



PUC Minas

DIRETORIA DE  
EDUCAÇÃO CONTINUADA

Pós Graduação *Lato Sensu*

**Ciência de Dados e Big  
Data / Introdução às  
linguagens estatísticas**



## Quem sou eu?

- Arthur Pereira de Gouveia e Silva
- Casado, 38 anos, pai de uma linda garota de dois anos de idade
- Torcedor “azul grená” do FC Barcelona
- Tolkien, Batman, Poe, Portnoy, Peart, I. Cavalera, Metal, Python, Rubik, Android, Google, Microsoft, Linux, Star Wars (May 4th be with you), PSOne, PS2, PS3, PS4,
- Telecom → Automação → Manutenção → Saneamento → Gestão → Estatística → Data Mining → Big Data



## Como me encontrar



@arthurg



gouveia.arthur@gmail.com



[www.linkedin.com/in/arthur-gouveia/](https://www.linkedin.com/in/arthur-gouveia/)



@arthur\_gouveia



[github.com/arthur-gouveia](https://github.com/arthur-gouveia)



## E quem são vocês?

- Sabem programar?
- Quais linguagens?





# Programação

- Motivação
- Introdução às linguagens de programação
- Introdução à programação
  - Algoritmos
  - Dados, operações e operadores
- Introdução à linguagem Python
  - Ambiente de desenvolvimento e boas práticas
  - Variáveis, condicionais, loops, estruturas de dados, funções
  - Acessando dados, módulos para ciência de dados
- Introdução à linguagem R
  - R e Rstudio
  - SWIRL



# Avaliação

- **25%** Conclusão dos exercícios em Python
- **25%** Conclusão das lições no Swirl
- **50%** Trabalho final em grupo
  
- Critérios de avaliação:
  - Clareza e simplicidade no código
  - Pontualidade nas entregas
  - Resultados corretos retornados pelo código
  
- **Trabalho final:** Desenvolver uma aplicação que se conecta a um banco de dados e a arquivos texto para leitura dos dados e gera um relatório dos dados lidos.

# Motivação





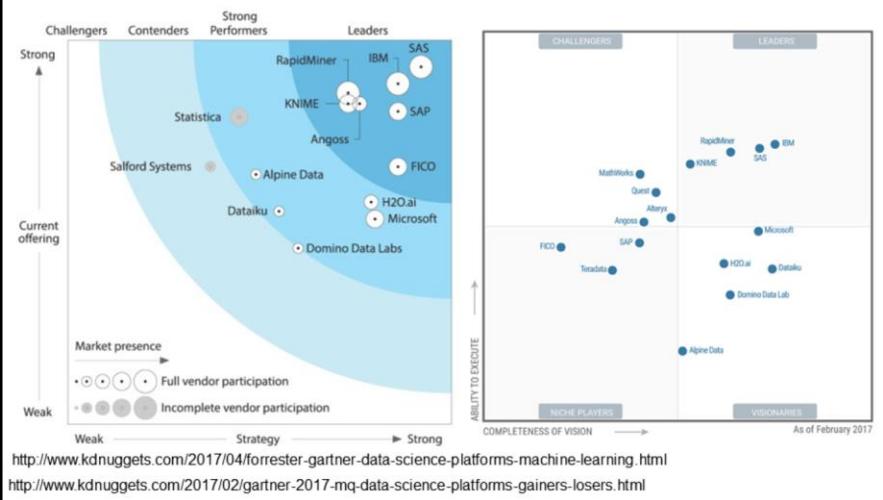
# Motivação

- Por que eu deveria aprender a programar?
  - Para desenvolver programas...
  - Para melhorar meu raciocínio lógico
  - Para entender melhor como meu computador funciona
  - Para gravar filmes no meu videocassete
  - Para ser um cientista de dados?



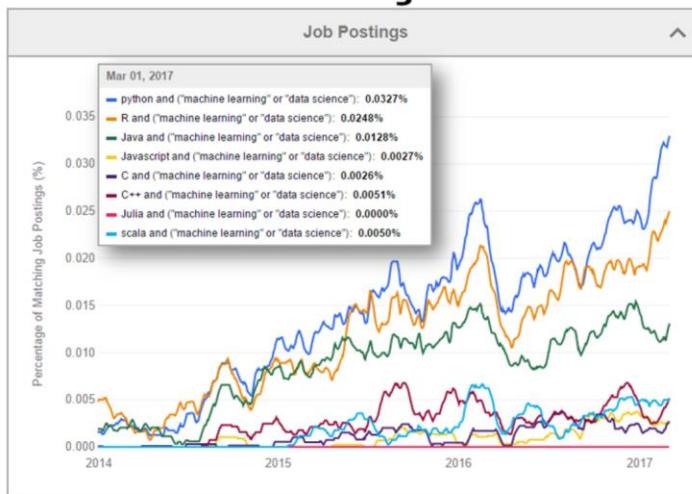


# Motivação





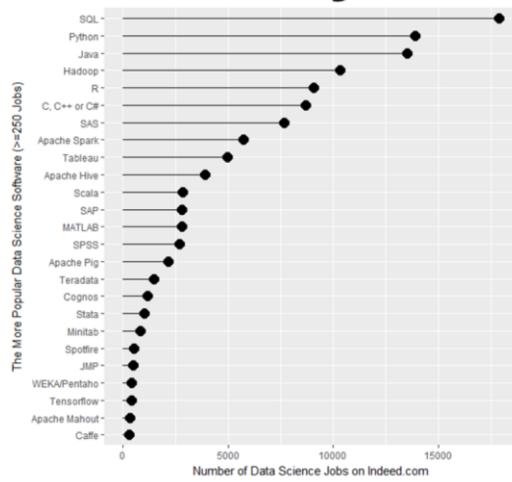
# Motivação



[https://www.indeed.com/jobtrends/q-python-and-\(%22machine-learning%22-or-%22data-science%22\)-q-R-and-\(%22machine-learning%22-or-%22data-science%22\)-q-Java-and-\(%22machine-learning%22-or-%22data-science%22\)-q-Javascript-and-\(%22machine-learning%22-or-%22data-science%22\)-q-C-and-\(%22machine-learning%22-or-%22data-science%22\)-q-C++-and-\(%22machine-learning%22-or-%22data-science%22\)-q-Julia-and-\(%22machine-learning%22-or-%22data-science%22\)-q-scala-and-\(%22machine-learning%22-or-%22data-science%22\).html](https://www.indeed.com/jobtrends/q-python-and-(%22machine-learning%22-or-%22data-science%22)-q-R-and-(%22machine-learning%22-or-%22data-science%22)-q-Java-and-(%22machine-learning%22-or-%22data-science%22)-q-Javascript-and-(%22machine-learning%22-or-%22data-science%22)-q-C-and-(%22machine-learning%22-or-%22data-science%22)-q-C++-and-(%22machine-learning%22-or-%22data-science%22)-q-Julia-and-(%22machine-learning%22-or-%22data-science%22)-q-scala-and-(%22machine-learning%22-or-%22data-science%22).html)



# Motivação



<http://r4stats.com/2017/02/28/r-passes-sas/>

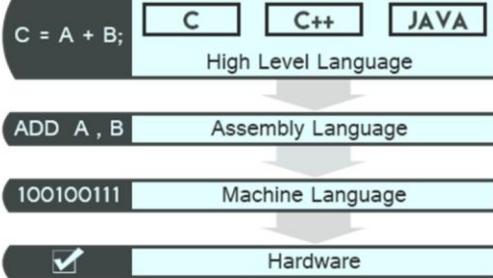
# **Introdução às linguagens de programação**





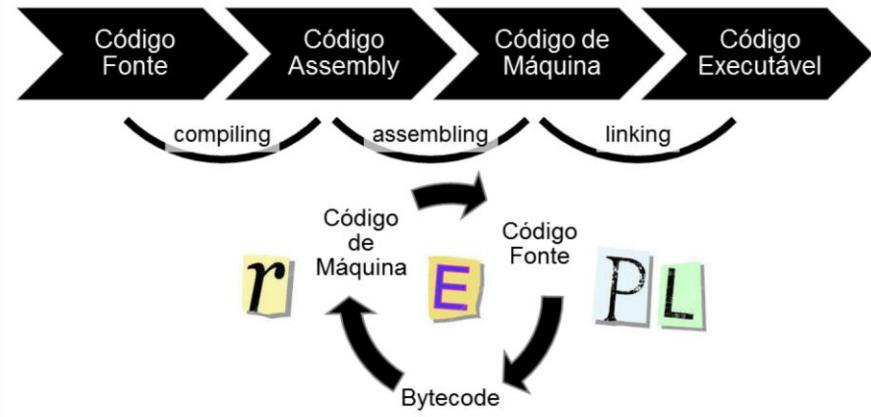
# Introdução às linguagens de programação

$C = A + B$





# Linguagens compiladas x interpretadas





## E para Data Science?

*all science is  
trial and error:*

$$\cancel{E = ma^2}$$

$$\cancel{E = mb^2}$$

$$E = mc^2$$



## Python & R



Python é um gênero de répteis da família Pythonidae. Pode ser encontrado na Ásia e África. Popularmente são denominadas de pitão (português europeu) ou piton (português brasileiro).

Nenhuma das serpentes desta família possui dentes inoculadores de veneno, porém possuem presas afiadas curvadas pra dentro para agarrar sua presa. Os pitons variam de 4,5 a 6 metros de comprimento.

A letra R (erre) é a décima oitava letra do alfabeto latino. Durante um longo período de tempo o "R" foi escrito "P" como no alfabeto cirílico.

O seu nome no alfabeto fenicio era "rech". Seu significado era o de uma cabeça, representada pela adaptação do hieróglifo egípcio de uma cabeça. Transformou-se no "rô" dos gregos. Os romanos modificaram o rô acrescentando um pequeno traço para diferenciá-lo do no nosso P.



## Python & R



GLLLLL! NÃO SEI  
POQUE NÃO  
OLENDO A  
MAREM





## Python & R



1989: Criação por Guido van Rossum

1994: grupo comp.lang.python e lançamento da v 1.0

2000: Python 2.0

2008: Python 3.0

2010: Python 2.7

2016: Python 3.6



1992: Concepção

1995: Versão inicial

2000: Versão 1.0

2004: Versão 2.0.0

2013: Versão 3.0.0

2016: Versão 3.3.2



Douglas Bates

Hello world:

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



```
public class HelloWorld {
    public static void main(String[] args) {
        // Prints "Hello, World" to the terminal window.
        System.out.println("Hello, World");
    }
}
```

Hello world:

```
> print("hello world")
"hello world"
```

## Introdução à programação

The screenshot shows a Google search results page for the query "algoritmo". At the top, there is a banner for the Diretoria de Educação Continuada IEC + PREPES PUC Minas, featuring a blue wavy line graphic and the institution's logo. Below the banner, the search bar contains "algoritmo". The search interface includes a microphone icon for voice search and a magnifying glass icon for search. Below the search bar, a navigation bar offers links to "Todas", "Vídeos", "Imagens", "Notícias", "Livros", "Mais", "Configurações", and "Ferramentas". A note indicates approximately 9,920,000 results found in 0.36 seconds. The main search result is a box titled "algoritmo" with the subtitle "substantivo masculino". It lists two definitions: 1. "mat sequência finita de regras, raciocínios ou operações que, aplicada a um número finito de dados, permite solucionar classes semelhantes de problemas." and 2. "inf conjunto das regras e procedimentos lógicos perfeitamente definidos que levam à solução de um problema em um número finito de etapas." Below the definitions, there is an "Origem" section with the etymology "ETIM lat.medv. *algorismus*, com infl. do gr. *arithmós* 'número'". There is also a "Traduzir algoritmo para o" dropdown menu set to "Bielorrusso" which shows the translation "1. алгарытм". At the bottom of the result box is a "Mostrar menos" link.



## Exemplos



<https://www.youtube.com/watch?v=R07JITOPicE>



## Tipos de dados

- **Numéricos:** Inteiro, Ponto flutuante
  - 1; -3; 42; 3.14159265; -0.37; 9e-10
- **Textuais:** String, caractere
  - "Python rules!"; "a"; "42"; "887169805.3872932"
- **Lógicos:** Booleanos
  - True; False
- **Vetoriais:** Listas, Arrays, Vetores...
  - [1, 2, 5, 89]; ["X", "exemplo", "45"]; [1; 3.905; "texto"; False]; [42]

<https://docs.python.org/3/library/stdtypes.html>



# Operações com dados

- **Exponenciação:** Somente para dados numéricos
  - $2^3$ ;  $5^2$ ;  $89.342^{16}$
- **Multiplicação e Divisão:**
  - $6 \times 8$ ;  $25 / 5$ ;  $3 \% 2$ ;  $4 \times "k"$
- **Adição e Subtração:**
  - $8 + 34$ ;  $5 - 37$ ;  $"4" + "2"$ ;  $[1, 3, 4] + [8, 9, 82]$
- **Comparação:**
  - $42 == 3$ ;  $"ab" < "ba"$ ;  $3 != "3"$ ;  $5 >= 5$ ;  $[30, 2, 3] < [30, 2, 4]$

<https://docs.python.org/3/reference/expressions.html#operator-precedence>



Tutorial: <https://docs.python.org/3/tutorial/index.html>



# Introdução ao Python

- Download do Python: <https://www.python.org/>
- **Recomendo** o Anaconda:  
<https://www.continuum.io/downloads>
- **Recomendo ainda mais** o miniconda:  
<https://conda.io/miniconda.html>
- **Faça parte da comunidade!** <http://python.org.br/>;  
<https://t.me/pythonbr>;  
<https://www.facebook.com/groups/python.brasil>;  
<https://t.me/datasciencepython>;  
<https://github.com/datascience-python>

Configuração e uso do Python: <https://docs.python.org/3/using/index.html>



## Ambiente de desenvolvimento Python

- Instalou o conda? Vamos testar.
- Abra um terminal ou prompt de comando
- Digite **conda info**
- Se mostrou informações do sistema, parabéns!
- Crie um ambiente de desenvolvimento: **conda create -n dev** → *dev é o nome do ambiente que estamos criando*
- Ative o ambiente: **activate dev**
- Abra o Python: **python**

“Colas” do anaconda: [https://conda.io/docs/\\_downloads/conda-cheatsheet.pdf](https://conda.io/docs/_downloads/conda-cheatsheet.pdf)

Gestão de ambientes conda: <https://conda.io/docs/using/envs.html>

Criação de ambientes em Python: <https://docs.python.org/3/library/venv.html>



## Ambiente de desenvolvimento Python

- Diga olá para o mundo!
- `>>> print('Olá mundo!')`
- Um ambiente mais “confortável” é o Jupyter. Digite `exit()` para sair do Python e voltar ao terminal.

May I Have Your  
Attention Please



## Ambiente de desenvolvimento Python

- **Instalação de módulos**
  - `conda install nome_do_pacote`
  - `pip install nome_do_pacote`
- Vamos instalar o Jupyter



- Mais informações em:
- <http://jupyter.org/>
- <https://nbviewer.jupyter.org/>

Instalando módulos: <https://docs.python.org/3/installing/index.html>



## Ambiente de desenvolvimento Python

- **Boas práticas de uso do Jupyter:**
- <https://www.svds.com/jupyter-notebook-best-practices-for-data-science/>
- O que eu uso:
  - Salvar .ipynb e .py
  - Executar o jupyter remotamente no servidor acessando-o de minha máquina:

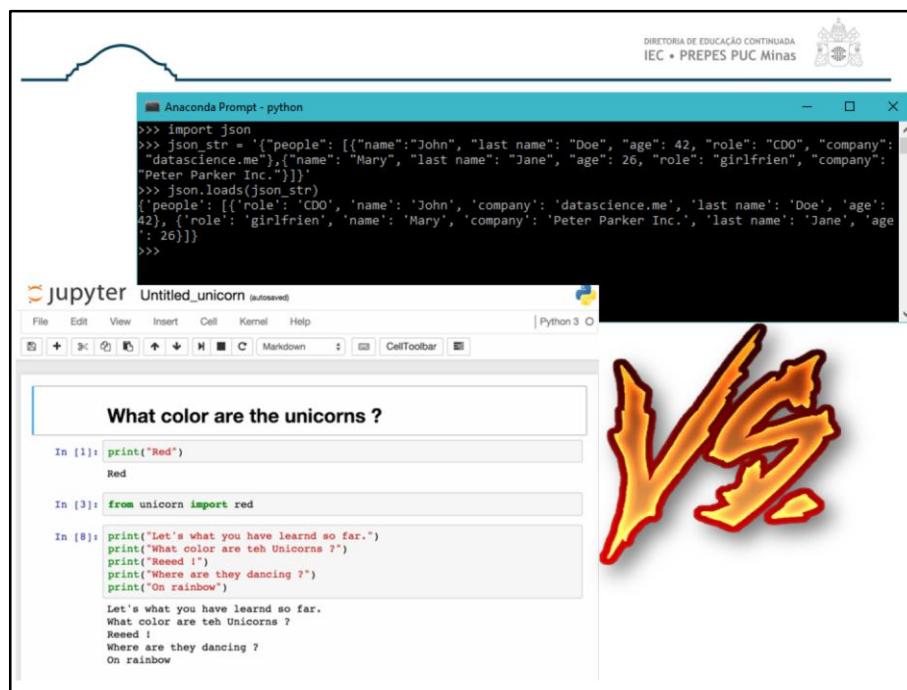
[http://jupyter-notebook.readthedocs.io/en/latest/public\\_server.html](http://jupyter-notebook.readthedocs.io/en/latest/public_server.html)

Dicas de uso do Jupyter: <https://medium.com/towards-data-science/jupyter-notebook-hints-1f26b08429ad>



## Ambiente de desenvolvimento Python

- Digite **conda install jupyter**
- Vários pacotes serão instalados. Ao fim da instalação digite **jupyter notebook**. O servidor inicia e abre o jupyter no seu browser.
- Navegue pelas pastas e abra o arquivo **Exercícios Python.ipynb**



IDEs para Python: <https://wiki.python.org/moin/IntegratedDevelopmentEnvironments> e <https://wiki.python.org.br/IdesPython>



## Importação de módulos

- Python vem com **baterias inclusas**, o que quer dizer que sua biblioteca padrão permite ao programador desenvolver aplicações completas sem precisar instalar bibliotecas de terceiros. Entretanto Python tem **caráter modular** e algumas funcionalidades, mesmo presentes na biblioteca padrão, não estão presentes imediatamente e **precisam ser importadas**. Por exemplo, as funções para lidar com o sistema de arquivos do S.O. estão presentes na biblioteca **os** que precisa ser importada antes de utilizar. Para importar um módulo basta utilizar a palavra reservada **import** seguida pelo nome do módulo.
- Por exemplo **import os**

Baterias inclusas: <https://docs.python.org/3/py-modindex.html> e <https://docs.python.org/3/library/index.html>



# Variáveis

Variáveis são espaços na memória do computador onde podemos armazenar valores. Em várias linguagens as variáveis são tratadas como “baldes” com nomes.

Em Python o conceito é um pouco diferente; variáveis são baldes com “etiquetas” e cada balde pode receber várias “etiquetas” diferentes.

Modelo de dados: <https://docs.python.org/3/reference/datamodel.html#objects-values-and-types>



## Variáveis

- Para salvar um valor na memória do computador damos um nome a este espaço de memória e atribuímos a ele o valor desejado. Por exemplo `a = 2`. Lê-se “a recebe 2”
- As variáveis em Python podem ter seus nomes formados por caracteres, números ou sublinhados porém o nome **não** pode começar com números. Ah! Caracteres com acento são aceitos em Python 3 ☺!!!
- Nomes válidos para variáveis: `salário`; `salário_médio`; `_temp02`; `Velocidade`; `time_11_jogadores`; `__hidden__`



## Operadores de comparação

- Em Python os valores lógicos são **True** e **False**.  
Perceba as letras maiúsculas...
- Os seguintes valores são avaliados como False em Python: **None**; **False**; zero de qualquer tipo numérico **0**, **0.0**, **0j**; qualquer sequência vazia como **''**, **()**, **[]**; qualquer mapeamento vazio como, por exemplo, **{}**
- Os operadores de comparação são:

Operador	Operação	Símbolo matemático
<code>==</code>	Igualdade	<code>=</code>
<code>&gt;</code>	Maior que	<code>&gt;</code>
<code>&lt;</code>	Menor que	<code>&lt;</code>
<code>!=</code>	Diferente	<code>≠</code>
<code>&gt;=</code>	Maior ou igual	<code>≥</code>
<code>&lt;=</code>	Menor ou igual	<code>≤</code>

<https://docs.python.org/3/library/stdtypes.html#comparisons>



# Operadores lógicos

- São 3: **not** (não), **and** (e), **or** (ou)
- Esses operadores obedecem a seguinte a tabela verdade:

V1	V2	Not V1	V1 and V2	V1 or V2
T	T	F	T	T
T	F	F	F	T
F	T	T	F	T
F	F	T	F	F

precedência

<https://docs.python.org/3/library/stdtypes.html#boolean-operations-and-or-not>



## Expressões lógicas

- Os operadores lógicos podem ser combinados em **expressões lógicas** e sua avaliação segue a ordem de precedência definida.

```
True or False and not True
True or False and False
True or False
True
```

- Também pode-se combinar operadores de comparação que têm maior precedência.

```
salário, idade = 1500, 25
salário > 2000 and idade > 20
1500 < 2000 and 25 > 20
True and False
False
```



## Variáveis do tipo string

- Uma string é um conjunto de caracteres (letras, símbolos, espaço etc.) como por exemplo **Um anel para todos governar**. Para separar uma string do restante do texto do seu programa utilizamos aspas. Python entende tanto aspas simples ' ' ou duplas " " para delimitar uma string. Desta forma escreveremos "**Um anel para todos governar**" ou mesmo '**Um anel para todos governar**'.
- Antes de falarmos sobre as operações com string vamos ver duas coisas muito importantes sobre elas: **como descobrir o tamanho de uma string e como acessá-la letra por letra**.

<https://docs.python.org/3/library/stdtypes.html#text-sequence-type-str>



# Comprimento e indexação de strings

- A função `len` retorna o tamanho de uma string.  
`len("a")` retorna 1, `len('Um')` retorna 2,  
`len("dois")` retorna 4 e `len('')` retorna 0.
- Outra coisa importante é acessar cada caractere de uma string. Sabendo o tamanho da string podemos acessar cada um de seus elementos através de um número inteiro dentro de colchetes `[]`. **Em Python as sequências começam pelo índice 0.**

0	1	2	3	4
T	e	x	t	o

<https://docs.python.org/3/library/stdtypes.html#string-methods>



## Operações com strings

- **Fatiamento:** O fatiamento amplia as possibilidades da indexação de strings e é indicado por dois pontos dentro dos colchetes [ : ]

```
>>> a = "ABCDEF"
```

```
>>> a[0:2]
```

AB

- **Concatenação:** Consiste em unir uma ou mais strings em uma única sequência de caracteres. A concatenação é feita utilizando os sinais de adição '+' ou multiplicação '\*'.



## Composição de strings

- Pode-se compor duas ou mais strings, ou uma string e um número através do sinal de %. %s para strings, %d para inteiros e %f para ponto flutuante.
- Uma forma mais poderosa é utilizar a função **str.format()**
- Em Python 3.6 existem as *fstrings*.

```
>>> preço = 3.5
>>> print(f'O preço é R$ {preço}')
'O preço é R$ 3.5'
```

Vale muito a pena ler <https://pyformat.info/>

<https://docs.python.org/3/library/string.html#custom-string-formatting>



# Condicionais

- Em Python, sempre que temos situações onde se uma coisa acontecer faça algo, usamos a instrução if.

- A sintaxe é:

```
if <uma coisa acontecer>:  
    algo
```

**Lembrete**

Em Python os diferentes blocos de execução PRECISAM ser identados (recuados). É assim que a linguagem sabe se um trecho de código está dentro ou fora do if. Outro ponto importante é indicar com dois pontos : onde o bloco if começa.

[https://docs.python.org/3/reference/compound\\_stmts.html#the-if-statement](https://docs.python.org/3/reference/compound_stmts.html#the-if-statement)



# Condicionais

Se uma coisa acontecer

Faça algo

Senão, se outra condição acontecer

Faça outra coisa

Senão

Faça uma terceira coisa

```
if <uma coisa acontecer>:  
    algo  
elif <outra condição acontecer>:  
    outra coisa  
else:  
    uma terceira coisa
```



# Repetições

O comando while repete seu bloco de código enquanto uma condição for verdadeira.

```
while <condição>:  
    algo
```

Imprime os valores de 1 a 3

```
>>> x = 1  
>>> while x <= 3:  
>>>     print(x)  
>>>     x += 1 # O que equivale a x = x + 1
```

[https://docs.python.org/3/reference/compound\\_stmts.html#the-while-statement](https://docs.python.org/3/reference/compound_stmts.html#the-while-statement)



# Repetições

Outra forma de fazer repetições é utilizar o *loop for*

```
>>> for i in range(4):
>>>     print(i)
0
1
2
3
```

Você pode "brincar" um pouco com a função range. Por exemplo range(10, 20, 2) retorna todos os números de 10 a 19 de dois em dois. range(20, 0, -1) gera números decrescentes de 20 a 1.

[https://docs.python.org/3/reference/compound\\_stmts.html#the-for-statement](https://docs.python.org/3/reference/compound_stmts.html#the-for-statement)

<https://docs.python.org/3/library/stdtypes.html#range>

<https://docs.python.org/3/tutorial/controlflow.html#more-control-flow-tools>



## Tipos específicos de dados

- Listas: [] (list)
- Dicionários: {} (dict)
- Tuplas: () (tuple)
- Conjuntos: {} (set)

<https://docs.python.org/3/tutorial/datastructures.html>



# Listas

A estrutura de dados mais comum em Python

```
>>> L = [1, 2, 3]
```

- Tamanho de L: `len(L)`
- Primeiro elemento de L: `L[0]`
- Último elemento de L: `L[-1]`
- L do segundo elemento até o final: `L[1:]`
- Adicionando dados a uma lista: `L.append(2)`
- Concatenando duas listas: `L += [4, 5, 6]`
- Ou então: `L.extend([7, 8, 9])`

<https://docs.python.org/3/library/stdtypes.html#list>



## Dicionários

São estruturas de dados semelhantes às listas, mas seus dados são pares chave-valor. Todos os dados em um dicionário devem estar associados a uma chave. Por exemplo um dicionário que representa produtos e preços:

```
>>> preço = {"TV": 3000, "Smartphone": 1900,  
             "Notebook": 4500}  
>>> preço["TV"]  
3000  
>>> preço['óculos'] = 350  
>>> preço  
{"TV": 3000, "Smartphone": 1900, "Notebook":  
4500, "óculos": 350}
```

<https://docs.python.org/3/library/stdtypes.html#mapping-types-dict>



# Tuplas

Tuplas são como listas, com a grande diferença de que são imutáveis. **Você não pode atribuir um valor a uma tupla.** Dessa forma elas são excelentes para armazenar valores constantes (como o resultado de uma consulta a um banco de dados) e para empacotamento e desempacotamento de dados.

```
>>> tupla = (42, "a", 3.1416, [10, 20, 30])
>>> # ou simplesmente separando os elementos por vírgulas
>>> tupla = 42, "a", 3.1416, [10, 20, 30]
>>> # Desempacotamento
>>> inteiro, string, real, lista = tupla
>>> a, b = b, a # Invertendo os valores das variáveis
```

<https://docs.python.org/3/library/stdtypes.html#tuple>



## Definição de funções

Funções são definidas utilizando a palavra reservada `def`

```
def soma(a, b):  
    print(a + b)
```

Funções retornam valores com a palavra `return`

```
def soma(a, b):  
    return a + b
```

[https://docs.python.org/3/reference/compound\\_stmts.html#function-definitions](https://docs.python.org/3/reference/compound_stmts.html#function-definitions)



## Valores padrão para os parâmetros

Na definição de uma função é possível atribuir valores padrão para seus parâmetros

```
>>> def exponenciação(a, b=2):  
>>>     return a ** b  
>>>  
>>> exponenciação(5)  
25  
>>> exponenciação(2, 3)  
8
```



## Mais sobre funções

Podemos passar uma função como parâmetro para outras:

```
>>> lista = [0, 2, -3, 5, -1, 4.2, -12]
>>> sorted(lista, key=abs)
[0, -1, 2, -3, 4.2, 5, -12]
```

Podemos acessar parâmetros pelo nome como no caso de key ao chamar a função sorted acima

Funções curtas e anônimas: `lambda x: x[-1]`

<https://docs.python.org/3/library/functions.html#sorted>

<https://docs.python.org/3/tutorial/controlflow.html#more-on-defining-functions>



## Acessando arquivos

Modo	Operações
r	leitura
w	escrita <b>[Apaga o conteúdo do arquivo se já existir]</b>
a	escrita, mas preserva o conteúdo se já existir
b	modo binário
+	atualização (leitura, escrita)

A abertura de um arquivo é feita com a função open que recebe o nome do arquivo e o modo. O exemplo a seguir mostra como criar um arquivo contendo os números de 1 a 100:

```
arquivo = open('números.txt', 'w')
for linha in range(100):
    arquivo.write("{}\n".format(linha+1))
arquivo.close()
```

<https://docs.python.org/3/library/io.html#module-io>

<https://docs.python.org/3/library/functions.html#open>



## Lendo um arquivo csv

```
import csv

dados = []
colunas = ['sepal_length', 'sepal_width',
           'petal_length', 'petal_width', 'class']
with open('iris.data') as iris:
    linhas = csv.DictReader(iris, fieldnames=colunas)
    for linha in linhas:
        for k, v in linha.items():
            if k != 'class':
                linha[k] = float(v)
    dados.append(linha)
```

<https://docs.python.org/3/library/csv.html>



## Acessando bancos de dados

Python consegue acessar vários tipos de bancos de dados mas o mais simples e que já vem integrado à linguagem é o SQLite. Sua principal vantagem é não precisar de um servidor dedicado e o programa que você está desenvolvendo pode criar o banco, as tabelas e transacionar dados nelas!



## Acessando bancos de dados

Para usar o SQLite em Python é necessário importar o módulo sqlite3

```
import sqlite3
```

A conexão com o banco é criada com

```
conexão = sqlite3.connect(<nome_do_banco>)
```

O cursor que irá "navegar" pelo banco executando comandos SQL é criado com

```
cursor = conexão.cursor()
```

<https://docs.python.org/3/library/sqlite3.html>



## Acessando bancos de dados

Comandos SQL podem ser passados usando  
`cursor.execute(<comando_SQL>)`  
`cursor.executemany(<comando_SQL>)`  
`cursor.fetchall(<comando_SQL>)`  
`cursor.fetchone(<comando_SQL>)`  
`Cursor.fetchall(<comando_SQL>)`

Se for necessário passar parâmetros para a consulta,  
como um where por exemplo, sempre use ? e não %s ou  
{ } com format. Assim evitamos ataque de injeção SQL



## Acessando bancos de dados

Sempre dê `cursor.commit()` após a execução de um comando pois só assim garantimos que os dados serão persistidos no banco.

Sempre feche a conexão e o cursor com  
`cursor.close()`  
`conexão.close()`



## Principais módulos para Ciência de Dados

Python está entre as principais linguagens utilizadas pelos cientistas de dados em todo o mundo e entre as linguagens mais solicitadas em vagas de emprego.

Mas a biblioteca padrão de Python, sozinha, não é muito útil para Ciência de Dados. **Precisamos** de bibliotecas de terceiros (porém todas gratuitas e de código aberto)

<https://medium.com/activewizards-machine-learning-company/top-15-python-libraries-for-data-science-in-2017-ab61b4f9b4a7>



# Principais módulos para ciência de dados



**Statsmodels** é um módulo Python que permite ao usuário explorar dados, estimar modelos estatísticos e realizar testes estatísticos. Uma lista extensa de estatísticas descritivas, testes estatísticos, funções gráficas e resultados estatísticos estão disponíveis.

**scikit-learn**: machine learning in Python. Tem slogan mais significativo que esse? 😊

scikit-learn tem funções para:

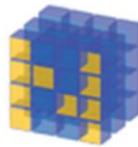
- Classificação
- Regressão
- Agrupamento (Clustering)
- Redução de dimensionalidade
- Seleção de modelos
- Pré-processamento

Statsmodels: <http://statsmodels.sourceforge.net/>

scikit-learn: <http://scikit-learn.org/stable/>



## Principais módulos para ciência de dados



**NumPy** é um módulo fundamental para computação científica em Python. A principal característica de NumPy é seu array N-dimensional extremamente poderoso. Vários outros módulos trabalham internamente com os famosos ndarrays do NumPy. Tanto que ao instalar o pandas, por exemplo, NumPy será instalado também.

Quando estiverem com tempo disponível leiam o quickstart do NumPy <https://docs.scipy.org/doc/numpy-dev/user/quickstart.html> e tentem executar e entender os códigos de lá.

<http://www.numpy.org/>



# Principais módulos para ciência de dados

## matplotlib

**Matplotlib** é um módulo para plotagens 2D que produz imagens de qualidade para publicações. Matplotlib se integra perfeitamente com o Jupyter através do comando mágico `%matplotlib inline`. Sempre que for plotar algum gráfico em um notebook Jupyter utilize esse comando logo no início do seu código.

A principal característica do Matplotlib é tornar simples as coisas complicadas e fazer as coisas complicadas possíveis. Ou seja, se você só quer plotar um boxplot isso é muito fácil, mas se quiser personalizar detalhes do gráfico isso é possível.

<http://matplotlib.org/>

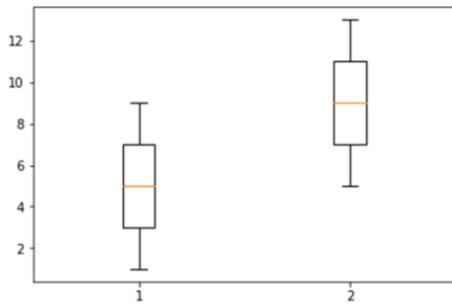
Folha de colas:

[https://s3.amazonaws.com/assets.datacamp.com/blog\\_assets/Python\\_Matplotlib\\_Cheat\\_Sheet.pdf](https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Python_Matplotlib_Cheat_Sheet.pdf)



# Matplotlib

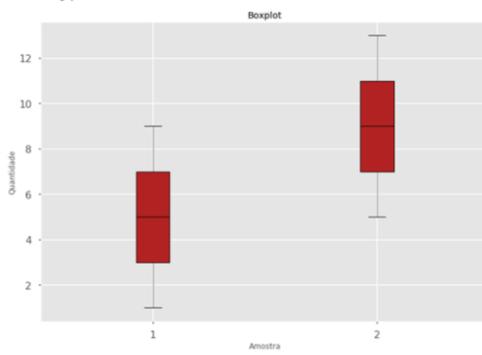
```
from matplotlib import pyplot as plt  
  
plt.figure()  
dados = [[1, 2, 3, 4, 5, 6, 7, 8, 9],  
         [5, 6, 7, 8, 9, 10, 11, 12, 13]]  
plt.boxplot(dados)  
plt.show()
```





# Matplotlib

```
from matplotlib import style
style.use('ggplot')
ax = plt.subplot()
bp = ax.boxplot(dados, patch_artist=True,
                 whiskerprops={'color': 'black', 'linestyle': '-', 'linewidth': 0.5},
                 boxprops={'color': 'black', 'facecolor': 'firebrick'},
                 medianprops={'color': 'black'})
ax.set_title('Boxplot')
ax.set_ylabel('Quantidade')
ax.set_xlabel('Amostra')
```

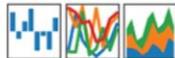




# Principais módulos para ciência de dados

## pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



**pandas: Python Data Analysis Library. Biblioteca de Análise de Dados em Python**

pandas é uma biblioteca de código aberto que fornece estruturas de dados de alta performance (baseadas em NumPy) e ferramentas para análise de dados.

Folha de colas do pandas (não use na prova!): [https://github.com/pandas-dev/pandas/tree/master/doc/cheatsheet/Pandas\\_Cheat\\_Sheet.pdf](https://github.com/pandas-dev/pandas/tree/master/doc/cheatsheet/Pandas_Cheat_Sheet.pdf)

A grande vantagem do pandas é fornecer o objeto DataFrame (baseado no R) que fornece uma estrutura semelhante a uma tabela em um banco SQL. Pandas permite até fazer inner e outer joins!

<http://pandas.pydata.org/>

Folha de colas oficial: [https://github.com/pandas-dev/pandas/blob/master/doc/cheatsheet/Pandas\\_Cheat\\_Sheet.pdf](https://github.com/pandas-dev/pandas/blob/master/doc/cheatsheet/Pandas_Cheat_Sheet.pdf)

Outra folha de colas:

[https://s3.amazonaws.com/assets.datacamp.com/blog\\_assets/PandasPythonForDataScience+\(1\).pdf](https://s3.amazonaws.com/assets.datacamp.com/blog_assets/PandasPythonForDataScience+(1).pdf)



# pandas

Grando um DataFrame a partir de um arquivo CSV

```
pd.read_csv('iris.data', header=None,  
names=['sepal_length','sepal_width','petal_length','petal_width','class'])
```

	sepal_length	sepal_width	petal_length	petal_width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
...	...	...	...	...	...



# pandas

Grando um DataFrame a partir de um banco de dados

```
conexão = sqlite3.connect('realestate.db')
df4 = pd.read_sql("select * from housing limit 100", conexão)
conexão.close()
df4
```

	mls	location	price	bedrooms	bathrooms	size	price_per_sq_ft	status
0	132842	Arroyo Grande	795000.0	3	3	2371.0	335.30	Short Sale
1	134364	Paso Robles	399000.0	4	3	2818.0	141.59	Short Sale
2	135141	Paso Robles	545000.0	4	3	3032.0	179.75	Short Sale
3	135712	Morro Bay	909000.0	4	4	3540.0	256.78	Short Sale
4	136282	Santa Maria-Orcutt	109900.0	3	1	1249.0	87.99	Short Sale



# pandas

## Filtrando, selecionando e fatiando um DataFrame

```
# Selecionando pelo nome da coluna
df['A']
# Selecionando pelo nome da coluna mas retornando um DataFrame ao invés de um
Series
df[['A']]
# Selecionando um valor pelo índice e nome da coluna
df.loc['2017-06-05', 'B']
# Selecionando apenas as linhas onde o valor da coluna A é maior que zero
df.loc[df.A > 0, :]
# Selecionando duas colunas
df[['A', 'C']]
# Fatiando as duas primeiras linhas
df.iloc[:2, :]
```

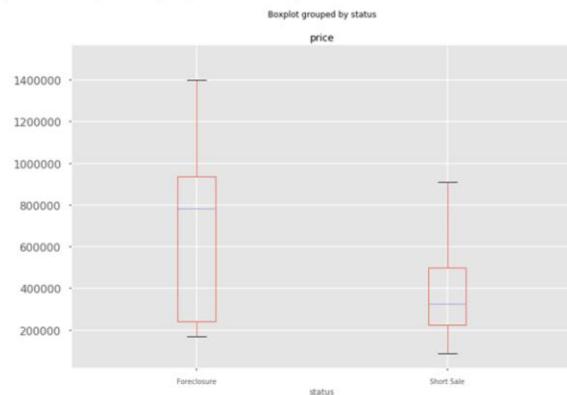
Prefira sempre as funções de seleção .loc, .iloc e .ix



# pandas

Plotando um boxplot do preço por status

```
df4.boxplot(column='price', by='status')
```



**Introdução à  
linguagem**

**R**



## Introdução ao R

# {swirl}

Learn R, in R.

<http://swirlstats.com/>

Instale o R (<http://cran.rstudio.com/>)

Instale o Rstudio

([www.rstudio.com/products/rstudio/download/](http://www.rstudio.com/products/rstudio/download/))



# Introdução ao R

- Abra o Rstudio
  - File
  - New Project
  - New Directory
- ```
> install.packages("swirl")
> library(swirl)
> install_course_github("swirldev", "R_Programming_E")
> swirl()
```

The screenshot shows the RStudio interface. The console pane displays the R startup message and the command sequence for installing the swirl package and starting the swirl() function. The environment pane shows an empty global environment. The file browser pane shows a directory structure under 'win-library 3.3' containing various R packages like bitops, crayon, curl, digest, evaluate, gtdr, httr, and RdDisplay.



## O básico do swirl

1. Quando você vir ... você só precisa pressionar [Enter]
2. Quando vir '[ANSWER:](#)', > ou quando for solicitado a selecionar um valor de uma lista é hora de entrar uma resposta e pressionar [Enter]
3. Saia do swirl a qualquer momento pressionando [Esc]. Você retornará ao prompt do R.
4. Quando estiver no prompt do R (>) dentro do swirl
  1. Digitando [skip\(\)](#) pula a pergunta atual.
  2. Digitando [play\(\)](#) o swirl ignora o que você digitar até você digitar [nxt\(\)](#) para retomar o swirl
  3. Digitando [bye\(\)](#) o swirl sai e seu progresso é salvo
  4. Digitando [main\(\)](#) você volta para o menu principal
  5. Digitando [info\(\)](#) essa ajuda é mostrada.



## Iniciando o curso

1. Digite seu nome e guarde essa informação. Você só poderá continuar os cursos digitando o mesmo nome.
2. Escolha o curso R Programming E
3. Escolha a lição desejada. Recomendo seguir a sequência.
4. Todos os cursos **exceto** 13: Simulation e 14: Dates and Times são mandatórios e fazem parte da avaliação

OBRIGADO



QUALQUER DÚVIDA  
PESQUISEM NO GOOGLE!  
BRINCADEIRA! VOCÊS TÊM MEUS CONTATOS