

## Objetivos

## Abordagem

desenvolver raciocínio lógico e capacidade  
escrever algoritmos utilizando a  
linguagem de programação Python

aprender estruturas de dados  
disponibilizadas pela linguagem e escolher  
soluções convenientes para problemas de  
diversos

render a ler e escrever em arquivos e  
criar expressões e programação  
thread quando necessário.

- Apresentação da teoria de forma graduada passando pelos tipos, estrutura de dados, controle de fluxo, laços, arquivos, tratamento de exceções

- Demonstração com exemplos práticos de cada tópico apresentado

- Aplicação prática na forma de exercícios serem resolvidos, exigindo o emprego de raciocínio lógico e conhecimentos adquiridos na linguagem

## Sérios de Avaliação

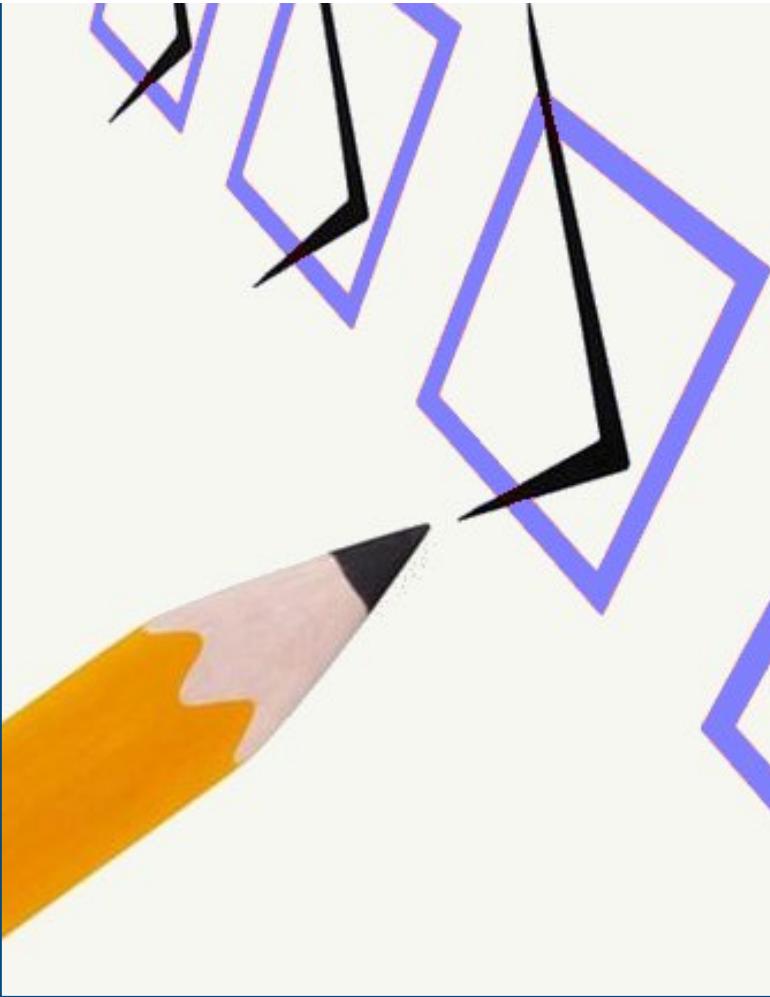
final de cada aula, será disponibilizada  
a lista de exercícios práticos  
da lista de exercícios valerá 3,5 (três  
meio e meia)

listas deverão ser entregues até o prazo  
terminado

tas entregues após o prazo determinado,  
serão consideradas

Iota final do módulo será a somatória da  
ntuação de cada lista de exercícios

ercícios que não estiverem em  
nprimento com as instruções não serão  
nsiderados



## ve-se

fina objetivos (metas e prazos) e  
riodicamente verifique os resultados. Faça  
stes quando necessário  
lique os conhecimentos adquiridos neste  
'so no seu cotidiano  
desafie, tente algo além do proposto  
lbalhe de forma colaborativa, isso o  
dará a melhorar seu desempenho  
atique. Quando terminar, pratique mais.  
oportunidades normalmente se  
resentam disfarçadas de trabalho árduo e  
or isso que muitos não as reconhecem  
ernardinho)  
or último e não menos importante, divirta-  
se sempre!



Did you  
know?



## Linguagens de Programação

### Java

Código Binário ou linguagem de máquina  
Assembly

### C / C++

Compiadas

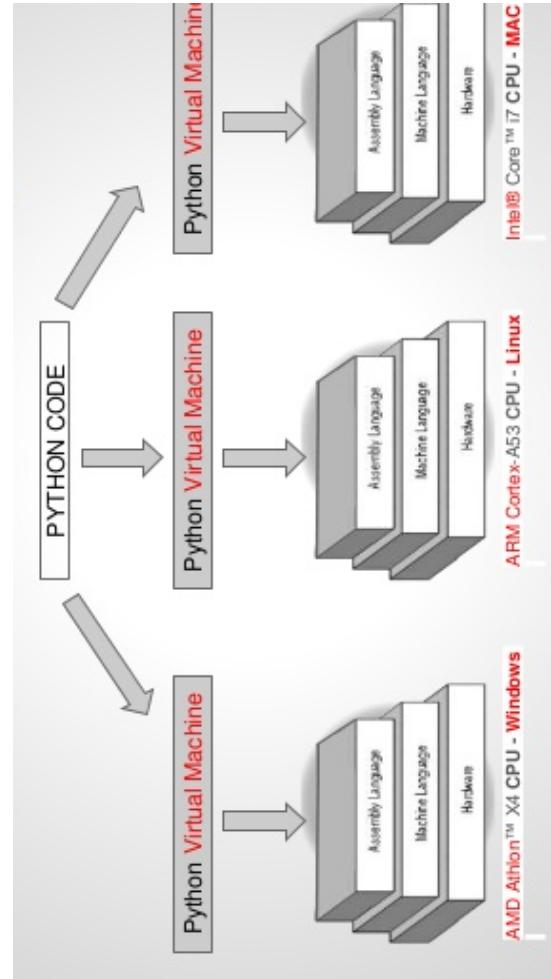
- C / C++
- Pascal
- Delphi

### Interpretadas

- Java
- C#
- PHP
- Javascript
- Python

### Código Intermediário

- Bytecode
- JIT Compilation
- Virtual Machine

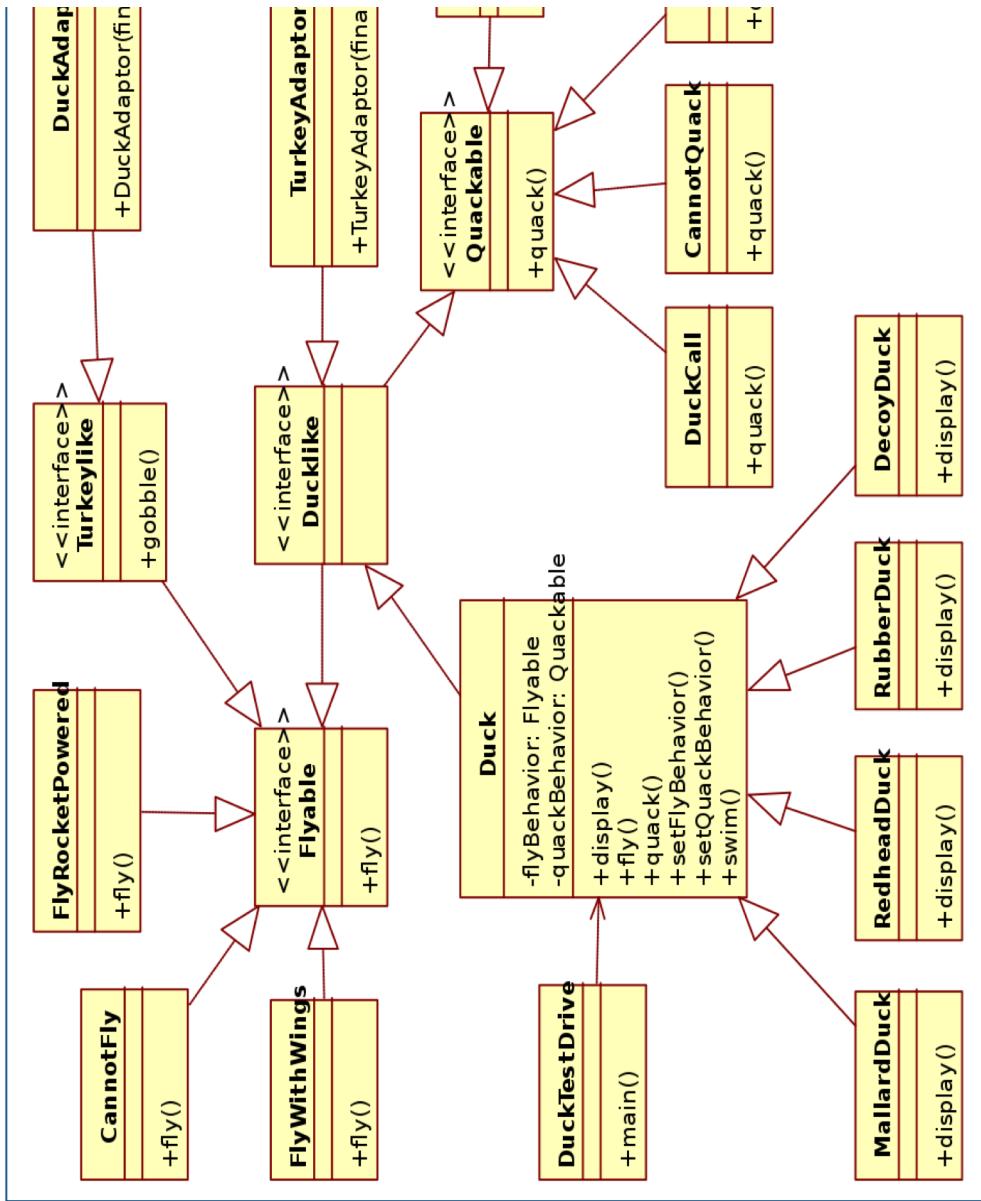


## Importância da Análise Orientada à Objetos

stração  
lomorfismo  
rança  
oplamento  
esão

utilização  
IL

ber construir e interpretar  
gramas (Casos de Uso, Classes,  
quencia)



## S Práticas de Programação

### de Conventions

<https://www.python.org/dev/peps/pep-0008/#an-Code>

Fácil de entender

Direto ao ponto

Eficiente

Elegante

mes Significativos (substantivos e verbos)  
cumentação  
mentários

stes Unitários

de Review

- Dicas
  - Antes de iniciar a programação tenha ce| do entendimento do problema
  - Adquira o hábito de comentar seu código momento em que estiver programando, preferencialmente antes de escrever a instrução
  - Seja breve e direto nos comentários, evit escrever o que não é relevante
  - Dê preferência aos comentários em inglé imaginou ler um código com comentários Latin? 😊
  - Não exagere, pois comentários em excesso podem atrapalhar ao invés de ajudar, prc comentar trechos de código

## Je é um Algoritmo?

oritmo é uma sequência finita de instruções bem definidas e não ambíguas, da uma das quais devendo ser executadas mecânica ou eletronicamente em um intervalo de tempo finito e com uma quantidade de esforço finita. (Wikipedia)

Aplicadamente um algoritmo é uma reita, um conjunto de instruções bem definidas para solucionar um problema решido.

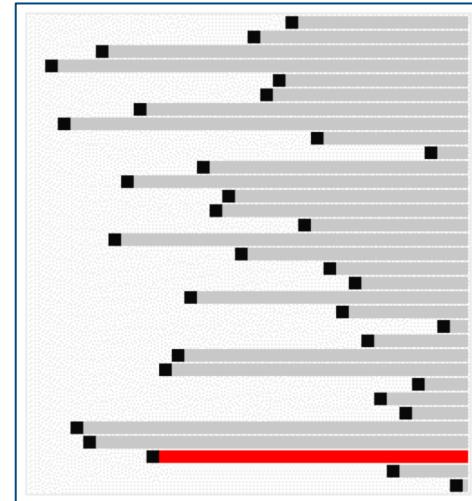
:a: Algoritmo não se aprende copiando ou usando algoritmos prontos e sim construindo e testando seus próprios oritmos



## Algoritmo BubbleSort

corra o vetor inteiro comparando  
mentos adjacentes (dois a dois)

que as posições dos elementos se eles  
iverem fora de ordem  
pita os dois passos acima com os  
meiros  $n-1$  itens, depois com os primeiros  
 $n-2$  itens, até que reste apenas um item



```
BubbleSort.py X
1  def bubbleSort(alist):
2      for passnum in range(len(alist) - 1, 0,
3          for i in range(passnum):
4              if alist[i] > alist[i + 1]:
5                  temp = alist[i]
6                  alist[i] = alist[i + 1]
7                  alist[i + 1] = temp
8
9  alist = [111, 245, 54, 26, 93, 17, 77, 31, 44, 55, 2
10 bubbleSort(alist)
11 print(alist)
12
```

```
BubbleSort X
C:\dev\workspace_git\DataScienceFacens\HelloWorld
[17, 20, 26, 31, 44, 54, 55, 77, 93, 111, 245]
Process finished with exit code 0
```

PRACTICE MAKES PERFECT



## 1 Breve Introdução

Python é uma linguagem de programação desenvolvida

- Guido van Rossum no final da década de 1980 com o objetivo de ser fácil e intuitiva, porém poderosa.

as principais características são:

Linguagem de programação de alto nível

Interpretada e de código-fonte aberto

Interativa

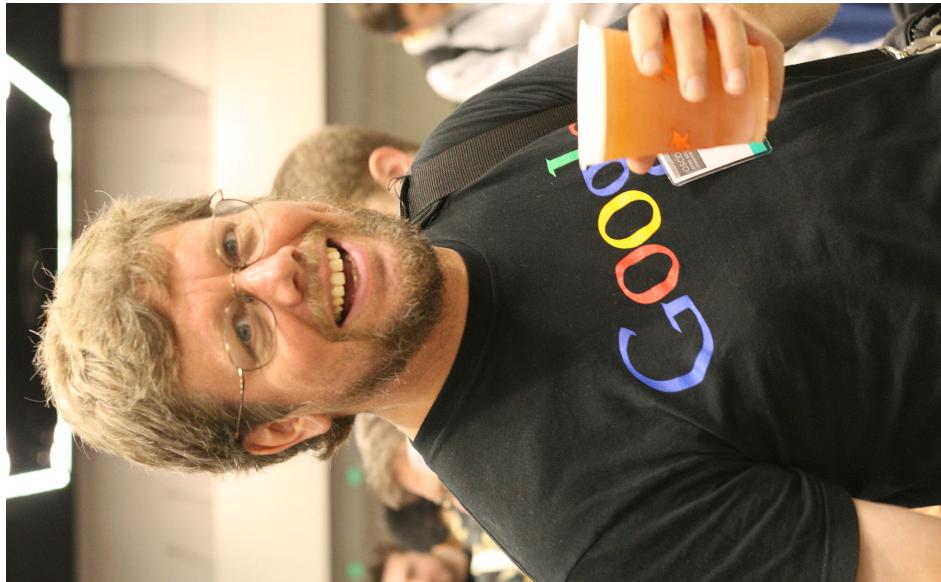
Multi-plataforma e Multi-paradigma

Sintaxe simples, fácil de aprender e de manter

Tipagem forte e dinâmica

Tudo em Python é um objeto: variáveis, funções, etc. Cada objeto tem um ID, tipo e valor

riosamente o nome não tem nenhuma relação com o ríbio de mesmo nome e sim uma homenagem ao grupo comédia britânico Monty Python!



## Ambiente de Desenvolvimento

### Versus Ambiente de desenvolvimento

objectos a serem considerados  
Estrutura e organização dos códigos fontes

Compilador e executor integrado

Ferramenta para depuração (debug)

Ferramenta para inspeção (variable inspect)

Gerenciador de pacotes

Consumo de recursos

### Ambientes de Desenvolvimento para Python

PyCharm

IDLE

Anaconda

- Visual Studio Code
- Atom

- Ibientes Interativos
  - IPython
  - Jupyter Notebook

<https://www.jetbrains.com/pycharm/>



## ion pip

tema de instalação e gerenciamento de pacotes do Python

liza o Python Package Index ou *PyPI* como repositório

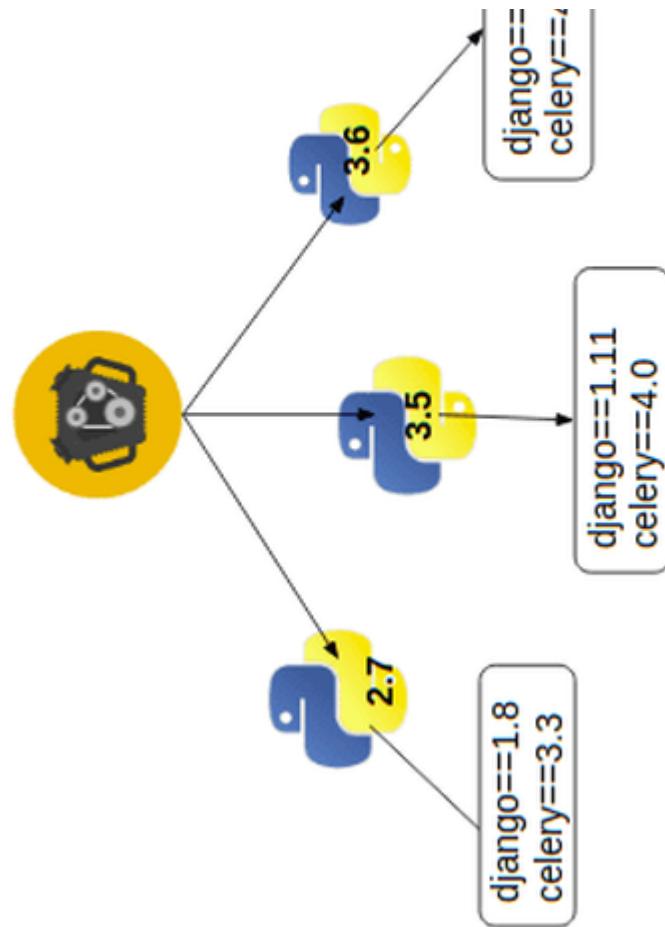
[os://pypi.org/](https://pypi.org/)

grande vantagem da utilização do *pip* é a facilidade de execução através de linha de comando:

**pip install algum-nome-de-pacote**

**pip uninstall algum-nome-de-pacote**





**python**  
virtualenv

## virtualenv

virtualenv é uma ferramenta que facilita o agrupamento de projetos, dependências e bibliotecas em um único lugar. Esse ambiente é específico para os projetos contidos nele e afere as dependências de outros projetos

É possível a existência de projeto X que usa a versão 1.0 da biblioteca Z e também eles é possível o projeto Y usando a versão 2.0 da mesma biblioteca Z

Um ambiente virtual é criado, os executáveis do Python, *pip*, etc são copiados para esse ambiente e todas as bibliotecas instaladas através do *pip* serão mantidas nesse ambiente, não afetando outros ambientes

Nessa forma é possível administrar ambientes completamente diferentes, sem que um interoutra, por exemplo um ambiente usa Python na versão 2.7 e a biblioteca NumPy na versão 3 e outro ambiente com Python na versão 3.6 e a biblioteca NumPy na versão 1.15, cada um desses ambientes com as mesmas bibliotecas em versões diferentes

Este recurso é bastante útil, quando se pretende migrar uma ou mais bibliotecas para versões recentes, sem quebrar a compatibilidade dos seus projetos com as versões anteriores dessas bibliotecas



**ANALOGIA<sup>®</sup>**

## conda

Anaconda é uma plataforma open source para Data Science, muito popular entre os cientistas de dados, estatísticos, cientistas da computação, entre outros

### Plataforma

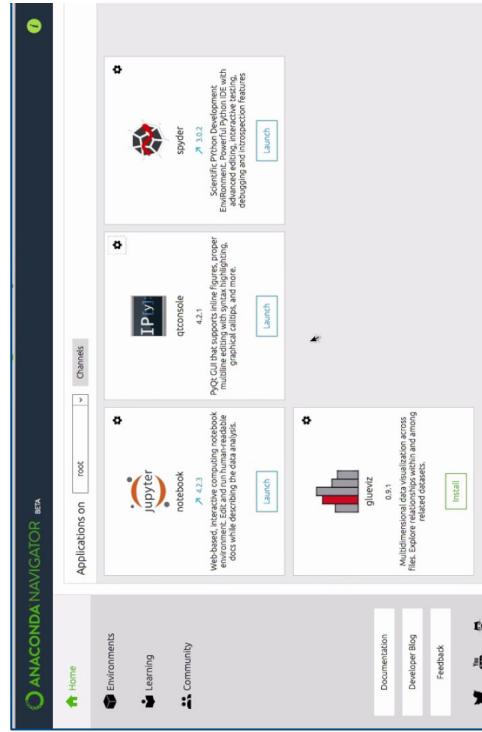
ssui um gerenciador de ambientes virtuais  
tualenv)

tala e se integra com o pip

iponibiliza um repositório com mais de 1000  
liotecas open source para os mais diversos fins  
<https://docs.anaconda.com/anaconda/packages/pkg-conda>)

tre os pacotes disponibilizados na instalação está o  
jupyter

<https://www.anaconda.com/download/>





## Jupyter

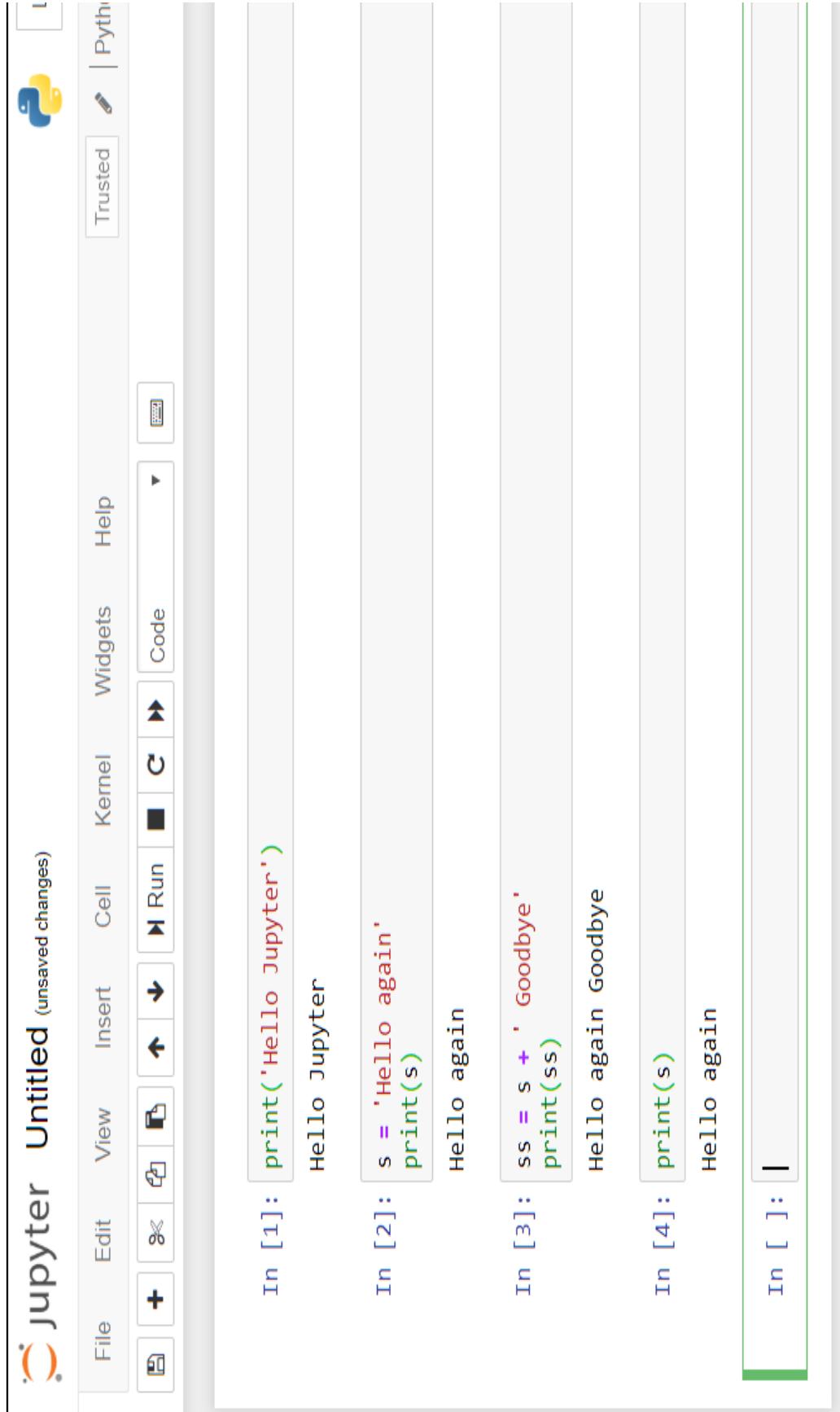
licação open source muito utilizada para ensinar linguagens de programação (*kernels*) de ser executada através de um navegador permite programar de forma iterativa, criar e compartilhar documentos que contenham trechos de código, equações, adicionar notas e visualizar suas ações de forma muito intuitiva e rápida. A integração nativa com ferramentas específicas de big data como Apache Spark por exemplo.

As principais ferramentas para limpeza de dados (data cleaning), transformação (transforming), manipulação numérica, modelagem estatística, visualização de dados, aprendizado de máquina e outras.

Para instalar o Jupyter é através do Anaconda, porém pode ser instalado através do **pip** (pip install jupyter)

[w.jupyter.org](http://w.jupyter.org)

ter



The screenshot shows a Jupyter Notebook interface with the following content:

- Cell 1:** `In [1]: print('Hello Jupyter')`  
Output: Hello Jupyter
- Cell 2:** `In [2]: s = 'Hello again'  
print(s)`  
Output: Hello again
- Cell 3:** `In [3]: ss = s + ' Goodbye'  
print(ss)`  
Output: Hello again Goodbye
- Cell 4:** `In [4]: print(s)`  
Output: Hello again
- Cell 5 (highlighted):** `In [ ]:` (empty cell)

**ipynb**

jupyter

The screenshot shows the Windows Start Menu search results for the query 'jupyter'. The results are as follows:

- Anaconda3 (64-bit) New
- Anaconda Navigator
- Anaconda Prompt New
- Jupyter Notebook New
- Reset Spyder Settings New
- Spyder New

A large blue arrow points upwards from the Anaconda Prompt result towards a blue-bordered terminal window.

**Anaconda Prompt - jupyter notebook**

```
(base) C:\Users\saolnsan>jupyter notebook
[I 22:52:25.485 NotebookApp] The port 8888 is already in use, trying another port.
[I 22:52:25.532 NotebookApp] JupyterLab beta preview extension loaded from C:\dev\python\anaconda3\lib\site-packages\jupyterlab
[I 22:52:25.533 NotebookApp] JupyterLab application directory is C:\dev\thon\anaconda3\share\jupyter\lab
[I 22:52:25.686 NotebookApp] Serving notebooks from local directory: C:\ers\saolnsan
[I 22:52:25.686 NotebookApp] 0 active kernels
[I 22:52:25.686 NotebookApp] The Jupyter Notebook is running at:
http://localhost:8889/?token=844164a39598434eb6a281e6e0bc15bbef
b6a281e6e0bc15bbeb483aea5f92c7
[I 22:52:25.686 NotebookApp] Use Control-C to stop this server and shutdown all kernels (twice to skip confirmation).
[C 22:52:25.689 NotebookApp]
```

Copy/paste this URL into your browser when you connect for the first time, to login with a token:  
[http://localhost:8889/?token=844164a39598434eb6a281e6e0bc15bbeb483aea5f92c7](http://localhost:8889/?token=844164a39598434eb6a281e6e0bc15bbefaea5f92c7&token=844164a39598434eb6a281e6e0bc15bbeb483aea5f92c7)

```
[I 22:52:25.845 NotebookApp] Accepting one-time-token-authenticated connection from ::1
```

## Jupyter - Markdown

• é um sistema simples de formatação de textos que busca tornar simplificar a escrita de textos técnicos

• um conjunto razoavelmente pequeno de códigos, fornece um conjunto deamentos de formatação (símbolos) que automaticamente convertidos para ML:

• ele é possível formatar elementos em íco, negrito, criar citações, listas lenadas e não ordenadas, tabelas, links e chos de código nativo de várias juagens

Jupyter fornece suporte a escrita de chos de código no formato MD o que ilita muito o aprendizado e é uma forma lito eficiente de documentar seus gramas

**Sample Document**

-----

Lorem ipsum dolor sit amet, consectetur adipisicinc elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Unordered list:

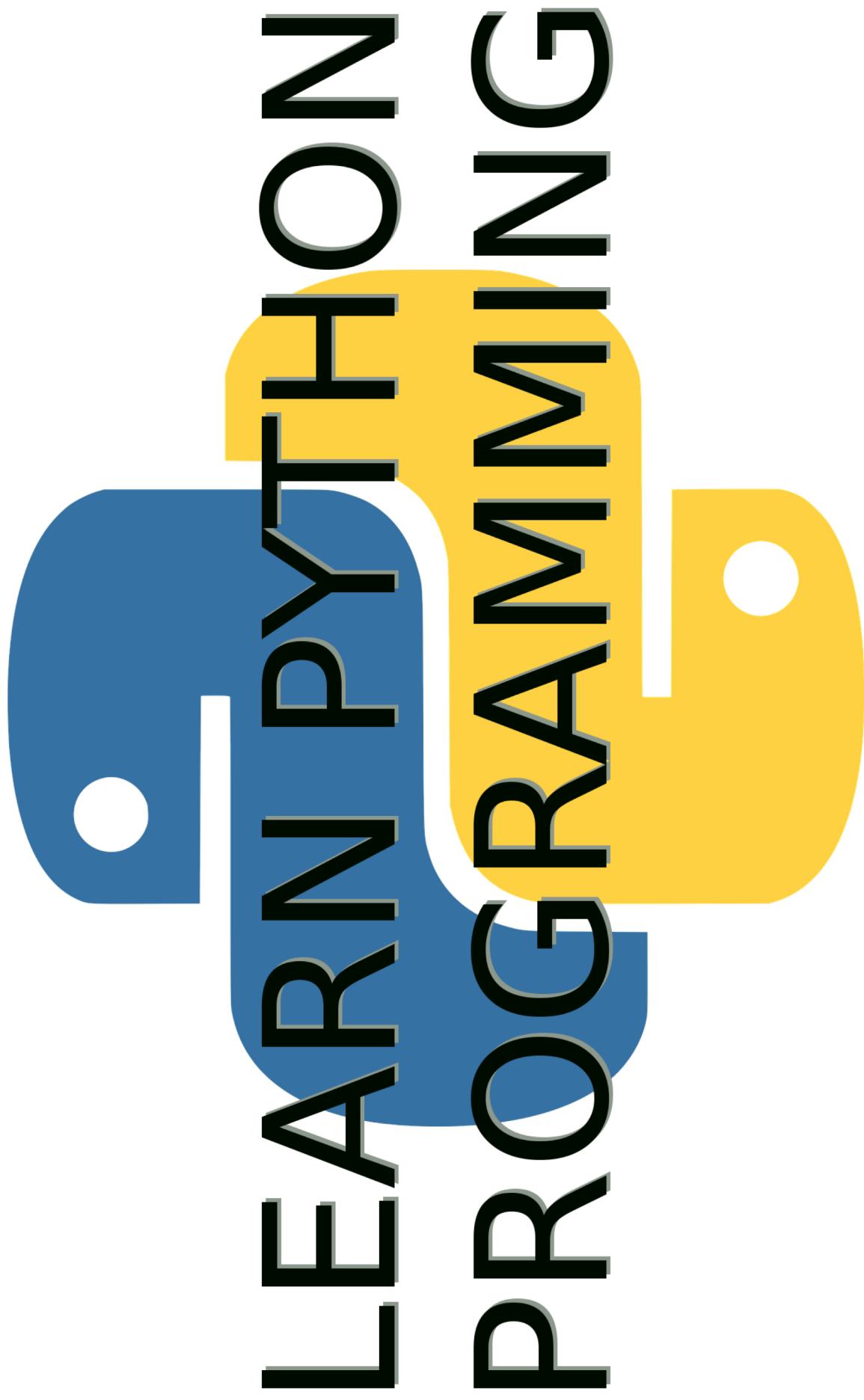
- Banana
- Apple
- Orange

Ordered list:

1. Milk
2. Juice
3. Water

Code block with GeSHi language tag:

```
<?php  
echo "Hello World";
```



## axe Básica

dentação é usada em Python para delimitar blocos de código. O número de espaços pode variar, porém os os *statements* (instruções) dentro de um mesmo bloco, devem possuir a mesma quantidade de espaços.

A primeira linha que compõe *statements* como *if*, *else*, *for*, *def*, e *class* deve ser finalizada com dois pontos : ( ponto e vírgula é opcional ao final de um *statement*)

mentários de apenas uma linha começam por # e de múltiplas linhas por """

Quivos de programas Python tem extensão .py

```
Hello.py X
1 # one line comment
2 if True:
3     print("Resposta")
4 else:
5     print("TRUE")
6     ...
7     multiple lines
8     comment
9     ...
10    print("Resposta")
11
12
```

## áveis

thon é dinamicamente tipado, você não precisa declarar o tipo das variáveis. A declaração acontece automaticamente no momento em que um valor é atribuído à variável.

Variáveis podem mudar de tipo, especialmente com uma nova atribuição de tipo diferente.

thon permite atribuir um único valor às várias variáveis ao mesmo tempo.

sim como permite atribuir múltiplos objetos a várias variáveis.

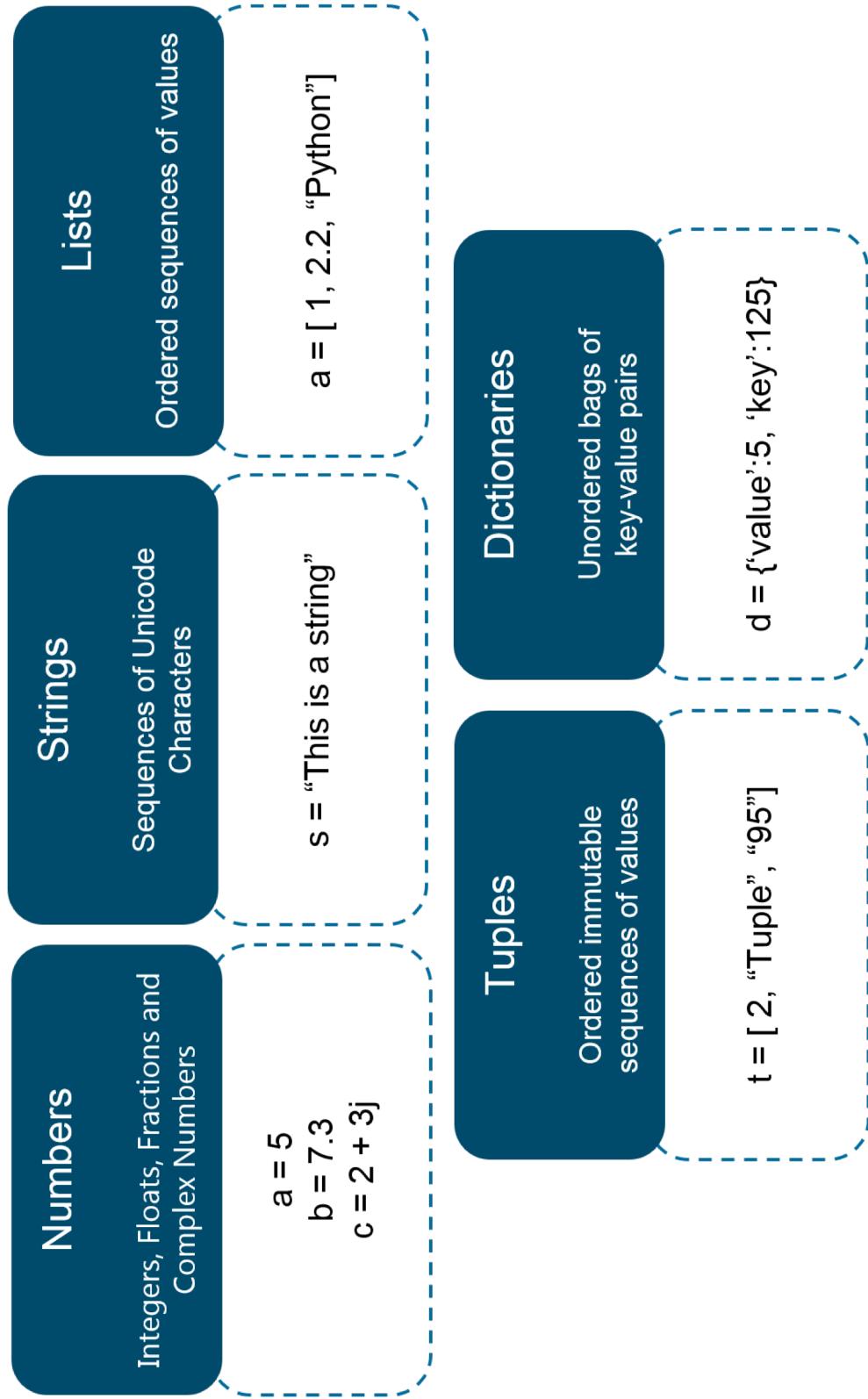
```
max = 10          # integer
value = 150.0     # float point
name = "Python"   # string
nothing = None    # null value
```

```
x = 1
x = "String value"
```

```
a = b = c = 1
```

```
a, b, c = 1, 2, "Python"
```

## S de Dados



## S Numericos

meros são objetos imutáveis em Python, isso significa que os valores não podem ser alterados

tem 3 tipos de dados para números em Python

`int (signed integer)`

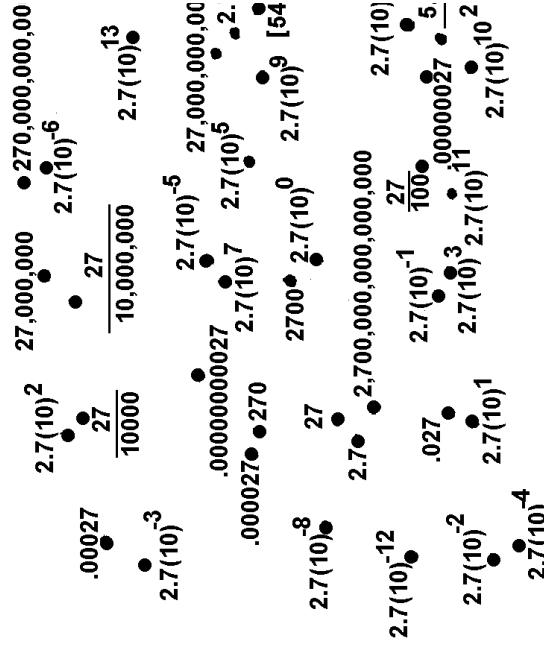
- Também chamado apenas de **integer**, comporta números positivos e negativos sem casas decimais

`float (floating point)`

- Representa números reais e são escritos com ponto decimal, dividindo um inteiro em partes fracionárias, também podendo ser escrito em notação científica com “e” indicando potencia de 10 (ex: `2.5e2`)

`complex`

- São escritos por dois valores reais, a parte real e a parte imaginária na forma (`real + IMAG J`). Neste caso o número i ( $\sqrt{-1}$ ) é designado pela letra j. (Não são muito utilizados)



## Funções mais comuns com tipos numéricos

Função	Descrição
<code>int (x)</code>	Converte x em um inteiro
<code>float (x)</code>	Converte x em um ponto-flutuante
<code>abs (x)</code>	Retorna o valor absoluto de x
<code>exp (x)</code>	Retorna o exponencial de x ( $e^x$ )
<code>log (x)</code>	Retorna o logaritmo natural de x (inverso da função exponencial)
<code>pow (x, y)</code>	Retorna o valor de x elevado à potencia y
<code>sqrt (x)</code>	Retorna a raiz quadrada de x
<code>round (x, y)</code>	Retorna x arredondado em y casas decimais

## 1gs

ing, assim como numbers, são objetos imutáveis em Python. Dessa forma um update só pode ser possível na alocação de um novo objeto com o novo conteúdo.

Python não suporta tipos “char”. Um char em Python é considerado uma String de tamanho 1

• A String pode ser atribuída de várias formas diferentes:

‘Uma String’

“Outro exemplo de String”

”String com multiplas linhas”

Ings iniciam sempre com indice 0

de se usar operações como slicing ([], [:]), concatenação (+), repetição (\*) e membership (in)



## Funções e métodos mais comuns com tipo String

Método	Descrição
isnumeric()	Converte um número em String
len(s)	Retorna o tamanho de uma String
isalpha()	Retorna False se a string contiver algum caractere que não seja letras
isdigit()	Retorna False se a string contiver algum caractere que não seja número
lower()	Retorna a string transformada em minúsculos
upper()	Retorna a string transformada em maiúsculos
replace(old, new)	Substitui uma porção da string por outro conteúdo
strip()	Retira espaços em branco no começo e no fim da string
title()	Retorna a string capitalizada (iniciais em maiúscula)
split(delimitador)	Separa uma string conforme um delimitador. É o inverso do join()
join(sequence)	Junta cada item da string com um delimitador especificado. É o inverso do split().

## 1S

tas são grupos de itens ou elementos indexados, não necessariamente do mesmo tipo

tas são mutáveis, isso significa que seu conteúdo pode ser alterado sem que um novo objeto seja criado

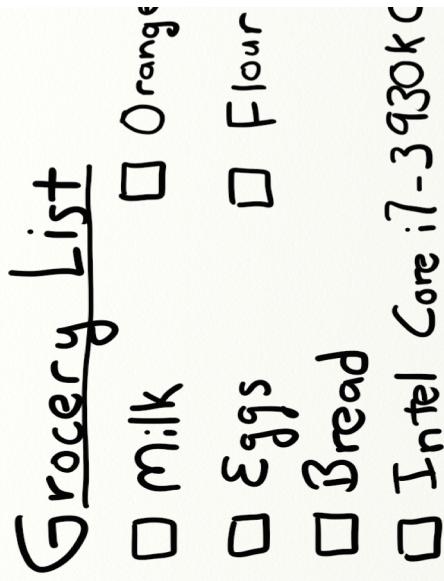
ra atribuir elementos a uma lista é necessário que estejam entre colchetes e separados por vírgula, ex:

```
l = [item1, item2, ... itemn]
```

índices em uma lista sempre se iniciam em 0 (zero)

tas podem conter sublistas que podem conter sublistas e assim por diante, tornando esse tipo de dado extremamente útil

de se usar operações como *slicing* ([], [:]), concatenação, repetição (\*) e *membership* (in)



## Funções e métodos mais comuns com tipo Lista

Função	Descrição
len(lista)	Retorna o tamanho total da lista
max(lista)	Retorna o maior elemento da lista
min(lista)	Retorna o menor elemento da lista
tuple(tupla)	Converte uma tupla em uma lista
append(obj)	Adiciona um objeto à lista
insert(index, obj)	Insere um objeto à lista em determinada posição
count(obj)	Retorna a quantidade de vezes que um determinado objeto ocorre na lista
index(obj)	Retorna o primeiro índice de ocorrência de um determinado objeto
remove(obj)	Remove um objeto da lista
reverse()	Reverte o ordenamento da lista
sort()	Ordena a lista

as

elas são tipos muito semelhantes às listas, porém imutáveis  
esse motivo, tuplas são mais eficientes que as listas, dessa  
ma quando não existe a necessidade de alteração de seu  
conteúdo, devem ser escolhidas preferencialmente  
mo são imutáveis, protegem os dados contidos nela de  
eracões acidentais

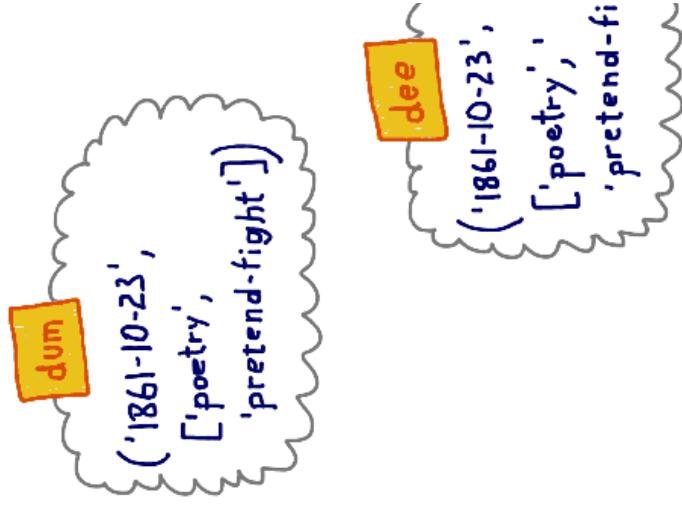
ra atribuir elementos a uma tupla é necessário que estejam  
re parêntesis e separados por vírgula, ex:

```
t = (item1, item2, ... Itemn)
```

ra tuplas de um elemento apenas é necessário incluir a vírgula  
ós o elemento

```
t = (item1,)
```

lces funcionam da mesma forma das listas, assim como os  
todos e funções



## Dicionários

Dicionários são uma implementação de hash table que consiste em uma lista de pares de chave-valor, sem ordenação aves, devem ser tipos imutáveis e usualmente são utilizadas ngs ou números

Ores podem ser qualquer tipo de objeto Python

ra atribuir elementos a um dicionário é necessário que estejam re chaves, separados por vírgula, e com os pares chave, valor jarrados por : (dois pontos)

```
dic = {"k1":1, 'k2':10, ... 'kn':n}
```

lementos de um dicionário podem ser acessados ou alterados avés de sua chave entre colchetes

```
dic[ 'k1' ] = 2
```



## Quais métodos mais comuns com tipo Dicionário

Nome	Descrição
len()	Retorna o número de elementos do dicionário
dict()	Retorna a representação em formato string do dicionário
keys()	Retorna a lista de chaves do dicionário
values()	Retorna a lista de valores do dicionário
items()	Retorna a lista de chave, valor do dicionário
get(key, default=None)	Retorna o valor de uma chave ou um valor default caso não seja encontrada
update(dict2)	Insere um elemento (chave, valor) no dicionário
clear()	Remove todos os elementos do dicionário

## Estruturas de Controle de Fluxo

demos dividir as estruturas de controle de fluxo em 2 tipos

Estruturas de Seleção

Estruturas de Repetição

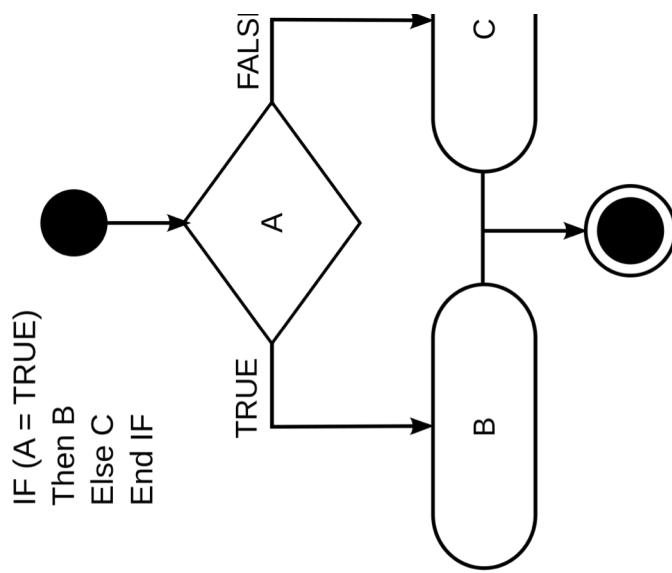
estrutura de Seleção são utilizadas para decidir qual fluxo de execução deverá ser tomado pelo programa, dada uma determinada condição ou um conjunto de condições

o representadas em Python, assim como em outras linguagens pelas instruções:

if

elif

else



## uturas de Controle de Fluxo

truturas de Seleção sempre utilizam processões booleanas para representar a determinada condição

le e False são Objetos imutáveis da sse bool e assume-se que qualquer or diferente de 0 (zero) E diferente de | são considerados True, todos os .ros valores considerados False

truturas de Seleção podem ser nhadas e combinadas juntamente n operadores (lógicos, aritméticos e acionais), podendo tornar-se mplexas estruturas de decisão

```

1 Selection.py x C:\dev\workspace_git\Datasci
2 var1 = 100
3 if var1:
4     print(var1)
5
6 Process finished with exit code 0

```

```

1 Selection.py x C:\dev\workspace_git\Datasci
2 var1 = 100
3 if var1:
4     print('OK')
5 else:
6     print('NOK')
7
8 Process finished with exit code 0

```

```

1 Selection.py x C:\dev\workspace_git\Datasci
2 var = 100
3 if var < 150:
4     if var == 150:
5         print("150")
6     elif var == 100:
7         print("100")
8     elif var <= 50:
9         print("<= 50")
10    print(">= 150")
11
12 Process finished with exit code 0

```

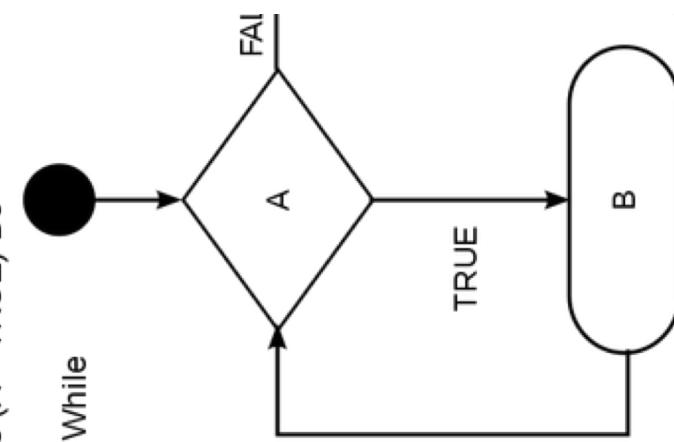
## Estruturas de Controle de Fluxo

Estruturas de Repetição são utilizadas para instruir o programa a repetir um determinado conjunto de código quanto uma determinada condição seja satisfatória ou utilizadas também para iterar em coleções de dados ou as

Python, existem 2 tipos de estruturas de repetição:

for

while



for é mais utilizada na iteração de listas ou quando há um mero conhecido de iterações a serem executadas  
while é mais recomendado quando uma determinada condição seja satisfatória para a execução do trecho de código

## uturas de Controle de Fluxo

```
Loops.py X
for letter in 'Python':
    print('The letter is:', letter)

frutas = ['banana', 'maça', 'laranja']
for fruta in frutas:
    print(fruta)

comidas = ('Pizza', 'Arroz', 'Feijão')
for index in range(len(comidas)):
    print(comidas[index])
```

```
Loops.py X
1 count = 0
2 while count < 5:
3     print(count)
4     count = count +
5
6
```

## uturas de Controle de Fluxo

---

truturas de Repetição oferecem instruções de controle em determinadas situações:

**ak** – permite a saída do fluxo antes da condição de finalização do laço ser atingida

**ntinue** – Instrui o interpretador a passar para a próxima iteração imediatamente, ignorando os comandos subsequentes

**ss** – permite que um determinado trecho de código seja escrito sintaticamente correto, porém em sua execução não será executada nenhuma instrução naquele momento

---

## ções

nplificadamente, uma função é um conjunto de instruções qual pode-se dar um nome

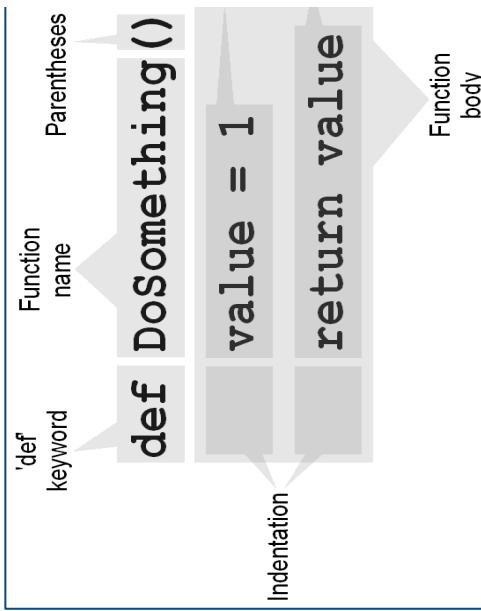
)movem o reaproveitamento, uma vez que um podem ser ocadas em diversos pontos do mesmo programa, sem a cessidade reescrita desse trecho de código

intaxe de definição de uma função é a seguinte:

```
def NOME(PARÂMETROS):
```

### COMANDOS

te a identação na parte de comandos, ela é fundamental a que o interpretador identifique quais são de fato, as truções que pertencem à função



## ↳ sobre Funções

parâmetros de uma função podem ser passados na mesma ordem em que foram definidos ou podem ser identificados através dação chamadora, assim não necessitando colocados na mesma ordem da definição função

râmetros de funções podem ter valores *fault*, dessa forma não é necessário sua usagem como argumento (desde sejam os mos argumentos em ordem reversa)

lções suportam parâmetros com número definido de valores. Para isso é necessário ≥ seja o último parâmetro da função e o argumento seja precedido por um \* (asterisco). Argumento será passado para a função no uma tupla de valores.

lções podem retornar valores, através da trucção *return*. Não é necessário declarar que função irá ter um retorno

```
Functions.py ×
1  def funcname(nome, idade, sexo, *outros):
2      print(type(nome), type(idade), type(sexo), type(*outros))
3      print(nome, idade, sexo, outros)
4
5      funcname('nome', 25, 'm')
6      funcname('nome', 25, 'f', 'outros1', 'outros2')
7
8
9
```

## Operadores Aritméticos

Operador	Descrição	Exemplo
+	Adição	$10 + 15 = 25$
-	Subtração	$25 - 15 = 10$
*	Multiplicação	$10 * 3 = 30$
/	Divisão	$30 / 4 = 7.5$
//	Parte inteira da divisão	$30 // 4 = 7$
%	Módulo (Resto da divisão)	$30 \% 4 = 2$
**	Exponenciação (Potência)	$3 ** 4 = 3 * 3 * 3 * 3 = 81$

## Operadores Relacionais

Op	Descrição	Exemplo
	Igual	<code>10 &gt; 15 é falso; 'Python' == 'python' é falso</code>
	Diferente	<code>5 != 7 é verdadeiro, 'Python' != 'python' é verdadeiro</code>
	Maior	<code>5 &gt; 5 é falso; 5 &gt; 2 é verdadeiro</code>
	Menor	<code>7 &lt; 12 é verdadeiro; 7 &lt; 5 é falso</code>
	Maior ou igual	<code>5 &gt;= 5 é verdadeiro; 5 &gt;= 6 é falso</code>
	Menor ou igual	<code>5 &lt;= 5 é verdadeiro; 5 &lt; 7 é verdadeiro</code>

## Operadores Lógicos

Operador	Descrição
<b>and</b>	Retorna True se os dois operandos forem True
<b>or</b>	Retorna True se um dos operandos for True
<b>not</b>	Negativa ou reverte a estado do operando

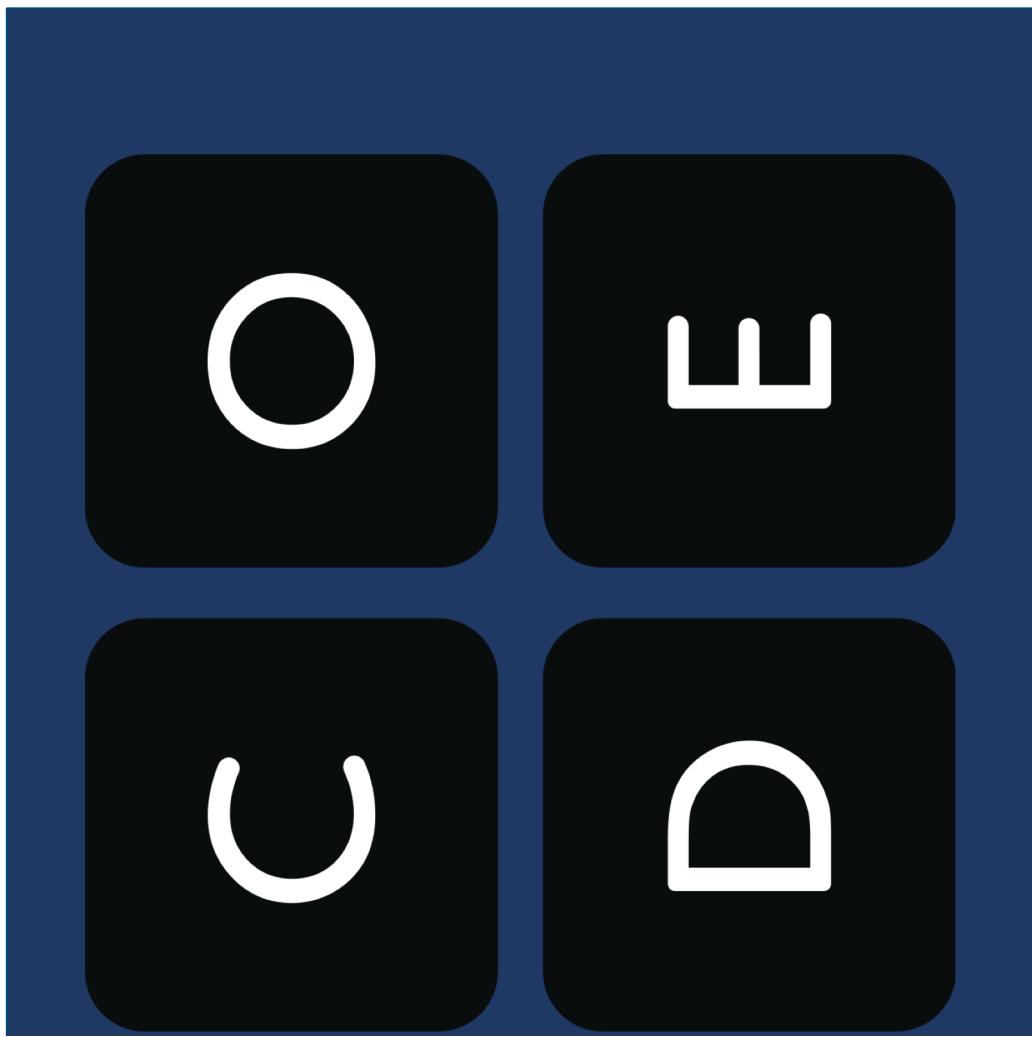
## Operadores de Membro

Operador	Descrição
<b>in</b>	True se encontra a expressão pesquisada

## Operadores de Identidade

Operador	Descrição
<b>is</b>	True se os operadores apontarem para o mesmo endereço de memória

## Challenges Time



## exercícios

Escreva uma função que receba três parâmetros (a, b e c) e retorne o maior valor entre os argumentos assados

Escreva uma função que receba uma lista de números e torne uma tupla com duas listas, a primeira contendo dos os números pares e a segunda contendo todos os números ímpares ordenados

## exercícios

Escreva uma função que receba dois valores inteiros positivos, e retorne a soma da sequência numérica inteira entre o primeiro e o segundo parâmetros, inclusive

Ado um número inteiro não negativo ( $n$ ), escreva duas funções sendo a primeira usando estruturas de repetição FOR e a segunda usando estruturas de repetição WHILE, que retornem o factorial de  $n$  ( $n!$ )

## Referências

- <https://www.python.org/>
- <https://python.org.br/>
- <http://www.diveintopython.net/>
- <https://www.w3schools.com/python>
- <https://www.codecademy.com/tracks/python>
- <https://www.jetbrains.com/pycharm/>





**Facens**  
AQUI TEM ENGENHARIA