



# *Introdução: Python*

*Prof. Dr. Max E. Vizcarra Melgar*



- 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;
- Possui tipagem dinâmica e uma de suas principais características é permitir a fácil leitura do código e exigir poucas linhas de código se comparado ao mesmo programa em outras linguagens.
- <https://www.python.org/>
- <https://python.org.br/>

<b>Paradigma</b>	Multiparadigma: Orientação a objetos Programação imperativa Programação funcional
<b>Surgido em</b>	1991 (28–29 anos) <sup>[1]</sup>
<b>Última versão</b>	3.8.0 (14 de outubro de 2019; há 3 meses <sup>[2]</sup> )
<b>Criado por</b>	Guido van Rossum <sup>[1]</sup>
<b>Estilo de tipagem:</b>	Dinâmica, forte
<b>Influenciada por</b>	ABC, <sup>[3]</sup> ALGOL 68, C <sup>[3]</sup> , Haskell, Icon, Java, Lisp, Modula-3 <sup>[3]</sup> , Perl, Smalltalk
<b>Influenciou</b>	Boo, D, Falcon, Fantom, Groovy, JavaScript, Nimrod, Py, Ruby, Squirrel, Swift
<b>Principais implementações</b>	CPython, IronPython, Jython, PyPy
<b>Extensão do arquivo:</b>	.py, .pyc, .pyd, .pyo, .pyw, .pyz
<b>Página oficial</b>	<a href="http://www.python.org">www.python.org</a>



- Aprendizagem rápida;
- Menor quantidade de código;
- Sintaxe fácil de ler;
- Usado em todas as empresas de tecnologia de ponta;
- Enorme quantidade de bibliotecas open-source

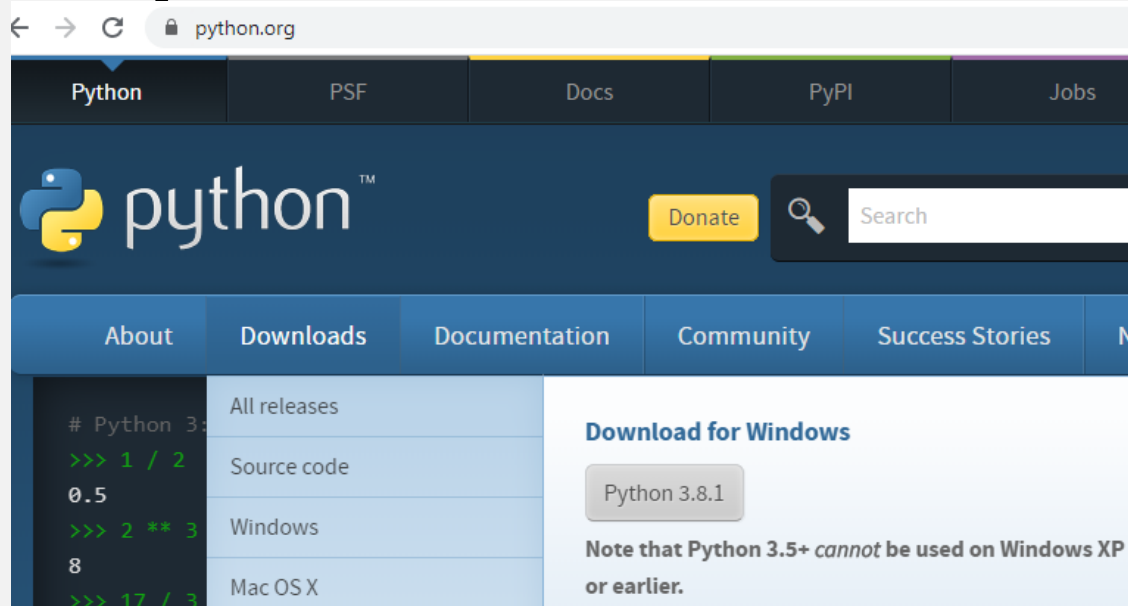
<b>Paradigma</b>	Multiparadigma: Orientação a objetos Programação imperativa Programação funcional
<b>Surgido em</b>	1991 (28–29 anos) <sup>[1]</sup>
<b>Última versão</b>	3.8.0 (14 de outubro de 2019; há 3 meses <sup>[2]</sup> )
<b>Criado por</b>	Guido van Rossum <sup>[1]</sup>
<b>Estilo de tipagem:</b>	Dinâmica, forte
<b>Influenciada por</b>	ABC, <sup>[3]</sup> ALGOL 68, C <sup>[3]</sup> , Haskell, Icon, Java, Lisp, Modula-3 <sup>[3]</sup> , Perl, Smalltalk
<b>Influenciou</b>	Boo, D, Falcon, Fantom, Groovy, JavaScript, Nimrod, Py, Ruby, Squirrel, Swift
<b>Principais implementações</b>	CPython, IronPython, Jython, PyPy
<b>Extensão do arquivo:</b>	.py, .pyc, .pyd, .pyo, .pyw, .pyz
<b>Página oficial</b>	<a href="http://www.python.org">www.python.org</a>



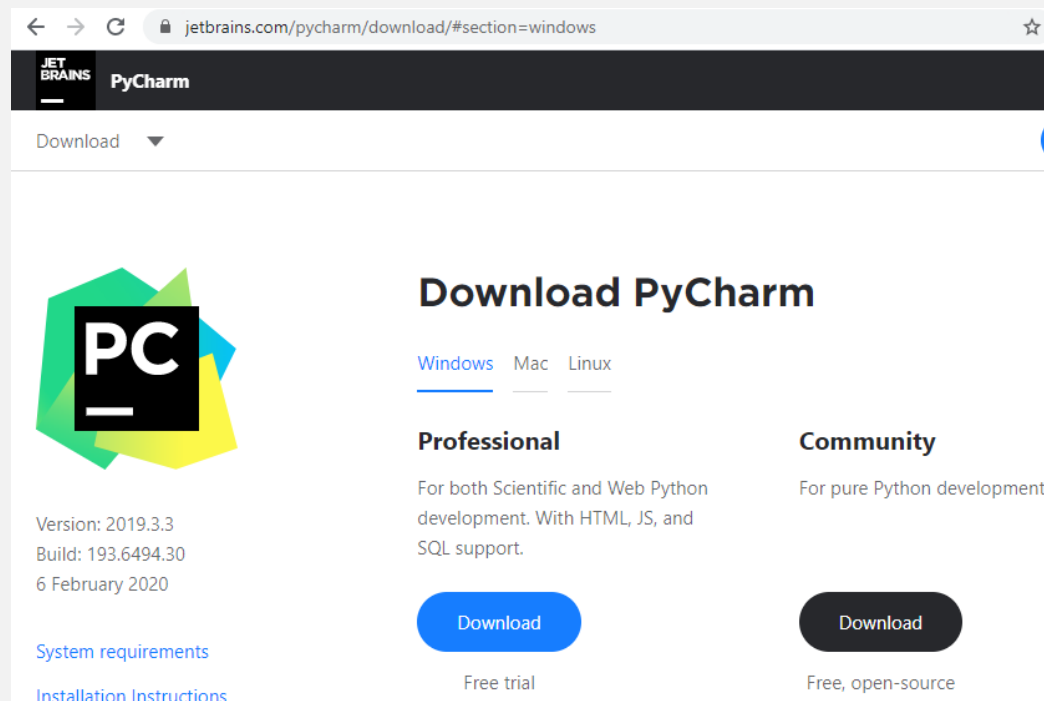
# Python 2 vs Python 3

- Python 2 sem patch de segurança ara 2020;
- Ainda existem muitas empresas que trabalham com Python 2;
- Versões MUITO similares!
- Pacotes liberados para Python 2 e 3 ou somente para Python 3.
- USAREMOS Python 3!!

# Instalação em Windows

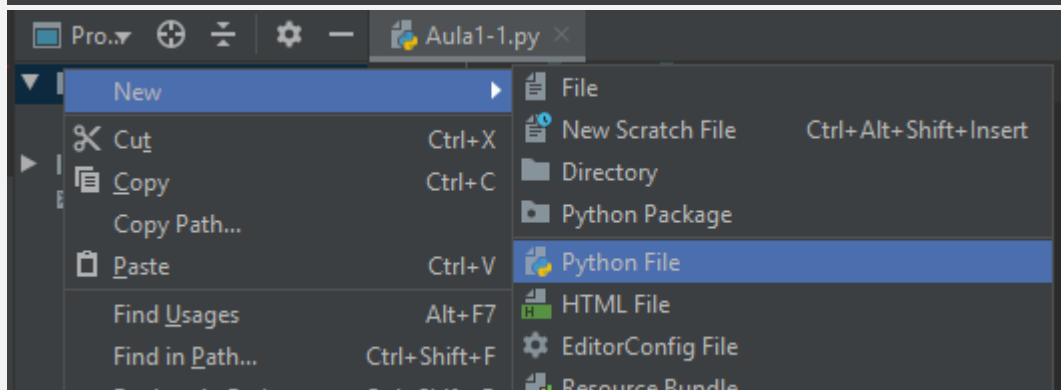
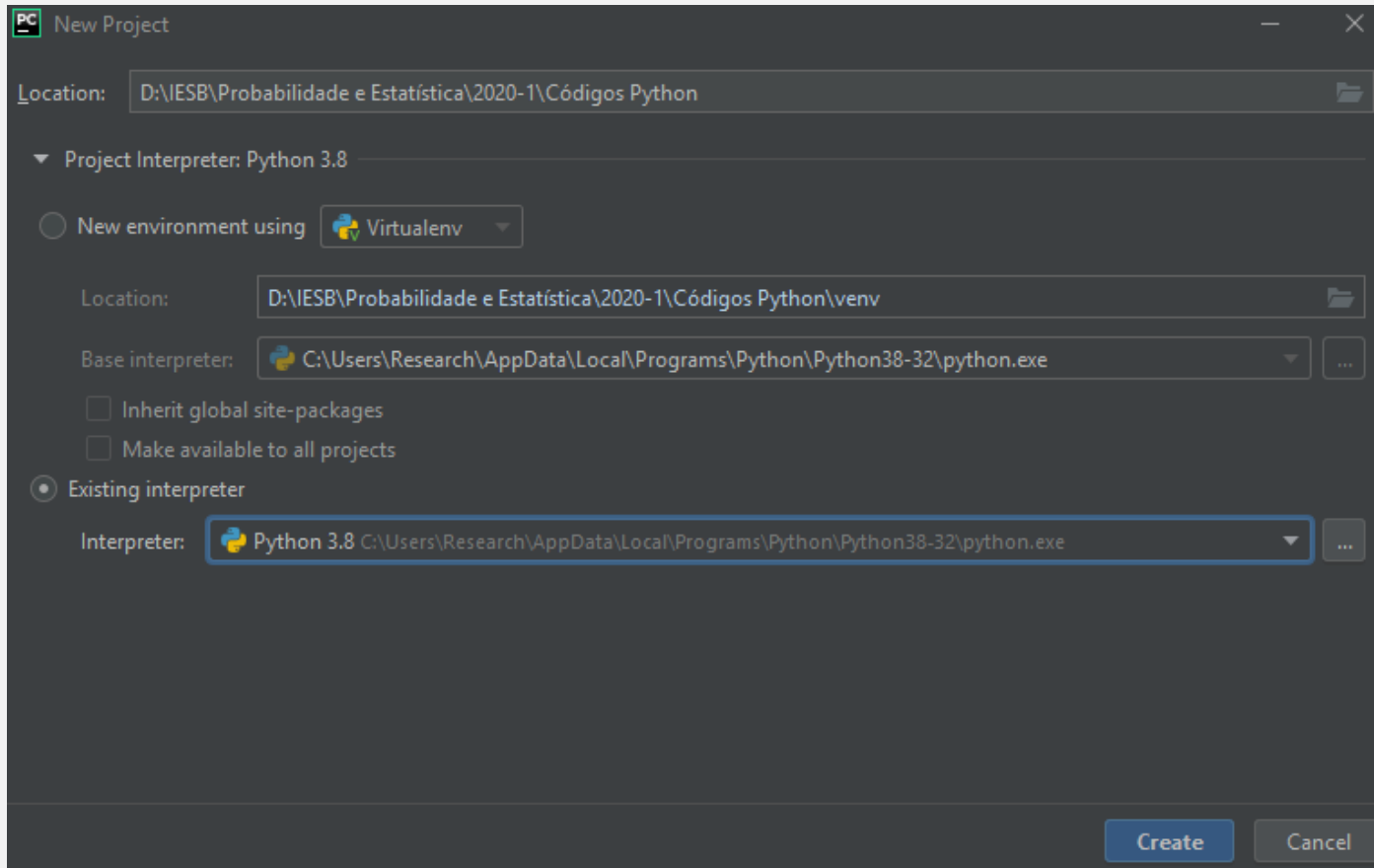


The screenshot shows the Python.org homepage. The navigation bar includes links for Python, PSF, Docs, PyPI, and Jobs. The main header features the Python logo, a 'Donate' button, and a search bar. Below the header, there are tabs for About, Downloads, Documentation, Community, Success Stories, and News. The 'Downloads' tab is active, showing a list of releases with links for All releases, Source code, Windows, and Mac OS X. A 'Download for Windows' section is visible, featuring a button for Python 3.8.1 and a note that Python 3.5+ cannot be used on Windows XP or earlier.



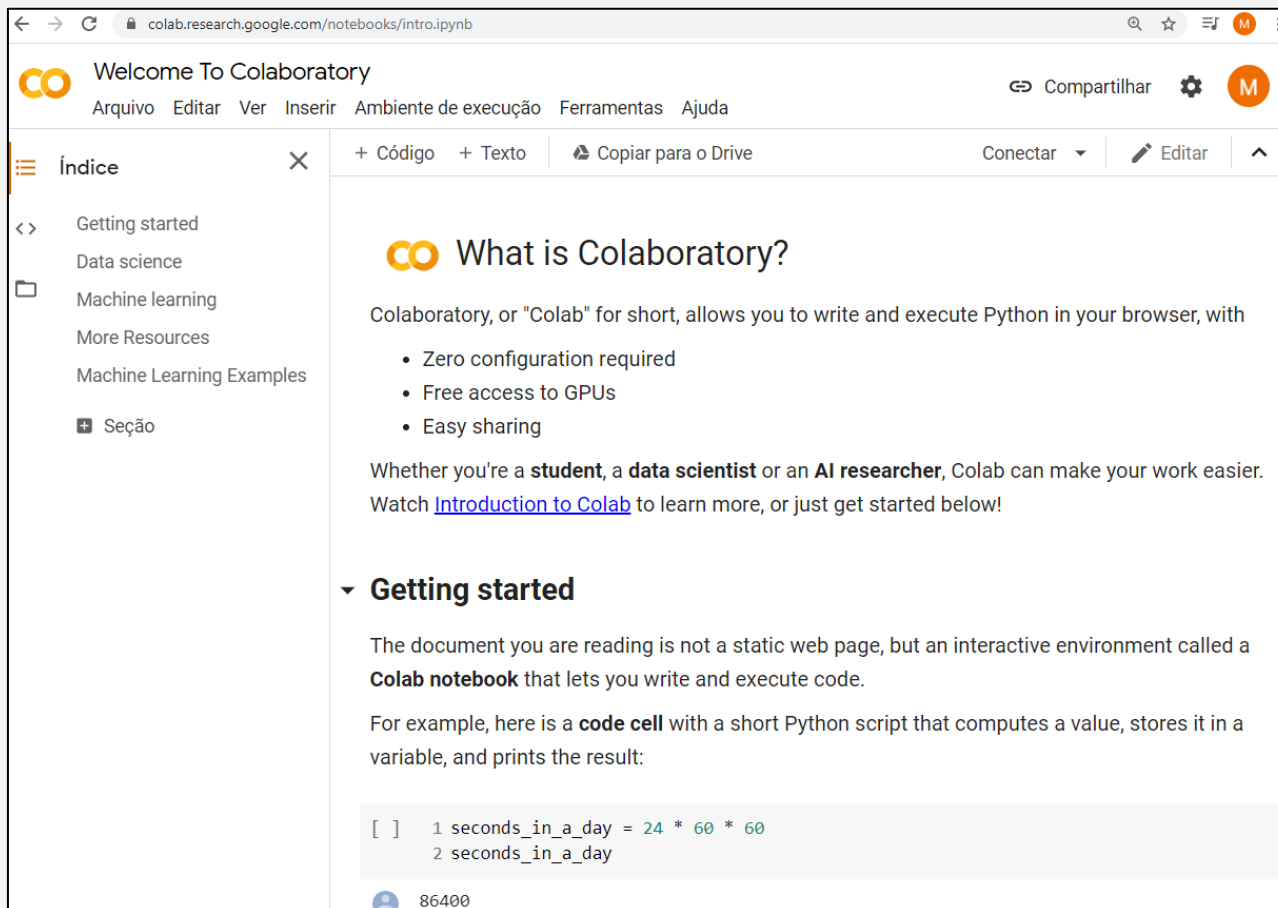
The screenshot shows the JetBrains PyCharm download page. The page has a dark header with the JetBrains logo and 'PyCharm' text. Below the header, there is a 'Download' button. The main content area is titled 'Download PyCharm' and includes tabs for Windows, Mac, and Linux. The 'Windows' tab is selected. The page is divided into two sections: 'Professional' and 'Community'. The 'Professional' section describes it as a tool for both Scientific and Web Python development, with HTML, JS, and SQL support. The 'Community' section describes it as a tool for pure Python development. Both sections have 'Download' buttons. The 'Professional' button is blue and labeled 'Free trial', while the 'Community' button is dark blue and labeled 'Free, open-source'. On the left side, there is a large 'PC' logo and version information: Version: 2019.3.3, Build: 193.6494.30, 6 February 2020. Links for 'System requirements' and 'Installation Instructions' are also present.

# Instalação em Windows



# Online (versão que usaremos)

1. Assista o vídeo: MELHOR FORMA DE APRENDER PYTHON (Google Colab Notebook) <https://www.youtube.com/watch?v=Gojqw9BQ5qY>
3. Crie sua conta GOOGLE (Gmail) e GitHub!
4. Execute Github and Google Colab → <https://www.youtube.com/watch?v=IAqSGivFmEM>
5. Acesse: <https://colab.research.google.com/notebooks/intro.ipynb>



colab.research.google.com/notebooks/intro.ipynb

Welcome To Colaboratory

Arquivo Editar Ver Inserir Ambiente de execução Ferramentas Ajuda

Compartilhar

Índice

- Getting started
- Data science
- Machine learning
- More Resources
- Machine Learning Examples
- Seção

+ Código + Texto Copiar para o Drive

Conectar Editar

## What is Colaboratory?

Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier. Watch [Introduction to Colab](#) to learn more, or just get started below!

### Getting started

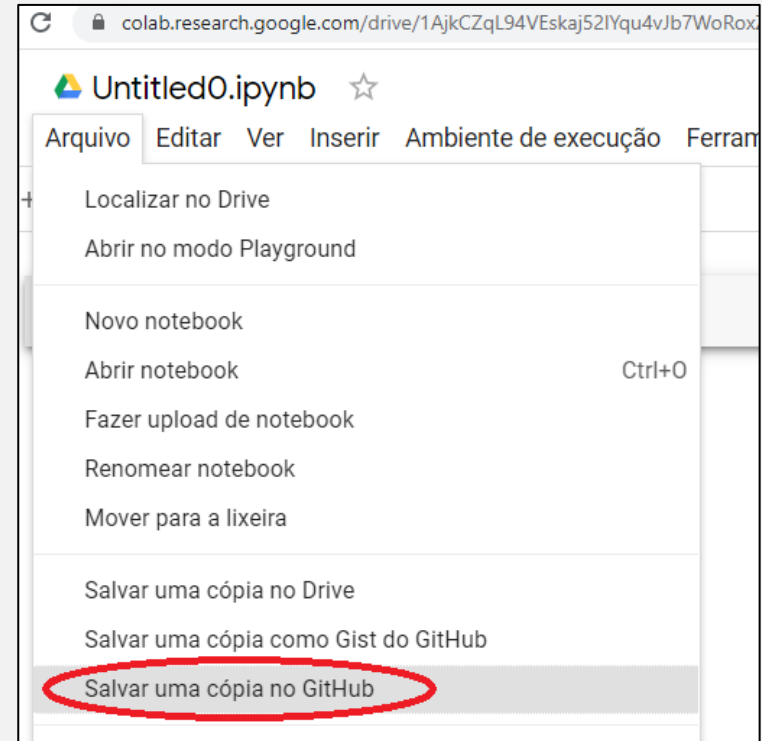
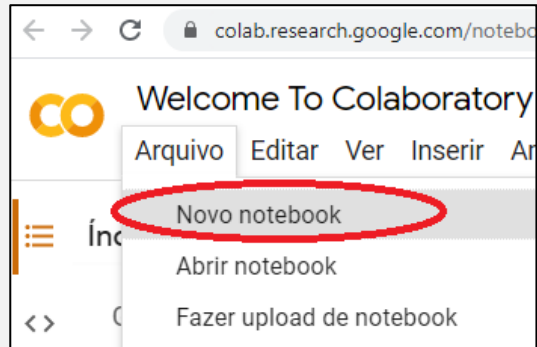
The document you are reading is not a static web page, but an interactive environment called a **Colab notebook** that lets you write and execute code.

For example, here is a **code cell** with a short Python script that computes a value, stores it in a variable, and prints the result:

```
[ ] 1 seconds_in_a_day = 24 * 60 * 60
    2 seconds_in_a_day
```

86400

# Online (versão que usaremos)



## Copiar para o GitHub

Repositório: [\[link\]](#)  
 [redacted]/IESB-Probability-Sttistics ▾

Ramificação: [\[link\]](#)  
 master ▾

Caminho do arquivo  
 Teste1.ipynb

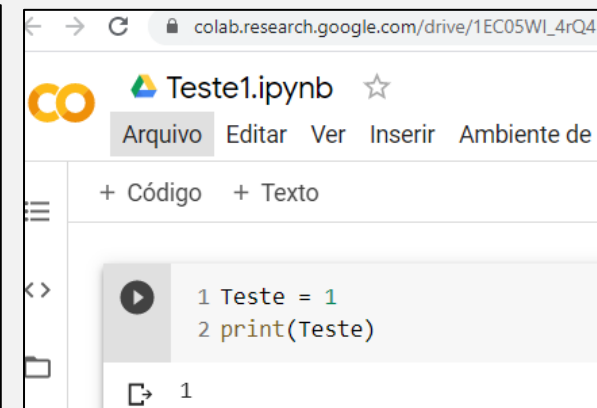
Mensagem de confirmação

Criado usando o Colaboratory

☒ Incluir um link para o Colaboratory

CANCELAR

OK





# Variáveis

Name	Type	Description
Integers	int	Whole numbers, such as: 3 300 200
Floating point	float	Numbers with a decimal point: 2.3 4.6 100.0
Strings	str	Ordered sequence of characters: "hello" 'Sammy' "2000" "楽しい"
Lists	list	Ordered sequence of objects: [10,"hello",200.3]
Dictionaries	dict	Unordered Key:Value pairs: {"mykey" : "value" , "name" : "Frankie"}
Tuples	tup	Ordered immutable sequence of objects: (10,"hello",200.3)
Sets	set	Unordered collection of unique objects: {"a","b"}
Booleans	bool	Logical value indicating True or False

# Operadores aritméticos

Operação	Operador
adição	+
subtração	-
multiplicação	*
divisão	/

Operação	Operador
exponenciação	**
parte inteira	//
módulo	%

# Variáveis Dinâmicas



```
int my_dog = 1;
```

```
my_dog = "Sammy" ; //RESULTS IN ERROR
```

Example of Static Typing  
(C++)

```
my_dogs = 2
```

```
my_dogs = [ "Sammy", "Frankie" ]
```

This is okay in  
Python!

## Vantagens:

- Fácil de trabalhar;
- Desenvolvimento mais rápido.

## Desvantagens:

- Bugs por tipos de entradas inesperadas;
- Ficar ligado em type()

The screenshot shows a Python IDE with a file named 'Aula1-1.py'. The code in the editor is:

```
1 a=10
2 print(a)
3 print(type(a))
4 a=9.5
5 print(a)
6 print(type(a))
```

Below the editor, the output of the program is displayed:

```
10
<class 'int'>
9.5
<class 'float'>
```

# Strings

Character :	h	e	l	l	o
Index :	0	1	2	3	4
Reverse Index:	0	-4	-3	-2	-1

[start:stop:step]

```
a="abcdefghijk"  
print(a[2:])  
print(a[6:10])  
print(a[::2])  
print(a[::3])  
print(a[2:7:2])  
print(a[::-1])
```

Strings são IMUTÁVEIS! Use Listas para conseguir mudar!

```
# Strings:  
a="Hello\nWorld"  
print(a)  
print(len(a))  
a="Hello\nWorld"  
print(a[0] + a[8] + a[5] + a[-1] + a[-3])
```

Aula1-1 ×

C:\Users\Research\AppData\Local\Programs\Python

Hello

World

11

Hr

dr

Aula1-1 ×

C:\Users\Research\AppData\Local\Programs\Python

cdefghijk

ghij

acegik

adgj

ceg

kjihgfedcba

# Strings

## Immutability

```
name = "Sam"
```

```
name[0] = 'P'
```

-----  
-----  
TypeError

```
a="Essa é uma string!"
b=a.split()
print(b)
print(b[0]+b[1]+b[2]+b[3])
b=a.split("a")
print(b)
print(b[0]+b[1]+b[2])
c=b[0]
print(c)
print(c[0]+c[1]+c[2])
```

Aula1-1 x

```
C:\Users\Research\AppData\Local\
['Essa', 'é', 'uma', 'string!']
Essa é uma string!
['Ess', ' é um', ' string!']
Ess é um string!
Ess
Ess
```

```
a="Hello"
b="\n"
c="World"
d=a+b+c
print(d)
```

Aula1-1 x

```
C:\Users\Research\AppData\Local\
Hello
World
```

```
a="Sam"
b="P"+a[1:]
print(b)
b=b+b
print(b)
b=5*b
print(b)
```

Aula1-1 x

```
C:\Users\Research\AppData\Local\
Pam
PamPam
PamPamPamPamPamPamPamPamPamPam
```

```
a="Hello World"
a.
(m) capitalize(self) str
(m) casefold(self) str
(m) center(self, width, fillchar) str
(m) count(self, x, __start, __end) str
(m) encode(self, encoding, errors) str
(m) endswith(self, suffix, start, end) str
(m) expandtabs(self, tabsize) str
(m) find(self, sub, __start, __end) str
(m) format(self, args, kwargs) str
(m) format_map(self, map) str
(m) index(self, sub, __start, __end) str
(m) isalnum(self) str
```

# Strings



```
a="Essa é uma mensagem: {} {} {}"  
msg0="Aviso 0"  
msg1="Aviso 1"  
msg2="Aviso 2"  
print(a.format(msg0,msg1,msg2))  
b=a.format(msg0,msg1,msg2)  
print(b)
```

Aula1-1 ×

```
C:\Users\Research\AppData\Local\Programs\Python\Python38-32  
Essa é uma mensagem: Aviso 0 Aviso 1 Aviso 2  
Essa é uma mensagem: Aviso 0 Aviso 1 Aviso 2
```

```
a="Essa é uma mensagem: {2} {0} {1}"  
msg0="Aviso 0"  
msg1="Aviso 1"  
msg2="Aviso 2"  
b=a.format(msg0,msg2,msg1)  
print(b)
```

Aula1-1 ×

```
C:\Users\Research\AppData\Local\Programs\Python\Python38-32  
Essa é uma mensagem: Aviso 1 Aviso 0 Aviso 2
```

Float formatting follows "{value:width.precision f}"

```
resultado=100/777  
print(resultado)  
print("O resultado foi: {r:15.4f}".format(r=resultado))
```

Aula1-1 ×

```
C:\Users\Research\AppData\Local\Programs\Python\Python38-32  
0.1287001287001287  
O resultado foi:           0.1287
```

```
nome="Professor Max"  
print(f"O nome dele é: {nome}") # Python > 3.6  
idade=30  
print(f"O nome dele é {nome} e sua idade é {idade}")
```

Aula1-1 ×

```
C:\Users\Research\AppData\Local\Programs\Python\Python38-32  
O nome dele é: Professor Max  
O nome dele é Professor Max e sua idade é 30
```

# Listas

```
lista=[1, "dois", 3.0, "d"]
print(lista)
print(lista[0])
print(lista[1])
print(lista[2])
print(lista[3])
lista[1]="DOIS"
print(lista)
lista[0]=lista[0]+5
print(lista)
lista[2]=lista[2]/2
print(lista)
```

```
lista=[1, "dois", 3.0, "d"]
lista.append("XX")
print(lista)
lista.append(88)
print(lista)
lista.pop() # retira o último
print(lista)
ultimo=lista.pop()
print(lista)
print(ultimo)
lista.pop(1) # indico a posição
print(lista)
```

```
lista=[5, 1, 3, 8, 4, 2.5, 2, 3.6]
print(lista)
lista.sort()
print(lista)
lista=["k", "b", "d", "j"]
print(lista)
lista.sort()
print(lista)
lista=[5, 1, "K", "c", 4, 2.5, "g", 3.6]
print(lista)
lista.sort()
print(lista)
```

Aula1-1 ×

C:\Users\Research\AppData\Local

[1, 'dois', 3.0, 'd']

1

dois

3.0

d

[1, 'DOIS', 3.0, 'd']

[6, 'DOIS', 3.0, 'd']

[6, 'DOIS', 1.5, 'd']

Aula1-1 ×

C:\Users\Research\AppData\Local\Progr

[1, 'dois', 3.0, 'd', 'XX']

[1, 'dois', 3.0, 'd', 'XX', 88]

[1, 'dois', 3.0, 'd', 'XX']

[1, 'dois', 3.0, 'd']

XX

[1, 3.0, 'd']

Aula1-1 ×

C:\Users\Research\AppData\Local\Programs\Pyth

[5, 1, 3, 8, 4, 2.5, 2, 3.6]

[1, 2, 2.5, 3, 3.6, 4, 5, 8]

['k', 'b', 'd', 'j']

['b', 'd', 'j', 'k']

[5, 1, 'K', 'c', 4, 2.5, 'g', 3.6]

Traceback (most recent call last):

# Listas

```
lista=[5, 1, 3, 8, 4, 2.5, 2, 3.6]
print(lista)
lista.reverse()
print(lista)
lista=["k", "b", "d", "j"]
print(lista)
lista.reverse()
print(lista)
```

Aula1-1 ✕

```
C:\Users\Research\AppData\Local\Program
[5, 1, 3, 8, 4, 2.5, 2, 3.6]
[3.6, 2, 2.5, 4, 8, 3, 1, 5]
['k', 'b', 'd', 'j']
['j', 'd', 'b', 'k']
```

```
lista=[5, 1, 3, 8, 4, 2.5, 2, 3.6]
print(lista)
lista.sort()
lista.reverse()
print(lista)
print(lista[2:5])
lista=["k", "b", "d", "j"]
print(lista)
lista.sort()
lista.reverse()
print(lista)
print(lista[2:])
```

Aula1-1 ✕

```
C:\Users\Research\AppData\Local\Program
[5, 1, 3, 8, 4, 2.5, 2, 3.6]
[8, 5, 4, 3.6, 3, 2.5, 2, 1]
[4, 3.6, 3]
['k', 'b', 'd', 'j']
['k', 'j', 'd', 'b']
['d', 'b']
```





# Dicionários

`{'key1':'value1','key2':'value2'}`

**Dicionários:** objetos chamados pelo nome chave e não são ordenados.

**Listas:** objetos chamados pela posição

```
# Dicionários
precos_frutas={'banana':1.99, 'cebola':2.5}
print(precos_frutas)
print(precos_frutas['cebola'])
precos_frutas['tomate']=3.99
print(precos_frutas)
precos_frutas['banana']=2.99
print(precos_frutas)
precos_frutas['cheefe']='Prof Max'
print(precos_frutas['cheefe'][0])
print(precos_frutas.keys())
print(precos_frutas.values())
print(precos_frutas.items())
precos_frutas['lista_aqui']=['g', 8, 3.6]
print(precos_frutas['lista_aqui'][0])
precos_frutas['lista_aqui'][0]=precos_frutas['lista_aqui'][0].upper()
precos_frutas['outro_dic']={'YYY':500, 'XXX':6.99}
print(precos_frutas)
print(precos_frutas['outro_dic']['YYY'])
```

Aula1-1 ×

C:\Users\Research\AppData\Local\Programs\Python\Python38-32\python.exe "D:/IESB/Probabilidade e Estatística/2020-1/Códigos Python/Aula1-1.

```
{'banana': 1.99, 'cebola': 2.5}
2.5
{'banana': 1.99, 'cebola': 2.5, 'tomate': 3.99}
{'banana': 2.99, 'cebola': 2.5, 'tomate': 3.99}
P
dict_keys(['banana', 'cebola', 'tomate', 'cheefe'])
dict_values([2.99, 2.5, 3.99, 'Prof Max'])
dict_items([('banana', 2.99), ('cebola', 2.5), ('tomate', 3.99), ('cheefe', 'Prof Max')])
G
{'banana': 2.99, 'cebola': 2.5, 'tomate': 3.99, 'cheefe': 'Prof Max', 'lista_aqui': ['G', 8, 3.6], 'outro_dic': {'YYY': 500, 'XXX': 6.99}}
500
```

# Tuples

Similares com as listas, porém são IMUTÁVEIS! → Não podem ser reescritos!  
Integridade de dados!

```
# Tuples
tup=(2,'a',3.5,'a','b','c','b','a') # É Tuple
lis=[1,2,3] # É List
print(type(tup))
print(type(lis))
print(tup.count('a')) # Qtos 'a' temos no tuple
print(tup.index('b')) # Posição onde aparece pela 1 vez
print(tup[2:])
print(len(tup))
tup[3]='z'
```

Aula1-1 ×

```
C:\Users\Research\AppData\Local\Programs\Python\Python38-32\
<class 'tuple'>
<class 'list'>
3
4
(3.5, 'a', 'b', 'c', 'b', 'a')
8
Traceback (most recent call last):
  File "D:/IESB/Probabilidade e Estatística/2020-1/Códigos F
    tup[3]='z'
TypeError: 'tuple' object does not support item assignment
```

Coleções não ordenadas de **elementos únicos!**

```
# Sets
meuset=set()
print(type(meuset))
meuset.add("r")
meuset.add(3)
meuset.add("z")
meuset.add("9.5")
print(meuset)
lista=["e", 2, 3, 4, 2, 3, 2, 4, "a", "e", "b", "a"]
meuset=set(lista)
print(meuset) # Rode o código várias vezes!
meuset2=set('Mississippi')
print(meuset2)
```

Aula1-1 ×

```
C:\Users\Research\AppData\Local\Programs\Python\Python38-
<class 'set'>
{'9.5', 'z', 3, 'r'}
{'e', 2, 3, 4, 'a', 'b'}
{'i', 'M', 'p', 's'}
```



# Boolean

True or False!

```
# Booleans  
a=True  
print(type(a))  
b=None  
print(type(b))  
c=False  
print(c)  
print(3>1)
```

```
Aula1-1 ×  
C:\Users\Research\A  
<class 'bool'>  
<class 'NoneType'>  
False  
True
```

```
# Files
meuarquivo=open('Texto.txt') # Ou 'C:\\Users\\ ...'
print(meuarquivo)
print(meuarquivo.read())
print(meuarquivo.read()) # Cursor foi ao final do arquivo
meuarquivo.seek(0) # Posicionar o cursor em 0
print(meuarquivo.read())
meuarquivo.seek(0)
stringarquivo=meuarquivo.read()
print(stringarquivo)
meuarquivo.seek(0)
paragrafosarquivo=meuarquivo.readlines() # Lista de paragrafos
print(paragrafosarquivo)
meuarquivo.close()
arq=open('OutroTexto.txt','w')
arq.write('Esse é outro texto')
arq.close() #veja na pasta esse novo arquivo
```

Aula1-1 ×

```
C:\Users\Research\AppData\Local\Programs\Python\Python38-32\python.
<_io.TextIOWrapper name='Texto.txt' mode='r' encoding='cp1252'>
Esse é um Texto
Para ser avaliado em Python

Esse é um Texto
Para ser avaliado em Python
Esse é um Texto
Para ser avaliado em Python
['Esse é um Texto\n', 'Para ser avaliado em Python']
```

## Reading, Writing, Appending Modes

- **mode='r'** is read only
- **mode='w'** is write only (will overwrite files or create new!)
- **mode='a'** is append only (will add on to files)
- **mode='r+'** is reading and writing
- **mode='w+'** is writing and reading (Overwrites existing files or creates a new file!)

```
with open('Texto2.txt',mode='w') as meuarquivo:
    meuarquivo.write('Novo texto!')
with open('Texto2.txt', mode='r') as meuarquivo:
    print(meuarquivo.read())
print('-----')
with open('Texto2.txt',mode='a') as meuarquivo:
    meuarquivo.write('\nNova linha!')
with open('Texto2.txt', mode='r') as meuarquivo:
    print(meuarquivo.read())
```

Aula1-1 ×

```
C:\Users\Research\AppData\Local\Programs\Python\Pytho
Novo texto!
-----
Novo texto!
Nova linha!
```



# Fim do Capítulo!

## **Basic Practice:**

<http://codingbat.com/python>

## **More Mathematical (and Harder) Practice:**

<https://projecteuler.net/archives>

## **List of Practice Problems:**

[http://www.codeabbey.com/index/task\\_list](http://www.codeabbey.com/index/task_list)

## **A SubReddit Devoted to Daily Practice Problems:**

<https://www.reddit.com/r/dailyprogrammer>

## **A very tricky website with very few hints and touch problems (Not for beginners but still interesting)**

<http://www.pythonchallenge.com/>