode ser verdadeiro. No exemplo anterior, se o conectivo fosse o ou-exclusivo, a sentença composta só seria verdadeira se fosse à praia ou ao campo, mas não ambos.

O último dos operadores é o operador unário não lógico, representado por \neg . Sua função é simplesmente inverter valor lógico da expressão a qual se aplica.

FONTE: <http://www.cristiancechinel.pro.br/my_files/algorithms/bookhtml/node45.html>. Acesso em: 20 nov. 2019.

Em uma visão prática no Python os operadores lógicos são, conforme vistos anteriormente, escritos em linguagem natural, ou seja, os comandos são AND, OR e NOT.

Veja a seguir como funciona a combinação desses operadores:

QUADRO 22 - EXEMPLOS DE OPERADORES DE COMPARAÇÃO

```
x = True
y = False
# Saída: x and y is False
print('x and y is',x and y)
# Saída: x or y is True
print('x or y is',x or y)
# Saída: not x is False
print('not x is',not x)
```

FONTE: O autor

7.1 RESOLVENDO PROBLEMAS COM PYTHON

Antes de encerrarmos nosso capítulo é muito importante que seus conhecimentos básicos em programação estejam bem sólidos para darmos continuidade. Por isso iremos exercitar um pouco para que possamos dar continuidade em seus conhecimentos de programação.

Exemplo 1: escrever o resultado das seguintes operações utilizando Python:

- A. $(20 - 15)/2$
- B. $20 - 15/2$
- C. $2*5/20 + 30/15^2$
- D. $2*(5/20) + 30/(15^2)$
- E. $500*20/100$

QUADRO 23 - "SOLUÇÃO DO EXEMPLO 1"

```
print("Letra A:", (20 - 15)/2 )
print("Letra B:", 20 - 15/2 )
print("Letra C:", 2*5/20 + 30/15**2 )
print("Letra D:", 2*(5/20) + 30/(15**2) )
print("Letra E:", 500*20/100)
```

FONTE: O autor

Exemplo 2: receba os valores de RAIO e calcule a área de uma circunferência, considerando a fórmula $\text{ÁREA} = \pi * \text{RAIO}^2$

QUADRO 24 - "SOLUÇÃO DO EXEMPLO 1"

```
pi    = 3.1416 #você ainda tem a opção de utilizar o valor exato utilizando a
biblioteca math
raio = 5
area = pi * (raio**2)
```

FONTE: O autor

Exemplo 3: o diretor da escola Rui Filho está preocupado com o aquecimento global e as fortes altas e baixas durante o ano. Por isso instituiu que, ao medir a temperatura e umidade relativa, deve-se tomar algumas decisões. Crie um programa Python que considere as decisões abaixo:

- Temperatura maior do que 100: aulas canceladas.
- Temperatura menor do que -25: aulas canceladas, mas o mundo deve ter acabado.
- Temperatura menor do que 0: aulas canceladas.
- Temperatura maior ou igual a 92 e umidade maior do que 75: aulas canceladas.
- Temperatura maior que 88 e umidade maior ou igual do que 85: aulas canceladas.
- Temperatura igual a 75 e umidade menor ou igual do que 65: atividades externas.
- Outros casos: aula normal.

QUADRO 25 - "SOLUÇÃO DO EXEMPLO 3"

```
Temperature = int(input("Entre com a temperatura:"))
Humidity = int(input("Entre com a umidade relativa:"))

if Temperature >= 100:
    print("Aulas canceladas")
elif Temperature >= 92:
    if Humidity > 75:
        print("Aulas canceladas")
elif Temperature > 88:
    if Humidity >= 85:
        print("Aulas canceladas")
elif Temperature == 75:
    if Humidity <= 65:
        print("Atividades Externas")
elif Temperature <= -25:
    print ("Aulas canceladas, mas o mundo acabou")
elif Temperature < 0:
    print("Aulas canceladas")
else:
    print("Aula normal")
```

FONTE: O autor

Exemplo 4: crie um programa Python para imprimir a quantidade de números pares de 100 até 200, incluindo-os.

QUADRO 26 - "SOLUÇÃO DO EXEMPLO 4"

```
contador = 100
while contador <= 200:
    if contador % 2 == 0:
        print(contador)
    contador = contador + 1
```

FONTE: O autor

Exemplo 5: na matemática, o factorial de um número natural n , representado por $n!$, é o produto de todos os inteiros positivos menores ou iguais a n . A notação $n!$ foi introduzida por Christian Kramp em 1808. Crie um programa para calcular o factorial de um número ao recebê-lo.

QUADRO 27 - SOLUÇÃO DO EXEMPLO 5

```
numero = int(input("Desejo saber o Fatorial de: ") )

fatorial = 1
conta    = 1

while conta <= numero:
    fatorial *= conta
    conta += 1

print(fatorial)
```

FONTE: O autor



Os exemplos dos códigos utilizados neste subtópico estão disponíveis em:
<https://colab.research.google.com/drive/117MgMvW8Ybm3hjzzj9JrHkOEDZXsZkab>.

LEITURA COMPLEMENTAR

Blocos de instrução e estruturas de desvio condicional e de repetição

Francisco Cunha Neto

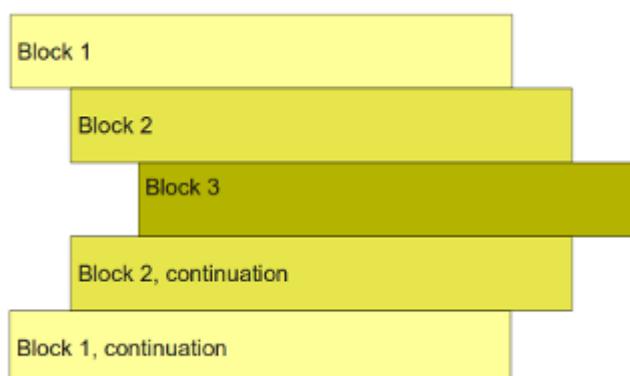
Em Python blocos de instrução são definidos pela identação, diferente da maioria das linguagens que usam as chaves {} para marcar os blocos de instrução. Assim Python requer uma identação padronizada, se em outras linguagens como C ou Java a identação é utilizada somente para melhor visualização, em Python ela define onde começa e onde termina um bloco de instruções. Esses blocos de instruções em Python são chamados de suítes.

O que são as PEP do Python: PEP significa Python Enhancement Proposal. Um PEP é um documento de design que fornece informações para a comunidade Python ou descreve um novo recurso para Python ou seus processos ou ambiente. O PEP deve fornecer uma especificação técnica concisa do recurso e uma justificativa para o recurso. <http://legacy.Python.org/dev/peps/pep-0001/#what-is-a-pep>. Acesso em: 20 nov. 2019

Aqui vamos chamar a atenção para a PEP 8 que é o guia de estilos para códigos em Python, a PEP 8 recomenda o uso de 4 espaços por nível de identação, atenção a PEP 8 é bem clara quanto à preferência pelo uso de espaços no lugar da tabulação. Nós recomendamos o uso dos estilos com base na PEP 8, mas, você pode usar o estilo que desejar desde que a identação tenha um tamanho padrão para separar os blocos de instrução e de preferência esse padrão seja seguido no código inteiro.

Resumindo sobre blocos de instrução

1. Os blocos de instrução são construídos por nível de identação.
2. A identação deve se manter constante por todo o código.
3. Para a identação é recomendado o uso de 4 espaços.



Estrutura Condisional IF

A estrutura de controle if analisa uma ou mais condição e conforme essa condição seja verdadeira ela realiza um bloco de instruções imediatamente após ela. Sua estrutura é mostrada a seguir:

```
if condições :  
    suite  
segue o programa
```

Vamos criar um programa chamado idade.py usando um if simples que verifica se uma pessoa pode ou não votar. A regra é que se a pessoa tiver média mais de 16 anos ela pode votar.

```
print('Programa para calcular se uma pessoa pode votar')  
print()  
  
nome = input('Entre com o nome da pessoa: ')  
print()  
  
a1 = int(input("Entre com o ano de nascimento: "))  
print()  
  
a2 = int(input("Entre com ano atual: "))  
print()  
  
idade = a2 - a1  
  
if idade > 16:  
    print(nome, 'pode votar')
```

Podemos ainda colocar um outro bloco de instruções para ser executado caso as condições seja falsa usando a cláusula else conforme a estrutura abaixo:

```
if condição :  
    suite  
else :  
    suite  
  
segue o programa
```

Vamos agora alterar o programa media para verificar se um aluno passou direto ou vai para prova final. A regra é se o aluno tiver nota igual ou maior que 7 ele já passou se não ele vai para a prova final:

```
print('Programa para verificar se o aluno passou direto')
print()
nome = input('Entre com o nome do aluno: ')
print()
nota1 = float(input("Entre com a primeira nota: "))
print()
nota2 = float(input("Entre com a segunda nota: "))
print()

media = (nota1 + nota2)/2
if media >= 7:
    print('{0} aprovado com media = {1:4.2f}'.format(nome, media))
else:
    print('{0} vai para a prova final com media = {1:4.2f}'.format(nome, media))
```

Além dessas estruturas podemos verificar outras condições entre if e else com elif na seguinte estrutura:

```
if condição:
    suite
elif condição:
    suite
elif condição:
    suite
else:
    suite
segue o programa
```

Vamos novamente modificar o programa media.py, agora as regras são se o aluno tiver nota igual ou maior que 7 ele está aprovado, se o aluno tiver nota menor que 7 e maior ou igual a 3 ele vai para a prova final se não ele já está reprovado:

```

print('Programa para verificar se o aluno passou direto')

print()
nome = input('Entre com o nome do aluno: ')

print()
nota1 = float(input("Entre com a primeira nota: "))

print()
nota2 = float(input("Entre com a segunda nota: "))

media = (nota1 + nota2)/2

print()
if media >= 7:
    print('{0} aprovado com media = {1:4.2f}'.format(nome, media))
elif media < 7 and media >= 3:
    print('{0} vai para prova final com media = {1:4.2f}'.format(nome, media))
else:
    print('{0} ficou reprovado com media = {1:4.2f}'.format(nome, media))
print()

```

ESTRUTURA DE REPETIÇÃO WHILE

A estrutura de controle while serve para repetir todo um bloco de instruções enquanto sua condição for verdadeira. A estrutura do while é:

```

while condição:
    suite
    segue o programa

```

```

numero = 0
contador = 1
while numero < 1 or numero > 9:
    print()
    numero = int(input('Entre com um número inteiro entre 1 e 9: '))

    if numero < 1 or numero > 9:
        print()
        print('O número deve ser número inteiro entre 1 e 9')
    print('\nTabuada de', numero)
    print()
    while contador <= 9:
        resultado = numero * contador
        print("{0} x {1} = {2}".format(numero, contador, resultado))
        contador += 1
    print()

```

Em programas feitos com while temos que prestar bastante atenção para que a condição de parada do loop seja satisfeita em algum momento, ou teremos um loop infinito o que pode causar problemas ao usuário pois o programa nunca termina.

OPERADOR FOR

Quem está acostumado com outras linguagens de programação como C pode estranhar um pouco o loop for do Python. Ele é basicamente feito para percorrer os tipos de dados de coleções (que veremos daqui a pouco) do Python. Para utilizar o for para realizar um loop indo do número zero a nove por exemplo devemos usar a função range(). Basicamente a forma da estrutura de repetição for é:

```
for [var] in [string, lista, tupla]:  
    suite  
    continua o programa
```

No exemplo abaixo usamos um for no ambiente iterativo para imprimir cada uma das letras da palavra Python

```
for c in 'Python':  
    print(c)
```

OS COMANDOS RANGE E BREAK

A forma básica da função range é: range(start, stop, step).

Essa função retorna uma série numérica no intervalo entre seus parâmetros start e stop usando o parâmetro step como passo, se start não é incluído ele é definido como 0 se step não é incluído ele é definido como 1 sendo o único parâmetro obrigatório o stop. A seguir iremos ver range() em ação junto com a estrutura de repetição for.

Quando a instrução break é encontrada dentro de um loop for ou while ela interrompe o loop e passa para as instruções imediatamente depois desses.

```
numero = 0
flag = True

# Aqui o usuário tem três tentativas para entrar com o número certo
for i in range(0, 3):
    print()
    numero = int(input('Entre com um número inteiro entre 1 e 9: '))

if numero < 1 or numero > 9:
    print()
    print('O número deve ser número inteiro entre 1 e 9')
    flag = False
else:
    flag = True
    break

if flag:
    print('\nTabuada de', numero)
    print()

    for contador in range(1, 10):
        resultado = numero * contador
        print("{0} x {1} = {2}".format(numero, contador, resultado))
else:
    print('\nVocê excedeu o número de tentativas para entrar com um número')
print()
```

FONTE: <<http://bit.ly/39NIOXL>>. Acesso em: 20 nov. 2019.

RESUMO DO TÓPICO 3

Neste tópico, você aprendeu que:

- O Python tem uma sintaxe particular e como ela funciona.
- O Python é uma linguagem dinâmica em relação aos tipos de dados
- Realizar operações no Python é simples operadores lógicos.
- O Google Colab é uma ferramenta de apoio para nosso livro.



Ficou alguma dúvida? Construímos uma trilha de aprendizagem pensando em facilitar sua compreensão. Acesse o QR Code, que levará ao AVA, e veja as novidades que preparamos para seu estudo.



AUTOATIVIDADE



- 1 Crie um programa em Python que leia o tempo de duração de um evento expressa em segundos e mostre-o expresso em horas, minutos e segundos.
- 2 Utilizando Python leia uma temperatura em graus Celsius e apresente-a convertida em graus Fahrenheit. A fórmula de conversão é: $F = (9 * C + 160) / 5$, na qual F é a temperatura em Fahrenheit e C é a temperatura em Celsius.
- 3 Crie um programa em Python que calcule a quantidade de litros de combustível gasta em uma viagem, utilizando um automóvel que faz 12Km por litro. Para obter o cálculo, o usuário deve fornecer o tempo gasto na viagem e a velocidade média durante ela. Desta forma, será possível obter a distância percorrida com a fórmula $DISTANCIA = TEMPO * VELOCIDADE$.
- 4 Receba dois números e indique qual deles é o maior. Você também precisa indicar se forem iguais.
- 5 O IMC – Índice de Massa Corporal é um critério da Organização Mundial de Saúde para dar uma indicação sobre a condição de peso de uma pessoa adulta. A fórmula é $IMC = peso / (altura)^2$. Elabore um programa em Python que leia o peso e a altura de um adulto e mostre sua condição de acordo com o que está a seguir:
 - 18,5 “abaixo do peso”.
 - Entre 18,5 e 25 “peso normal”.
 - Entre 25 e 30 “acima do peso”.
 - Acima de 30 “obeso”.

UNIDADE 2

CONHECENDO AS ESTRUTURAS DE DADOS DO PYTHON

OBJETIVOS DE APRENDIZAGEM

A partir do estudo desta unidade, você deverá ser capaz de:

- contextualizar o paradigma de programação vetorial;
- utilizar as principais bibliotecas para armazenamento de dados;
- criar listas e utilizá-las como suporte na programação;
- instalar e utilizar bibliotecas externas;
- utilizar o NumPy para manipulação de array;
- utilizar o Pandas para manipulação de dados.

PLANO DE ESTUDOS

Esta unidade está dividida em três tópicos. No final de cada um deles você encontrará autoatividades com o objetivo de reforçar o conteúdo apresentado.

TÓPICO 1 – LISTAS

TÓPICO 2 – NUMPY ARRAY

TÓPICO 3 – PANDAS – SÉRIES E DATAFRAMES



Preparado para ampliar seus conhecimentos? Respire e vamos em frente! Procure um ambiente que facilite a concentração, assim absorverá melhor as informações.

1 INTRODUÇÃO

Prezado aluno, até agora você aprendeu as estruturas básicas de programação em Python, teve uma visão geral sobre a linguagem e criou seus primeiros programas utilizando esta linguagem.

Um ponto importante que falamos, foi sobre um dos motivos de estarmos utilizarmos Python e que também é uma característica das linguagens de programação para Big Data: a programação vetorial.

A programação vetorial, também chamada de programação de arrays, programação de matrizes, entre outros nomes, se refere ao fato de que um conjunto de valores armazenados em um vetor no Python pode ser reconhecido e manipulado como um único objeto, sem a necessidade de grandes iterações.

Para compreender melhor vamos resolver o seguinte problema: Dado o vetor $X = [75\ 82\ 23\ 66\ 90\ 99\ 89\ 40\ 70\ 72\ 97\ 67\ 98\ 4\ 40\ 1\ 11\ 26\ 15\ 41]$, multiplique todos os elementos deste vetor por 10.

Utilizando apenas os conhecimentos em lógica de programação e das estruturas que vimos na unidade anterior a solução provável seria assim:

QUADRO 1 - EXEMPLO DE PERCORRER UM VETOR

```
import numpy as np
X = np.array([75,82,23,66,90,9,89, 40, 70, 72, 97, 67, 98, 4 ,40, 1, 11, 26, 15, 41])
for i in range(0,20):
    X[i]=X[i]*10
print(X)
```

FONTE: O autor

No exemplo anterior, se olharmos pela notação grande O, o desempenho do programa escrito será de $O(n)$, isto significa que no pior caso o programa irá percorrer todos os dados.

A grande questão é que nos programas escritos em Python a grande maioria das operações que são realizadas, manipulam todo um conjunto. E, quando falamos de conjunto não é o exemplo anterior com 20 exemplos, mas milhares de registros.

Deste modo, o Python trata esse tipo de iteração de maneira interna, com estruturas de dados específicas para isso. Veja o mesmo exemplo utilizando a lógica vetorial do Python.

QUADRO 2 - EXEMPLO DE PROGRAMAÇÃO VETORIAL

```
import numpy as np
X = np.array([75,82,23,66,90,9, 89, 40, 70, 72, 97, 67, 98, 4 ,40, 1, 11, 26, 15, 41])
X=X*10
print(X)
```

FONTE: O autor



O código fonte do conteúdo que foi desenvolvido neste tópico está disponível em: <https://colab.research.google.com/drive/1siWWmsChWjyO6kFaltMj4fxALWY05Erd>.

Note que a partir do exemplo anterior o código ficou mais enxuto, no entanto, o mais importante é que otimizamos a performance dele. Isto significa que com um conjunto grande de dados será visivelmente a diferença entre a abordagem vetorial e a programação tradicional.

A partir de agora, iremos aprender os principais tipos de dados do Python utilizados para manipular dados. Nosso estudo será iniciado com o conteúdo sobre listas

2 LISTAS

Chamamos de canivete suíço porque as listas são utilizadas em diversos contextos desta linguagem de programação. Principalmente no início dos estudos, é comum utilizá-lo para resolver problemas, dada a sua simples maneira de declaração e de manipulação dos dados.

O objeto de lista Python é a sequência mais geral fornecida pela linguagem. As listas são coleções ordenadas de objetos de tipo arbitrário e não possuem tamanho fixo (LUTZ, 2013). Ou seja, as listas também são mutáveis, as listas podem ser modificadas no local, atribuindo deslocamentos, bem como uma variedade de chamadas de método de lista.

Uma lista é uma estrutura de dados que contém uma coleção ordenada de itens, ou seja, você pode armazenar uma sequência de itens em uma lista.

Segundo Swaroop (2013, p. 82, tradução nossa), para imaginar como isso funciona:

[...] basta pensar em uma lista de compras na qual você tem itens para comprar, exceto que provavelmente você tem cada item em uma linha separada na sua lista de compras, enquanto no Python você coloca vírgulas entre eles.

A lista de itens deve estar entre colchetes, para que o Python entenda que você está especificando uma lista. Depois de criar uma lista, você pode adicionar, remover ou procurar itens na lista. Como podemos adicionar e remover itens, dizemos que uma lista é um tipo de dados mutável, ou seja, esse tipo pode ser alterado.

Uma vez que a lista de compras foi utilizada, vamos fazer uma lista de compras em Python. Lembrando que as listas sempre devem estar entre []. Por exemplo: minhalista = [elemento1, elemento2, elemento3]. Vamos ao exemplo:

QUADRO 3 - EXEMPLO DE LISTAS DE COMPRAS EM PYTHON

```
lista_de_compras = ['maçã', 'banana', 'couve', 'mexirica']
print ('Minha lista tem ', len(lista_de_compras), 'itens')
print ('O primeiro item a ser comprado é ', (lista_de_compras[0]))
print ('O segundo item a ser comprado é ', (lista_de_compras[1]))
print ('O terceiro item a ser comprado é ', (lista_de_compras[2]))
print ('O quarto item a ser comprado é ', (lista_de_compras[3]))
```

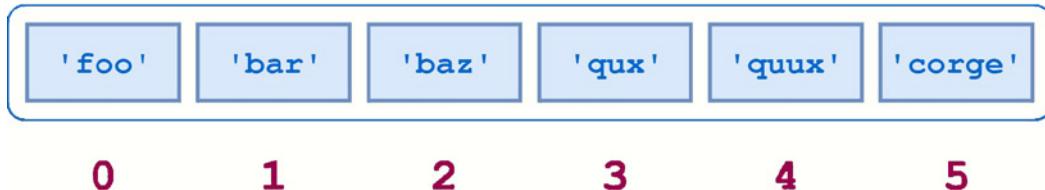
FONTE: O autor



O código fonte do conteúdo que foi desenvolvido neste tópico está disponível em: <https://colab.research.google.com/drive/1siWWmsChWjyO6kFaltMj4fxALWY05Erd>.

Ao desenvolver o primeiro exemplo de listas foram vistas duas coisas importantes, como saber o tamanho de uma lista, através do comando `len()`. Outro ponto importante é a indexação dentro do Python e das listas.

FIGURA 1 - EXEMPLO DE LISTAS DE COMPRAS EM PYTHON



FONTE: <<https://files.realPython.com/media/t.c11ea56e8ca2.png>>. Acesso em: 5 dez. 2019.

Uma vez que uma lista é uma variável que armazena diversos valores, os índices servem para permitir o acesso a estes elementos em específico. Dentro de uma lista o primeiro elemento inicia-se em 0 e o último elemento será o tamanho da lista -1. Note que no exemplo que fizemos havia 4 posições e o último elemento estava na posição 2.

2.1 TIPOS DE DADOS DAS LISTAS

Conforme estudamos na unidade anterior, o Python tem tipagem dinâmica dos dados, isto significa que o tipo de dados é mutável. No que se refere às listas, isso também acontece, de forma ainda mais dinâmica, podem existir listas de qualquer tipo de dados.

Além disso, as listas podem possuir múltiplos tipos de dados, a maneira mais simples de pensar é que cada posição dentro de uma lista representa uma variável e pode ter seu tipo específico.

Vamos ver um quadro que realize um exemplo com cada um dos tipos de dados.

QUADRO 4 - EXEMPLO DE LISTAS COM SEUS RESPECTIVOS TIPOS

```
#Exemplo de lista com inteiros
lista_inteiros = [20, 30, 40, 50 ,60]
print(lista_inteiros)

#Exemplo de lista com float
lista_float = [1.7, 9.9 , 25.7, 25.6]
print(lista_float)

#Exemplo de lista com booleanos
lista_booleanos = [True, False, True, True]
print(lista_booleanos)

#Exemplo de lista com chars
lista_char = ["b","i","g","d"]
print(lista_char)

#Exemplo de lista com strings
lista_strings = ["banco de dados","sistemas operacionais","programação"]
print(lista_strings)
```

FONTE: O autor

Em relação aos tipos e as listas, há o tipo caractere e o tipo string. Uma string, pode ser compreendida como uma lista de caracteres, ou seja, as letras que compõem um texto podem ser manipuladas pelo índice.

No exemplo a seguir, veremos que apesar do tipo de dados de uma lista ser string, é possível acessar os valores como se fosse uma lista.

QUADRO 5 - ACESSANDO ELEMENTOS DE UMA STRING COMO UMA LISTA

```
#Exemplo de string sendo manipulada como listas
texto = "curso de programação para big data"
print(type(texto))
print(texto[0])
print(texto[1])
print(texto[2])
print(texto[3])
print(texto[4])
```

FONTE: O autor

Algo interessante e que se deve ter muito cuidado com as listas é com o seu fator dinâmico, que é o fato de que uma lista pode armazenar dados de diversos tipos ao mesmo tempo. Por outro lado, esta é uma grande vantagem para manipular dados, dos mais diversos tipos. Confira o exemplo a seguir:

QUADRO 6 - LISTA DE VÁRIOS TIPOS DE DADOS

```
#Exemplo de lista com vários tipos de dados
lista = [30, True, 2.4, "João", [1,2,3,4] ]
print(type(lista))
#Exemplo de item inteiro
print(lista[0])
print(type(lista[0]))
#Exemplo de item Booleano
print(lista[1])
print(type(lista[1]))
#Exemplo de item float
print(lista[2])
print(type(lista[2]))
#Exemplo de item string
print(lista[3])
print(type(lista[3]))
#Exemplo de item lista
print(lista[4])
print(type(lista[0]))

print(lista[4][1])
print(type(lista[4][1]))
```

FONTE: O autor

Conforme é visto no exemplo acima, é possível armazenar em uma lista itens que contenham os mais diversos tipos de dados. Uma atenção especial ao último elemento desta lista também é do tipo lista.

Sempre que houver uma lista dentro de uma lista, é importante saber que a obtenção dos registros dentro da mesma será através dos índices. No exemplo anterior, lista[4][1]. O primeiro índice, [4], refere-se à lista principal que criamos inicialmente em lista = [30, True, 2.4, "João", [1,2,3,4]] e o segundo índice, refere-se ao segundo elemento dentro da lista interna [1,2,3,4].

A partir de agora será visto os principais comandos de manipulação de listas em Python, em consequência realizaremos maiores exemplos visando esclarecer o funcionamento dos índices em listas.

2.2 MANIPULAÇÃO DE LISTAS

Até o momento vimos apenas como criar uma lista com valores e obtê-los para exibir na tela. A partir de agora vamos ver os comandos particulares para a manipulação das listas.

- **Append**

Quando se fala em inserção se fala em aumentar o tamanho de uma lista, inserindo, ali, um novo valor. Novamente, fazendo uma analogia com uma lista de compras, é comum que ao se fazer uma lista de mercado cada novo item seja inserido ao final da lista.

Neste exemplo de uma lista de mercado, na qual os itens são adicionados um atrás do outro, utilizaremos o comando `append`. Vejamos um exemplo:

QUADRO 7 - UTILIZANDO APPEND PARA INSERIR

	<pre>lista_de_compras = [] lista_de_compras.append("egg") lista_de_compras.append("milk") lista_de_compras.append("apple") lista_de_compras.append("bullet") lista_de_compras.append("bread") print(lista_de_compras)</pre>
--	---

FONTE: O autor

- **Insert**

Em um outro contexto de inserção, no qual a ordem da inserção pode importar o comando `append` não será o ideal. Por exemplo, assumindo que dado a lista de nomes `["Ari", "Beto", "Fernando", "Rodrigo"]` inserir o nome “Cláudio” de tal maneira que a lista permaneça ordenada.

Para resolver esse problema o ideal é utilizar o comando `insert(posição, valor)`. Que permite inserir um determinado valor em uma posição específica, deslocando os demais itens da lista.

QUADRO 8 - INSERINDO ELEMENTO EM UMA POSIÇÃO ESPECÍFICA NA LISTA

```
lista_nomes = ["Ari", "Beto", "Fernando", "Rodrigo"]
lista_nomes.insert(2,"Cláudio")
print(lista_nomes)
```

FONTE: O autor

Note que no comando `insert` foram passados dois parâmetros 2 e “Cláudio”. Seguindo a sintaxe, o primeiro parâmetro é a posição na qual o valor será inserido e o segundo o valor que será inserido em determinada lista.

- **Extended**

Quando o caso for ler uma lista para popular outra lista, há a opção de percorrer item a item e utilizar o comando `append()`. No entanto, há um custo computacional, bem como de limpeza de código. Vejamos um exemplo de junção de listas.

QUADRO 9 - MESCLANDO DUAS LISTAS

```
lista_de_compras = ['maçã', 'banana', 'couve', 'mexerica']
lista_extra     = ["arroz, feijao"]
print(lista_de_compras)

lista_de_compras.extend(lista_extra)
print(lista_de_compras)
```

FONTE: O autor

- **Remoção**

A processo de remoção pode ser feito pelo nome do elemento ou pela posição do item. Quando for necessário utilizar o comando `lista.remove(valor-do-item)` e por índice a remoção é dada por `del lista[índice]`.

O comando `pop(posição)` funciona de maneira muito similar ao `del`, uma vez que para remover um item é necessário conhecer sua posição. Vamos fazer um exemplo utilizando a remoção por ambos os comandos.

QUADRO 10 - EXCLUINDO ELEMENTOS DE UMA LISTA

```

lista = ['maçã', 'banana', 'couve', 'maçã', 'mexerica', 'arroz','maçã']
print(lista)
#utilizando remove
lista.remove("maçã")
print(lista)
#utilizando pop
lista.pop(3)
print(lista)
#utilizando del
del lista[2]
print(lista)
#utilizando clear
lista.clear()
print(lista)

```

FONTE: O autor

- **Alteração**

A alteração de elementos dentro de uma lista seguirá conforme já fizemos anteriormente, acessando o índice da lista. Um elemento pode ser recuperado, depois atualizado, ou, simplesmente substituído. Confira no exemplo.

QUADRO 11 - ALTERAÇÃO DE ELEMENTOS DE UMA LISTA

```

lista = ['maçã', 'banana', 'couve', 'maçã', 'mexerica', 'arroz','maçã']
print(lista)

lista[0] = lista[0]+ " novo texto"
print(lista)

lista[4] = "novo valor"
print(lista)

```

FONTE: O autor

2.3 PERCORRENDO UMA LISTA

Ainda que diversas operações serão realizadas dentro de uma lista, a principal operação será a leitura de uma lista, afinal após a leitura todas as demais operações serão realizadas.

Neste subtópico serão vistas diversas maneiras de recuperar dados armazenados dentro de uma lista e é importante ressalvar que cada uma pode ser a melhor, dependendo do contexto da aplicação que está sendo desenvolvida. Muitas das estratégias vistas aqui, serão úteis na hora de percorrer outras estruturas.

A maneira mais simples de percorrer qualquer estrutura é através do comando for, por ele pode-se iterar por cada item dentro da lista.

Vamos a um exemplo básico de percorrer e escrever os elementos de uma lista.

QUADRO 12 - PERCORRER ELEMENTOS DE UMA LISTA

```
lista = ['maçã', 'banana', 'couve', 'maçã', 'mexerica', 'arroz','maçã']
for elemento in lista:
    print(elemento)
```

FONTE: O autor

Existem casos que se faz necessário obter a posição de determinado elemento durante a iteração. Para isso pode ser utilizado o seguinte código.

QUADRO 13 - PERCORRER ELEMENTOS DE UMA LISTA

```
lista = ['maçã', 'banana', 'couve', 'maçã', 'mexerica', 'arroz','maçã']
for indice, valor in enumerate(lista):
    print (indice, valor)
```

FONTE: O autor

Para os casos que se faz necessário inverter uma lista ou até mesmo de ordenar uma lista. Estes dois exemplos estão a seguir.

QUADRO 14 - PERCORRER ELEMENTOS DE UMA LISTA

```
lista = ['maçã', 'banana', 'couve', 'maçã', 'mexerica', 'arroz','maçã']
print("Lista Inversa")
for elemento in reversed(lista):
    print(elemento)
print("Lista Ordenada")
for elemento in sorted(lista):
    print(elemento)
```

FONTE: O autor

Com o que foi visto da estrutura será possível navegar pelas listas de diversas maneiras. No entanto, como vimos anteriormente, o Python possui estruturas internas que nos facilitam e otimizam a manipulação de dados, dentre estas: o fatiamento.

Para extrairmos um conjunto de elemento de uma lista, utilizamos inicialmente, a mesma notação utilizada para a obtenção de um único elemento, porém, ao invés de definirmos somente o elemento que desejamos, informaremos agora, o intervalo desejado. É importante observar que a notação de fatiamento é uma evolução da notação de obtenção de um único elemento, a diferença está somente na adição de alguns caracteres. Para retornarmos um elemento de uma lista, devemos passar entre colchete, qual o índice do elemento (FILHO, 2015, s.p.).

Vamos utilizar a mesma lista anterior e recuperar os valores dela utilizando o conceito de fatiamento.

QUADRO 15 - OBTER ELEMENTOS DE UMA LISTA POR FATIAMENTO

```
lista = ['maçã', 'banana', 'couve', 'mexerica', 'arroz', 'feijão']
print(lista[0:1])
print(lista[0:2])
print(lista[0:3])
print(lista[2:4])
print(lista[0:len(lista)])
```

FONTE: O autor

O importante é perceber que o início do fatiamento é inclusivo e o fim não inclusivo.

3 LENDO ARQUIVO DE TEXTO EM PYTHON

Na vida de quem trabalha com ciência de dados e Python é muito comum a familiaridade com sua principal fonte de trabalho: os dados. Quando trabalhamos com tipos de dados como listas, geralmente estamos manipulando dados externos.

São diversos tipos de arquivos e fontes utilizados, mas iremos começar utilizando algo mais primitivo, um arquivo de texto.

Para inicializar, crie um documento de texto, em qualquer editor, insira os dados a seguir e salve-o como texto.txt.

QUADRO 16 - EXEMPLO DE ARQUIVO DE TEXTO

```
lavar roupa
levar crianças na escola
pagar contas
ir à academia
ir à reunião de pais
```

FONTE: O autor

Note que durante o livro tivemos várias opções de utilização do Python e durante a codificação vem sendo utilizado o Colab Notebook para disponibilizar os códigos. Por isso veremos duas opções de leitura de arquivo.

Utilizando o Python em seu computador, é possível abrir o arquivo e exibir o conteúdo do arquivo de texto com o comando a seguir.

QUADRO 17 - EXEMPLO DE ARQUIVO DE TEXTO

```
arquivo = open("texto.txt", "r")
print(arquivo.read())
```

FONTE: O autor

O Colab Notebooks utiliza uma plataforma on-line, na qual você compartilha os documentos da sua conta Google com o seu documento Colab Notebook, inclusive os códigos do notebook ficam salvos na sua conta google. Para manipularmos o arquivo dentro do Colab Notebook, primeiramente, envie o arquivo criado para a pasta inicial do Drive <<https://drive.google.com/drive/my-drive>>. Depois disso, é necessário instalar a biblioteca Pydrive através do pip e importar as bibliotecas para acessar documentos.

QUADRO 18 - PREPARANDO AMBIENTE PARA IMPORTAR TEXTOS

```
!pip install -U -q PyDrive
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials
```

FONTE: O autor

Uma vez importadas as bibliotecas, será realizada a autorização de acesso do código aos arquivos.

QUADRO 19 - PREPARANDO AMBIENTE PARA IMPORTAR TEXTOS

```
auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)
```

FONTE: O autor

Executando esse código, você será redirecionado para a tela de autorização. Basta seguir o link de redirecionamento, copiar o código de autorização e colar no console do Colab Notebook.

FIGURA 2 - AUTORIZANDO ACESSO AO GOOGLE COLAB

```
➡ Go to this URL in a browser: https://accounts.google.com/o/oauth2  
Enter your authorization code:  
.....  
Mounted at /content/drive/
```

FONTE: O autor

Apesar de acessarmos uma plataforma web para executar o Colab, o backend é um sistema Linux. Segundo Silva (2019, s.p.):

Em sistemas GNU/Linux, ou Unix-like de uma maneira geral, quando se desejar acessar um sistema de arquivos seja uma partição, um disquete, um CD-ROM ou diferentes dispositivos tais como zip drive, pen-drives dentre outros, é necessário, inicialmente, montar o sistema de arquivos antes de começar a usá-lo. Atualmente, a maioria dos sistemas proveem mecanismos de automontagem, onde o sistema faz automaticamente a montagem para o usuário. Entretanto, algumas vezes temos que fazer esse processo manualmente, o que é feito utilizando-se o comando `mount`.

Deste modo é necessário realizar a montagem do seu drive para que o seu código possa reconhecer e ler os arquivos lá armazenados. Isso é feito por:

QUADRO 20 - EXEMPLO DE ARQUIVO DE TEXTO

```
from google.colab import drive  
drive.mount('/content/drive/')
```

FONTE: O autor

Uma vez realizado a autorização, o código para acessar um documento de texto armazenado na nuvem será igual ao código local. Veja o exemplo, note que o arquivo está na pasta raiz do drive.

QUADRO 21 - EXEMPLO DE ARQUIVO DE TEXTO

```
arquivo = open('/content/drive/My Drive/texto.txt', 'r')  
print(arquivo.read())
```

FONTE: O autor

Durante essa unidade ainda iremos trabalhar mais com manipulação de arquivos e integração com estruturas de dados do Python. A seguir veremos um conjunto de exemplos de como manipular arquivos com Python e na sequência será feito a integração com as estruturas estudadas na unidade.

QUADRO 22 - MANIPULANDO ARQUIVO DE TEXTO

```
arquivo = open('/content/drive/My Drive/texto.txt', 'r')
#retorna os 3 primeiros caracteres do arquivo
print(arquivo.read(3))

#realiza a leitura das linhas
print(arquivo.readline())

#criamos uma nova instância do arquivo com permissões de escrita
arquivo2 = open('/content/drive/My Drive/texto.txt', 'w')

#retorna os 3 primeiros caracteres do arquivo
arquivo2.write("Now the file has more content!")
arquivo2.close()

#criando um novo arquivo
arquivo3 = open("novotexto.txt", "x")

#removendo um arquivo
import os
os.remove("novotexto.txt")
```

FONTE: O autor

O objetivo deste subtópico foi ter uma visão geral da manipulação de arquivos, principalmente a configuração do Colab Notebook para fazer essas operações. Na próximo subtópico faremos exemplos práticos para integrar listas e manipulação de arquivos.

4 RESOLVENDO PROBLEMAS COM LISTAS

Desenvolvemos alguns exemplos sobre como manipular listas em diversas situações. Agora, iremos realizar exemplos completos utilizando situações cotidianas, seguido de algumas dicas de outros comandos em linguagem Python.



Como uma maneira de deixar a explicação mais intuitiva o passo a passo da resolução dos exercícios está na forma de `#comentário` dentro do código fonte. Fique atento, pois cada comentário descreve o que está sendo feito na próxima linha.

Exemplo 1: agora que aprendemos a abrir arquivos e criar listas. Crie um programa em Python que leia um arquivo entrada.txt, transforme-o em uma lista, adicione os itens “limpar o carro” e “levar o cachorro para passear” ao final da lista e salve-a no arquivo.

QUADRO 23 - RESOLUÇÃO DO EXEMPLO 1 UNIDADE 2

```
#abre o arquivo
arquivo = open('/content/drive/My Drive/texto.txt', 'r')
#cria lista vazia
lista = []

#percorre o arquivo e adiciona na lista
for line in arquivo:
    lista.append(line.strip())

#adiciona os items solicitados
lista.append("limpar o carro")
lista.append("levar o cachorro para passear")

#cria uma instancia de leitura do arquivo
arquivoescrita = open('/content/drive/My Drive/texto.txt', 'w')

#adiciona os items alterados no arquivo
for item in arquivo:
    arquivoescrita.writelines(item)
arquivoescrita.close()
```

FONTE: O autor

Exemplo 2: crie um programa que receba o nome, telefone de maneira dinâmica. Após receber estes dados o usuário deverá ter a opção de continuar cadastrando mais dados (s/n). Após cadastrar todos, os dados devem ser exibidos na tela e salvos em um arquivo de texto.

QUADRO 24 - RESOLUÇÃO DO EXEMPLO 2 UNIDADE 2

```
#variavel para armazenar os dados
cadastro = []

#variavel que recebe se os cadastros irão
continua = "s"

#loop enquanto o valor não for n cadastros serão cadastrados
while(continua!="n"):
    #a lista recebe o nome
    nome = str(input("Informe o seu nome: "))
    cadastro.append(nome)

    #a lista recebe a idade
    idade = str(input("Informe a idade: "))
    cadastro.append(idade)

    #a lista recebe o telefone
    telefone = str(input("Informe o telefone: "))
    cadastro.append(telefone)

    #a lista recebe o email
    email = str(input("Informe o e-mail: "))
    cadastro.append(email)

    #a variável continua é atualizada
    continua = str(input("Continua (s/n)?: "))

#abre um arquivo ou cria um novo
arquivoescrita = open('/content/drive/My Drive/salvadados.txt', 'w')
#adciona os items alterados no arquivo
for item in arquivo:
    arquivoescrita.writelines(item)
arquivoescrita.close()

print(cadastro)
```

FONTE: O autor

Exemplo 3: crie um programa que receba o nome de um determinado produto, seu respectivo valor. Deve haver uma condição de saída(s/n), ao final deve ser exibido o valor total vendido.

QUADRO 25 - RESOLUÇÃO DO EXEMPLO 3 UNIDADE 2

```

#variaveis para armazenar os dados
produtos = []
precos = []

#variavel que recebe se os cadastros irão
continua = "s"

#loop enquanto o valor não for n cadastros serão cadastrados
while(continua!="n"):
    #a lista recebe o nome do produto
    nome = str(input("Informe o nome do produto: "))
    cadastro.append(nome)

    #a lista recebe o valor
    valor = str(input("Informe o valor que foi vendido: "))
    precos.append(valor)

    #a variavel continua é atualizada
    continua = str(input("Continua (s/n)?: "))
    #inicializa soma com 0
    soma = 0
    #percorre cada item e soma ao valor inicial
    #observe que como ao utilizar input o valor é uma string o comando int() faz a
    #transformação da variável para integer
    for item in precos:
        soma += int(item)

    print("O total é:", soma)

```

FONTE: O autor

Exemplo 4: crie um programa que leia uma lista e retorne o maior elemento da mesma.

QUADRO 26 - RESOLUÇÃO DO EXEMPLO 4 UNIDADE 2

```

lista = [5, 4, 6, 7, 1, 3, 0]
maior = lista[0]
for a in lista:
    if a > maior:
        maior = a
print("O maior valor da lista é:",maior)

```

FONTE: O autor

Exemplo 5: crie um programa que percorra uma lista e remova todos os items duplicados na mesma.

QUADRO 27 - RESOLUÇÃO DO EXEMPLO 5 UNIDADE 2

```

lista = [30,50,20,30,60,30,20,10,50,20,100,200]
#cria um set para auxiliar, para entender melhor leia o texto a seguir

duplicados = set()
unicos = []
for x in lista:
    if x not in duplicados:
        unicos.append(x)
        duplicados.add(x)

print(dup_items)

```

FONTE: O autor



CONJUNTOS – UM TIPO DE DADOS SIMILAR ÀS LISTAS

Definido em Python é uma estrutura de dados equivalente a conjuntos em matemática. Pode consistir em vários elementos; a ordem dos elementos em um conjunto é indefinida. Você pode adicionar e excluir elementos de um conjunto, você pode iterar os elementos do conjunto, você pode realizar operações padrão em conjuntos (união, intersecção, diferença). Além disso, você pode verificar se um elemento pertence a um conjunto.

Ao contrário das matrizes, onde os elementos são armazenados como lista ordenada, a ordem dos elementos em um conjunto é indefinida (além disso, os elementos do conjunto geralmente não são armazenados em ordem de aparição no conjunto; isso permite verificar se um elemento pertence a um conjunto mais rápido do que apenas passando por todos os elementos do conjunto).

Qualquer tipo de dado imutável pode ser um elemento de um conjunto: um número, uma string, uma tupla. Tipos de dados mutáveis (variáveis) não podem ser elementos do conjunto. Em particular, list não pode ser um elemento de um conjunto (mas tuple pode), e outro conjunto não pode ser um elemento de um conjunto. A exigência de imutabilidade decorre da maneira como os computadores representam conjuntos na memória.

Você pode definir um conjunto tão simples quanto nomeando todos os seus elementos entre colchetes. A única exceção é o conjunto vazio, que pode ser criado usando a função set(). Se set(..) tiver uma lista, uma string ou uma tupla como parâmetro, retornará um conjunto composto por seus elementos. Por exemplo:

```

A = {1, 2, 3}
A = set('qwerty')
print(A)

```

irá imprimir {'e', 'q', 'r', 't', 'w', 'y'} como saída.

A ordem dos elementos não é importante. Por exemplo, o programa

```
A = {1, 2, 3}
B = {3, 2, 3, 1}
print(A == B)
```

irá imprimir True, porque A e B são conjuntos iguais.

Cada elemento pode entrar no conjunto apenas uma vez. set('Hello') retorna o conjunto de quatro elementos: {'H', 'e', 'l', 'o'}.

Você pode obter o número de elementos no conjunto usando a função len. Você também pode iterar todos os elementos do conjunto (em uma ordem indefinida!) Usando o loop for:

```
primes = {2, 3, 5, 7, 11}
for num in primes:
    print(num)
```

Você pode verificar se um elemento pertence a um conjunto usando a palavra-chave in: expressões como a in A retornar um valor do tipo bool. Da mesma forma, há a operação oposta not in. Para adicionar um elemento ao conjunto, há o método add:

```
A = {1, 2, 3}
print(1 in A, 4 not in A)
A.add(4)
```

Existem dois métodos para remover um elemento de um conjunto: discard e remove. Seu comportamento varia apenas no caso, se o item excluído não estava no conjunto. Neste caso o discard do método não faz nada e o método remove lança o KeyError exceção. Finalmente, o pop remove um elemento aleatório do conjunto e retorna seu valor. Se o conjunto estiver vazio, pop gera a exceção KeyError. Você pode transformar um conjunto em lista usando a list funções.

As operações de conjuntos realizados por um elemento set, bem como o respectivo comando são mostrados a seguir:

A B A.union (B)	Retorna um conjunto que é a união dos conjuntos A e B
A = B A.update (B)	Adiciona todos os elementos da matriz B ao conjunto A
A e B A.intersection (B)	Retorna um conjunto que é a interseção dos conjuntos A e B
A & = B A.intersection_update (B)	Deixa no conjunto A apenas os itens que pertencem ao conjunto B
A - B Diferença (B)	Retorna a diferença definida de A e B (os elementos incluídos em A , mas não incluídos em B).
A - = B A.difference_update (B)	Remove todos os elementos de B a partir do conjunto A .
A ^ B A. diferença simétrica (B)	Retorna a diferença simétrica dos conjuntos A e B (os elementos pertencentes a A ou B , mas não a ambos os conjuntos simultaneamente).

A ^ = B	Escreve em A a diferença simétrica dos conjuntos A e B
A . s y m m e t r i c _ difference_update (B)	
A <= B	Retorna true se A é um subconjunto de B
A. subconjunto (B)	
A> = B	Retorna true se B é um subconjunto de A
A <B	Equivalente a A <= B and A != B
A> B	Equivalente a A >= B and A != B

FONTE: <<https://www.snakify.org/pt/lessons/sets/>>. Acesso em: 20 nov. 2019.

RESUMO DO TÓPICO 1

Neste tópico, você aprendeu que:

- As listas são uma importante estrutura do Python.
- Existem funções para manipular manipular listas em Python.
- Listas em Python seguem a mesma lógica das listas no mundo real.
- É possível manipular arquivos com Python.
- É simples integrar listas em Python com arquivos externos.
- Existe a estrutura set muito similar a uma lista.

AUTOATIVIDADE



- 1 Utilizando listas crie um programa que leia um conjunto de valores e retorne o menor elemento da mesma.
- 2 Seja dada a lista `lista = [30,50,20,30,60,30,20,10,50,20,100,200]`. Crie um programa que crie um clone desta lista, ou seja, uma lista com os mesmos valores.
- 3 Os índices são a maneira de acessar os valores dentro de uma lista. Seja dada a lista `nums = ["Tarefa A", "Tarefa B", "Tarefa C", "Tarefa D"]`. Crie um programa que exiba os elementos dessa lista, junto com seus respectivos índices.
- 4 Uma lista é uma estrutura de dados composta por itens organizados de forma linear. Utilizando o conceito de listas, crie um programa utilizando o conceito de CRUD (CREATE, READ, UPDATE, DELETE). Mas que utilize a própria lista como estrutura de armazenamento.
- 5 Uma lista é uma estrutura de dados composta por itens organizados de forma linear. Utilizando o conceito de listas, crie um programa utilizando o conceito de CRUD (CREATE, READ, UPDATE, DELETE). Mas que utilize uma lista e após grave os resultados em um arquivo de texto.

NUMPY

1 INTRODUÇÃO

O Python é uma linguagem de programação que tem inúmeras aplicações e tem sido amplamente adotada por todos os tipos de comunidades, da ciência de dados aos negócios.

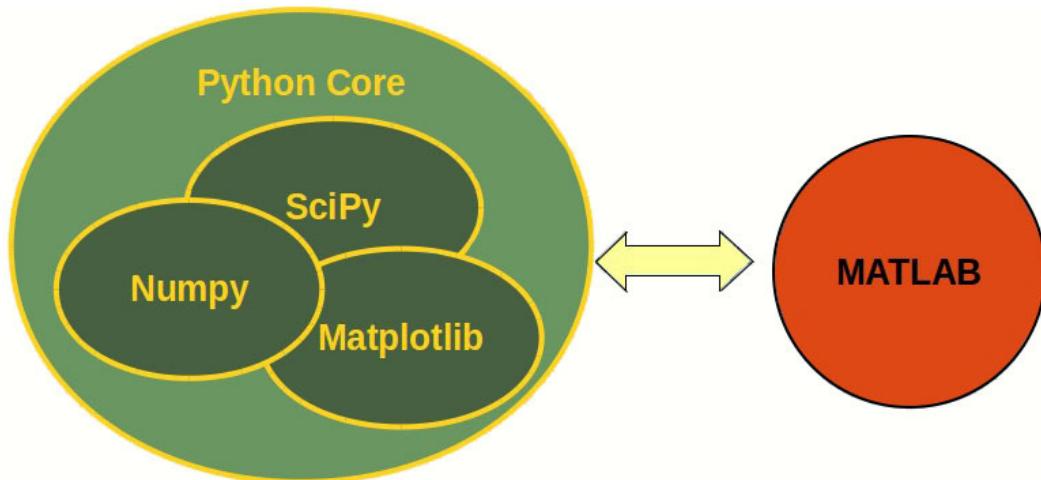
Essas comunidades valorizam o Python por sua sintaxe precisa e eficiente, curva de aprendizado relativamente plana e boa integração com outras linguagens (por exemplo, C / C ++). A popularidade da linguagem resultou na produção de vários pacotes Python para visualização de dados, aprendizado de máquina, processamento de linguagem natural, análise de dados complexa e muito mais.

Dentre os pacotes e bibliotecas, as que mais se destacam são as ditas bibliotecas científicas, aquelas utilizadas como suporte à ciência de dados, durante este tópico vamos conhecer a mais tradicional destas bibliotecas: NumPy.

2 NUMPY A BIBLIOTECA MAIS FAMOSA

O NumPy é o pacote fundamental para a computação científica em Python. É uma biblioteca Python que fornece um objeto de matriz multidimensional, vários objetos derivados (como matrizes e matrizes mascaradas) e uma variedade de rotinas para operações rápidas em matrizes. Tais rotinas incluem manipulação matemática, lógica, de formas, classificação, seleção, entrada/saída, transformadas discretas de Fourier, álgebra linear básica, operações estatísticas básicas, simulação aleatória e muito mais (WHAT IS NUMPY?, 2019, s.p., tradução nossa).

FIGURA 3 - NUMPY



FONTE: <<http://bit.ly/36R0saX>>. Acesso em: 10 dez. 2019.

Para Santiago Jr. (2018, s.p.):

O NumPy fornece um grande conjunto de funções e operações de biblioteca que ajudam os programadores a executar facilmente cálculos numéricos. Esses tipos de cálculos numéricos são amplamente utilizados em tarefas como:

- Modelos de Machine Learning: ao escrever algoritmos de Machine Learning, supõe-se que se realize vários cálculos numéricos em Array. Por exemplo, multiplicação de Arrays, transposição, adição, etc. O NumPy fornece uma excelente biblioteca para cálculos fáceis (em termos de escrita de código) e rápidos (em termos de velocidade). Os Arrays NumPy são usados para armazenar os dados de treinamento, bem como os parâmetros dos modelos de Machine Learning.
- Processamento de Imagem e Computação Gráfica: imagens no computador são representadas como Arrays Multidimensionais de números. NumPy torna-se a escolha mais natural para o mesmo. O NumPy, na verdade, fornece algumas excelentes funções de biblioteca para rápida manipulação de imagens. Alguns exemplos são o espelhamento de uma imagem, a rotação de uma imagem por um determinado ângulo etc.
- Tarefas matemáticas: NumPy é bastante útil para executar várias tarefas matemáticas como integração numérica, diferenciação, interpolação, extrapolação e muitas outras. O NumPy possui também funções incorporadas para álgebra linear e geração de números aleatórios. É uma biblioteca que pode ser usada em conjunto do SciPy e Matplotlib. Substituindo o MATLAB quando se trata de tarefas matemáticas.

Segundo VanderPlas (2016, p. 33, tradução nossa):

O NumPy (Numerical Python) fornece uma interface eficiente para armazenar e operar em buffers de dados densos. De certa forma, as matrizes NumPy são como o tipo de lista interno do Python, mas as matrizes NumPy fornecem operações de armazenamento e dados muito mais eficientes à medida que as matrizes aumentam de tamanho.

As matrizes NumPy formam o núcleo de quase todo o ecossistema de ferramentas de ciência de dados em Python; portanto, o tempo gasto aprendendo a usar o NumPy com eficácia será valioso, independentemente do aspecto da ciência de dados que lhe interessa.

A partir do próximo subtópico aprenderemos a instalar e utilizar essa poderosa biblioteca.

3 INSTALANDO E UTILIZANDO NUMPY

Nas versões atuais da instalação do Python o NumPy costuma vir de modo nativo, ou seja, não é necessário ser instalado. De todo modo, é possível fazer essa verificação dentro do próprio Python. Execute o código a seguir e deve retornar a versão do Numpy instalada.

QUADRO 28 - EXEMPLO DE ARQUIVO DE TEXTO

```
import numpy
print(numpy.__version__)
```

FONTE: O autor

Caso não esteja instalado basta utilizar o comando pip e instalar através do pip3 install numpy.

4 NUMPY ARRAYS

O Numpy contém diversas funcionalidades e métodos, no entanto iremos focar no seu objeto principal, que é o NumPy Array. Assim como nas demais linguagens de programação um arranjo (array) é um objeto que possui N dimensões e permite armazenar qualquer tipo de dado.

A maneira simples de criar um array é passando os valores. Note também que é possível apelidar o NumPy como np para facilitar a codificação.

QUADRO 29 - CRIANDO UM ARRAY NUMPY

```
import numpy
print(numpy.__version__)
```

FONTE: O autor

A maneira mais simples de criar um array pelo NumPy é chamando o objeto array através do `numpy.array()`. Vejamos um exemplo.

QUADRO 30 - CRIANDO UM ARRAY NUMPY

```
import numpy as np
x = np.array([1, 2, 3, 4, 5, 6])
print("Conteúdo de x:", x)
print("Tipo de dados de x:", type(x))
```

FONTE: O autor

Em relação aos tipos de dados, um NumPy array é igual ao que já aprendemos, ou seja, dinâmico. O objeto permite armazenar conteúdo de diversos tipos de dados, veja no exemplo como é possível criar um array com diversos tipos de dados armazenados.

QUADRO 31 - CRIANDO UM ARRAY COM VÁRIOS TIPOS DE DADOS

```
import numpy as np
x = np.array([1, "batman", 3.5, 4, 5, False])
print("Conteúdo de x:", x)
print("Tipo de dados de x:", type(x))
```

FONTE: O autor

No entanto, apesar de permitir o armazenamento dinâmico, um array NumPy permite que os dados tenham um tipo específico. Isto pode ser percebido ao escrever o array `print(x)`, no qual o tipo de dados é denotado por `dtype`. O `dtype` descreve como os bytes no bloco de memória de tamanho fixo correspondente a um item da matriz devem ser interpretados.

5 ARRAYS MULTIDIMENSIONAIS

Até o momento os arrays criados tiveram uma forma muito similar às listas, pois se tratavam dos ditos arrays unidimensionais, que se comportam como vetores armazenando valores em uma única dimensão.

A partir de agora iremos explorar as dimensões dos arrays, a começar dos arrays bidimensionais. Apesar do nome complexo, um array bidimensional representa a forma mais comum de armazenarmos dados em um computador: uma tabela.

TABELA 1 - COMO ARMAZENAMENTO DE DADOS

1. PRODUTOS E FORNECEDORES		Essa é a primeira etapa da planilha. Aqui, você deve fazer o cadastro de todos. Para isso, você vai precisar definir algumas variáveis.					
PRODUTOS	FORNECEDORES	Análise da formação dos preços produtos para revenda					
Produto	Fornecedor	Custo unitário com o produto	Quantidade	Custo total com o produto	Mark-up	Preço unitário com markup	
Produto 1	Fornecedor 1	R\$ 200,00	30	R\$ 6.000,00	100,00%	R\$ 400,00	
Produto 2	Fornecedor 1	R\$ 100,00	50	R\$ 5.000,00	100,00%	R\$ 200,00	
Produto 3	Fornecedor 1	R\$ 50,00	50	R\$ 2.500,00	100,00%	R\$ 100,00	
Produto 4	Fornecedor 1	R\$ 10,00	100	R\$ 1.000,00	100,00%	R\$ 20,00	
Produto 5	Fornecedor 1	R\$ 150,00	20	R\$ 3.000,00	100,00%	R\$ 300,00	
Produto 6	Fornecedor 1	R\$ 250,00	2	R\$ 500,00	100,00%	R\$ 500,00	
Produto 7	Fornecedor 1	R\$ 300,00	5	R\$ 1.500,00	100,00%	R\$ 600,00	

FONTE: <<http://bit.ly/35Pnqyb>>. Acesso em: 20 nov. 2019.

Desde o estudo sobre listas, vimos que os elementos são armazenados e podem ser acessados através de um índice sequencial iniciado em 0. A partir de agora teremos duas dimensões, isso significa que cada elemento será referenciado por um par ordenado, no caso dos arrays bidimensionais.

Para ficar mais claro como funciona, considere um array a , que contenha duas dimensões, cada uma de tamanho 3.

QUADRO 32 - ARMAZENAMENTO EM UM ARRAY BI-DIMENSIONAL

	Coluna 0	Coluna 1	Coluna 2
Linha 0	“João” $a[0][0]$	25 $a[0][1]$	True $a[0][2]$
Linha 1	“Fernando” $a[1][0]$	10 $a[1][1]$	False $a[1][2]$
Linha 2	“Marcel” $a[2][0]$	17 $a[2][1]$	True $a[2][2]$

FONTE: O autor

Agora que já compreendemos como a indexação funciona de modo abstrato, vamos construir esse mesmo array utilizando NumPy.

QUADRO 33 - CRIANDO UM ARRAY BIDIMENSIONAL

```
import numpy as np
a = np.array([["João",25,True], ["Fernando", 10, False], ["Marcel", 17, True]])
print(a)
```

FONTE: O autor

No exemplo anterior foi criado uma matriz igual a Tabela 1 e posteriormente exibido todo o conteúdo. No entanto, não estamos percorrendo este array. Veja que para acessar o item “joão” é necessário passar a posição a[0][0], logo para percorrer um array serão necessárias mais de uma iteração. Vamos entender melhor com o exemplo.

QUADRO 34 - CRIANDO UM ARRAY COM VÁRIOS TIPOS DE DADOS

```
import numpy as np
a = np.array([["João",25,True], ["Fernando", 10, False], ["Marcel", 17, True]])
for linha in a:
    #print(linha)
    for coluna in linha:
        print(coluna)
```

FONTE: O autor

Com o exemplo anterior a primeira iteração irá primeiro percorrer as linhas e depois as colunas, perceba que se descomentar o trecho comentado, o código irá exibir a linha na forma de um vetor. O NumPy possui uma gama de recursos, vamos ver algumas das suas principais funções.

Zeros: permite construir arrays contendo apenas valores 0.

QUADRO 35 - NUMPY ZEROS

```
#array unidimensional
a = np.zeros(10)
print(a)
#array bidimensional
a = np.zeros([2,4])
print(a)
#array bidimensional
a = np.zeros([10,10])
print(a)
```

FONTE: O autor

Ones: permite construir arrays contendo apenas valores 1.

QUADRO 36 - NUMPY ONE

```
#array unidimensional
a = np.ones(10)
print(a)
#array bidimensional
a = np.ones([2,4])
print(a)
#array bidimensional
a = np.ones([10,10])
print(a)
```

FONTE: O autor

6 OPERAÇÕES COM ARRAYS MULTIDIMENSIONAIS

O NumPy é utilizado para processar diversos tipos de dados e, como vimos, pode possuir muitas dimensões. Na documentação do Python, esse tipo de objeto é referido como ndarray (n-dimensions array — array de n dimensões).

Como há um conjunto de métodos e funções que operam em cima dos dados armazenados em um ndarray, para que um objeto possa ser considerado deste tipo e que não haja problemas na execução de tais funções é necessário que o array contenha um mesmo tipo de dados, geralmente tendo um tamanho fixo.

Um objeto ndarray possui um conjunto de métodos que operam na matriz ou que são utilizados para construir matrizes, no geral o retorno de tais métodos é um objeto do tipo matriz (Numpy Array). Vamos conferir cada uma destas operações.

Transpose: em matemática, matriz transposta é a matriz que se obtém da troca de linhas por colunas de uma determinada matriz. Desta forma, transpor uma matriz é a operação que leva na obtenção de sua transposta.

QUADRO 37 - MATRIZ TRANSPOSTA

```
a = np . array ([[ 1 , 2 , 3],[4 ,5,6 ]])
#array original
print(a)
#array transposto
print(a.T)
```

FONTE: O autor

Shape: o shape retorna o tamanho de um array. Especificamente, retorna o tamanho de suas dimensões.

QUADRO 38 - TAMANHO DO ARRAY

```
a = np . array ([[ 1 , 2 , 3],[4 ,5,6 ]])  
#array original  
print(a)  
#tamanho das dimensões  
print(a.shape)
```

FONTE: O autor

Sum: em arrays que armazenam numéricos, retorna a soma de todos os valores.

QUADRO 39 - SOMA DO ARRAY

```
a = np . array ([[ 1 , 2 , 3],[4 ,5,6 ]])  
#array original  
print(a)  
#soma dos valores  
print(a.sum())
```

FONTE: O autor

Max: em arrays que armazenam numéricos, retorna o maior valor armazenado.

QUADRO 40 - MAIOR VALOR DO ARRAY

```
a = np . array ([[ 1 , 2 , 3],[4 ,5,6 ]])  
#array original  
print(a)  
#soma dos valores  
print(a.max())
```

FONTE: O autor

Eye: conhecido como matriz identidade na álgebra linear, com este comando é possível criar um array com 1 na diagonal e horizontal.

QUADRO 41 - CRIANDO MATRIZ IDENTIDADE

```
np.eye(5)
#irá resultar em
#array([[1., 0., 0., 0., 0.],
#       [0., 1., 0., 0., 0.],
#       [0., 0., 1., 0., 0.],
#       [0., 0., 0., 1., 0.],
#       [0., 0., 0., 0., 1.]])
```

FONTE: O autor

Random: ao receber as dimensões do array, irá criar um array com valores aleatórios em escala 1. Para aumentar a escala basta multiplicar por um valor maior.

QUADRO 42 - CRIANDO MATRIZ COM VALORES ALEATÓRIOS

```
np.random.rand(5, 4)*100
```

FONTE: O autor

Arange: similar ao comando range visto anteriormente, cria um array em um intervalo de valores.

QUADRO 43 - CRIANDO MATRIZ COM VALORES SEQUENCIAIS

```
np.arange(1, 10)
```

FONTE: O autor

Um objeto array/ndarray possui muitos métodos que operam na matriz ou com ela de alguma maneira, geralmente retornando um resultado da matriz. Esses métodos são explicados brevemente a seguir. A documentação de cada método tem uma descrição mais completa. Seguindo os mesmos exemplos anteriores o NumPy ainda conta com as seguintes funções de manipulação:

- ndarray.item: copie um elemento de uma matriz para um escalar padrão do Python e retorne-o.
- ndarray.tolist: retorne a matriz como uma lista aninhada profunda a.ndim de escalares Python.
- ndarray.itemset: inserir escalar em uma matriz (escalar é convertido no tipo de matriz, se possível).
- ndarray.tostring: construa bytes Python contendo os bytes de dados brutos na matriz.
- ndarray.max: retorne o máximo ao longo de um determinado eixo.
- ndarray.argmax: retorna índices dos valores máximos ao longo do eixo especificado.

- ndarray.min: retorne o mínimo ao longo de um determinado eixo.
- ndarray.argmin: retorna índices dos valores mínimos ao longo do eixo especificado de a.
- ndarray.ptp ([eixo, saída, arquivo de manutenção]) Valor pico a pico (máximo - mínimo) ao longo de um determinado eixo.
- ndarray.clip ([min, max, out]) retorna uma matriz cujos valores são limitados a [min, max].
- ndarray.round: retorne a com cada elemento arredondado para o número especificado de casas decimais.
- ndarray.mean: retorna a média dos elementos da matriz ao longo do eixo especificado.
- ndarray.std: retorna o desvio padrão dos elementos da matriz ao longo do eixo especificado.
- ndarray.prod : retornar o produto dos elementos da matriz sobre o eixo especificado.

Em diversas aplicações, mas principalmente na hora de criar os algoritmos de machine learning, será necessário que os dados sigam determinado comportamento. Isso significa que as dimensões de um array podem não estar de acordo com o esperado por determinado método. Por isso, os arrays podem ser redimensionados ao decorrer do código. O processo de ajuste de tamanho é realizado pelo comando reshape. Para que esse processo funcione, o tamanho da matriz inicial deve corresponder ao tamanho da matriz remodelada.

QUADRO 44 - AJUSTANDO O TAMANHO DE UM ARRAY

```
#criando array
a = np.arange(1, 10)
print(a)

#ajustando os irems em nova dimensão
ar = a.reshape((3, 3))
print(ar)

#criando array
x = np.array([1, 2, 3])

#ajustando os irems em nova dimensão
x.reshape((1, 3))
```

FONTE: O autor

Uma vez compreendido como que funciona o ajuste de elementos dentro de um array, outro elemento muito utilizado é o fatiamento de arrays, muito similar ao mesmo procedimento em listas.

Segundo VanderPlas (2016, p. 44, tradução nossa):

Assim como podemos usar colchetes para acessar elementos individuais da matriz, também podemos usá-los para acessar subarrays com a notação de fatia, marcada pelo caractere de dois pontos (:). A sintaxe de fatiamento NumPy segue a da lista padrão do Python; para acessar uma fatia de uma matriz x , use o seguinte: $x[\text{start}:\text{stop}:\text{step}]$.

Se algum deles não for especificado, eles serão padronizados com os valores $\text{start} = 0$, $\text{stop} = \text{tamanho da dimensão}$, $\text{step} = 1$. Vamos analisar o acesso aos subarrays em uma dimensão e em várias dimensões.

QUADRO 45 - UTILIZANDO ARRAY RESHAPE

```
#cria um array com elementos de 0 até 9
x = np.arange(10)
print(x)
# mostra os 5 primeiros elementos
print(x[:5])

#mostra os 5 elementos depois de 5
print(x[5:])

# mostra todos elementos entre 4 e 7, incluindo 4, excluindo 7
print(x[4:7] )

# seguindo a sintaxe, considera apenas o intervalo 2
print(x[::-2])

# inicializa em 1, e navega com intervalo 2
print(x[1::2])

# mosstra todos os elementos em ordem inversa
print(x[::-1])
```

FONTE: O autor

Quando se trata de arrays multidimensionais, a sintaxe será a mesma. No entanto, a partir de agora, como consideraremos duas dimensões, quando o $:$ for colocado em uma dimensão sem nenhum valor, todos os itens desta dimensão serão considerados.

QUADRO 46 - UTILIZANDO ARRAY RESHAPE EM MULTIDIMENSIONAL

```
#cria um array com valores randomicos 3X4
a = np.random.randint(10, size=(3, 4))
print(a)

# mostra o conteúdo em duas linhas e três colunas
print(a[2:, :3])

# utiliza apenas os passos 2
print(a[3:, ::2])

#exibe toda a primeira coluna(0)
print(a[:, 0])

#exibe toda a primeira linha (0)
print(a[0, :])

#exibe toda a primeira linha (0) de uma outra maneira
print(a[0])
```

FONTE: O autor

Agora que conhecemos melhor sobre a manipulação de arrays vamos começar a interagir com objetos externos. No caso do NumPy ele tem suas próprias funções para salvar e recuperar arquivos.

QUADRO 47 - SALVANDO E RECUPERANDO UM ARRAY

```
a = np.array([1,2,3,4,5,6])
np.save("a.dat",a)
b = np.load("a.dat")
print(b)
```

FONTE: O autor

Como o processo de codificação muitas vezes envolve diversos arrays, muitas vezes é necessário uni-los ou separá-los em partes. O processo de união, é dado pela concatenação de arrays.

QUADRO 48 - CONCATENANDO ARRAYS

```
a = np.array([1,2,3,4,5,6])
np.save("a.dat",a)
b = np.load("a.dat")
print(b)
```

FONTE: O autor



Ainda que exista um método para salvar e recuperar dados dentro de um array NumPy, é possível salvá-lo em um documento, bem como recuperar um documento, conforme visto no Tópico 1.

7 MANIPULANDO ALGEBRICAMENTE UM ARRAY

Iniciamos a Unidade 2 falando sobre um dos maiores benefícios do Python, como um pré-requisito para uma linguagem para Big Data, que é a programação vetorial. A partir de agora vamos compreender melhor como é realizada a manipulação de arrays sem a necessidade de iterações.

Quando tratamos de arrays e operações, um limiar são operações simples, ou seja, todo o vetor será considerado em relação a este limiar. Vamos considerar o exemplo a seguir, realizando múltiplas operações com um mesmo vetor.

QUADRO 49 - OPERAÇÕES ALGÉBRICAS COM ARRAY

```
a = np.array([[1,2,3,4],[5,6,7,8]])  
  
#array original  
print("a =", a)  
  
#array somando 10 a cada elemento  
print("a + 5 =", a + 10)  
  
#array subtraindo 10 a cada elemento  
print("a - 5 =", a - 10)  
  
#array multiplicando por 2 a cada elemento  
print("a * 2 =", a * 2)  
  
#array dividindo por 2 a cada elemento  
print("a / 2 =", a / 2)  
  
#array fazendo divisão por inteiro 2 a cada elemento  
print("a // 2 =", a // 2)  
  
#array array negativo ou multiplicado por (-1) cada elemento  
print("-a =", -a)  
  
#array a2  
print("a ** 2 =", a ** 2)  
  
#array retornando o resto da divisão por 2  
print("a % 2 =", a % 2)
```

FONTE: O autor

Pode ser visto que a realização de operações algébricas com limiares é simples. No entanto, ao realizar o mesmo com operações que envolvem dois ou mais arrays, o processo envolve um pouco de matemática.

Considere, por exemplo, o array `a = np.array([[1,2],[3,4]])` e o `b = np.array([[1,2],[3,4], [5,6]])`. Note o que acontece ao tentar fazer o produto entre esses dois arrays.

FIGURA 4 - MULTIPLICAÇÃO COM ARRAYS DE DIMENSÕES DIFERENTES

```
[13] a = np.array([[1,2],[3,4]])
    b = np.array([[1,2],[3,4], [5,6]])
    a*b

D -----
ValueError                                Traceback (most recent call last)
<ipython-input-13-5fa80fceea6f> in <module>()
      1 a = np.array([[1,2],[3,4]])
      2 b = np.array([[1,2],[3,4], [5,6]])
----> 3 a*b

ValueError: operands could not be broadcast together with shapes (2,2) (3,2)

SEARCH STACK OVERFLOW
```

FONTE: O autor

Note que o compilador acusou que as dimensões dos arrays que estão sendo multiplicados, são diferentes. Isto acontece, pois, matematicamente, a multiplicação de matrizes é um processo que só pode ser feito se o número de colunas da primeira matriz é igual ao número de linhas da segunda matriz.

O exemplo não teria problema, caso os dados fossem `a = np.array([[1,2],[3,4]])` e `b = np.array([[2,2]])`. No entanto, esse seria um exemplo da multiplicação de uma matrix 2X2 por um vetor de dois elementos.

Para realizar a multiplicação matricial, segundo a matemática, utilizaremos o comando `dot()`. O comando `dot` aplica o conceito de que, ao receber as matrizes $A = (a_{ij})_{mxn}$ e $B = (b_{jk})_{npx}$, o produto de $A \cdot B = \text{matriz } D = (d_{ik})_{mpx}$, onde $d_{ik} = a_{i1} \cdot b_{1k} + a_{i2} \cdot b_{2k} + \dots + a_{in} \cdot b_{nk}$.

Apesar de matematicamente ser uma operação relativamente complexa, em Python se define a utilizar a função `dot`. Veja como funciona.

QUADRO 50 - MULTIPLICAÇÃO DE MATRIZES

```
a = np.array([[1,2],[3,4],[5,6]])
b = np.array([[1,1,1],[2,2,2]])
a.dot(b)
```

FONTE: O autor

As demais operações com matrizes, seguirão conforme a multiplicação, sempre com uma fundamentação matemática. Vejamos alguns exemplos destas operações.

QUADRO 51 - OPERAÇÕES COM MATRIZES

```
a = np.array([1, 2, 3])
b = np.array([(1.5, 2, 3), (4, 5, 6,)])
c = np.array([(1.5, 2, 3), (4, 5, 6,)], [(3, 2, 1), (4, 5, 6)]))

#subtração de arrays
r1 = a - b
print(r1)

r2 = np.subtract(a, b)
print(r2)

#adição de arrays
r3 = b + a
print(r3)

r4 = np.add(b, a)
print(r4)

#divisão de arrays
r5 = a / b
print(r5)

r6 = np.divide(a, b)
print(r6)

#produto de arrays
r7 = a * b
print(r7)

r8 = (np.multiply(a, b))
print(r8)

#exponenciação
print(np.exp(b))

#raiz
print(np.sqrt(b))

#seno
print(np.sin(a))

#coseno
print(np.cos(b))

#logaritmo
print(np.log(a, b))
```

FONTE: O autor

8 NUMPY NA PRÁTICA

Podemos ver durante o Tópico 2 que o NumPy é uma biblioteca robusta e cheia de recursos. Para tentar colocar todo o conteúdo em prática, bem como aprender algo mais sobre essa biblioteca vamos fazer alguns exemplos.

Exemplo 1: crie uma matriz 2D com 1 na borda e 0 dentro.

QUADRO 52 - UNIDADE 2 TÓPICO 2 EXEMPLO 1

```
a = np.arange(10)
a.reshape(2, -1)
```

FONTE: O autor

Exemplo 2: dado o array $a = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]$, converta em uma matriz 2D com 2 linhas.

QUADRO 53 - UNIDADE 2 TÓPICO 2 EXEMPLO 2

```
a = np.arange(10)
a.reshape(2, -1)
```

FONTE: O autor

Exemplo 3: crie um vetor com 100 elementos aleatórios e ordenados.

QUADRO 54 - UNIDADE 2 TÓPICO 2 EXEMPLO 3

```
a = np.random.random(100)
a.sort()
print(a)
```

FONTE: O autor

Exemplo 4: sejam dados $a = np.array([1,2,3,4,5])$ e $b = np.array([5,6,7,8,9])$, remover de A os elementos que existam em b.

QUADRO 55 - UNIDADE 2 TÓPICO 2 EXEMPLO 4

```
a = np.array([1,2,3,4,5])
b = np.array([5,6,7,8,9])
np.setdiff1d(a,b)
```

FONTE: O autor

Exemplo 5: crie uma matriz 2D da forma 5x3 para conter números decimais aleatórios entre 5 e 10.

QUADRO 56 - UNIDADE 2 TÓPICO 2 EXEMPLO 5

```
a = np.arange(9).reshape(3,3)

rand_a = np.random.randint(low=5, high=10, size=(5,3)) + np.random.
random((5,3))

rand_a = np.random.uniform(5,10, size=(5,3))
print(rand_a)
```

FONTE: O autor

RESUMO DO TÓPICO 2

Neste tópico, você aprendeu que:

- A instalação do Numpy é feita através do PIP.
- A criação e manipulação de arrays é feita pelo Numpy.
- É possível criar arrays multidimensionais com Numpy.
- É possível realizar operações matemáticas com arrays.
- O conceito de array do Numpy é associado com a matemática.

AUTOATIVIDADE



- 1 Crie um array de 10 posições com valores aleatórios e retorne os números pares e ímpares dentro deste array.
- 2 Leia uma matriz de 3×3 elementos. Calcule a soma dos elementos que estão na diagonal principal.
- 3 Crie uma cartela de bingo com a dimensão de 6×5 e valores de 1 até 100. Exiba os valores na tela.
- 4 Crie um programa que receba duas dimensões e crie uma matriz com as mesmas utilizando valores aleatórios.
- 5 Crie uma matriz na qual a borda sejam os valores 0 e os valores internos sejam 1.

1 INTRODUÇÃO

Estamos estudando nesta unidade as principais maneiras de armazenar e manipular dados dentro do Python. Até o momento vimos o conceito de listas e Numpy Arrays. Ao conhecer o Pandas o objetivo não é comparar as estruturas, mas sim utilizar cada uma em seu melhor caso.

Quando evoluir seus estudos para os métodos de inteligência artificial será importante o domínio sobre as estruturas de armazenamento, bem como a manipulação de dados.

Dando continuidade a este estudo, iremos estudar o Pandas, uma das mais poderosas bibliotecas para armazenamento, manipulação e análise de dados utilizando Python.

FIGURA 5 - PANDAS



FONTE: <<http://bit.ly/2ND4ZXh>>. Acesso em: 20 nov. 2019.

2 CONHECENDO O PANDAS

Segundo McKinney (2019, p. 11, tradução nossa):

Pandas é um pacote Python que fornece estruturas de dados rápidas, flexíveis e expressivas, projetadas para tornar o trabalho com dados relacionais ou rotulados fácil e intuitivo. O objetivo é ser o bloco de construção de alto nível fundamental para a análise prática e prática dos dados do mundo real em Python. Além disso, ele tem o objetivo mais amplo de se tornar a ferramenta de análise / manipulação de dados de código aberto mais poderosa e flexível disponível em qualquer idioma. Já está a caminho desse objetivo. Pandas é adequado para muitos tipos diferentes de dados:

- Dados tabulares com colunas de tipo heterogêneo, como em uma tabela SQL ou planilha do Excel.
- Dados de séries temporais ordenados e não ordenados (não necessariamente com frequência fixa).
- Dados da matriz arbitrária (de forma homogênea ou heterogênea) com rótulos de linha e coluna.
- Qualquer outra forma de observação / estatística.

Ainda sobre as características, o Pandas pode ser considerado:

[...] um banco de dados nosql na memória, que possui construções semelhantes a sql, suporte estatístico e analítico básico, além de capacidade de representação gráfica. Por ser construído sobre o Cython, possui menos sobrecarga de memória e é executado mais rapidamente. Muitas pessoas estão usando Pandas para substituir o Excel, executar ETL, processar dados tabulares, carregar arquivos CSV ou JSON e muito mais. Embora tenha crescido fora do setor financeiro (para análise de dados de séries temporais), agora é uma biblioteca de manipulação de dados de uso gera.

Como o Pandas tem alguma linhagem de volta ao NumPy, ele adota alguns padrões que os programadores normais de Python podem não estar cientes ou familiarizados. Certamente, pode-se usar o Cython para realizar análises de dados de digitação rápida com um dialeto semelhante ao Python, mas com os Pandas, você não precisa. Este trabalho é feito para você. Se você estiver usando Pandas e as operações vetorializadas, estará chegando perto da velocidade do nível C, mas escrevendo Python (HARRISON, 2016, p. 5-6, tradução nossa).

Agora que temos uma visão geral sobre o Pandas, vamos conhecer um pouco mais sobre suas estruturas e seu funcionamento.

Antes de iniciarmos com a prática, assim como o NumPy, verifique se já tem o Pandas instalado em sua máquina. através do código a seguir:

QUADRO 55 - VERIFICA INSTALAÇÃO DO PANDAS

```
import Pandas
Pandas.__version__
```

FONTE: O autor

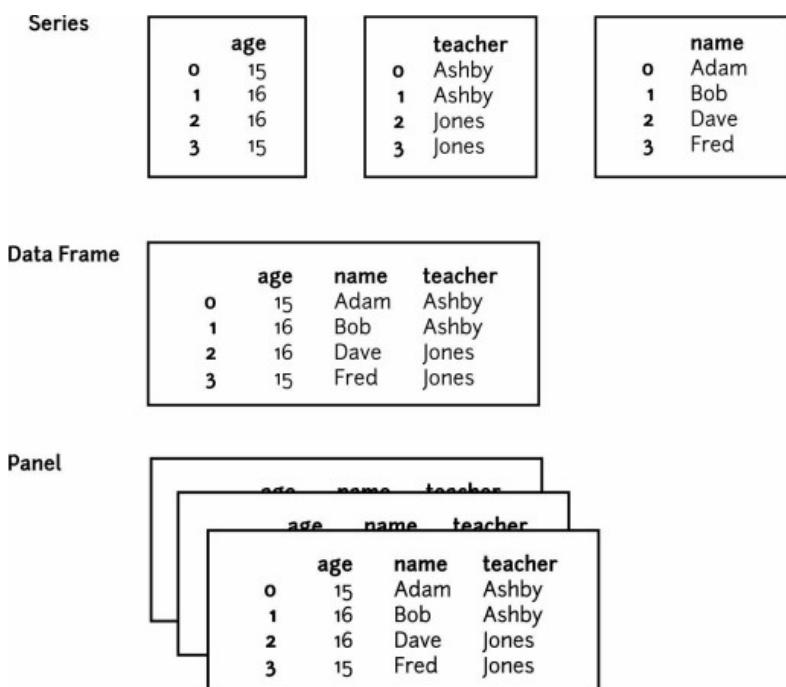
Caso não esteja instalado basta fazê-lo através do pip: pip3 install Pandas.

No nível muito básico, os objetos do Pandas podem ser vistos como versões aprimoradas de matrizes estruturadas NumPy nas quais as linhas e colunas são identificadas com rótulos, em vez de índices inteiros simples. O Pandas fornece uma série de ferramentas, métodos e funcionalidades úteis sobre as estruturas básicas de dados, mas quase tudo o que se segue exigirá uma compreensão do que são essas estruturas (VANDERPLAS, 2016, p. 98, tradução nossa).

Portanto, antes de prosseguirmos, vamos apresentar essas três estruturas de dados fundamentais do Pandas:

- Series: 1 dimensão.
- DataFrame: 2 dimensões .
- Panel: múltiplas dimensões.

FIGURA 6 - ESTRUTURAS DE DADOS PANDAS



FONTE: Harrison; Prentiss (2016, p. 12)

3 PANDAS SÉRIES

Uma série (*series*) é a estrutura básica de armazenamento do Pandas, pode ser associada a uma lista ou a um array unidimensional. Assim como nas demais estruturas, cada item dentro de uma série tem um índice associado.

Vejamos um exemplo simples da criação de uma série, e note que o comando Series é escrito com S maiúsculo.

QUADRO 58 - VERIFICA INSTALAÇÃO DO PANDAS

```
import pandas as pd  
s = pd.Series([1,2,3,4,5])  
print(s)
```

FONTE: O autor



Todo conteúdo desenvolvido nos quadros, durante este tópico, está disponibilizado on-line em: <https://colab.research.google.com/drive/12pQ7778srD0zlHxPfmde9elkXc29E9NrG>.

Caso necessário, as séries permitem com que sejam definidos os índices, para além dos valores tradicionais que são sequenciais inicializando em 0.

QUADRO 59 - VERIFICA INSTALAÇÃO DO PANDAS

```
s = pd.Series ([1,2], index = ['a', 'b'])  
print(s['a'])  
print(s[0])
```

FONTE: O autor



Dicionários – Trabalhando com JSON dentro do Python

Antes de evoluirmos com a exploração do conteúdo do Pandas, é importante conhecermos essa estrutura de armazenamento do Python.

Todos os tipos de dados compostos que estudamos em detalhes até agora – strings, listas e tuplas – são coleções sequenciais. Isto significa que os itens na coleção estão ordenados da esquerda para a direita e eles usam números inteiros como índices para acessar os valores que eles contêm.

Dicionário é um tipo diferente de coleção. Ele é um tipo de mapeamento nativo do Python. Um mapa é uma coleção associativa desordenada. A associação, ou mapeamento, é feita a partir de uma chave, que pode ser qualquer tipo imutável, para um valor, que pode ser qualquer objeto de dados do Python.

Como exemplo, vamos criar um dicionário para traduzir palavras em inglês para Espanhol. Para este dicionário, as chaves são strings.

Uma maneira de criar um dicionário é começar com o dicionário vazio e adicionar pares chave-valor. O dicionário vazio é denotado {}

```
eng2sp = {}
eng2sp['one'] = 'uno'
eng2sp['two'] = 'dos'
eng2sp['three'] = 'tres'
```

A primeira atribuição cria um dicionário chamado eng2sp. As outras atribuições adicionam novos pares chave-valor para o dicionário. O lado esquerdo define o dicionário e a chave a ser associada. O lado direito define o valor a ser associado a essa chave. Nós podemos imprimir o valor atual do dicionário da maneira usual. Os pares chave-valor do dicionário são separados por vírgulas. cada par contém uma chave e um valor separado por dois pontos.

A ordem dos pares pode não ser a que você esperava. O Python usa algoritmos complexos, projetados para acesso muito rápido, para determinar onde os pares chave-valor são armazenadas em um dicionário. Para os nossos propósitos, podemos pensar que esta ordenação é imprevisível.

Outra maneira de criar um dicionário é fornecer uma lista de pares chave-valor usando a mesma sintaxe que a saída anterior.

```
eng2sp = {'three': 'tres', 'one': 'uno', 'two': 'dos'}
print(eng2sp)
```

Não importa em que ordem escrevemos os pares. Os valores em um dicionário são acessados com chaves, não com índices, por isso não há necessidade de se preocupar com a ordenação.

O exemplo mostra como usar uma chave para pesquisar o valor correspondente.

Método	Parâmetros	Descrição
keys	nenhum	Retorna uma vista das chaves no dicionário.
values	nenhum	Retorna uma vista dos valores no dicionário.
items	nenhum	Retorna uma vista dos pares chave-valor no dicionário.
get	key	Retorna o valor associado com a chave; ou None.
get	key,alt	Retorna o valor associado com a chave; ou alt.

O método keys retorna o que o Python 3 chama de **vista** (view) das chaves. Podemos iterar sobre a vista ou transformar a vista em uma lista usando a função list de conversão.

```
inventory = {'apples': 430, 'bananas': 312, 'oranges': 525, 'pears': 217}

for akey in inventory.keys():    # the order in which we get the keys is
    print("Got key", akey, "which maps to value", inventory[akey])

ks = list(inventory.keys())
print(ks)

for k in inventory:
    print("Got key", k)
```

FONTE: <<https://panda.ime.usp.br/pensepy/static/pensepy/11-Dicionarios/dicionarios.html>>. Acesso em: 20 nov. 2019.

Um dicionário é uma estrutura que mapeia chaves arbitrárias para um conjunto de valores arbitrários e uma Série é uma estrutura que mapeia chaves digitadas para um conjunto de valores digitados. Essa digitação é importante: assim como o código compilado específico do tipo por trás de uma matriz NumPy o torna mais eficiente do que uma lista Python para determinadas operações, as informações de tipo de uma série Pandas o tornam muito mais eficiente do que os dicionários Python para determinadas operações.

QUADRO 60 - VERIFICA INSTALAÇÃO DO PANDAS

```
population_dict = {'Camboriú': 38332521,
'Indaial': 26448193,
'Blumenau': 19651127,
'Florianópolis': 19552860,
'Brusque': 12882135}
population = pd.Series(population_dict)
print(population)
print(population['Camboriu'])
print(population['Camboriu':'Blumenau'])
```

FONTE: O autor

4 PANDAS DATAFRAME

Um DataFrame no Pandas pode ser compreendido como uma tabela, pois é uma estrutura bi-dimensional que permite armazenar dados. Um DataFrame pode ser associado a uma planilha ou até mesmo a uma tabela em um banco de dados, afinal seu armazenamento e visualização é similar a tais.

Veja um exemplo de como criar uma lista de nomes e depois transformá-la em um DataFrame.

QUADRO 61 - CRIANDO UM PRIMEIRO DATAFRAME

```
import pandas as pd

lst = ['João','Fernando','Gilberto','Felipe','Maria','Eduarda','Mikaela','Nicole']

df = pd.DataFrame(lst)
print(df)
```

FONTE: O autor

Um DataFrame é uma estrutura dinâmica, para criá-lo basta ter dados que possam ser projetados em 2D e estejam em um formato ou tipo de dados aceito pelo Python. Veja um exemplo de conversão de um dicionário para DataFrame.

QUADRO 61 - DICIONÁRIO PARA DATAFRAME

```
data = {'Nome':['Henrique', 'Joana', 'Felipe', 'Jane'],
        'Idade':[20, 21, 19, 18]}

df = pd.DataFrame(data)

print(df)
```

FONTE: O autor

Com o NumPy não é diferente, é possível criar e manipular objetos e, posteriormente ,transformá-los em DataFrames.

QUADRO 63 - NUMPY PARA DATAFRAME

```
import numpy as np

ex1 = pd.DataFrame(np.random.rand(3, 2), columns=['foo', 'bar'], index=['a', 'b', 'c'])
print(ex1)

ex2 = pd.DataFrame(np.zeros(3, dtype=[('A', 'i8'), ('B', 'f8')]))
print(ex2)
```

FONTE: O autor

5 ACESSANDO E MANIPULANDO DADOS COM DATAFRAMES

Uma das funções mais importantes de um DataFrame é a facilidade de se conectar e manipular arquivos externos, seja uma tabela em um banco de dados, um documento de texto ou uma planilha do Excel.

Durante estudos de Big Data e Inteligência Analítica, você aprenderá as diversas bases de dados e também sobre ambientes como Keagle e UCI que oferecem diversos conjuntos de dados para estudos na área de *machine learning* e Big Data.



Para dar continuidade a esta seção é necessário a utilização de arquivos CSV, por isso é recomendado realizar o download do dataset disponibilizado em: <https://www.kaggle.com/shivachandel/kc-house-data/data#>.

Através do comando `read_csv` é possível transformar um documento csv em um DataFrame, caso esteja utilizando um Colab Notebook é necessário realizar a montagem do disco. Com o emprego do comando `head()` é mostrado o cabeçalho do documento, composto pelo nome das colunas e um conjunto de 5 registros para que o programador tenha uma visão geral do que está armazenado.

QUADRO 64 - ABRINDO DOCUMENTOS COM DATAFRAME

```
dados = pd.read_csv('/content/drive/My Drive/kc_house_data.csv', sep=',')
dados.head()
```

FONTE: O autor

Assim como nas listas e nos arrays o acesso aos dados é dado por colchetes []. No entanto, ao invés de utilizar apenas os índices numéricos é possível explorar o nome das colunas.

QUADRO 65 - EXIBINDO DADOS DE UM DATAFRAME

```
#Exibe os dados da coluna id
print(dados["id"].head())

#Exibe os dados da coluna date
print(dados["date"].head())

#cria um array com nome de varias colunas
colunas = ["price","sqft_lot","condition","yr_built"]

#Exibe os dados das colunas utilizando o array criado
print(dados[colunas].head())
```

FONTE: O autor

Assim como no NumPy um DataFrame tem opções de obter o valor máximo, mínimo, médio, soma, entre outros valores estatísticos de maneira independente. Complementarmente, é possível obter todos os dados estatísticos através do comando describe().

QUADRO 66 - DADOS ESTATÍSTICOS DO DATAFRAME

```
#Exibindo a média do DataFrame
print(dados.mean())

#Exibindo a soma do DataFrame
print(dados.sum())

#Exibindo o valor minimo do DataFrame
print(dados.min())

#Exibindo o valor máximo do DataFrame
print(dados.max())

#Exibindo os dados estatísticos do DataFrame
print(dados.describe())
```

FONTE: O autor

O DataFrame também possui funções de visualização dos dados. Na continuidade do livro e do curso. Vamos entender um pouco da função plot().

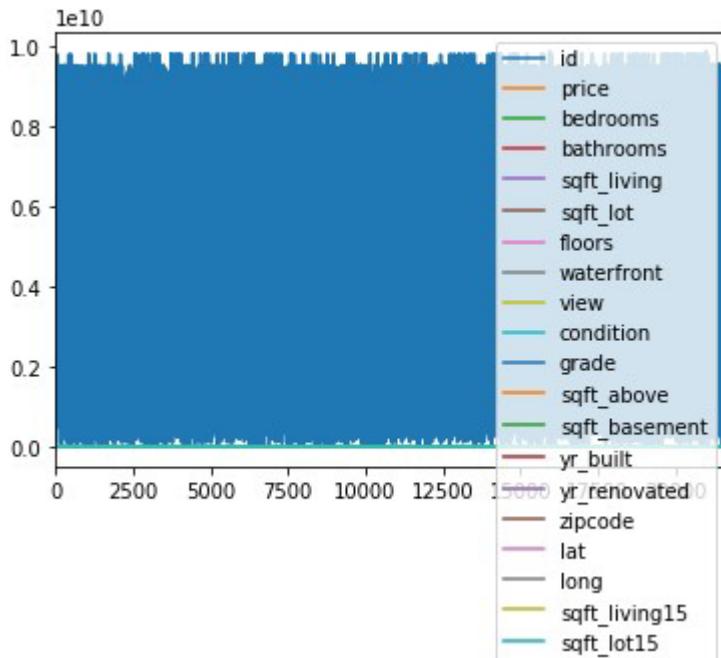
QUADRO 67 - PLOTAGEM DO DATAFRAME

```
dados.plot()
```

FONTE: O autor

Note que ao plotar todo o DataFrame há um volume de dados muito grande e por conta das diversas escalas, há sobreposição dos dados.

FIGURA 7 - PLOTAGEM DO DATAFRAME



FONTE: O autor

Note que ao plotar todo o DataFrame há um volume de dados muito grande e por conta das diversas escalas, há sobreposição dos dados.

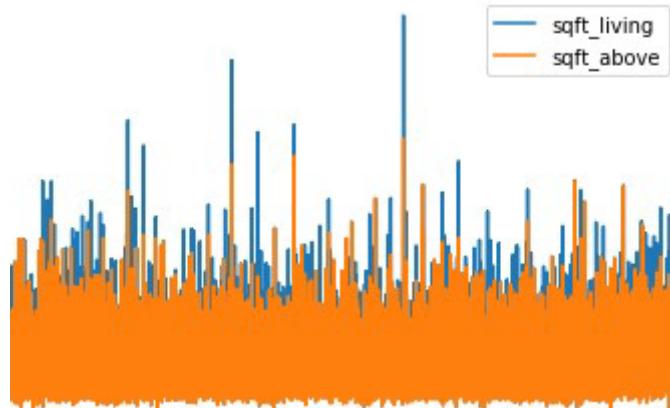
QUADRO 68 - PLOTAGEM DO DATAFRAME COM SELEÇÃO DE DADOS

```
dados["sqft_living"].plot()
col = ["sqft_living", "sqft_above"]
dados[col].plot()
```

FONTE: O autor

É possível notar que com dados em escala próxima se pode visualizar e analisar o que está sendo visto. Esta é apenas uma visão geral sobre a plotagem e no futuro será visto uma biblioteca em particular para plotar qualquer tipo de dado, não somente DataFrame.

FIGURA 8 - PLOTAGEM DO DATAFRAME COM SELEÇÃO DE DADOS



FONTE: O autor

No que se refere a recuperação dos dados ainda é possível realizar consultas em um DataFrame através do método `query()`. Este método permite explorar as colunas através de operadores condicionais. Vejamos alguns exemplos.

QUADRO 69 - UTILIZANDO QUERY EM DATAFRAMES

```
dados.head()
dados.query("bedrooms==3")
dados.query("floors!=1")
dados.query("sqft_living>2000")
dados.query("grade<=7")
```

FONTE: O autor

Agora que realizamos a manipulação de um documento CSV, para salvar o DataFrame no formato csv basta utilizar o comando `dados.to_csv('/content/drive/My Drive/saida.csv')`.

Finalizando nosso estudo sobre o Pandas, fique com uma Leitura Complementar sobre alguns comandos dessa incrível biblioteca de manipulação de dados.

LEITURA COMPLEMENTAR

28 comandos úteis de Pandas que talvez você não conheça

Paulo Vasconcellos

Pandas é a minha biblioteca favorita do Python. Seja para Data Visualization ou para Data Analysis, a praticidade e funcionalidade que essa ferramenta oferece não é encontrada em nenhum outro módulo. Quando comecei a usá-lo, não tinha conhecimento das variadas funções que Pandas oferece para resolver diversas tarefas, o que me fazia criar uma série de loop fore while para resolver o problema. Hoje, mais maduro no uso do “Pandão”, decidi trazer uma lista dos meus comandos favoritos. Confira:

Importando o Pandas por convenção

Aqui vai uma informação que talvez você já conheça: sempre que importar o Pandas, utilize a regra de convenção. Isso faz com que pessoas que lerem seu código no futuro — incluindo você mesmo — possa identificar a biblioteca mais facilmente. Por regra, Pandas deve ser importado sob o nome de pd, assim:

```
import pandas as pd
```

Series e DataFrame

Posso estar sendo meio óbvio falando sobre Series e DataFrame para alguém que já está acostumado a usar o Pandão, mas quero deixar claro para aqueles que estão começando a principal diferença entre esses dois tipos de Estrutura de Dados.

- **Series** nada mais é que um array de 1 dimensão. Você pode considerar um Series também como uma coluna de uma tabela. Exemplo:

```
>>> s = pd.Series([3, -5, 7, 4], index=['a', 'b', 'c', 'd'])
```

INDEX

INDEX	
A	3
B	-5
C	7
D	4

Saída do código acima: um array de valores indexados

- Um **DataFrame** é simplesmente um conjunto de Series. Trata-se de uma estrutura de dados de 2 dimensões — colunas e linhas — que transforma os dados em uma bela tabela. Exemplo:

```
#Criando um dicionário onde cada chave será uma coluna do
DataFrame >>> data = {
'País': ['Bélgica', 'Índia', 'Brasil'],
'Capital': ['Bruxelas', 'Nova Delhi', 'Brasília'],
'População': [123465, 456789, 987654]
}
#Criando o DataFrame
>>> df = pd.DataFrame(data, columns=['País','Capital','População'])
```

INDEX	País	Capital	População
1	Bélgica	Bruxelas	123465
2	Índia	Nova <u>De</u> lhi	456789
3	Brasil	Brasília	987654

Saída do código: um lindo DataFrame

Agora sim! Confira os melhores comandos para utilizar no Pandas

- Abrindo e escrevendo arquivos CSV:

```
#Para ler arquivos CSV codificados em ISO
>>> pd.read_csv('nome_do_arquivo.csv', encoding='ISO-8859-1')
#Para escrever arquivos CSV
>>> pd.to_csv('nome_do_arquivo_para_salvar.csv')
```

- Abrindo arquivos de Excel:

```
>>>      xlsx      =      pd.ExcelFile('seu_arquivo_excel.xlsx')
>>> df = pd.read_excel(xlsx, 'Planilha 1')
```

- Removendo linhas e colunas:

```
#Removendo linhas pelo index
s.drop([0, 1])
#Removendo colunas utilizando o argumento axis=1
df.drop('País', axis=1)
```

- Coletando informações básicas sobre o DataFrame:

```
#Quantidade de linhas e colunas do DataFrame
>>> df.shape
#Descrição do Index
>>> df.index
#Colunas presentes no DataFrame
>>> df.columns
#Contagem de dados não-nulos
>>> df.count()
```

- Criando uma nova coluna em um DataFrame:

```
>>> df['Nova Coluna'] = 0
```

- Renomeando colunas de um DataFrame:

#Se seu DataFrame possui 3 colunas, passe 3 novos valores em uma lista
df.columns = ['Coluna 1', 'Coluna 2', 'Coluna 3']

- Resumo dos dados:

#Soma dos valores de um DataFrame

```
>>> df.sum()
```

#Menor valor de um DataFrame

```
>>> df.min()
```

#Maior valor

```
>>> df.max()
```

#Index do menor valor

```
>>> df.idxmin()
```

#Index do maior valor

```
>>> df.idxmax()
```

#Resumo estatístico do DataFrame, com quartis, mediana, etc.

```
>>> df.describe()
```

#Média dos valores

```
>>> df.mean()
```

#Mediana dos valores

```
>>> df.median()
```

- Aplicando funções:

#Aplicando uma função que substitui a por b

```
df.apply(lambda x: x.replace('a', 'b'))
```

- Ordenando valores:

#Ordenando em ordem crescente

```
df.sort_values()
```

#Ordenando em ordem decrescente

```
df.sort_values(ascending=False)
```

- Operações aritméticas em Series:

```
>>> s = pd.Series([1, 2, 3, 4, 5], index=['a', 'b', 'c', 'd', 'e'])
```

#Somando todos os valores presentes na Series por 2

```
>>> s.add(2)
```

#Subtraindo 2 de todos os valores

```
>>> s.sub(2)
```

#Multiplicando todos os valores por 2

```
>>> s.mul(2)
```

#Dividindo valores por 2

```
>>> s.div(2)
```

- Indexação por Boolean:

#Filtrando o DataFrame para mostrar apenas valores pares
df[df['População'] % 2 == 0]

- Selecionando valores:

#Selecionando a primeira linha da coluna país
df.loc[0, 'País']

E aí, gostou dos códigos? Tinha algum que não conhecia? Deixa nos comentários sua sugestão de novos comandos para atualizarmos aqui. Ah, e não deixa de seguir o blog caso queira saber mais conteúdos sobre Pandas e Data Science. Até mais e obrigado pelos peixes!

FONTE: <<https://paulovasconcellos.com.br/28-comandos-%C3%BAteis-de-Pandas-que-talvez-voc%C3%AA-n%C3%A3o-conhe%C3%A7a-6ab64beefa93>>. Acesso em: 20 nov. 2019.

RESUMO DO TÓPICO 3

Neste tópico, você aprendeu que:

- Com o PIP é possível instalar o Pandas.
- A estrutura simples do Pandas é o conceito de séries.
- É possível criar séries a partir dos mais diversos tipos de dados.
- Os DataFrames são um tipo complexo de dados similares a tabelas.
- Os DataFrames permitem a realização de consultas.
- Se pode plotar gráficos a partir de um DataFrame.



Ficou alguma dúvida? Construímos uma trilha de aprendizagem pensando em facilitar sua compreensão. Acesse o QR Code, que levará ao AVA, e veja as novidades que preparamos para seu estudo.



AUTOATIVIDADE



- 1 Carregue um arquivo csv e transforme-o em um Dataframe:
- 2 Crie um DataFrame baseado em um dicionário:
- 3 Salve o DataFrame em um arquivo externo:

UNIDADE 3

CONCEITOS AVANÇADOS DE PROGRAMAÇÃO EM PYTHON

OBJETIVOS DE APRENDIZAGEM

A partir desta unidade, você deverá ser capaz de:

- aprender recursos para otimização de código;
- conhecer o conceito de reúso de código;
- aprender a programar orientado a objetos em Python;
- utilizar funções para simplificar a carga de trabalho;
- criar conexão com o banco de dados em nuvem;
- realizar operações com Python e banco de dados;
- plotar gráficos com Matplotlib;
- conhecer mais sobre as bibliotecas do Python.

PLANO DE ESTUDOS

Esta unidade está dividida em três tópicos. No decorrer da unidade você encontrará autoatividades com o objetivo de reforçar o conteúdo apresentado.

TÓPICO 1 – TÉCNICAS DE PROGRAMAÇÃO EM PYTHON

TÓPICO 2 – CONEXÃO COM BANCO DE DADOS

TÓPICO 3 – BIBLIOTECAS COMPLEMENTARES



Preparado para ampliar seus conhecimentos? Respire e vamos em frente! Procure um ambiente que facilite a concentração, assim absorverá melhor as informações.

TÉCNICAS DE PROGRAMAÇÃO EM PYTHON

1 INTRODUÇÃO

O desenvolvimento de sistemas é um processo complexo, envolve diversas técnicas e áreas da ciência da computação. Nossa livro tem como foco a programação para Big Data, mas, em muitos momentos, recursos são utilizados em ambos os cenários.

Tendo em vista facilitar o desenvolvimento de código, iniciaremos nosso estudo abordando o conceito de funções. Uma função é um tipo de procedimento ou rotina. Algumas linguagens de programação fazem uma distinção entre uma função, que retorna um valor, e um procedimento, que executa alguma operação, mas não retorna um valor.

Em Python você aprenderá que a criação de funções, assim como outros recursos da linguagem, funciona de maneira dinâmica.

2 FUNÇÕES EM PYTHON

No caso do Python, como sabemos, ele tem um viés dinâmico em seus recursos, deste modo as funções podem variar de comportamento. É perfeitamente possível escrever programas usando os tipos de dados e estruturas de controle que abordamos nas partes anteriores. No entanto, muitas vezes queremos fazer essencialmente o mesmo processamento repetidamente, mas com uma pequena diferença, com um valor inicial diferente.

Segundo Summerfield (2010), o Python fornece um meio de encapsular suítes como funções que podem ser parametrizadas pelos argumentos a serem transmitidos. Um exemplo genérico da criação de uma função pode ser visto a seguir:

QUADRO 1 - SINTAXE DE FUNÇÕES NO PYTHON

```
def minhafuncao(entrada):
    saida = entrada
    return saida

print(minhafuncao("a"))
print(minhafuncao(100))
print(minhafuncao(True))
```

FONTE: O autor

Na função criada no Quadro 1 podemos notar o recebimento do argumento entrada entre parênteses e retorno da função por meio do comando `return`. Podemos notar que a função se comporta de maneira dinâmica em relação ao tipo de dado recebido, bem como com o tipo de dado retornado pela função.

Dependendo de como for codificada, uma função pode tratar o tipo de dados recebido e seu retorno com o já visto comando `type` que retorna o tipo de dados.

Complementarmente, uma maneira mais correta do ponto de vista de boas práticas de codificação, é possível explicitar o tipo de dados de uma função. Vejamos um exemplo de uma função que recebe uma lista, uma posição, e retorna o elemento referente à posição nessa lista.

QUADRO 2 - DEFININDO TIPOS DE DADOS EM FUNÇÕES

```
lista = [10, 20, 30, 40, 50]
def retorna_item(l: list, index: int) -> int:
    return l[index]

#passa os parametros conforme o esperado
retorna_item(lista,2)

#retorna_item(lista,2.5)

#retorna_item(1,2)
```

FONTE: O autor

Para compreender melhor o código do Quadro 2, você pode executar os códigos que estão em comentário. Com isso mensagens de erro serão mostradas indicando que o tipo de dado passado é diferente do tipo de dado definido pela função.



O código FONTE: do conteúdo desenvolvido neste tópico está disponível em:
https://colab.research.google.com/drive/1LFgSzXW_juDYvTxWMkRZ7KNHfN3KDj81.

Outro procedimento comum durante a criação de funções é a definição de valores padrão para os argumentos passados para uma função. Ao deixar um argumento como padrão, será esse o valor utilizado caso a função não receba o parâmetro.

QUADRO 3 - CRIANDO FUNÇÕES COM ARGUMENTOS PADRÃO

```
def escreve(nome, idade=0):
    print("Nome:", nome)
    print("Idade", idade)

escreve("João")
escreve("Felipe", 19)
```

FONTE: O autor

No universo de Big Data é muito comum uma etapa de experimentação, na qual avaliamos os métodos de *machine learning*, desenvolvemos algoritmos, avaliamos técnicas de pré-processamento.

Nessa etapa, o emprego de funções é muito comum, através de programação procedural. Neste contexto, o principal emprego de funções será na manipulação de dados. para isso vamos ver alguns exemplos de manipulação utilizando funções.

Exemplo1: crie uma função que receba um objeto qualquer (lista, array ou DataFrame), percorra os elementos escrevendo seus itens na tela e escreva o tipo do objeto.

QUADRO 4 - EXEMPLO 1, TÓPICO 1, UNIDADE 3

```
import numpy as np
import random
import pandas as pd

#cria a função
def percorre(objeto):
    #escreve o tipo do objeto

    print("O objeto é do tipo:", type(objeto))

    print("O objeto contém os seguintes elementos:")
    for item in objeto:
        print(item)

#cria objetos com dados aleatórios
df = pd.DataFrame(np.random.randint(0,100,size=(100, 1)), columns=list('A'))
lst = random.sample(range(0, 100), 10)
arr = np.random.randint(1,101,10)

#chama a função
percorre(df['A'])
percorre(arr)
percorre(lst)
```

FONTE: O autor

Exemplo 2: escreva uma função para imprimir uma caixa de estrelas, como a apresentada na sequência, em vários pontos.

```
*****
*      *
*      *
*****
```

QUADRO 5 - EXEMPLO 2, TÓPICO 1, UNIDADE 3

```
def plotar():
    print('*' * 15)
    print('*', '*'*11, '*')
    print('*', '*'*11, '*')
    print('*' * 15)
plotar()
```

FONTE: O autor

Exemplo 3: crie uma função que permita inserir conteúdo dentro de um DataFrame a partir de outro DataFrame e de uma posição específica.

QUADRO 6 - EXEMPLO 3, TÓPICO 1, UNIDADE 3

```
import pandas as pd
import numpy as np

def inserir(idx, df, df2):
    dfA = df.iloc[:idx, ]
    dfB = df.iloc[idx:, ]

    df = dfA.append(df2).append(dfB).reset_index(drop = True)

    return df

#cria um primeiro Dataframe
df = pd.DataFrame(np.random.randn(100, 4), columns=list('ABCD'))
print(df)

#cria um segundo Dataframe
df2 = pd.DataFrame(np.random.randn(5, 4), columns=list('ABCD'))
print(df2)

#Utiliza a função
df = inserir_linha(1, df, df2)

print(df)
```

FONTE: O autor

A partir de agora as funções farão parte do nosso curso e sempre que necessário iremos utilizá-las. O principal objetivo da criação de funções é o reuso de código, ou seja, não ter que fazer a mesma coisa em vários lugares. Mas podemos citar outros benefícios:

- Código mais limpo e de fácil organização.
- Documentar ações complexas.
- Em cenários de big data, implementar e documentar métodos matemáticos.
- Evitar que blocos de códigos fiquem muito extensos.

3 ORIENTAÇÃO A OBJETOS EM PYTHON

Desde que iniciamos nosso livro sempre nos referimos aos elementos do Python com objetos, isto porque o Python é uma linguagem de programação dita orientada a objetos.



O fato de uma linguagem de programação ser orientada a objetos não significa que ao utilizá-la você está programando utilizando o paradigma ou um projeto orientado a objetos. Neste livro, por exemplo, todos os exemplos feitos até agora utilizam o paradigma procedural/estruturada, ou seja, um código sequencial auxiliado por funções.

A orientação a objetos é um paradigma de programação empregado por diversas linguagens tradicionais C, C#, Java, PHP, bem como no Python.

A programação orientada a objetos tem como principal objetivo a reutilização de código FONTE: em projetos de sistemas. A reutilização de software é o processo de criar sistemas de software utilizando recursos existentes, em vez de criar sistemas de software a partir do zero (KRUEGER, 1992).

Santachè (2015) elenca os seguintes pontos positivos de se reutilizar o código:

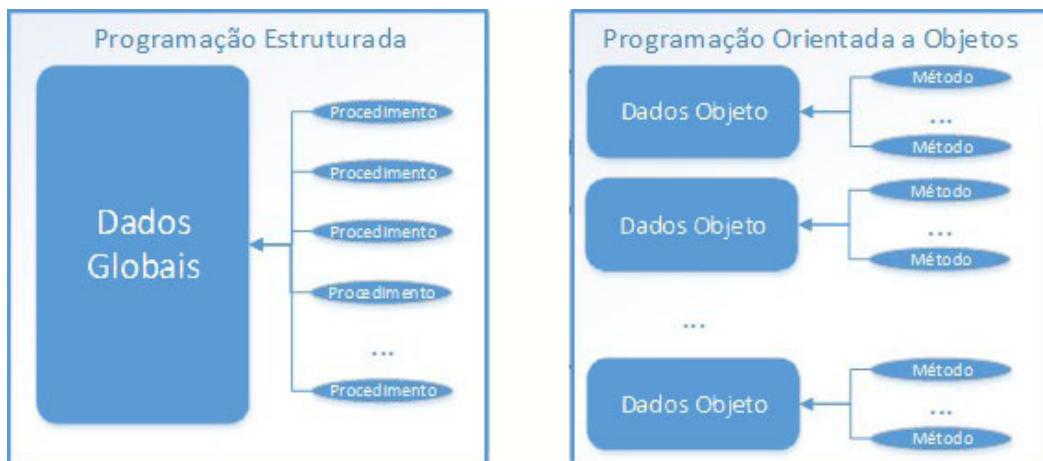
- Aumento de produtividade.
- Redução de custos de desenvolvimento e manutenção.
- Reuso pode promover sucessivas melhorias no produto.
- Frequência de reuso pode indicar qualidade.

Para um dos principais autores da área de engenharia de software, Sommerville (2011), a análise orientada a objetos concentra-se no desenvolvimento de um modelo orientado a objetos do domínio da aplicação. Os objetos nesse modelo refletem as entidades e as operações associadas ao problema a ser resolvido.

Segundo Gasparotto (2019, s.p.):

[...]no paradigma estruturado, temos procedimentos (ou funções), que são aplicados globalmente em nossa aplicação. No caso da orientação a objetos, temos métodos que são aplicados aos dados de cada objeto. Essencialmente, os procedimentos e métodos são iguais, sendo diferenciados apenas pelo seu escopo.

FIGURA 1 - PROGRAMAÇÃO ESTRUTURADA VS PROGRAMAÇÃO ORIENTADA A OBJETOS



FONTE: <<http://bit.ly/2TaCfZ9>>. Acesso em: 11 dez. 2019.

Um dos grandes diferenciais da programação orientada a objetos em relação a outros paradigmas de programação que também permitem a definição de estruturas e operações sobre essas estruturas está no conceito de herança, mecanismo através do qual definições existentes podem ser facilmente estendidas. Juntamente com a herança deve ser enfatizada a importância do polimorfismo, que permite selecionar funcionalidades que um programa irá utilizar de forma dinâmica, durante sua execução (RICARTE, 2001, p. 3).

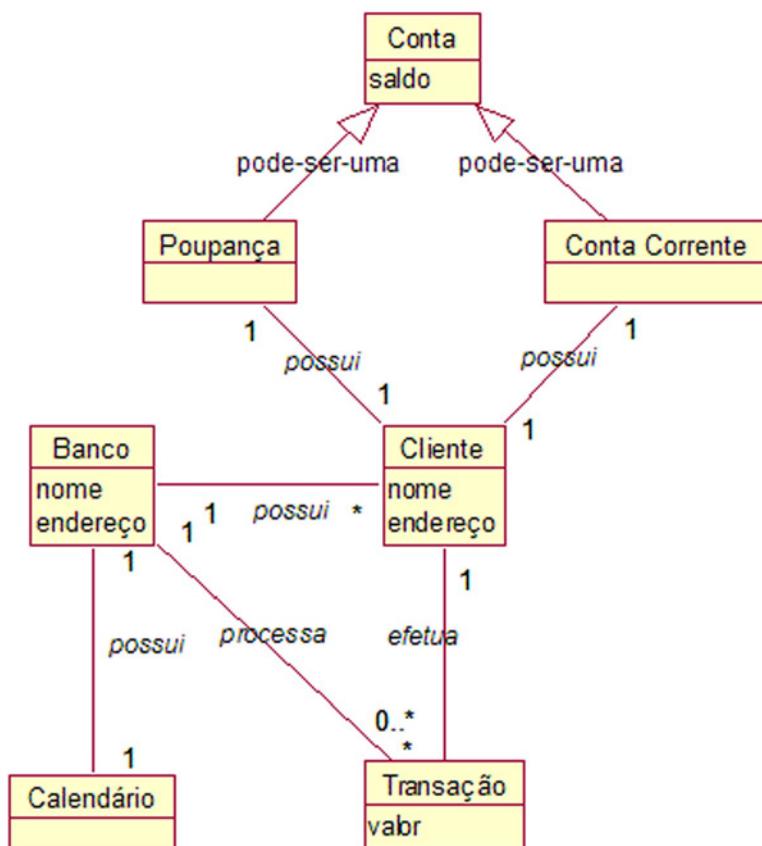
Tanto do ponto de vista de projeto quanto do ponto de vista prático da programação orientada a objetos existem dois elementos principais: classes e objetos.

Uma classe é um roteiro para se definir objetos, nela o código é estruturado com as principais características e comportamentos de um objeto (métodos e atributos). A classe geralmente é projetada por um diagrama (de classes) durante uma etapa da modelagem conceitual de sistemas denominada UML (Unified Modeling Language). De modo geral, uma classe pode ser associada com uma receita de bolo.

Uma classe possui dois elementos principais: métodos e atributos. Que, segundo Guilherme Somera (2006, p. 13), podem ser definidos como:

- Atributos: O conjunto de atributos descreve as propriedades da classe. Cada atributo é identificado por um nome e tem um tipo associado. Em uma linguagem de programação orientada a objetos pura, o tipo é o nome de uma classe. Na prática, a maior parte das linguagens de programação orientada a objetos oferecem um grupo de tipos primitivos, como inteiro, real e caráter, que podem ser usados na descrição de atributos. O atributo pode ainda ter um valor_default opcional, que especifica um valor inicial para o atributo.
- Métodos: Os métodos definem as funcionalidades da classe, ou seja, o que será possível fazer com objetos dessa classe. Cada método é especificado por uma assinatura, composta por um identificador para o método (o nome do método), o tipo para o valor de retorno e sua lista de argumentos, sendo cada argumento identificado por seu tipo e nome.

FIGURA 2 - EXEMPLO DE DIAGRAMA DE CLASSES

FONTE: <<http://bit.ly/2TaDFTt>>. Acesso em: 26 nov. 2019.



Vale lembrar que nosso objetivo neste Livro Didático é aprender conceitos da linguagem Python com ênfase em aplicações de Big Data. Para maiores informações sobre modelagem UML, bem como conceitos avançados de modelagem de classes, deixamos dois livros que são referência nesta área como dica de leitura.

- BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML**: guia do usuário. Amsterdã: Elsevier, 2006.
- SOMMERVILLE, Ian. **Software engineering**. Boston: Addison-Wesley/Pearson, 2011.

Do ponto de vista prático, ao criar uma classe em sua estrutura pode ser muito similar aos conceitos já vistos. Os atributos são criados como variáveis e os métodos como funções. O método de inicialização do Python é o mesmo que qualquer outro método, exceto que ele possui um nome especial: `__init__`. Os dois sublinhados iniciais e finais significam “este é um método especial que o intérprete Python tratará como um caso especial”. “Nunca nomeie uma função própria com sublinhados duplos à esquerda e à direita” (PHILLIPS, 2015, p. 39, tradução nossa). Vejamos um exemplo de criação de uma classe em Python.

QUADRO 7 - EXEMPLO DE CRIAÇÃO DE CLASSE

```
class MinhaClasse:

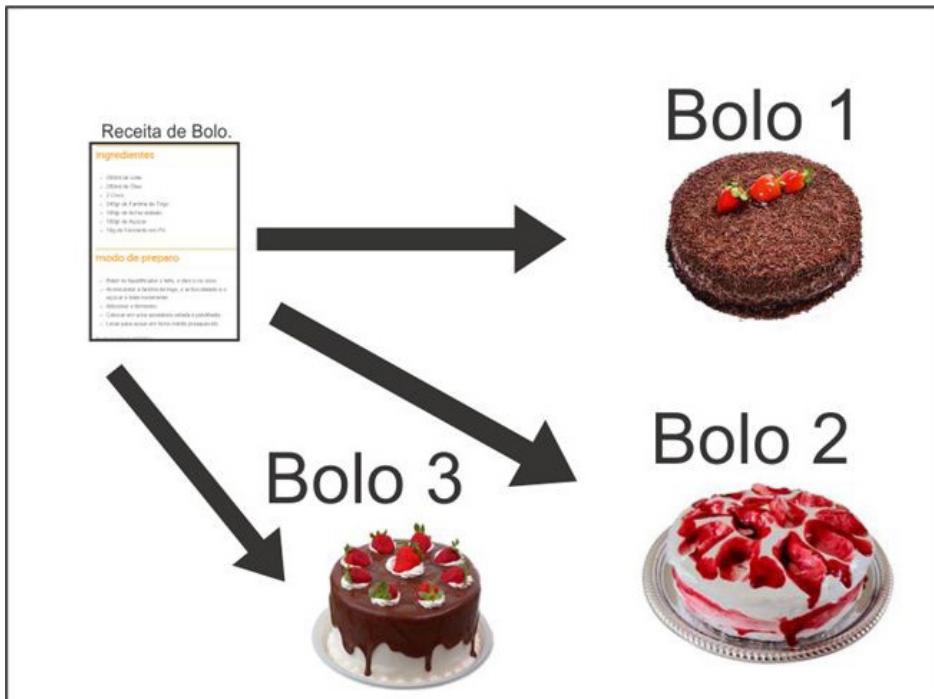
    def __init__(self, a, b):
        self.a = a
        self.b = b

    def soma(self):
        return self.a + self.b
```

FONTE: O autor

Os objetos são compreendidos nas instâncias das classes. Ou seja, tudo que for definido durante a criação de uma classe o objeto terá. Na associação com a receita de bolo, um objeto é um bolo feito a partir de uma receita (classe).

FIGURA 3 - CLASSES E OBJETOS – RECEITA DE BOLO



FONTE: <<http://bit.ly/303TI72>>. Acesso em: 26 nov. 2019.

No processo de programação orientada a objetos o termo instanciar se refere ao momento que um objeto é criado. Ou seja, que uma instância de uma classe é criada. Vamos instanciar um objeto da classe criado anteriormente.

QUADRO 8 - INSTANCIANDO UM OBJETO

```
obj = MinhaClasse(5, 10)
print(obj.soma())
```

FONTE: O autor

Observe que a partir de agora a variável obj se tornou um objeto da classe MinhaClasse(). A partir de agora todos os métodos dessa classe serão acessados pelo objeto.

Segundo Gasparotto (2019, s.p.): “O encapsulamento é uma das principais técnicas que define a programação orientada a objetos. Trata-se de um dos elementos que adicionam segurança à aplicação em uma programação orientada a objetos pelo fato de esconder as propriedades, criando uma espécie de caixa preta”.

No Python, o encapsulamento acontece por intermédio do conceito de público e privado. Conceitualmente, tanto atributos quanto métodos podem ter essas características. Um método público de uma classe pode ser acessado por outras classes, enquanto um método privado somente pela classe de origem.

QUADRO 9 - ENCAPSULAMENTO EM PYTHON

```
class A:
    a = 1 # atributo publico
    __b = 2 # atributo privado a class A

class B(A):
    __c = 3 # atributo privado a B

    def __init__(self):
        print self.a
        print self.__c

a = A()
print a.a # imprime 1

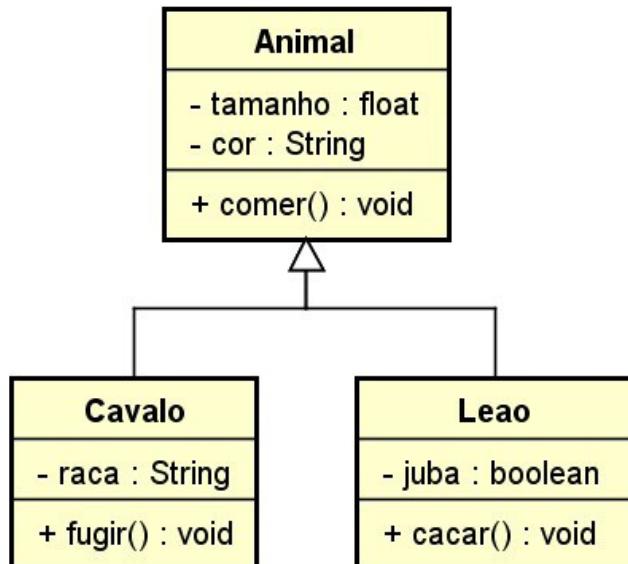
b = B()
print b.__b # Erro, pois __b é privado a classe A.
print b.__c # Erro, __c é um atributo privado, somente chamado pela classe.

print b._B__c # Imprime __c = 3, muito pouco utilizada, mas existe.
```

FONTE: <<http://bit.ly/2R2nKUB>>. Acesso em: 27. nov. 2019.

Um aspecto importante no que se refere ao aproveitamento de código fonte na programação orientada a objetos é o conceito de herança. De modo geral, este conceito, uma classe é criada com base em outra classe, ou seja, herda da outra classe.

FIGURA 4 - HERANÇA EM ORIENTAÇÃO A OBJETOS



FONTE: <<http://bit.ly/2QGAoKl>>. Acesso em: 27 nov. 2019.

Quando você faz isso, a nova classe obtém todas as variáveis e métodos da classe da qual está herdando (chamada de classe base). Em seguida, ele pode definir variáveis e métodos adicionais que não estão presentes na classe base e pode substituir alguns dos métodos da classe base. Ou seja, ele pode reescrevê-los para se adequar a seus próprios propósitos (HEINOLD, 2012, p. 132, tradução nossa).

Ao declarar uma classe que irá herdar outra em Python, basta passar o nome da classe que será herdada entre parênteses. Vejamos um exemplo de como funciona.

QUADRO 10 - HERANÇA EM PYTHON

```
#a classe principal é criada
class Pai:
    def __init__(self, a):
        self.a = a
    def metodo1(self):
        print(self.a*2)
    def metodo2(self):
        print(self.a+'!!!')

class Filho(Pai):
    #a inicialização é alterada para receber mais um parametro
    def __init__(self, a, b):
        self.a = a
        self.b = b
    #o método 1 é sobreescrito (override)
    def metodo1(self):
        print(self.a*7)
    #o método 1 é sobreescrito (override)
    def metodo3(self):
        print(self.a + self.b)

p = Pai('Olá')
c = Filho('Olá', 'Adeus')
print('Método 1 do Pai: ', p.metodo1())
print('Método 2 do Pai: ', p.metodo2())

print('Método 1 do Filho: ', c.metodo1())
print('Método 2 do Filho: ', c.metodo2())
print('Método 3 do Filho: ', c.metodo3())
```

FONTE: O autor

No exemplo anterior podemos absorver dois novos conteúdos de orientação a objetos. Um deles é denominado polimorfismo. Polimorfismo se refere ao fato de um método ter o mesmo nome, mas conter comportamentos diferentes em classes diferentes.

No momento em que na classe Filho() alteramos o método1 e método3 herdados da Classe pai realizamos um procedimento denominado sobreescriver (*overwrite*). O mesmo aconteceu com a inicialização, no qual alteramos os parâmetros de entrada.

Agora que aprendemos um pouco sobre como a orientação a objetos em Python, vamos colocar em prática com alguns exemplos.

Exemplo 1: crie uma classe Pets que contém instâncias de cães; essa classe é completamente separada da classe Dog.

QUADRO 11 - EXEMPLO 4, TÓPICO 1, UNIDADE 3

```

class Pets:

    dogs = []

    def __init__(self, dogs):
        self.dogs = dogs

class Dog:

    species = 'mamífero'

    def __init__(self, name, age):
        self.name = name
        self.age = age

    def description(self):
        return self.name, self.age

    def speak(self, sound):
        return "%s fala %s" % (self.name, sound)

    def eat(self):
        self.is_hungry = False

class RussellTerrier(Dog):

    def run(self, speed):
        return "%s rucorrens %s" % (self.name, speed)

class Bulldog(Dog):

    def run(self, speed):
        return "%s corre %s" % (self.name, speed)

# Cria uma instância de Dogs
my_dogs = [
    Bulldog("Tom", 6),
    RussellTerrier("Fletcher", 7),
    Dog("Larry", 9)
]

# Inicializa a Classe Pets
my_pets = Pets(my_dogs)

# Saída
print("Eu tenho {} cachorros.".format(len(my_pets.dogs)))
for dog in my_pets.dogs:
    print("{} tem {}".format(dog.name, dog.age))

print("Eles são todos {}".format(dog.species))

```

FONTE: O autor

Exemplo 2: neste exemplo faremos um jogo da velha orientado a objetos proposto por Heinold (2012, p. 136, tradução nossa):

A classe contém duas variáveis, um número inteiro representando o jogador atual e uma lista 3×3 representando o quadro. [...] Existem quatro métodos:

- `get_open_spots` - retorna uma lista dos lugares no tabuleiro que ainda não foram marcados pelos jogadores.
- `is_valid_move` - pega uma linha e uma coluna representando uma movimentação em potencial e retorna True se a movimentação for permitida e False caso contrário.
- `make_move` - pega uma linha e uma coluna representando uma jogada em potencial, chama `is_valid_move` para ver se a jogada está boa e, se estiver, define a matriz do tabuleiro de acordo e muda o jogador.
- `check_for_winner` - varre a lista do tabuleiro e retorna 1 se o jogador 1 venceu, 2 se o jogador 2 venceu, 0 se não houver jogadas restantes e nenhum vencedor e -1 se o jogo continuar.

QUADRO 12 - EXEMPLO 5, TÓPICO 1, UNIDADE 3

```
def __init__(self):
    self.B = [[0,0,0],
              [0,0,0],
              [0,0,0]]
    self.player = 1

def get_open_spots(self):
    return [[r,c] for r in range(3) for c in range(3)
           if self.B[r][c]==0]

def is_valid_move(self,r,c):
    if 0<=r<=2 and 0<=c<=2 and self.B[r][c]==0:
        return True
    return False

def make_move(self,r,c):
    if self.is_valid_move(r,c):
        self.B[r][c] = self.player
        self.player = (self.player+2)%2 + 1
    def check_for_winner(self):
        for c in range(3):
            if self.B[0][c]==self.B[1][c]==self.B[2][c]!=0:
                return self.B[0][c]
        for r in range(3):
            if self.B[r][0]==self.B[r][1]==self.B[r][2]!=0:
                return self.B[r][0]
        if self.B[0][0]==self.B[1][1]==self.B[2][2]!=0:
            return self.B[0][0]
```

```

if self.B[2][0]==self.B[1][1]==self.B[0][2]!=0:
    return self.B[2][0]
if self.get_open_spots()==[]:
    return 0
return -1
def print_board():
    chars = ['-','X','O']
    for r in range(3):
        for c in range(3):
            print(chars[game.B[r][c]], end=' ')
    print()
game = tic_tac_toe()
while game.check_for_winner() == -1:
    print_board()
    r,c = eval(input('Entre com a posição, jogador ' + str(game.player) + ': '))
    game.make_move(r,c)
print_B()
x = game.check_for_winner()
if x==0:
    print("Esta plotado na tela")
else:
    print('Jogador', x, 'venceu!')

```

FONTE: Adaptado de Heinold (2016)

Para finalizar nossa abordagem sobre orientação a objetos no Python vamos deixar algumas dicas fornecidas por Dan Bader (2019) sobre orientação a objetos com Python.



- **Tutorial de Programação Orientada a Objetos em Python 3:** neste tutorial do Real Python, você aprenderá os fundamentos da programação orientada a objetos (OOP) no Python e como trabalhar com classes, objetos e construtores. O tutorial também vem com vários exercícios de POO para revisar seu progresso de aprendizado.
 - Disponível em: <http://bit.ly/2ug1u1X>
- **Documentação Oficial:** esta é uma introdução muito boa à mecânica básica de classes e Programação Orientada a Objetos em Python. O tutorial foi aprimorado ao longo dos anos e vale a pena ler se você ainda não o viu.
 - Disponível em: <https://docs.python.org/3/tutorial/classes.html>.
- **Repositório de Padrões no GitHub:** uma coleção de padrões e expressões comuns de design de Programação Orientada a Objetos em Python. Os exemplos são puramente baseados em códigos e há poucas explicações ou informações básicas. No entanto, em

alguns casos, apenas ver um exemplo de implementação mínima pode ser útil.

o Disponível em: <http://www.github.com/faif/python-patterns>.

• **Padrões de Projeto em Python:** este é um bom tutorial introdutório aos padrões de projeto orientados a objetos no Python. Esse documento ressalta a maneira simples de implementação de padrões em Python.

o Disponível em: [toptal.com/python/python-design-patterns](https://www.toptal.com/python/python-design-patterns).

RESUMO DO TÓPICO 1

Neste tópico, você aprendeu que:

- No Python as funções trabalham de maneira dinâmica;
- As funções em Python podem ou não receber parâmetros;
- Em Python uma função pode ou não ter um retorno;
- Python é uma linguagem orientada a objetos.

AUTOATIVIDADE



- 1 Crie uma função que receba três números e retorne a soma deles.
- 2 Escreva uma função que verifique se uma palavra é um palíndromo.
- 3 A orientação a objetos é um recurso das linguagens de programação que permite o aproveitamento de códigos. Dentre diversos recursos há uma funcionalidade que permite que classes obtenham métodos e atributos de outras. Selecione a alternativa CORRETA sobre qual é este recurso:
 - a) () Automação.
 - b) () Doação.
 - c) () Objeto.
 - d) () Herança.
- 4 Crie uma classe animal sendo principal e filhos que herdam suas características.
- 5 Baseado nas classes criadas no exercício anterior crie as instâncias (objetos) para cada uma das classes.

CONEXÃO COM BANCO DE DADOS

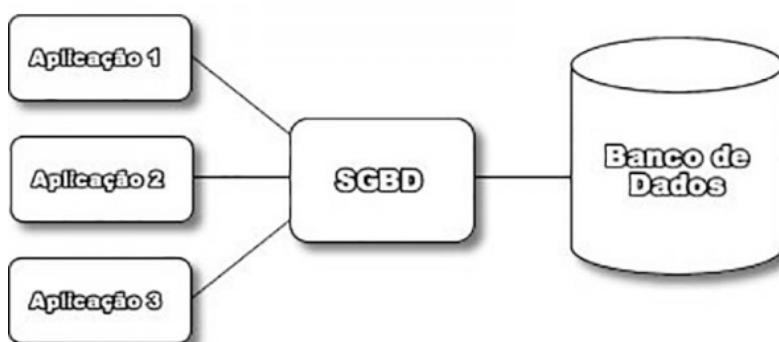
1 INTRODUÇÃO

Durante as unidades anteriores aos poucos fomos manipulando arquivos externos com a finalidade de compreender a interação entre as estruturas de dados do Python. Em muitos casos, principalmente na etapa de experimentação, é comum que sejam utilizados dados em arquivos (.txt ou .csv) a fim de conhecer datasets específicos.

No entanto, haverá momentos que sistemas mais complexos serão construídos, e para isso serão utilizados os sistemas gerenciadores de banco de dados com o objetivo de obter e armazenar os dados com maior integridade.

Um SGBD (Sistema Gerenciador de Banco de Dados) é um software que faz a interface entre usuários e programas com os dados em si. Tais programas são responsáveis por encapsular os dados de tal modo que não possam ser acessados diretamente, garantindo a segurança e integridade dos dados armazenados. Para fazer o acesso aos dados são utilizadas linguagens específicas do SGBD.

FIGURA 5 - ARQUITETURA DE UM SGBD



FONTE: <<http://bit.ly/36BPAxH>>. Acesso em: 27 nov. 2019.

O Python tem compatibilidade com a maioria dos SGBDs do mercado. Principalmente, pela facilidade de instalar novas bibliotecas com um comando é possível se conectar e manipular dados.



Nosso objetivo é mostrar a integração da linguagem Python com banco de dados. Os conceitos de banco de dados são bem mais profundos do que o visto aqui. Por isso fica a recomendação de dois livros e um artigo, caso deseje aprofundar seus conhecimentos sobre os conceitos de banco de dados.

- DATE, Christopher J. **Introdução a sistemas de bancos de dados**. Amsterdã: Elsevier, 2004.
- ELMASRI, Ramez et al. **Sistemas de banco de dados**. São Paulo: Pearson, 2005.
- NOGUEIRA, R. R. **Passo a passo para a modelagem de dados**. SQL Magazine, Rio de Janeiro, v. 1, p. 15-25, 2016.

Para dar início à etapa prática iremos instalar o PostgreSQL e realizar a integração com banco de dados.

2 UTILIZANDO POSTGRESQL COM PYTHON

O PostgreSQL é um sistema de gerenciamento de banco de dados relacional (SGBDR) gratuito e de código aberto, constantemente atualizado e com uma forte contribuição da comunidade.

Para obter e instalar o PostgreSQL basta fazer acesso ao instalador de acordo com o sistema operacional que esteja utilizando:

- Windows: instalador disponível em: <https://www.postgresql.org/download/windows>
- MacOs: instalador disponível em: <https://www.postgresql.org/download/macosx/>
- Linux: pode ser obtido através do instalador de pacotes pelo apt-get install postgresql. Ou se obter os arquivos binários em: <https://www.postgresql.org/download/linux>

Como estamos desenvolvendo nosso código utilizando o Google Colab, que é uma plataforma em nuvem, e para que possamos continuar assim, de tal forma que o conteúdo possa ser disponibilizado e utilizado por todos em qualquer dispositivo. Iremos utilizar uma plataforma que disponibilize o acesso ao PostgreSQL on-line.

Existe a possibilidade de criar um banco de dados no servidor onde estamos executando o Colab, através do <https://cloud.google.com/>. No entanto, apesar de existirem vouchers é uma ferramenta paga.

FIGURA 6 - CRIAÇÃO DE UMA INSTÂNCIA NO GOOGLE CLOUD

Teste o Google Cloud Platform gratuitamente

Etapa 1 de 2

País

Brasil

Termos de Serviço

Li e concordo com os [Termos de Serviço da avaliação gratuita do Google Cloud Platform](#)

Necessário para continuar

CONTINUAR

[Política de Privacidade](#) | [Perguntas frequentes](#)



FONTE: O autor

Visando possibilitar a criação de bancos de dados na nuvem, com uma opção de que todos possam acessar, iremos utilizar o projeto ElephantSQL. Uma plataforma on-line para criação e manutenção de bancos de dados PostgreSQL, a ferramenta pode ser acessada em: <https://www.elephantsql.com/>. Em sua opção FREE é possível criar um banco de dados de até 20 Mb de maneira gratuita sem a necessidade fornecer dados de cartão de crédito, sendo que essa será opção utilizada para construir os exemplos deste livro.

FIGURA 7 - ELEPHANTSQ

We are here to help you. Database administrator support is included in all paid plans.

Shared Instances

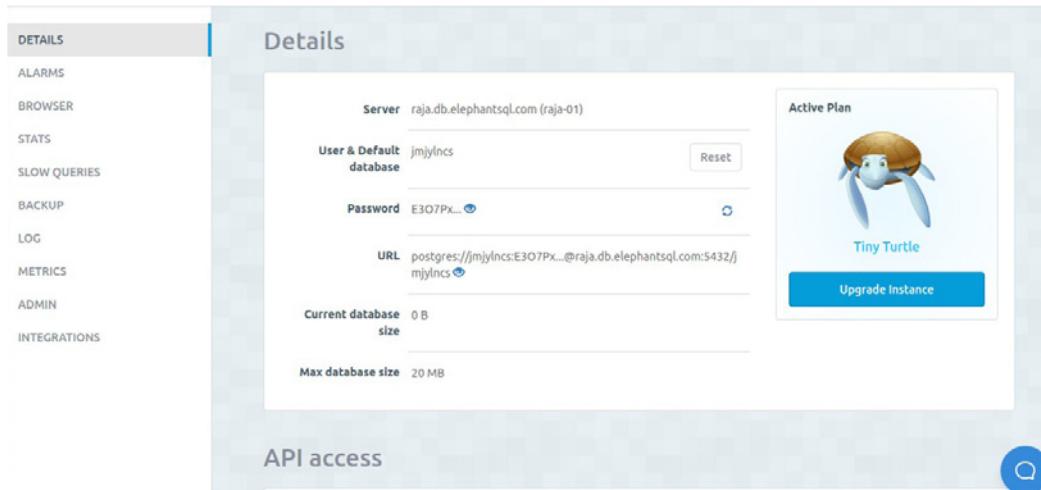
TINY TURTLE	SIMPLE SPIDER	CRAZY CAT	PRETTY PANDA
Shared high performance server			
20 MB data	500 MB data	1 GB data	2 GB data
5 concurrent connections	10 concurrent connections	15 concurrent connections	20 concurrent connections
FREE	\$ 5 PER MONTH	\$ 10 PER MONTH	\$ 19 PER MONTH
GET STARTED	Try now for \$5/month	Try now for \$10/month	Try now for \$19/month

FONTE: O autor

Durante o desenvolvimento dos exemplos, você poderá utilizar os dados disponibilizados no Colab deste tópico. Porém, recomendamos que crie sua própria conta para que possa executar o processo completo e, por consequência, aprender mais sobre a manipulação de dados.

Uma vez realizado o cadastro, a etapa mais importante é obter os dados de acesso ao servidor PostgreSQL. Estes dados estão disponíveis na aba details. Destes dados serão utilizados server, password e user.

FIGURA 8 - DADOS DE ACESSO ELEPHANTSQL



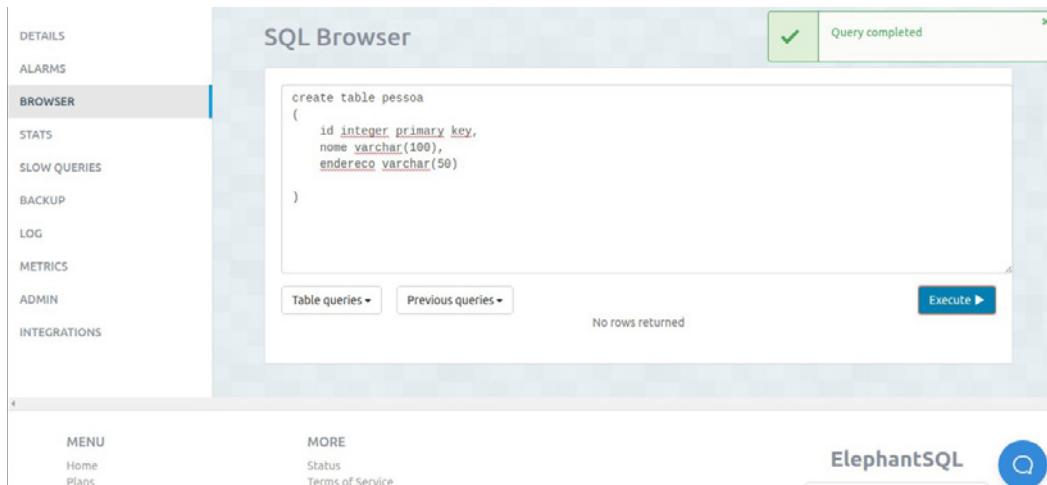
FONTE: O autor

O PostgreSQL é um SGBD, isto significa que ele é um servidor que armazena os dados. Tal servidor precisa ser acessado através de uma interface aos dados, com o usuário e senha.

Através do navegador é possível executar comandos de acesso aos dados pela aba Browser. Será nessa caixa de texto que executaremos os comandos de manipulação de dados, denominados SQL (*Structured Query Language* – Linguagem de Consulta Estruturada). É utilizando comandos SQL que criaremos a estrutura de um banco de dados, bem como faremos a manipulação dos dados armazenados.

No exemplo a seguir é executando um comando de criação de tabela. Na qual é criado uma tabela denominada pessoa com identificador (do tipo inteiro), nome (do tipo string) e endereço (do tipo string).

FIGURA 9 - EXECUÇÃO DE SQL



The screenshot shows the SQL Browser interface. On the left, there's a sidebar with tabs: DETAILS, ALARMS, BROWSER (which is selected), STATS, SLOW QUERIES, BACKUP, LOG, METRICS, ADMIN, and INTEGRATIONS. The main area is titled "SQL Browser" and contains a code editor with the following SQL script:

```
create table pessoa
(
    id integer primary key,
    nome varchar(100),
    endereco varchar(50)
)
```

Below the code editor, there are two buttons: "Table queries" and "Previous queries". To the right of the code editor, it says "No rows returned". At the bottom right of the main area is a blue "Execute" button with a play icon. In the top right corner of the main window, there's a green checkmark icon and the text "Query completed".

At the bottom of the interface, there's a "MENU" section with "Home" and "Plans" links, and a "MORE" section with "Status" and "Terms of Service" links. On the far right, there's a logo for "ElephantSQL" and a magnifying glass icon.

FONTE: O autor



Estamos trazendo uma visão prática da integração entre Python e o PostgreSQL. No decorrer do curso você terá conceitos avançados em bancos de dados. Complementarmente deixamos duas dicas de leitura sobre o PostgreSQL.

- MILANI, André. **PostgreSQL**: guia do programador. São Paulo: Novatec Editora, 2008.
- **Ferramentas para o PostgreSQL**: disponível no endereço https://wiki.postgresql.org/wiki/Ferramentas_para_o_PostgreSQL.

Agora que já temos o nosso banco de dados sendo executado em um servidor PostgreSQL iremos conectar com o código Python. O primeiro passo antes de iniciarmos é instalar a biblioteca necessária para conexão com PostgreSQL. Para isso é necessário executar pip3 install psycopg2.

Para conectar o servidor de banco de dados com o código Python é necessário saber os dados de acesso. Vejamos um exemplo de conexão.

QUADRO 13 - CONEXÃO DO PYTHON COM BANCO DE DADOS

```

import psycopg2 as ps

def conecta():
    #endereço do servidor
    host      = 'raja.db.elephantsql.com'

    #nome do banco criado, nesse caso é o mesmo nome do usuário
    bancodedados = 'jmjylncs'

    #porta da conexão, no PostgreSQL o padrão é 5432
    porta = '5432'

    #nome do usuário
    usuario = 'jmjylncs'

    #senha
    senha = 'E3O7Pxx_Vi6rmRix2yb-CJKD_t0TAtbO'

    try:
        conn=
        ps.connect(host=host,database=bancodedados,user=usuario,password=senha,port=porta)
        except ps.OperationalError as e:
            raise e
        else:
            print('Conectado com Sucesso!')
        return conn

conecta()

```

FONTE: O autor



Os exemplos desenvolvidos durante esse tópico estão disponibilizados no seguinte endereço: <http://bit.ly/36Hqs8S>.

Note que foi criada uma função denominada `conecta()`. Tal função retorna um ponteiro de conexão para o servidor, como o procedimento de se conectar é recorrente em manipulação de dados, criar uma função auxilia na codificação.

Agora que aprendemos a conectar com o banco de dados, vamos inserir um registro utilizando o Python. Em SQL o comando para inserção é denominado `INSERT`, veja como fica a integração com Python.

QUADRO 14 - INSERINDO REGISTROS PYTHON COM BANCO DE DADOS

```

import psycopg2

try:
    conn = conecta()
    cursor = conn.cursor()

    sql_insert = """ INSERT INTO pessoa (id, nome, endereco) VALUES
(%s,%s,%s)"""

    id    = int(input("Informe o id:"))
    nome  = input("Informe o Nome:")
    endereco = input("Informe o Endereço")

    record_to_insert = (id, nome, endereco)
    cursor.execute(sql_insert, record_to_insert)

    conn.commit()
    count = cursor.rowcount
    print (count, "Registro Inserido com Sucesso")

except (Exception, psycopg2.Error) as error :
    if(conn):
        print("Falha ao inserir, ocorreu o erro:", error)

finally:
    #encerra a conexão com o servidor
    if(conn):
        cursor.close()
        conn.close()
        print("Conexão encerrada")

```

FONTE: O autor

Um modo de conferir como ficou a inserção, basta na interface de consulta do servidor PostgreSQL executar o seguinte comando `select * from pessoa`. Que irá retornar todos os registros armazenados na tabela `pessoa`.

Para realizar a consulta dentro do Python iremos utilizar o mesmo comando. A instrução SQL é executada e os registros do banco de dados são retornados em um objeto do Python. Vamos conferir como fica esse exemplo.

QUADRO 15 - CONSULTANDO REGISTROS PYTHON COM BANCO DE DADOS

```

import psycopg2

try:
    conn = conecta()
    cursor = conn.cursor()
    sql = "select * from pessoa"

    cursor.execute(sql)

#retorna todos os registros em um único objeto

    pessoas = cursor.fetchall()

    print("Registros Armazenados em Pessoa")
    for row in pessoas:
        print("ID :", row[0], )
        print("Nome", row[1])
        print("Endereço:", row[2], "\n")

except (Exception, psycopg2.Error) as error :
    print ("Erro ao realizar consulta", error)

finally:
    if(conn):
        cursor.close()
        conn.close()
        print("Conexão encerrada")

```

FONTE: O autor



Ao executar os comandos `type(pessoas)` e `type(pessoas[0])` você irá perceber que serão retornadas lista e tupla, respectivamente. Isto significa que ao realizar uma consulta e trazer registros de um banco de dados esse objeto será do tipo tupla, que é imutável. Ou seja, você não poderá alterá-lo, caso seja necessário terá que armazenar em outra estrutura, alterar e atualizar no banco de dados.

Já que falamos sobre a alteração e sobre o funcionamento das tuplas. Vamos realizar um exemplo da alteração dos registros no banco de dados através do Python. Em SQL, o comando de atualização é o UPDATE.

QUADRO 16 - ATUALIZANDO REGISTROS PYTHON COM BANCO DE DADOS

```

try:
    connection = conecta()

    cursor = connection.cursor()

    #exibe os dados antes de fazer a atualização
    print("Dados Antes de Atualizar ")
    sql_select_query = """select * from pessoa where id = %s"""
    cursor.execute(sql_select_query, (1, ))
    record = cursor.fetchone()
    print(record)

    #comando que executa a SQL que faz a atualização, passando por parametro
    o ID
    sql_update_query = """Update pessoa set nome = %s where id = %s"""
    cursor.execute(sql_update_query, ("Tina Tunner", 1))
    connection.commit()
    count = cursor.rowcount
    print(count, " Registro atualizado com sucesso ")

    #exibe os dados após fazer a atualização
    print("Dados após a atualização ")
    sql_select_query = """select * from pessoa where id = %s"""
    cursor.execute(sql_select_query, (1,))
    record = cursor.fetchone()
    print(record)

except (Exception, psycopg2.Error) as error:
    print("Ocorreu um erro ao atualizar:", error)

finally:
    if (connection):
        cursor.close()
        connection.close()
        print("Conexão Encerrada")

```

FONTE: O autor

Note que como dito anteriormente não podemos acessar diretamente a tupla como se estivesse acessando uma lista ou um array. Por isso para atualizar os dados foi utilizado o comando UPDATE que foi executando no PostgreSQL.

Para finalizarmos as operações principais iremos excluir os registros. Em SQL, o comando DELETE é utilizado para excluir registros. É muito importante que esse comando sempre venha acompanhado da instrução WHERE, que aplica uma condição na exclusão, caso contrário todos os registros de uma tabela serão excluídos.

QUADRO 17 - EXCLUINDO REGISTROS PYTHON COM BANCO DE DADOS

```
try:  
    connection = conecta()  
  
    cursor = connection.cursor()  
  
    #exibe os dados antes de deletar o registro  
    print("Dados exibidos ")  
    sql_select_query = """select * from pessoa where id = %s"""  
    cursor.execute(sql_select_query, (1, ))  
    record = cursor.fetchone()  
    print(record)  
  
    #comando que executa a SQL que faz a remoção, passando por parametro o  
    ID  
    sql = """Delete from pessoa where id = %s"""  
    cursor.execute(sql, (1, ))  
    connection.commit()  
    count = cursor.rowcount  
    print(count, " Registro removido com sucesso ")  
  
    #exibe os dados após fazer a remoção  
    print("Dados após a remoção ")  
    sql_select_query = """select * from pessoa where id = %s"""  
    cursor.execute(sql_select_query, (1, ))  
    record = cursor.fetchone()  
    print(record)  
  
except (Exception, psycopg2.Error) as error:  
    print("Ocorreu um erro ao excluir:", error)  
  
finally:  
    if (connection):  
        cursor.close()  
        connection.close()  
        print("Conexão encerrada")
```

FONTE: O autor

RESUMO DO TÓPICO 2

Neste tópico, você aprendeu que:

- É possível instalar o PostgreSQL em vários sistemas Operacional.
- O PostgreSQL pode ser acessado remotamente na nuvem.
- O Python se conecta com o PostgreSQL.
- É possível criar programas que armazenam e manipulam dados em um SGBD utilizando Python.
- Em Python os dados vindos de um SGBD são armazenados em tuplas.

AUTOATIVIDADE



1 Crie uma tabela funcionário com os seguintes atributos:

- ID
- Nome
- Salário

R.:

2 Utilizando a tabela criada, crie uma função para cadastrar funcionário:

R.:

3 Utilizando a tabela criada, crie uma função para listar os funcionários cadastrados

R.:

4 Utilizando a tabela criada, crie uma função para editar o cadastro de um funcionário a partir de um ID:

R.:

5 Utilizando a tabela criada, crie uma função para excluir o cadastro de um funcionário a partir de um ID.

R.:

BIBLIOTECAS COMPLEMENTARES

1 INTRODUÇÃO

No decorrer do curso de Big Data e Inteligência Analítica você utilizará com constância a linguagem Python para desenvolvimento de diversas tarefas, desde a coleta de dados, criação de modelos analíticos, bem como visualização de dados.

Este tópico tem como objetivo aprender um pouco sobre algumas das bibliotecas que irão ser utilizadas no decorrer do curso e que também são referência no estudo de Big Data.

2 MATH

A matemática é fundamental para a ciência da computação, pode-se dizer que a computação nasceu da matemática. No que se refere à Big Data e os métodos de *machine learning* o conhecimento dos métodos matemáticos se torna essencial.

Como um recurso muito utilizado na linguagem Python temos a biblioteca Math. A biblioteca Math contém um conjunto de recursos para auxiliar em operações matemáticas, bem como na computação científica.

Para Heinold (2012, p. 219, tradução nossa), os principais métodos da biblioteca são:

- sin, cos, tan: funções trigonométricas;
- asin, acos, atan: funções trigonométricas;
- sinh, cosh, tanh: funções hiperbólicas;
- asinh, acosh, atanh: funções hiperbólicas inversas;
- log, log10: funções logarítmicas;
- exp: funções exponenciais;
- e, pi: constantes do número epteroiano e do π .

Para compreender melhor as funções anteriores, bem como outras funções da biblioteca Math, vejamos um exemplo.

QUADRO 18 - UTILIZANDO A BIBLIOTECA MATH

```

import math

#potência
print(math.pow(2, 3))

#O método ceil () retorna o valor do teto de x - o menor número inteiro que não
é menor que x.
data = 21.6
print(math.ceil(21.6))

#trabalhando com flutuantes
numero = -15.1
print ('O número fornecido é:', numero)
print ('O valor do piso é:', math.floor (numero))
print ('O valor do teto é:', math.ceil (numero))
print ('O valor absoluto é:', math.fabs (numero))

numero = 1e-5
print ('O número fornecido (x) é:', numero)
print ('e ^ x (usando a função exp ()) é:', math.exp (numero) -1)
print ('log (fabs (x), base) é:', math.log (math.fabs (numero), 10))

#funções trigonométricas
angulo = 90
radianos = math.radians (angulo)

print ('O ângulo fornecido é:', radianos)
print ('sin (x) é:', math.sin (radianos))
print ('cos (x) é:', math.cos (radianos))
print ('tan (x) é:', math.tan (radianos))

```

FONTE: O autor



Os códigos utilizados neste tópico estão disponíveis em: <https://colab.research.google.com/drive/1x4-PB2m4y53HxSfTm8edo8dTHeojqC2U>.

3 MATPLOTLIB

Durante o processo de construção de conhecimento através de um grande volume de dados, a análise gráfica se torna importante. A visualização dos dados permite avaliar o comportamento deles, visualizar padrões e discrepâncias.

O Matplotlib é a principal biblioteca do Python para plotagem e visualização dos dados. Atualmente existem diversas outras bibliotecas, mas a grande maioria depende do Matplotlib para funcionar.

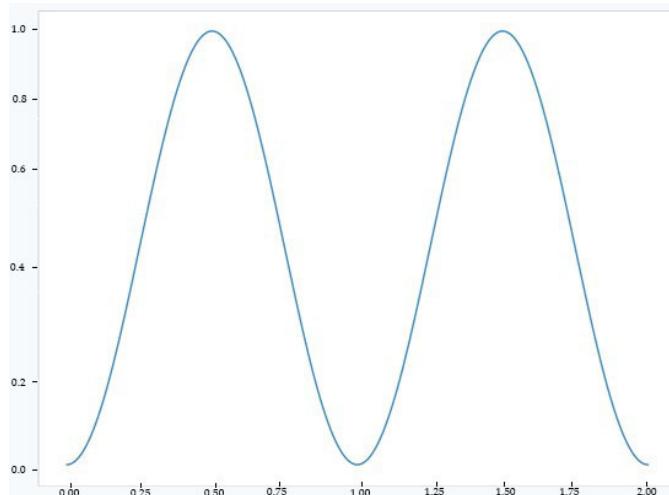
Vamos fazer um exemplo simples criando uma função seno em um intervalo de 200 números e plotar no gráfico.

QUADRO 19 - PLOTANDO A FUNÇÃO SENO

```
from matplotlib import pyplot
%matplotlib inline
from matplotlib import rcParams
rcParams['figure.figsize']=(12,9)
from math import sin, pi
x = []
y = []
for i in range(201):
    x_point = 0.01*i
    x.append(x_point)
    y.append(sin(pi*x_point)**2)
pyplot.plot(x, y)
pyplot.show()
```

FONTE: O autor

GRÁFICO 1 - PLOTANDO A FUNÇÃO SENO



FONTE: O autor

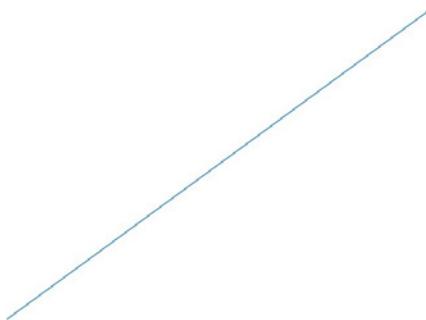
O interessante do Matplotlib é que a exibição dos dados é feita conforme os próprios dados. No caso anterior e no exemplo a seguir são passadas apenas uma dimensão, note como a plotagem dos dados se comporta.

QUADRO 20 - PLOTANDO A DADOS DE UM INTERVALO

```
import matplotlib.pyplot as plt
plt.plot(range(0,21))
plt.ylabel('Números de 0 a 20')
plt.show()
```

FONTE: O autor

GRÁFICO 2 - PLOTANDO A DADOS DE UM INTERVALO



FONTE: O autor

O Matplotlib também pode receber dois conjuntos de valores, desde que tenham o mesmo número de elementos. Para isso as abscissas e ordenadas serão dadas pela combinação dos elementos destes objetos.

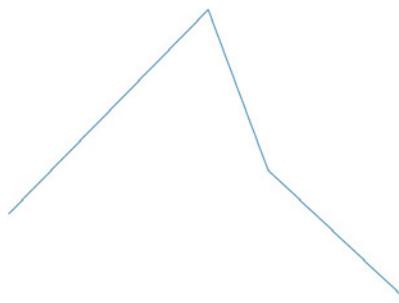
QUADRO 21 - PLOTANDO A DADOS DE DOIS OBJETOS

```
import matplotlib.pyplot as plt
import random
x = random.sample(range(30), 4)
y = random.sample(range(30), 4)
plt.plot(x,y)
plt.show()
```

FONTE: O autor

Se você executou o código anterior, reparou que em cada execução gera um gráfico diferente. Isto acontece porque os valores são aleatórios. Mas a parte principal, caso você altere, é reparar que os vetores x e y devem ter o mesmo número de elementos.

GRÁFICO 3 - PLOTANDO A DADOS DE DOIS OBJETOS



FONTE: O autor

O Matplotlib permite ajustar o gráfico para diversos cenários de dados, alterar a maneira de exibição. São diversos parâmetros, como cores, formato, entre outros. Vejamos um exemplo.

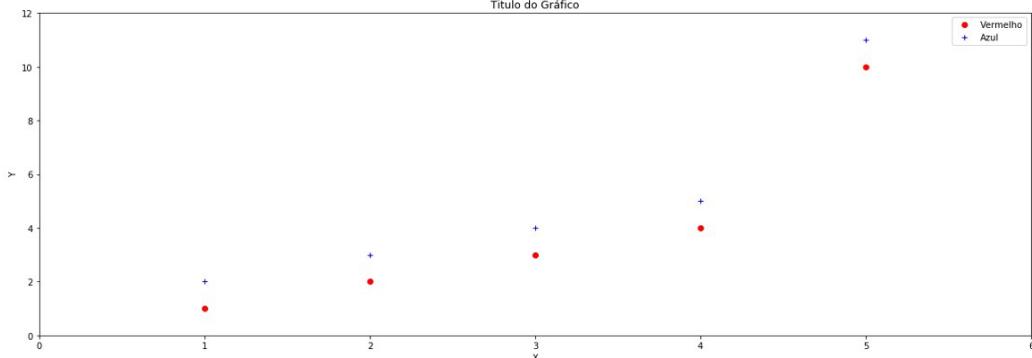
QUADRO 22 - CONFIGURANDO O PLOT

```
plt.figure(figsize=(20,7))
plt.plot([1,2,3,4,5], [1,2,3,4,10], 'ro', label='Vermelho')
plt.plot([1,2,3,4,5], [2,3,4,5,11], 'b+', label='Azul +')
plt.title('Tituto do Gráfico')
plt.xlabel('X')
plt.ylabel('Y')
plt.xlim(0, 6)
plt.ylim(0, 12)
plt.legend(loc='best')
plt.show()
```

FONTE: O autor

Neste exemplo, assim como no anterior, utilizamos dois vetores, só que nesse caso, com valor fixo. A configuração dos dados foi realizada através da escolha dos marcadores (markers), uma vez utilizado b+ significa que terá a cor azul (blue) e o símbolo +. A outra configuração foi o ro, indicando que a cor é vermelho (red) e o símbolo é um círculo indicado pela letra O.

GRÁFICO 4 - CONFIGURANDO O PLOT



FONTE: O autor

O processo de plotagem de dados dentro do cenário de Big Data e análise será constante, haverá momentos que os dados atenderam a sua necessidade como programador para compreender o comportamento dos métodos, outrora, para que clientes e gestores compreendam melhor determinados conteúdos. Para todos os casos, o Matplotlib tem opções e maneiras de flexibilizar a visualização dos dados.



Para todas as configurações possíveis consulte a documentação do Matplotlib.
Disponível em: <https://matplotlib.org>.

Vamos encerrar nosso subtópico sobre Matplotlib com um exemplo de aplicação de *machine learning* em um dataset. O exemplo em questão mostra a distribuição de dados que possuem diferentes categorias, cada categoria indicada por uma cor específica.

QUADRO 23 - EXEMPLO DE PLOT EM MACHINE LEARNING

```

import numpy as np
import matplotlib.pyplot as plt

from sklearn.datasets import make_multilabel_classification as make_ml_clf

COLORS = np.array(['!',
                  '#FF3333', # red
                  '#0198E1', # blue
                  '#BF5FFF', # purple
                  '#FCD116', # yellow
                  '#FF7216', # orange
                  '#4DBD33', # green
                  '#87421F' # brown
                 ])

RANDOM_SEED = np.random.randint(2 ** 10)

def plot_2d(ax, n_labels=1, n_classes=3, length=50):
    X, Y, p_c, p_w_c = make_ml_clf(n_samples=150, n_features=2,
                                    n_classes=n_classes, n_labels=n_labels,
                                    length=length, allow_unlabeled=False,
                                    return_distributions=True,
                                    random_state=RANDOM_SEED)

    ax.scatter(X[:, 0], X[:, 1], color=COLORS.take((Y * [1, 2, 4]
                                                    ).sum(axis=1)),
               marker='.')
    ax.scatter(p_w_c[0] * length, p_w_c[1] * length,
               marker='*', linewidth=.5, edgecolor='black',
               s=20 + 1500 * p_c ** 2,
               color=COLORS.take([1, 2, 4]))
    ax.set_xlabel('Amostra 0 ')
    return p_c, p_w_c

_, (ax1, ax2) = plt.subplots(1, 2, sharex='row', sharey='row', figsize=(8, 4))
plt.subplots_adjust(bottom=.15)

p_c, p_w_c = plot_2d(ax1, n_labels=1)
ax1.set_title('n_categorias=1, tamanho=50')
ax1.set_ylabel('Amostra 1')

```

```

plot_2d(ax2, n_labels=3)
ax2.set_title('n_categorias=3, tamanho=50')
ax2.set_xlim(left=0, auto=True)
ax2.set_ylim(bottom=0, auto=True)

plt.show()

print('O dataset gerado contém um total aleatório de %d amostras, divididas
nas seguinte maneira:' % RANDOM_SEED)
print('Classes', 'P(C)', 'P(w0|C)', 'P(w1|C)', sep='\t')
for k, p, p_w in zip(['vermelho', 'azul', 'amarelo'], p_c, p_w_c.T):
    print("%s\t%.2f\t%.2f\t%.2f" % (k, p, p_w[0], p_w[1]))

```

FONTE: <<http://bit.ly/35JWSOt>>. Acesso em: 27 nov. 2019.

4 BIBLIOTECAS PARA WEB

A Web é a grande responsável pelo aumento do volume de dados nos últimos anos, respectivamente pelo emprego de tecnologias como Big Data. Nesse contexto, é interessante conhecer as principais bibliotecas do Python que permitem coletar dados da Web e também desenvolver páginas na internet.

Para compreender o emprego de duas das principais dessas bibliotecas vamos ficar com uma leitura complementar que executa um exemplo de uma aplicação de coleta e visualização de dados da Web.

LEITURA COMPLEMENTAR

Criando uma API de filmes em cartaz usando Python e Heroku

Rodrigo Ferreira

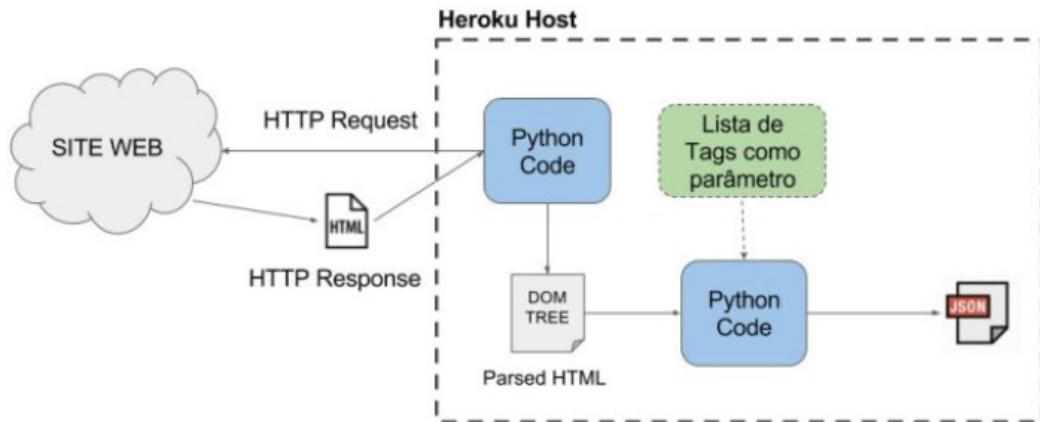
Hoje vou sair ligeiramente da minha zona de conforto e me arriscar no mundo Python. Não é uma linguagem que possuo intimidade mas por ter tido contato com algumas bibliotecas relacionadas a Inteligência Artificial, fiquei interessado em aprender mais sobre a mesma. Neste post vou mostrar como disponibilizar uma API REST criada com Python e o microframework web Flask no serviço Heroku, e o melhor: de forma 100% gratuita!

Essa história recente com o python começou com uma necessidade pessoal. Eu queria criar uma página web que exibisse diariamente os filmes que estão em cartaz nos cinemas brasileiros, a ideia em si é bem simples e jurava que ia encontrar algum site ou blog que liberasse um Feed RSS, serviço REST ou web service de forma atualizada e de fácil consulta, mas me decepcionei com o que encontrei que muitas vezes trazia apenas o que existia em outros países, ou era algum projeto no Github que estava há alguns anos sem atualização ou simplesmente não funcionava.

Partindo deste ponto, cansei de procurar e resolvi criar um serviço REST que retornasse um JSON com a lista de filmes em cartaz nos cinemas brasileiros. Para isso precisava pegar estas informações de algum site/blog que estivesse sempre atualizado. Então a missão é:

1. Encontrar uma lista de filmes pública, de fácil acesso e atualizada.
2. Preparar o ambiente com VsCode, Python e /módulos necessários.
3. Código web do Flask.
4. Fazer o parse do HTML da FONTE: usando Python para separar apenas as informações pertinentes aos filmes em cartaz.
5. Transformar essa saída em um JSON.
6. Disponibilizar o código-FONTE: no Github.
7. Publicar no Heroku para acesso público.

Pode parecer muita coisa, mas é simples, o esquema da imagem a seguir mostra o que desejamos conseguir.



1 ENCONTRAR A LISTA DE FILMES ON-LINE

Existem vários no país mas acabei escolhendo o AdoroCinema por já conhecer e gostar bastante, a lista neste link, por exemplo, traz os filmes em cartaz. No momento que crie esse post, Guardiões da Galáxia 2 no topo. *I'm groot!*

2 PREPARAR O AMBIENTE

Já temos a FONTE: dos dados precisamos carregar essa página no nosso app Python, percorrer o HTML e coletar os dados que queremos. A lib que vai fazer esse trabalho pesado para nós é a BeautifulSoup, ela vai além de apenas ler um HTML, ela permite fazer parse de outros formatos também e gerar uma árvore dos dados, serve ao nosso propósito.

Fiz tudo usando o Visual Studio Code versão 1.11, mas fique à vontade para usar outro editor de código. A versão do Python que tenho instalada é a 3.5.3 e as versões das bibliotecas usadas foram as seguintes:

```
beautifulsoup4==4.5.3
Flask==0.12
```

Para instalar as duas libs acima, usei o gerenciador de módulos pip, que faz o mesmo trabalho que o npm no Node.js. Basta executar os comandos a seguir na linha de comando:

```
pip install beautifulsoup4
pip install Flask
```

E para quem usar o VSCode recomendo a extensão [donjayamanne.python](#) para auxiliar no desenvolvimento, mas é um extra. Para instalar execute o comando abaixo no VSCode ou use o gerenciador de extensões:

```
ext install python
```

Pode ser que no momento que estejam lendo este post já existam versões mais recentes do Python, do Visual Studio Code e das libs mostradas. Caso queira testar se o python está funcionando, crie um arquivo “teste.py” com o código abaixo: Agora execute usando o comando “python teste.py” no cmd ou pelo VSCode. A saída irá mostrar: ['boldest']. O que foi feito aqui foi:

```

1 #1
2 from bs4 import BeautifulSoup
3 #2
4 import urllib.request
5 #3
6 soup = BeautifulSoup('Extremely bold','html.parser')
7 #4
8 tag = soup.b
9 #5
10 print(tag['class'])

```

Agora execute usando o comando “python teste.py” no cmd ou pelo VSCode. A saída irá mostrar: ['boldest']. O que foi feito aqui foi:

- #1 – Importar a biblioteca beautifulSoup
- #2 – Importar a biblioteca urllib(chamada http)
- #3 – Passar manualmente um HTML para ser feito o parse para o método BeautifulSoup
- #4 – Carregar a tag do DOM como um objeto para a variável python “tag”
- #5 – Exibir o atributo class do objeto capturado. Neste caso “boldest”

3 CÓDIGO WEB DO FLASK

O exemplo do tópico 2 mostra o funcionamento geral do que iremos fazer a partir daqui, mas existem alguns detalhes adicionais que vou explicar no código a seguir.

```

1 #PARTE1
2 from flask import Flask, jsonify, request
3 from bs4 import BeautifulSoup
4 from urllib.request import urlopen, Request
5 import os
6
7 #PARTE2
8 app = Flask(__name__)
9
10 #PARTE3
11 @app.route('/api/v1/filmes', methods=['GET'])
12 def filmes():
13     #MEU CÓDIGO AQUI
14     return "Todo pronto!"
15
16 #PARTE4
17 if __name__ == '__main__':
18     port = int(os.environ.get('PORT', 5000))
19     app.run(host='127.0.0.1', port=port)

```

#PARTE1: carregamos todos os módulos necessários, existem três carinhas aqui que ainda não tinha comentado: jsonify e os. Eles já vêm com a instalação padrão do Python e usaremos para transformar um array para o formato JSON e para acessar as portas do nosso ambiente no Heroku respectivamente.

#PARTE2: esse trecho simplesmente cria um app web usando Flask. A variável app recebe o objeto Flask. Usaremos a variável app mais abaixo para criar a rota e iniciar nosso serviço.

#PARTE3: com app.route dizemos qual o caminho de acesso e qual verbo HTTP será utilizado. Neste caso estamos apenas retornando uma lista de elementos então será criado um GET no caminho “/api/v1/filmes”, que fica ao seu critério escolher. Em sequência definimos uma função que será chamada ao acessar esse caminho, dentro dela terá apenas um return com a string “Tudo pronto!” por enquanto, no próximo tópico veremos o código que ficará neste local.

#PARTE4: agora definimos alguns parâmetros de ambiente, a variável port recebe uma porta disponível ou a 5000 por padrão, a lib os cuida disso. Precisei deixar dessa forma para funcionar no Heroku. O mesmo vale para o host, quando rodo local uso 127.0.0.1(localhost) mas quando publico no Heroku preciso colocar 0.0.0.0 para funcionar, nos bastidores ele irá setar o host correto.

Simples né? Isto foi apenas para testar e montar nosso ambiente base, vamos agora carregar o HTML do site AdoroCinema e capturar informações de verdade.

4 FAZER O PARSE DO HTML

Vamos agora carregar os dados dos filmes direto da página on-line do AdoroCinema, a estrutura HTML do link <http://www.adorocinema.com/filmes/numero-cinemas/> é esta:



The screenshot shows a movie page for 'Guardiões da Galáxia Vol. 2'. On the left is the movie poster. To the right, several data points are highlighted with red boxes:

- 1. Guardiões da Galáxia Vol. 2** nomeObj
- Lançamento 27/04/2017 (2h16min) dataObj
- De [James Gunn \(II\)](#)
- Com [Chris Pratt](#), [Zoe Saldana](#), [Dave Bautista](#)
- Gênero [Ação](#), [Ficção científica](#), [Comédia](#)
- Agora já conhecidos como os Guardiões da Galáxia, os guerreiros viajam ao longo do cosmos e lutam para manter sua nova família unida. Enquanto isso...** sinopseObj
- AdoroCinema**  4,5
- Imprensa**  3,7
- Leitores**  4,6

At the bottom are two buttons: **Ver o trailer** and **Sessões (448)**.

O site traz vários blocos como este mostrado na imagem acima, um embaixo do outro. Se conseguirmos trazer as informações de um bloco trazer dos outros é trivial. Os dados que queremos são:

- Nome do filme
- Data de Lançamento
- Sinopse
- Imagem do poster

Veja que cada um dos elementos eu circulei em vermelho e coloquei ao lado o nome da variável referente a eles que será utilizado no código em Python. Para capturar os dados precisamos da tag do DOM no HTML para passar como parâmetro para a biblioteca *BeautifulSoup*, a ferramenta perfeita para isso é o Inspetor de Elementos do navegador, todos os browsers atuais possuem um, seja Chrome, Firefox ou MS Edge. Veja um exemplo:



The screenshot shows a movie listing page with a card for '1. Guardiões da Galáxia Vol. 2'. The card includes a thumbnail, the movie title, and a brief description.

Below the card, the browser's developer tools are open, specifically the 'Elements' tab. The HTML code for the movie card is visible, with several elements highlighted in red:

```

<!-- sep -->
<div class="data_box">
  <div class="img_side_content">
    > a href="/filmes/filme-226995/" class="xXx ">...</a>
  <div class="content">
    <div class="titlebar_02 margin_10b">
      <span class="bold">
        1.
      </span>
      <h2 class="tt_18 d_inline">
        <a class="no_underline" title href="/filmes/filme-226995/">
          Guardiões da Galáxia Vol. 2
        </a> == $0
      </h2>
    </div>
    <!--/titlebar_02-->
    <ul class="list_item_p2v tab_col_first">...</ul>
    <p>...</p>
    <div class="margin_10v">...</div>
    <a class="button btn-primary" href="/filmes/filme-226995/trailer-19554075/" itemprop="trailer">...</a>
    <div class="button btn-group">...</div>
  </div>
<!--/content-->

```

The highlighted elements include the movie title ('Guardiões da Galáxia Vol. 2'), the URL ('/filmes/filme-226995/'), and the trailer URL ('/filmes/filme-226995/trailer-19554075/'). Below the code, the browser's status bar shows the selected element's class: 'div.titlebar_02.margin_10b'.

A linhas que irão capturar os valores de cada tag e armazenar em um vetor são:

```

1. #PARTE1
2. html_doc=urlopen("http://www.adorocinema.com/filmes/numero-cinemas/").
   read()
3. soup = BeautifulSoup(html_doc, "html.parser")
4.
5. #PARTE2
6. data = []
7. for DataBase in soup.find_all("div",class_="data_box"):
8.     nomeObj = DataBase.find("h2", class_="tt_18 d_inline").find("a", class_="no_
   underline")
9.     imgObj = DataBase.find(class_="img_side_content")
10.    sinopseObj = DataBase.find("div", class_="content").find("p")
11.    dataObj = DataBase.find("ul", class_="list_item_p2v tab_col_first").
      find("div", class_="oflow_a")
12.
13. #PARTE3
14.     data.append({ 'nome': nomeObj.text.strip(),
15.                   'poster' : imgObj.img['src'].strip(),
16.                   'sinopse' : sinopseObj.text.strip(),
17.                   'data' : dataObj.text.strip()})

```

Obs.: o código de acesso a cada tag depende do html do site, isto significa que qualquer mudança deve ser feita também no código.

Destrinchando este código em 3 partes, temos:

#PARTE1: carregamos o HTML do site na variável `html_doc` e o resultado do parse do `BeautifulSoup` na variável `soup`.

#PARTE2: criamos um array vazio chamado `data`, criamos um laço com `for` para percorrer todos os divs que possuam o atributo `class` igual a `data_Box`. Ou seja, cada um dos filmes da lista. Para cada dados que desejamos (nome, sinopse, data e imagem) chamamos o método `find` da variável `DataBase` criada no `for`, o parâmetro do `find` é a tag do DOM onde a informação se encontra. Veja que se tomarmos como exemplo o nome do filme (`nomeObj`), ele pode ser encontrado no HTML dentro da tag

#PARTE3: por último, mas não menos importante, jogamos o objeto montado dentro do array `data`. Isto será feito várias vezes até que o laço chegue ao fim, ou seja, não tenha mais filmes na lista.

5 GERAR A SAÍDA EM JSON

O pior já passou, agora é só transformar nosso vetor `data` que possui a lista de objetos(filmes), para o formato JSON. No `return` colocamos a linha abaixo:

```
1 return jsonify({'filmes': data})
```

Feito tudo isso execute a aplicação na linha de comando com “python web.py”. Renomeei seu arquivo para web.py pois será necessário agora que enviaremos para o Heroku. No último tópico mostrarei como ficou a lista de filmes em JSON.

6 COMPARTILHANDO NO GITHUB

Antes de colocar no Heroku o projeto foi disponibilizado no Github no endereço abaixo. O motivo é que uma das formas que o Heroku permite fazer o deploy da app é através da integração com o Github o que é bem mais prático.

- <https://github.com/rodrigorf/filmes-cartaz-json>

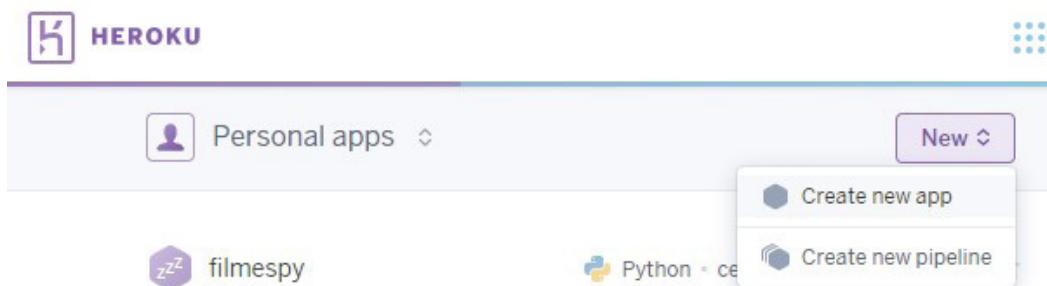
7 PUBLICANDO A API NO HEROKU

Antes de colocarmos no Heroku precisamos criar três arquivos de configuração que o servidor precisa, acabei descobrindo isso após me bater bastante ao subir o projeto da primeira vez. Os arquivos na raiz do projeto são **Procfile**, **requirements.txt** e **runtime.txt**.

- requirements.txt -> informa os módulos que devem ser importados no servidor para que a aplicação funcione, são eles: beautifulsoup4==4.5.3 e Flask==0.12.
- runtime.txt -> informa a versão do python que queremos rodar: python-3.4.3.
- Procfile -> arquivo que contém o comando de execução e o nome da app, desta forma -> web: python web.py

A aplicação está versionada e já retorna a lista de filmes em formato JSON, isto já deve servir para muitas pessoas, mas a ideia era criar algo que outros possam usar sem precisar necessariamente clonar um repositório no Github e compilar uma app em Python. Meu objetivo principal é ter uma URL pública que outros desenvolvedores possam utilizar também seja em Python, C#, Node.js, Java, ou qualquer linguagem. Vamos agora publicar em nossa hospedagem, o <http://heroku.com>, siga os passos abaixo:

1. Crie uma conta no site do Heroku.
2. Faça o login no seu painel em <https://dashboard.heroku.com>



Tela inicial do painel do Heroku

3. Clique em “Create new app”.
4. Na tela seguinte informe um nome e a região do servidor.
5. Ao avançar seremos levados diretamente a tela de deploy da app. Aqui faremos a conexão com o Github conforme imagem abaixo.

The screenshot shows the Heroku Dashboard for an app named "testepyfilmes". In the "Deployment method" section, there are three options: "Heroku Git", "GitHub", and "Dropbox". The "GitHub" button is highlighted with a red box. Below it, there are sections for connecting to pipelines and a "Connect to GitHub" button.

Escolhemos conectar com Github para pegar o código-FONTE: da app.

6. Clique em Github e autorize a conexão do Heroku. A aba abaixo é mostrada, procure o projeto em seus repositórios e clique em “Connect”.

The screenshot shows the "Connect to GitHub" dialog. It includes a search bar with the results "rodrigorf" and "filmes-cartaz", a "Search" button, and a "Connect" button at the bottom. There is also a message about organization access.

Conexão com o repositório do app que criamos

Lembrando que para que for fazer isso deverá primeiro clonar o projeto no Github.

7. Estando tudo certo e devidamente conectados com o Github já podemos fazer o deploy. O deploy pode ser automático, ou seja, todo push feito no branch dispara uma publicação, ou ser feito de forma manual. No nosso caso será manualmente.

The screenshot shows the "Manual deploy" dialog. It has a dropdown menu set to "master" and a "Deploy Branch" button.

8. Ao clicar em “Deploy Branch” o processo de carregamento do aplicativo será iniciado e um log do processo é mostrado, o Python é configurado, o PIP bem como os módulos existentes são instalados no servidor para que a aplicação funcione.

The screenshot shows the Heroku build log interface. At the top, there's a green checkmark icon and the text "Receive code from GitHub". Below that, it says "Build master" and "Hide build log". The main area displays the build process logs:

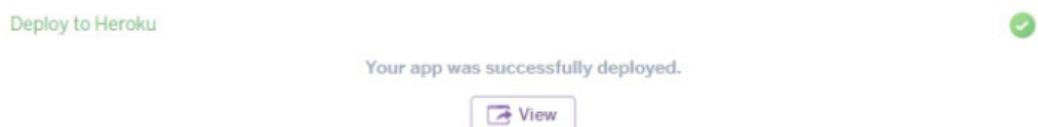
```

-----> Python app detected
-----> Installing python-3.4.3
-----> Installing pip

```

At the bottom, there are two buttons: a checked checkbox labeled "Autoscroll with output" and a link labeled "View build log".

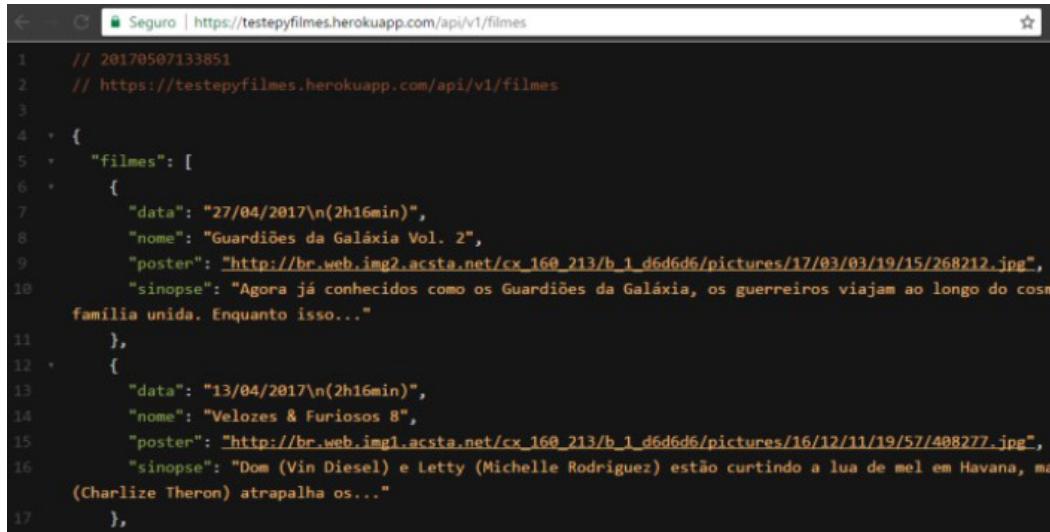
Caso tudo termine sem erros, é apresentada uma mensagem de sucesso e um botão para exibir a aplicação.



Clique em “View” para abrir a URL da sua aplicação, algo do tipo <https://minhaapp.herokuapp.com>. Verá uma mensagem de “NOT FOUND” isto ocorre pois criamos uma rota específica para nosso GET que retorna os filmes, lembra? o caminho da rota é “/api/v1/filmes”, então fica:

- <https://minhaapp.herokuapp.com/api/v1/filmes>

Finalmente é apresentado nosso JSON com a lista dos filmes carregadas em tempo real. Olha só que legal:



The screenshot shows a browser window with the URL <https://testepyfilmes.herokuapp.com/api/v1/filmes>. The page displays a JSON response with two movie entries. The first movie is "Guardiões da Galáxia Vol. 2" (data: 27/04/2017, nome: Guardiões da Galáxia Vol. 2, poster: http://br.web.img2.acsta.net/cx_160_213/b_1_d6d6d6/pictures/17/03/03/19/15/268212.jpg, sinopse: Agora já conhecidos como os Guardiões da Galáxia, os guerreiros viajam ao longo do cosmos em busca de sua família unida. Enquanto isso...). The second movie is "Velozes & Fúriosos 8" (data: 13/04/2017, nome: Velozes & Fúriosos 8, poster: http://br.web.img1.acsta.net/cx_160_213/b_1_d6d6d6/pictures/16/12/11/19/57/408277.jpg, sinopse: Dom (Vin Diesel) e Letty (Michelle Rodriguez) estão curtindo a lua de mel em Havana, mas Charlize Theron atrapalha os...).

```

1 // 20170507133851
2 // https://testepyfilmes.herokuapp.com/api/v1/filmes
3
4 +
5   "filmes": [
6     {
7       "data": "27/04/2017\n(2h16min)",
8       "nome": "Guardiões da Galáxia Vol. 2",
9       "poster": "http://br.web.img2.acsta.net/cx_160_213/b_1_d6d6d6/pictures/17/03/03/19/15/268212.jpg",
10      "sinopse": "Agora já conhecidos como os Guardiões da Galáxia, os guerreiros viajam ao longo do cosmos em busca de sua família unida. Enquanto isso..."
11    },
12    {
13      "data": "13/04/2017\n(2h16min)",
14      "nome": "Velozes & Fúriosos 8",
15      "poster": "http://br.web.img1.acsta.net/cx_160_213/b_1_d6d6d6/pictures/16/12/11/19/57/408277.jpg",
16      "sinopse": "Dom (Vin Diesel) e Letty (Michelle Rodriguez) estão curtindo a lua de mel em Havana, mas Charlize Theron atrapalha os..."
17    }
]

```

Caso a exibição no Chrome não esteja do seu gosto, eu instalei uma extensão chamada Json Viewer que permite aplicar alguns temas e formatar melhor o JSON. Bom é isso galera, o post ficou um pouco extenso pois tentei descrever da melhor forma possível cada passo.

Em breve quero deixar o projeto mais flexível para facilitar o carregamento a partir de outras FONTES de filmes, a ideia basicamente é criar um arquivo JSON de configuração onde o usuário informa apenas a URL e os XPATH de onde cada dado será puxado, por exemplo, para o AdoroCinema ficaria assim:

```

1 [
2   {
3     "url": "http://www.adorocinema.com/filmes/numero-cinemas/",
4     "pathNome": "//*[@id='col_main']/div[6]/div/div/div[1]/h2/a",
5     "pathData": "//*[@id='col_main']/div[6]/div/div/ul/li[1]/div",
6     "pathSinopse": "//*[@id='col_main']/div[6]/div/div/p",
7     "pathPoster": "//*[@id='col_main']/div[6]/div/a/img"
8   }
]

```

Infelizmente, a BeautifulSoup não dá suporte direto para XPath, talvez consiga fazer isso com a lib lxml, mas o bacana desse modelo é que qualquer um com um navegador moderno pode capturar o xpath a partir do inspetor de elementos e criar um JSON de configuração, permitindo que a própria comunidade crie as mais variadas FONTES de filmes sem muito esforço.

FONTE: <<https://códigosimples.net/2017/05/15/criando-uma-api-de-filmes-em-cartaz-usando-python/>>. Acesso em: 27 nov. 2019.

RESUMO DO TÓPICO 3

Neste tópico, você aprendeu que:

- Utilizando a biblioteca Math é possível realizar diversas operações matemáticas.
- O Matplotlib é uma biblioteca de suporte à criação de gráficos.
- É possível plotar dados e funções com Matplotlib.
- Com Matplotlib se pode criar gráficos por diversas perspectivas.
- Utilizando flask se pode fazer a integração de Python com a Web.



Ficou alguma dúvida? Construímos uma trilha de aprendizagem pensando em facilitar sua compreensão. Acesse o QR Code, que levará ao AVA, e veja as novidades que preparamos para seu estudo.



AUTOATIVIDADE



- 1 No contexto de Big Data, a Web é uma FONTE: rica de dados, pois através dela é possível coletar e analisar dados de sites, redes sociais, dentre outros. Selecione a alternativa CORRETA que contenha o nome da biblioteca do Python utilizada para coletar dados da Web.
 - a) () IPython.
 - b) () Scrapy.
 - c) () Math.
 - d) () Matplotlib.
- 2 O desenvolvimento para Web é um mercado muito forte e cada vez mais tem dominado espaço. Selecione a alternativa CORRETA que contenha o nome da biblioteca Python para desenvolvimento de páginas Web.
 - a) () Scikit-Learn.
 - b) () Math.
 - c) () Flask.
 - d) () Matplotlib.
- 3 Considere x uma sequência de 1 até 10 e y um conjunto de 10 números aleatórios. Crie um programa em Python que mostre o gráfico de y em função de x.
- 4 Considere x uma sequência de 1 até 20 e y um conjunto de 20 números aleatórios. Crie um programa em Python que mostre o gráfico de y em função de x. A linha desta função deverá ser vermelha.
- 5 Considere x uma sequência de 1 até 20 e y um conjunto de 20 números aleatórios, z um conjunto de 20 números aleatórios. Crie um programa em Python que mostre o gráfico de y em função de x e z em função de x. A linha de x deverá ser vermelha a linha de z deverá ser verde.

REFERÊNCIAS

- AD, H. **Qual distro Linux é a mais popular?** (sem achismo). Blog Diolinux, [s.l.], 8 maio 2019. Disponível em: <http://bit.ly/2vcCY2o>. Acesso em: 20 nov. 2019.
- BADER, D. Object-orientes programming (OOP) in Python 3. Real Python, [s.l.], c2019. Disponível em: <http://bit.ly/39ZE6WH>. Acesso em: 20 nov. 2019.
- BRESSERT, E. **SciPy and NumPy**: an overview for developers. Newton, MA: O'Reilly Media, Inc., 2012.
- CARDOSO, A.; LEITÃO, J.; TEIXEIRA, C. *Using the Jupyter Notebook as a Tool to Support the Teaching and Learning Processes*. In: INTERNATIONAL CONFERENCE ON INTERACTIVE COLLABORATIVE LEARNING – SPECIAL SESSION ON TALKING ABOUT TEACHING, 21., 2018, Kos Island, GRE. **Proceedings** [...]. Kos Island, GRE: ICL, 2018. p. 227-236.
- CECHINEL, C. Operadores lógicos. **Cristian Cechinel Personal Website**, Pelotas, c2019. Disponível em: <http://bit.ly/2scIfFX>. Acesso em: 20 nov. 2019.
- CIENTISTA de dados a profissão do futuro continua em alta. **Exame**, São Paulo, 27 maio 2019. Disponível em: <http://bit.ly/2NaHx3a>. Acesso em: 20 nov. 2019.
- DONADEL, B. S. et al. **Análise de funções de medida para o método K-MEANS**. 2018, 58f. Trabalho de Conclusão de Curso (Graduação em Matemática) – Universidade Federal de Santa Catarina, Florianópolis, 2018. Disponível em: <http://bit.ly/2TdFkb0>. Acesso em: 6 dez. 2019.
- EQUIPE DSA. A diferença entre engenheiro de dados, estatísticos, cientista de dados e engenheiro de software. **Data Science Academy**, São Paulo, 11 jul. 2019. Disponível em: <http://bit.ly/3073FAV>. Acesso em: 20 nov. 2019.
- FILHO, C R C. **Fatiando listas em Python**. eXcript, [s.l.], 28 abr. 2015. Disponível em: <http://bit.ly/2t6WBYH>. Acesso em: 17 nov. 2019.
- GASPAROTTO, H. (2019). POO: Os 4 pilares da Programação Orientada a Objetos. **DevMedia**, Rio de Janeiro, 2014. Disponível em: <http://bit.ly/2TdZcea>. Acesso em: 28 nov. 2019.
- HARRISON, M.; PRENTISS, M. **Learning the Pandas library**: Python tools for data munging, analysis, and visual. Scotts Valley, CA: Createspace Independent Publishing Platform, 2016. Disponível em: <http://bit.ly/37V9nbM>. Acesso em: 10 dez. 2019.
- HEINOLD, B. A practical introduction to Python programming. Emmitsburg,

MD: Mount St. Mary's University, 2012. Disponível em: <http://bit.ly/39VkfOU>. Acesso em: 10 dez. 2019.

KLUYVER, T. et al. *Jupyter Notebooks-a publishing format for reproducible computational workflows*. In: INTERNATIONAL CONFERENCE ON ELECTRONIC PUBLISHING, 20., 2016, Göttingen, GER. **Proceedings** [...]. Göttingen, GER: ELPUB, 2016. p. 87-90. Disponível em: <http://bit.ly/30b0oAh>. Acesso em: 6 dez. 2019.

KRUEGER, C. W. Software reuse. **ACM Computing Surveys**, New York, v. 24, n. 2, p. 131-183, 1992. Disponível em: <http://sunnyday.mit.edu/16.355/kruger.pdf>.

KUHLMAN, D. *A python book: beginning python, advanced python, and python exercises*. Lutz: Platypus Global Media, 2009.

LENO, M. A história do python. Mind Bending, [s.l.], 8 out. 2014. Disponível em: <http://mindbending.org/pt/a-historia-do-python>. Acesso em: 6 dez. 2019.

LUTZ, M. **Learning Python**: powerful object-oriented programming. Newton, MA: O'Reilly Media, Inc., 2013.

MATOS, D. Bibliotecas de data science em python, R e scala. **Ciência e Dados**, [s.l.], 25 jun. 2019. Disponível em: <http://bit.ly/3a07k7S>. Acesso em: 6 dez. 2019.

MCKINNEY, W. et al. **pandas**: powerful Python data analysis toolkit. Pandas, [s.l.], 2 nov. 2019. Disponível em: <http://bit.ly/2QGj3RD>. Acesso em: 10 dez. 2019.

PHILLIPS, D. **Python 3 object-oriented programming**. Birmingham: Packt Publishing Ltda., 2015.

REINSEL, D.; GANTZ, J.; RYDNING, J. *Data age 2025: the evolution of data to life-critical. don't focus on big data*. Framingham, MA: IDC, 2017. Disponível em: <https://go.ey.com/2FIL1FP>. Acesso em: 20 nov. 2019.

RICARTE, I. L. M. **Programação orientada a objetos**: uma abordagem com Java. Campinas: Universidade Estadual de Campinas, 2001. Disponível em: <http://bit.ly/39XSnDp>. Acesso em: 11 dez. 2019.

SANTANCHE, A. **Programação orientada a objetos**. Campinas: Universidade Estadual de Campinas, 2015. Disponível em: <http://bit.ly/39Xp300>. Acesso em: 11 dez. 2019.

SANTIAGO JR., L. Entendendo a biblioteca NumPy. **Ensina ai**, [s.l.] 30 set. 2018. Available at: <http://bit.ly/2TiaIFc>. Acesso em: 20 nov. 2019. 17 nov. 2019.

SIGRID, Google colab: guia do iniciante. **Machina Sapiens**, [s.l.], 7 dez. 2018. Disponível em: <http://bit.ly/2R3I1JJ>. Acesso em: 6 dez. 2019.

SILVA, J. M. **Desmistificando o comando mount**. Dicas-L. Jaguariúna, c2019.

Disponível em: <http://bit.ly/307A6iB>. Acesso em: 18 nov. 2019.

SOMMERVILLE, I. **Engenharia de software**. 9. ed. São Paulo: Pearson Prentice Hall, 2011. Disponível em: <http://bit.ly/39VkYt2>. Acesso em: 11 dez. 2019.

SUMMERFIELD, M. **Programming in Python 3**: a complete introduction to the Python language. 2. ed. Boston: Addison-Wesley Professional, 2010. Disponível em: <http://bit.ly/36GaI5P>. Acesso em: 11 dez. 2019.

SWAROOP, c. h. A byte of Python. [s.l.]: ebsshelf Inc., 2013. Disponível em: <http://bit.ly/2Tc2C1a>. Acesso em: 10 dez. 2019.

TENENBAUM, A. M.; LANGSAM, Y.; AUGENSTEIN, M. J. **Estruturas de dados usando C**. São Paulo: M. Books, 2004.

TIOBE index for november 2019. TIOBE (the software quality company), Eindhoven, NE, 2019. Disponível em: <https://www.tiobe.com/tiobe-index/>. Acesso em: 25 nov. 2019.

VANDERPLAS, J. **Python data science handbook**: essential tools for working with data. Newton, MA: O'Reilly Media, Inc., 2016. Disponível em: <http://bit.ly/36RSMWh>. Acesso em: 10 dez. 209.

VICTÓRIA, P. Qual a melhor linguagem para ciência de dados? **iMasters**, São Paulo, 25 fev. 2019. Disponível em: <http://bit.ly/37TsqTB>. Acesso em: 25 nov. 2019.

WHAT IS NUMPY? **SciPy.org**, [s.l.], 26 jul. 2019. Disponível em: <https://docs.scipy.org/doc/numpy/index.html>. Acesso em: 10 dez. 209.