Introdução a linguagem Python



Pythonidae (Python family)

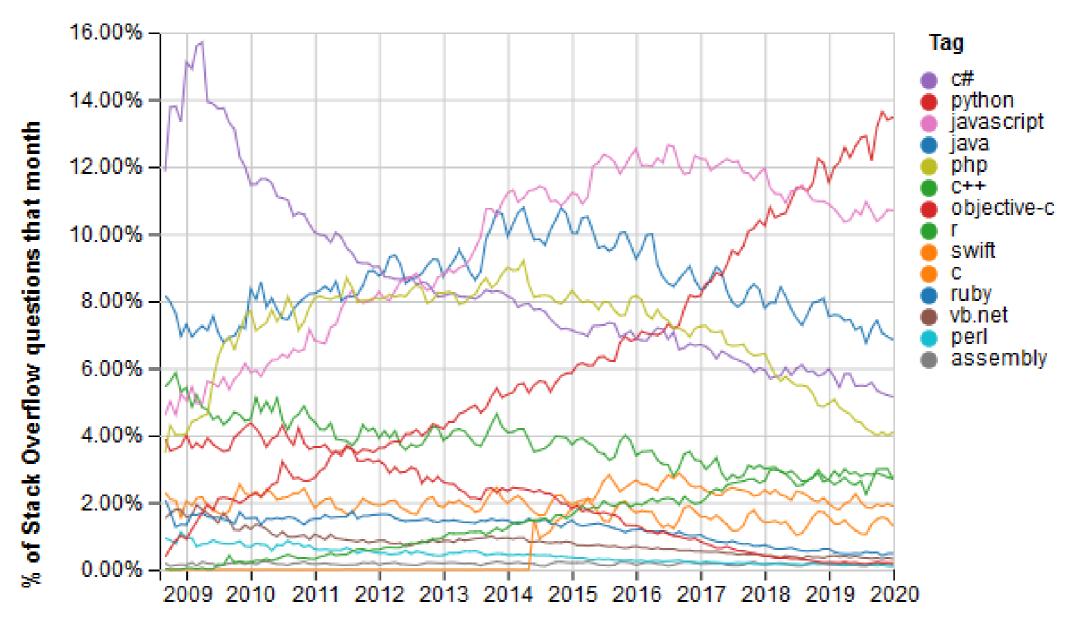






Professor: Alex Pereira

Porcentagem de Questões Por Mês / Linguagem



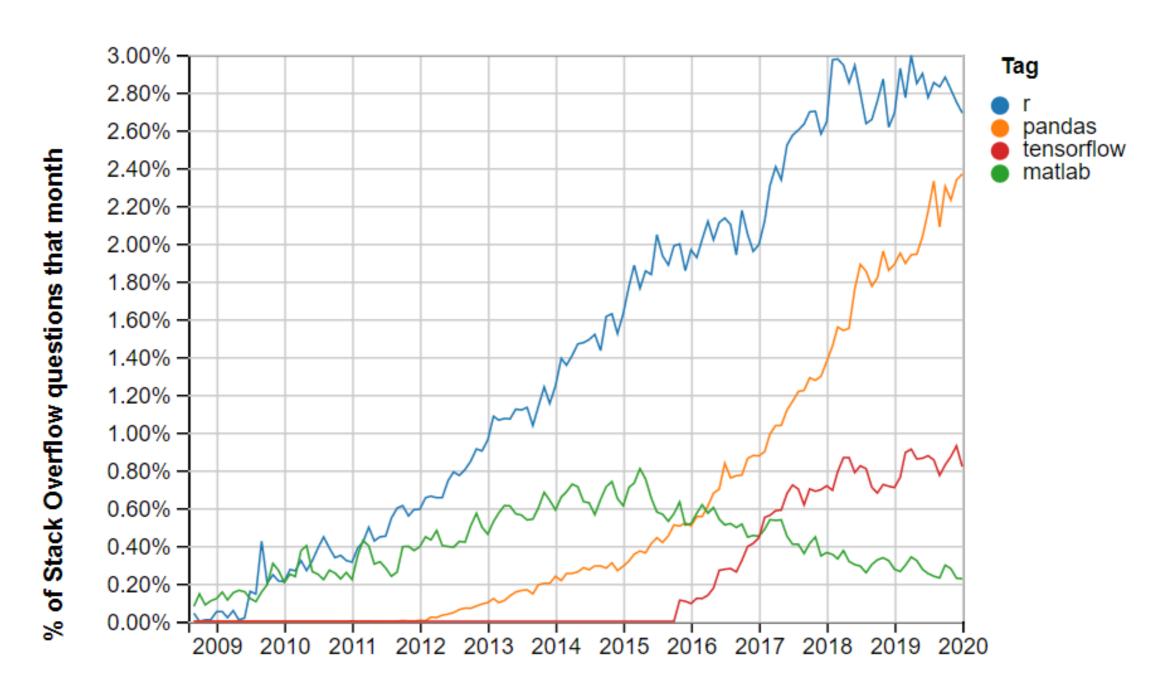
Fonte: <u>Stackoverflow</u>

TIOBE: Índice de Popularidade das linguagens

- Baseado na frequência de buscas
 - isso pode refletir o número de engenheiros hábeis, cursos e vagas de emprego no mundo todo
- Fontes de informação
 - Google, Bing, Yahoo!, Wikipedia, Amazon, YouTube e Baidu

Feb 2020	Feb 2019	Change	Programming Language	Ratings	Change
1,	1		Java	17.358%	+1.48%
2	2		С	16.766%	+4.34%
3	3		Python	9.345%	+1.77%
4	4		C++	6.164%	-1.28%

Python/Pandas vs R



Vantagens do Python

- Extenso suporte de bibliotecas
 - NumPy, Pandas, Flask e etc.
- Open source e desenvolvido em comunidade
- Fácil de aprender
- Estruturas de dados amigáveis
- Linguagem tipada dinamicamente
 - não é necessário definir previamente o tipo de dados com base no valor atribuído

Vantagens do Python

- Linguagem orientada a objetos
- Possui um console interativo
- Portável em diversos sistemas operacionais
- O código Python é significativamente menor que o código C ++ / Java equivalente.
 - Isso implica que há menos para digitar, depurar e manter.

Desvantagens do Python

- Velocidade / Desempenho computacional
 - Se precisar de alto desempenho, dê preferência para C ou C++.
 - ✓ Velocidade de execução pode não ser tão importante quanto velocidade de desenvolvimento*.
- Consumo de memória:
 - Python consome mais memória do que outras linguagens como C ou C++.
- Desenvolvimento Mobile, aplicativos embarcados e IoT:
 - Java e outras linguagens oferecem mais suporte para desenvolvimento de aplicativos de celular e IoT.

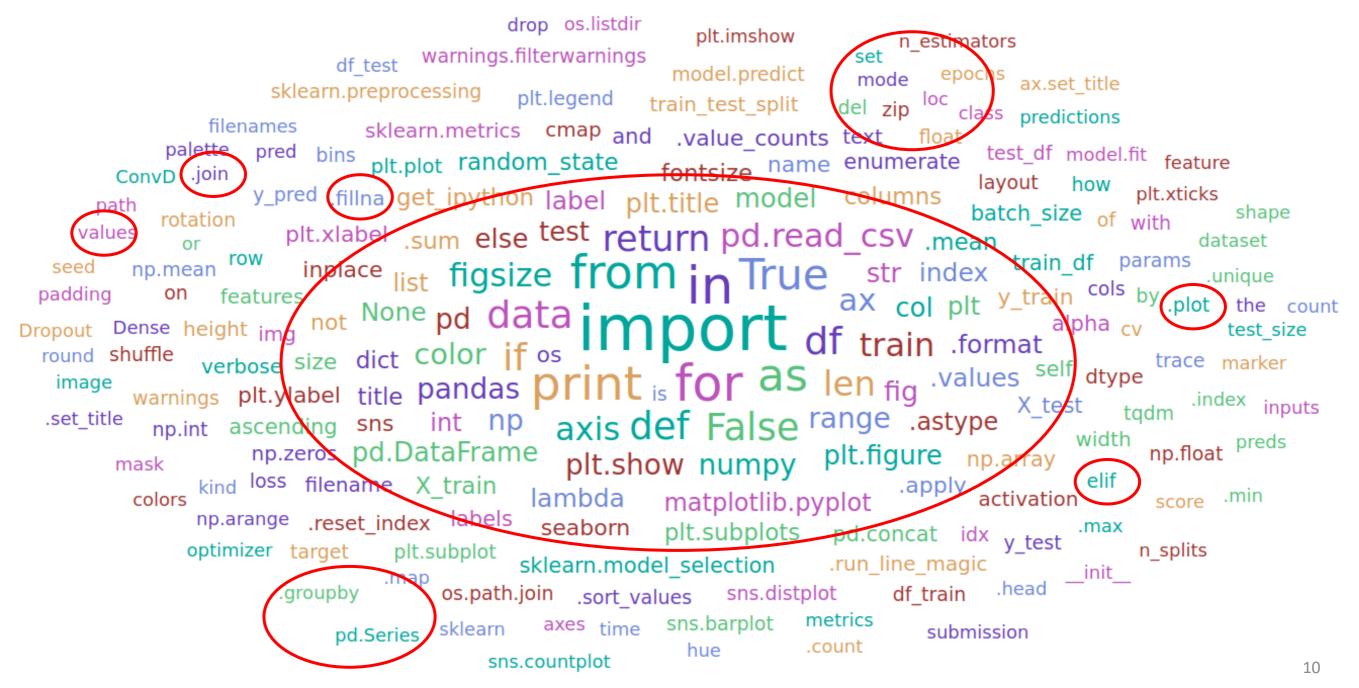
Capacitando-se em TI com eficiência

- Trade-off capacitação prévia vs Busca de informação sob demanda
 - Afiar o machado
- Como eu programo:
 - o interativamente, sem focar em memorizar sintaxe/nomes,
 ✓ e sabendo perguntar/pesquisar.
- Buscas e Stackoverflow
- Comece com pequenas iterações
- Participação ativa

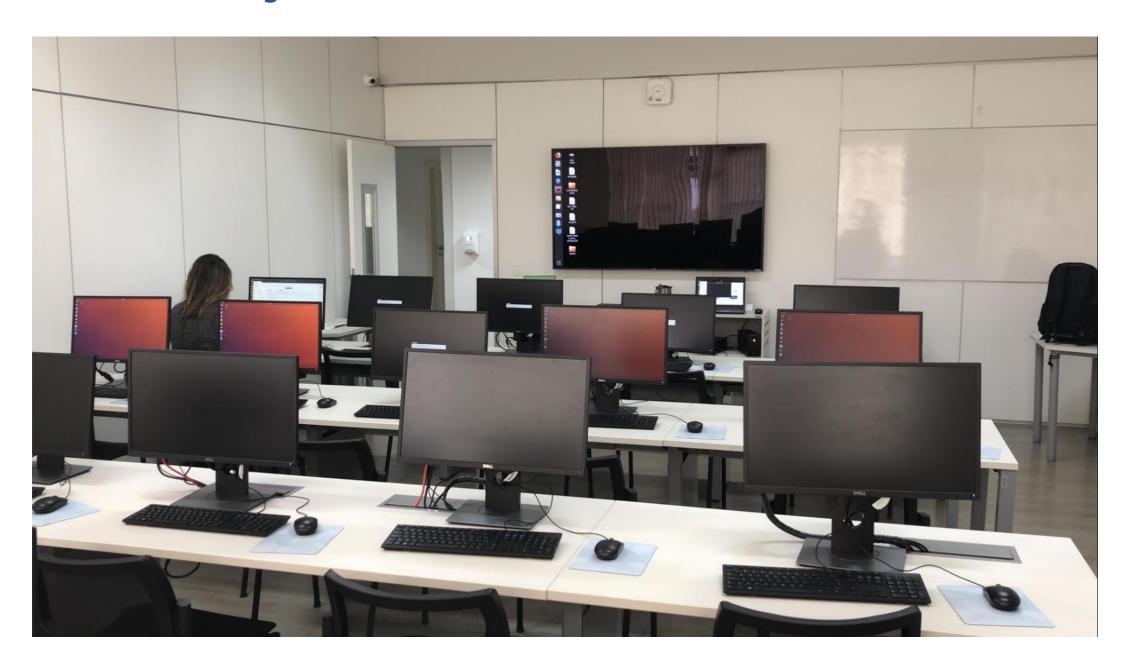
Nuvem de Palavras de Arquivos Python do Kaggle

```
drop os.listdir
                                                                                                                                                                                                                                                     n_estimators
                                                                                                                                                                                                    plt.imshow
                                                                                              df_test warnings.filterwarnings model.predict set mode epochs ax.set_title
                                                                           sklearn.preprocessing plt.legend train_test_split del zip loc class predictions
                                            filenames sklearn.metrics cmap and .value_counts text float palette pred bins plt.plot random_state fontsize name enumerate layout how plt.sticks
                                                               y_pred .fillna get_ipython label plt.title model columns batch_size of with
                             ConvD .join
                                          or plt.xlabel .sum else test return pd.read_csv .mean train df par
seed np.mean row inplace list figsize from in True str index train_df params unique ax col plt y_train cols by .plot the count alpha cv test_size round shuffle verbose size dict color if os warnings plt.ylabel title pandas print is for as len fig .values self dtype inplied in the count of train self train index inputs ax col plt y_train cols by .plot the count alpha cv test_size round shuffle verbose size dict color if os inputs is for as len fig .values self dtype inputs index inputs axis def False range .astype width property preds
                                                    np.zeros pd.DataFrame plt.show numpy plt.figure np.array elif np.arange recet index labels np.arange re
                                                    np.arange .reset_index labels seaborn plt.subplots pd.concat idx y_test .max
                                                  optimizer target plt.subplot sklearn.model_selection .run_line_magic
                                                                              .map .map .map os.path.join .sort_values sns.distplot df_train .head __init__
                                                                                              pd.Series sklearn axes time sns.barplot metrics
                                                                                                                                                                                                                                                                        submission
                                                                                                                                                                                                                                        .count
                                                                                                                                                                                             hue
                                                                                                                                           sns.countplot
```

Nuvem de Palavras de Arquivos Python do Kaggle



Concentração na Aula — Durante Aula Prática

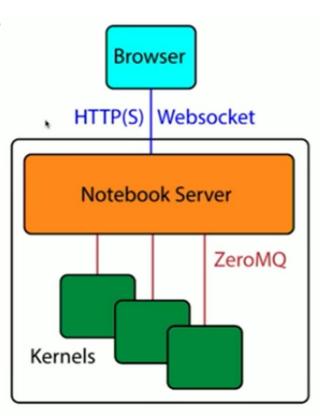


Concentração na Aula - Ulisses e o canto das sereias

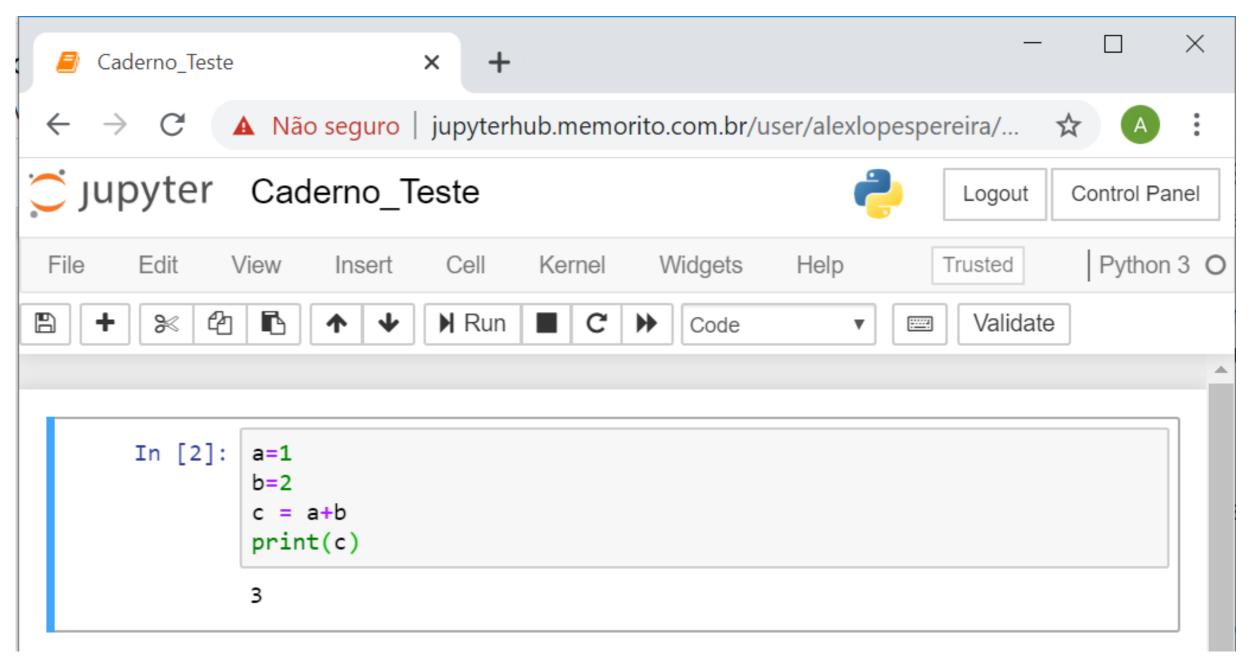


Introdução ao Jupyter Notebook

- O que é
 - Ferramenta de programação no navegador;
 - Códigos, instruções e resultados são mostrados juntos;
 - Útil para escrever códigos que contam uma história; e
 - Utilizado por estudantes, cientistas e pesquisadores.
- Como é implementado
 - É um servidor web local.
 - Abre uma página no navegador.
 - Suporta diversas linguagens de programação
 - ✓ Entre elas o Python.
- Boa prática de programação no Jupyter notebook
 - Programar iterativamente: validando cada pequeno resultado



Jupyter Notebook



Jupyter Notebook: Como usar?

- Preste atenção na demonstração (live coding) do professor.
 - Você terá tempo para praticar sozinho.
- Interação básica com o Jupyter Notebook
- Clicar em Play ou tecle SHIFT+ENTER para executar uma célula
 - Os números entre colchetes indicam a ordem de execução dos comandos.
 - Um asterisco entre colchetes indica que o código está sendo executado.
- Se você **reiniciar** o notebook o conteúdo das variáveis é **perdido**.
- Leia as mensagens de log de erro (elas são úteis).
- https://jupyter.enap.gov.br

Atalhos de Teclado muito úteis

- ESC
 - Sai do modo de edição e entra no modo de comandos
 - ✓ Pode-se sair do modo de edição clicando fora das células
- SHIFT+ENTER
 - Executa a célula atual e passa o cursor para a próxima célula;
- CTRL+ENTER
 - Executa a célula atual e mantém o cursos na mesma célula;
- B (Below) / A (Above) no modo de comando
 - Adiciona uma célula abaixo/acima da célula selecionada
- - Move células para baixo / cima

Sintaxe do Python

- Dois pontos sinaliza o início de uma sentença composta
 - O conteúdo das sentenças compostas é aninhado com espaços/tabs
 ✓ Sem chaves (sem a keyword end)
- Ponto e virgula para finalizar uma sentença é opcional

```
if x > 0
# adiciona 1 a x
x = x + 1;
puts x
puts 'x maior que zero'
else
puts 'x menor ou igual a zero'
end
```

```
3
x maior que zero
```

```
3
x maior que zero
```

Sintaxe do Python

- Referências para objetos não possuem um tipo associado
 - Não há problema em reatribuir uma referência a outro tipo
 - ✓ Por exemplo:

```
x = 2
print(type(x))
x = 'carro'
print(type(x))

<class 'int'>
<class 'str'>
```

```
x = 2
puts x.class
x = 'carro'
puts x.class

Integer
String
```

Sintaxe do Python

[2]

- Funções são declaradas com a palavra-chave
 - o def
- Argumentos são passados por referência para objetos
 - Na chamada de funções

```
def insert_1(a):
    a.append(1)
b = [2]
insert_1(b)
print(b)
[2, 1]
```

Dados do tipo Escalar

- Guardam valores únicos (single) ou simples
- None
 - A representação de valor nulo em Python
- str
 - Conjunto de caracteres. Guarda strings codificadas com UTF-8
- bytes
 - bytes ASCII (ou bytes codificados como Unicode)
- float
 - Número em formato ponto flutuante de 64-bit (Exemplo: 3.1290)
- bool
 - Um valor True ou False
- int
 - Um número inteiro

Formando uma string (texto)

- Strings podem ser definidas usando aspas simples ou duplas
 - 'aluno' ou "aluno" são válidos

```
a = 4.5
b = 2
frase = f"a={a}, b={b}"
print(frase)
```

```
a=4.5, b=2
```

```
a = 4.5
b = 2
frase = "a=#{a}, b=#{b}"
puts frase
```

Funções Populares de String

• join

 cria uma string concatenando uma lista de objetos com um serparador (vírgula, por exemplo)

strip

- remove caracteres (em branco*) do início e do fim de uma string
- upper / lower
 - Transforma os caracteres e maiúsculos / minúsculos
- replace
 - substitui todas as ocorrências de uma string por outra
- split
 - divide uma string em várias outras. O ponto de divisão é o local do separador especificado

Imports (require do Ruby)

- Um módulo é um arquivo de extensão .py
 - Útil para reusabilidade de código

```
# modulo_exemplo.py
def insert1(a):
     a.append(1)
from modulos.modulo_exemplo import insert1
b = [2]
insert1(b)
print(b)
[2, 1]
```

Operadores Binários

Operação	Descrição	
a + b	Soma a e b	
a - b	Subtrai a de b	
a * b	Multiplica a por b	
a / b	Divide a por b	
a // b	Divisão inteira de a por b. Desconsidera o resto.	
a ** b	Eleva a à potência b	
a & b	True se a e b são True	
a b	True se a ou b são True	

^{*} Em negrito os mais importantes

Operadores Binários

Operação	Descrição
a == b	True se a éigual a b
a != b	True se a não é igual a b
a < b, a <= b	True se a é menor (ou menor ou igual) a b
a > b, a >= b	True se a é maior (ou maior ou igual) a b
a is b	True se a e b referenciam o mesmo objeto Python
a is not b	True se a e b referenciam objetos Python distintos
a ^ b	Operação de XOR. True se a ou b são True, mas não ambos.

^{*} Em negrito os mais importantes

Conversão de Tipos

- •str, bool, int, e float
 - o também são funções para converter valores para esses tipos

```
s = '3.14159'
fval = float(s)
ival = int(fval)
print(fval)
print(ival)
3.14159
3
```

Laços (loops) do tipo for

- Uma maneira de iterar sobre uma coleção
- Usa-se o keyword **in** para referenciar a coleção.
 - o **val** foi um termo escolhido pelo programador
 - break interrompe a iteração

```
sequence = [1, 2, 0, 4, 6, 5, 2, 1]
tot_until_5 = 0
for val in sequence:
     if val == 5:
           break
     tot_until_5 = tot_until_5 + val
print(tot_until_5)
```

```
tot_until_5 = 0
[1,2,0,4,6,5,2,1].each do |val|
    if val == 5
         break
    end
    tot_until_5 = tot_until_5 + val
end
puts(tot_until_5)
13
```

Laço do tipo for com a função range

- range(5)
 - o retorna um iterator da sequência 0, 1, 2, 3, 4
- range(2, 6)
 - o retorna um iterator da sequência 2, 3, 4, 5

```
for val in range(3):
    print(val*val)
```

```
0
1
4
```

```
(0..2).each do |element|
puts element * element
end
```

```
014
```

Estruturas de Dados do Python (Nativas)

list (Lista) / array (Ruby)

- Sequência de tamanho variável e conteúdo mutável (alterável)
 - Pode conter objetos de vários tipos
 - Define-se uma lista com colchetes []
 - o append (inserir elementos), pop (remover elementos pelo índice)

```
al = [2, 4, 0, None]
print(al)
al.append(9)
print(al)
al.pop(1) # Remover pelo indice
print(al)
al.remove(0) # Use remove para remover pelo valor
print(al)
```

[2, 4, 0, None] [2, 4, 0, None, 9] [2, 0, None, 9] [2, None, 9]

Combinando Listas

- Use o operador + ou a função extend
 - A função extend é mais rápida do que o operador +

```
al = [2, 4, 0, None]
al_plus = al + ['a', 'b']
al.extend(['a', 'b'])
comp = al_plus == al
print(comp)
```

True

Slicing (fatiar)

• Use intervalos entre colchetes para fatiar sequências

```
al = [2, 4, 0, 3, 7, 10, 4, 5]
print(al[0:3]) # Do indice zero até o 3 (não incluso)
print(al[:4]) # Do indice zero ao 4
print(al[2:]) # Do indice 2 até o último
print(al[-1]) # 0 último element
print(al[::2]) # A cada dois elementos
print(al[::-1]) # Reverter/espelhar os elementos
al[1:3] = [8, 8]
print(al)
```

dict (Dicionário) / hash (Ruby)

- Uma coleção de key-value (chave-valor) de tamanho flexível
 - O key e o value podem ser (quase) qualquer objeto python

```
empty_dict = {} # Cria um dicionário vazio
d1 = {'tipo' : 'carro', 'ano':2020, 'ocup' : [1, 2]}
d1['cor'] = 1 # Cria um novo par chave-valor
d1['fabric'] = 3 # Cria um novo par chave-valor
print(d1)
print(d1['ocup'])
{'ocup': [1, 2], 'ano': 2020, 'fabric': 3, 'tipo': 'carro', 'cor': 1}
[1, 2]
```

dict (Dicionário) / hash (Ruby)

Outros métodos relacionados a dicionários

```
d1 = {'tipo' : 'carro', 'ano':2020, 'ocup' : [1, 2]}
print('tipo' in d1) # Testa: 'tipo' está em d1.keys()?
del d1['ano'] # Remove o par identificado por 'ano'
ret = d1.pop('ocup') # Remove do dict e retorna o valor
print(list(d1.keys()))
d1.update({'motor' : '18c', 'dono' : 12}) #atualiza
print(d1)
```

```
True
['tipo']
{'motor': '18c', 'dono': 12, 'tipo': 'carro'}
```

NumPy (Numerical Python)

Pacote Python para computação numérica

Recursos do Numpy

- Computação eficiente com array multi-dimensional
 - o armazena dados numa região contínua de memória;
 - As funções numpy escritas em C podem operar diretamente na memória.
- Funções matemáticas eficientes/rápidas em arrays
 - sem a necessidade de escrever loops (laços)
- Ferramentas para ler e escrever arrays do disco
- Álgebra linear, geração de números aleatórios, entre outros
- API C para integração com Código escrito em C/C++/Fortran

Operações vetorizadas em ndarrays

- Funções e operações em várias dimensões sem loops
 - Mais eficientes (preferíveis)

```
import numpy as np
data = np.random.randn(2, 3) # Gerar dados aleatórios. 2L x 3C
print(data)
print(data * 10)
print(data + data)
```

Instruções inline (na mesma linha)

```
a = 2
# Sintaxe: valor_se_verdadeiro if condicao else valor_se_falso
b = 0 if a > 10 else 1
print(b)
# [ OPERACAO_VAL for VAL in SEQUENCIA ]
resultado1 = [x for x in range(10)] # list comprehension
resultado2 = [x * x for x in range(10)]
print(resultado1)
print(resultado2)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

Comparação de Desempenho: Numpy array vs list

```
# import numpy as np => Importar se já não tiver importado
my_arr = np.arange(10000000) #cria array de números sequenciais
my_list = list(range(10000000))
# %time mede o tempo tomado pela execução da linha
%time my_arr2 = my_arr * 2
%time my_list2 = [x * 2 for x in <math>my_list]
print(my_arr2[1:5])
print(my_list2[1:5])
```

Wall time: 32 ms

Wall time: 1.15 s

[2 4 6 8]

[2, 4, 6, 8]

30

Criando ndarrays

```
data1 = [6, 7.5, 8, 0, 1]
arr1 = np.array(data1)
print(arr1)
data2 = [[1, 2, 3, 4], [5, 6, 7, 8]]
arr2 = np.array(data2)
print(arr2)
print(np.zeros(5)) # Função que cria um array de zeros
```

```
[6. 7.5 8. 0. 1.]
[[1 2 3 4]
[5 6 7 8]]
[0. 0. 0. 0. 0.]
```

Atributos do ndarrays (alguns)

```
# 0 tipo do dado é inferido. Mas também pode ser especificado
arr1 = np.array([6, 7.5, 8, 0, 1])
print(arr1.dtype) # tipo do dado
arr2 = np.array([[1, 2, 3, 4], [5, 6, 7, 8]], dtype=np.int32)
print(arr2.shape) # Forma/dimensões do array
print(arr2.ndim) # Quantidade de dimensões
```

float64

(2L, 4L)

```
41
```

Numpy data types (tipos de dados)

Tipo básico	Tipo Numpy disponív.	Comentários
Boolean	bool	Tamanho de 1 byte
Integer	int8, int16, int32, int64, int128, int	int tem o tamanho do int padrão da plataforma C
Unsigned Integer	uint8, uint16, uint32, uint64, uint128, uint	uint tem o tamanho do uint padrão da plataforma C
Float	float32, float64, float, longfloat	Float é sempre de precisão dupla (64 bits). longfloat é um float maior cujo tamanho depende da plataforma.
Complex	complex64, complex	A parte real e a imaginaria de um complex64 ocupam cada um 32 bits
String	str, Unicode	Unicode é sempre UTF32
Object	Object	Representa itens em um array de objetos Python

Conversão de tipos (cast)

```
num_str = np.array(['1.25', '-9.6', '42'], dtype=np.string_)
arr_float = num_str.astype(float) # Converte para float
print(arr_float)
arr1 = np.array([3.7, -1.2, -2.6, 0.5, 12.9, 10.1])
arr1_int = arr1.astype(np.int32) # Converte para int32
print(arr1_int)
[1.25 - 9.6 42.]
[3-1-2 0 12 10]
```

Aritmética com NumPy Arrays

```
arr = np.array([1., 2., 3.])
arr2 = np.array([4., 5., 6.])
print(arr * arr)
print(arr - arr)
print(1 / arr)
print(arr ** 0.5)
print(arr2 > arr)
[1. 4. 9.]
[0. \ 0. \ 0.]
[1. 0.5 0.33333333]
[1. 1.41421356 1.73205081]
[True True True]
```

Slicing (fatiar) NumPy Arrays

- Nas listas os slices são cópias;
- Nos arrays numpy os slices são views (visualizações)
 - para copiar, use a função: .copy() , exemplo: arr[1:4].copy()

```
arr = np.arange(10)
print(arr)
arr_slice = arr[5:8]
arr_slice[1] = 12345
print(arr)
li2 = list(range(10)) # 0 equivalente com uma lista
list_slice = li2[5:8]
list_slice[1] = 12345
print(li2)
```

Slicing (fatiar) de arrays bidimensionais

```
arr2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print(arr2d.shape)
print(arr2d[2])
                                                                axis 1
print(arr2d[0][2]) # Tanto faz
print(arr2d[0, 2]) # Tanto faz
                                                          0,0
                                                                 0, 1
                                                                         0,2
(3L, 3L)
[7 8 9]
                                             axis 0
                                                          1,0
                                                                 1,1
                                                                         1, 2
                                                          2,0
                                                                 2, 1
                                                                         2,2
```

Jupyter Hub e NBGrader

Jupyter Hub

- Login com sua conta do Github
- 1 container na nuvem por aluno, com
 - ✓ 1 processador
 - ✓ E até 4GB de memória RAM

NBGrader

- Ferramenta de automação de correção
 - ✓ De cadernos/notebooks Jupyter
- Etapas de uso do Jupyter auxiliado pelo NBGrader
 - ✓ Fetch: recebe um notebook publicado;
 - ✓ Submit: submete notebook para correção/revisão; e
 - ✓ Fetch Feedback: recebe feeback de uma submissão.

Teste com o NBGrader

- 1) Logar com sua conta no JupyterHub
 - Endereço: jupyter.enap.gov.br
- 2) Fetch do Assignment Aula1_Ex1.ipynb
- 3) Resolução do caderno Aula1_Ex2.ipynb
- 4) Submit do caderno Aula1_Ex1.ipynb
- 5) Fetch feedback do Assignment Aula1_Ex1.ipynb