
PYTHON

1 – Linguagem python aplicada Data Science

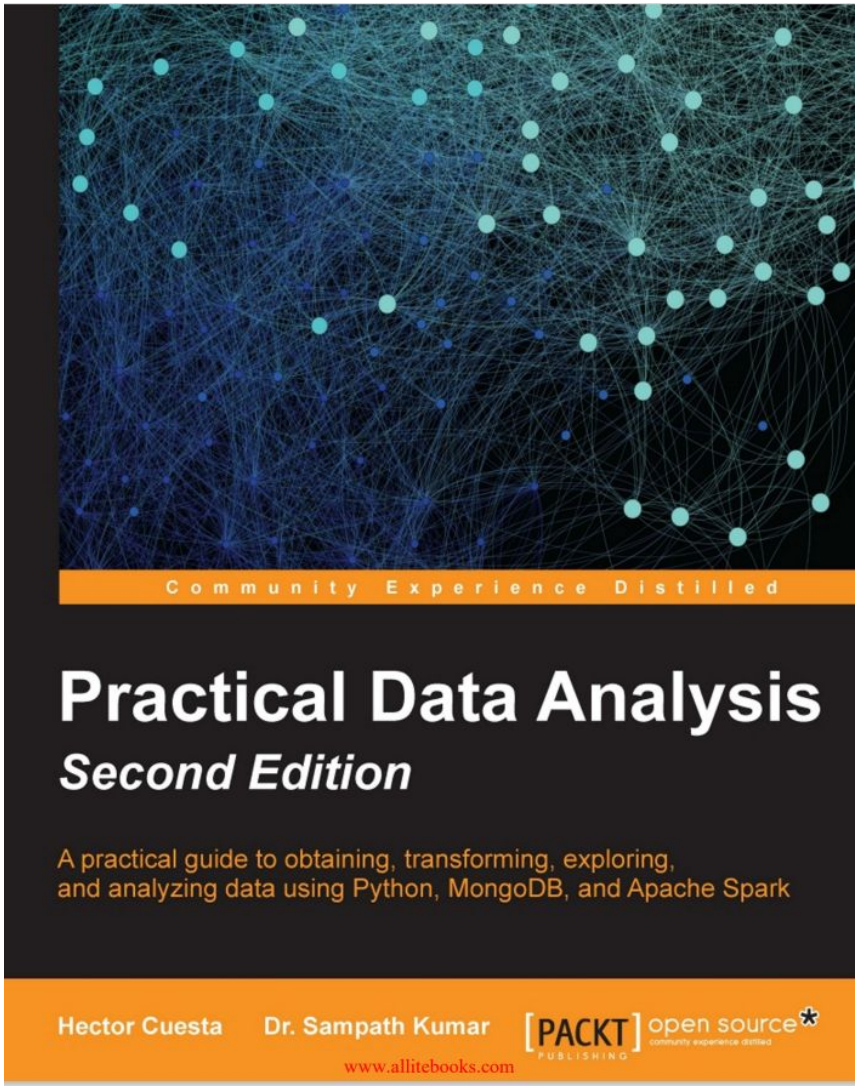
- Introdução na Linguagem Python
- Jupyter: Interface e Notebooks: Colab google
- Operações matemáticas
- Variáveis
- Programando com Python
- Importação e limpeza de dados em Python
- Databases em Python
- Arrays e Matrizes com NumPy
- Estruturas e Operações com Pandas
- Manipulação de dados com NumPy

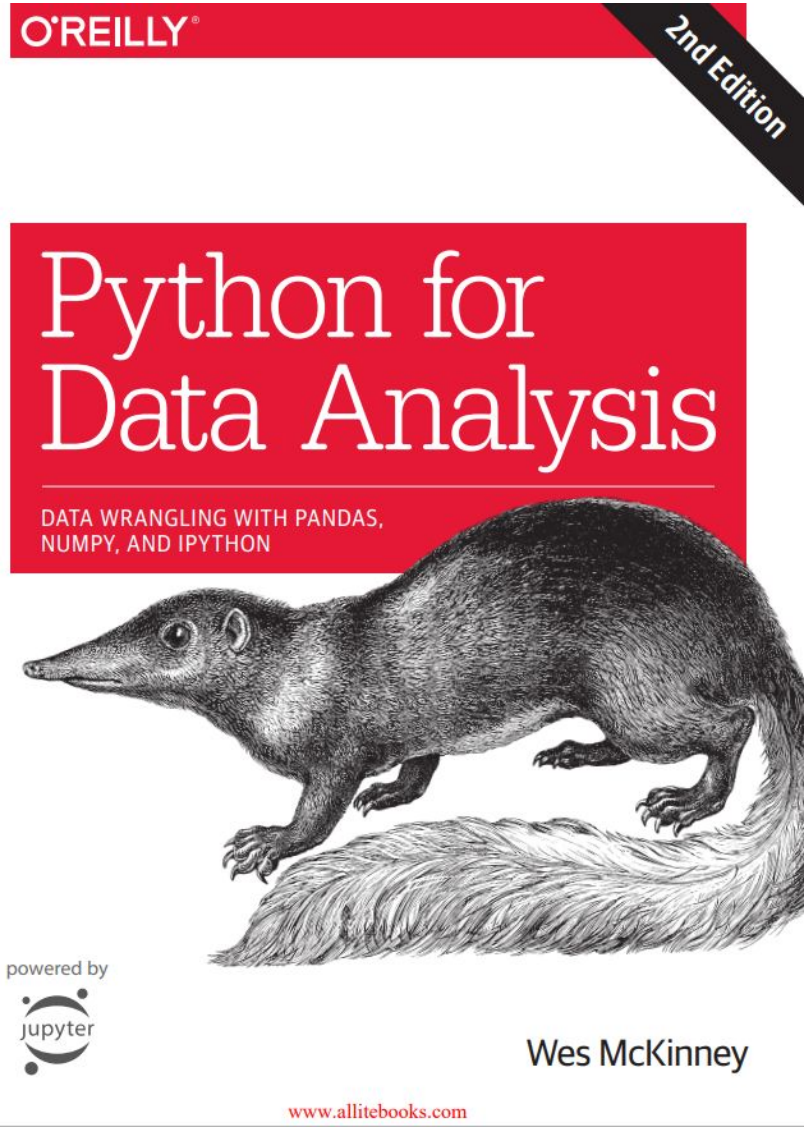
2 - Visualização de Dados

- Matplotlib
- Seaborn

3 - Exploração e Análise de Dados

- Distribuições
- Histogramas
- Box-plots, quartiles, scatter plots, heat maps and others
- Aplicando regressão linear e estatística


Community Experience Distilled
Practical Data Analysis
Second Edition
A practical guide to obtaining, transforming, exploring,
and analyzing data using Python, MongoDB, and Apache Spark
Hector Cuesta
Dr. Sampath Kumar
[PACKT] open source
PUBLISHING
www.allitebooks.com


O'REILLY
2nd Edition
Python for
Data Analysis
DATA WRANGLING WITH PANDAS,
NUMPY, AND IPYTHON
powered by
jupyter
Wes McKinney
www.allitebooks.com

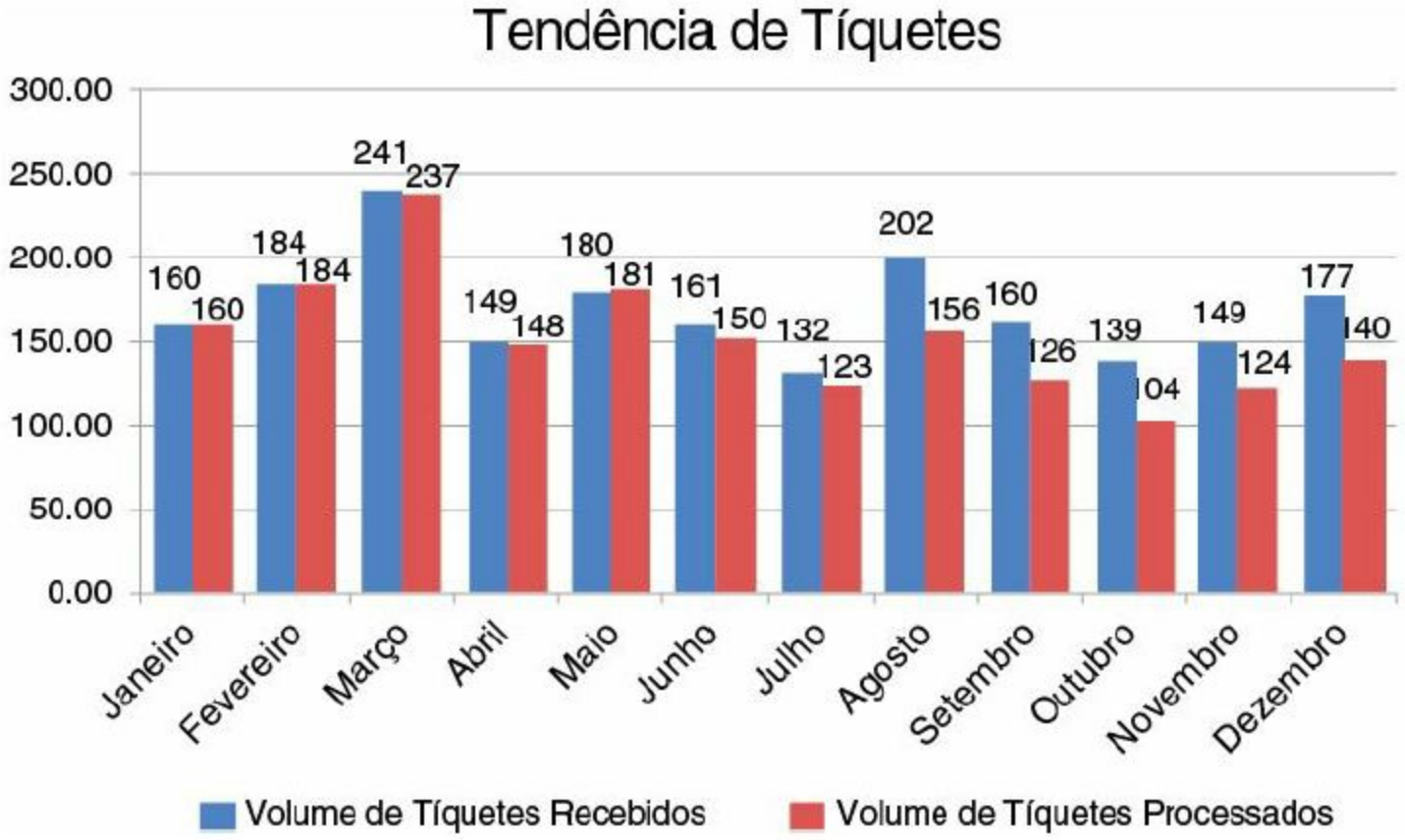


FIGURA 0.2 Exemplo 1 (antes): apresentação de dados

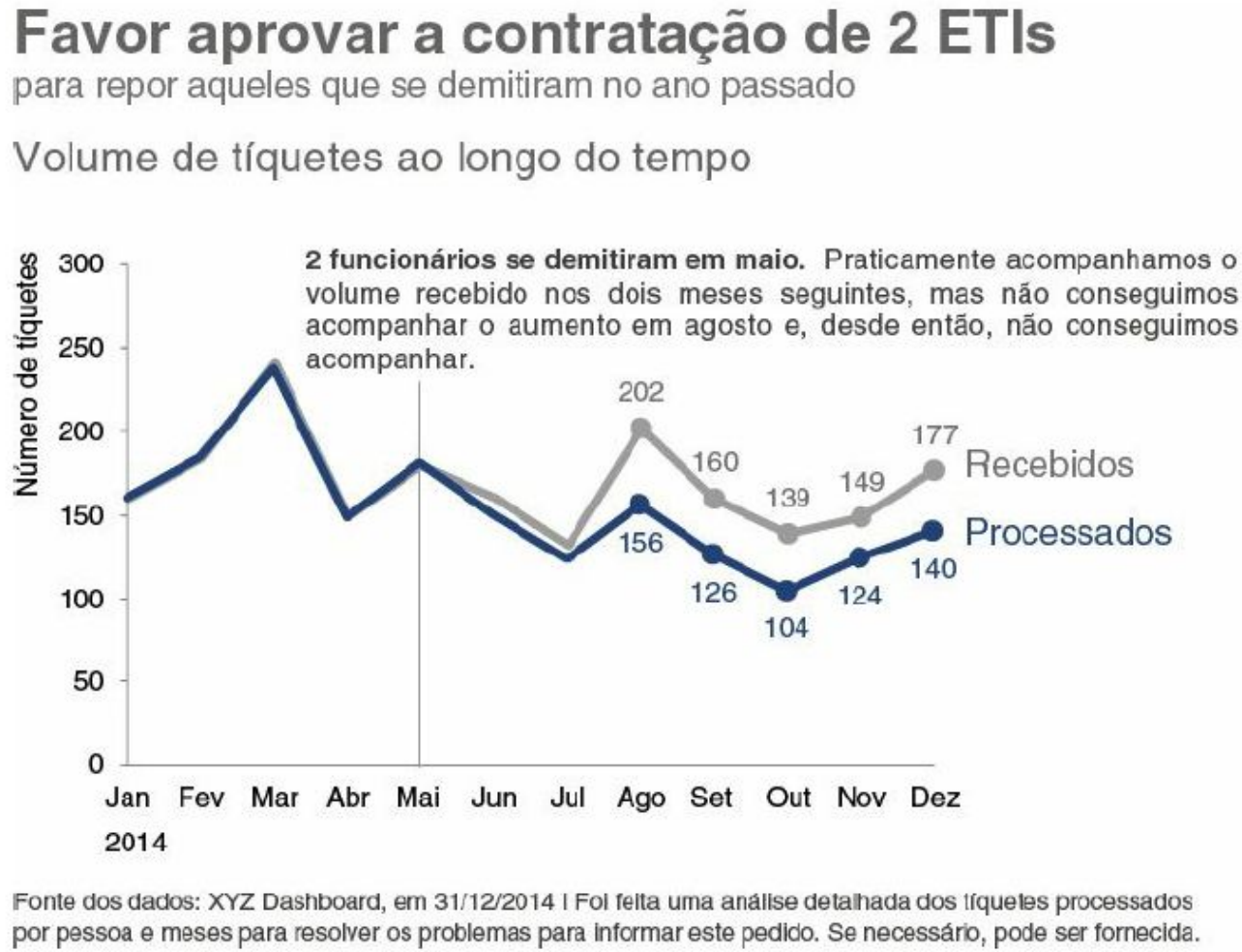


FIGURA 0.3 Exemplo 1 (depois): storytelling com dados

FIGURA 0.4 Exemplo 2 (antes): apresentação de dados

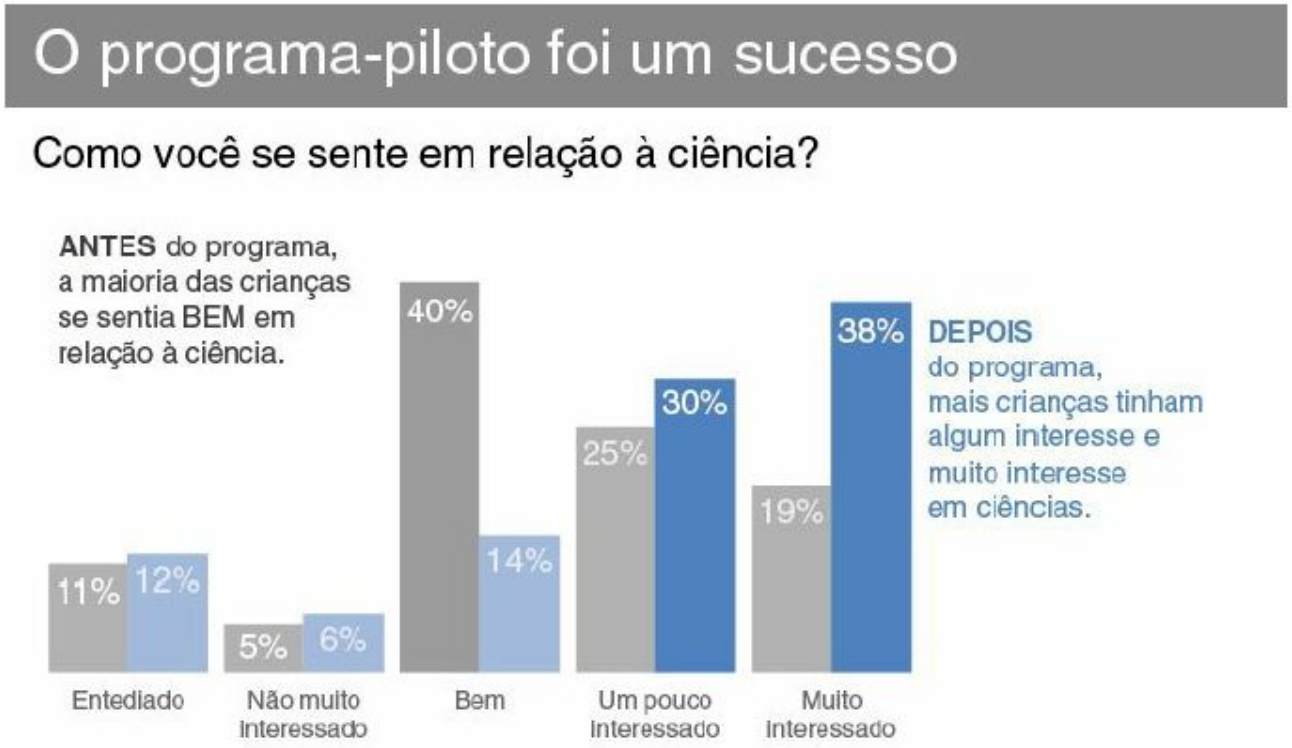


FIGURA 0.5 Exemplo 2 (depois): storytelling com dados

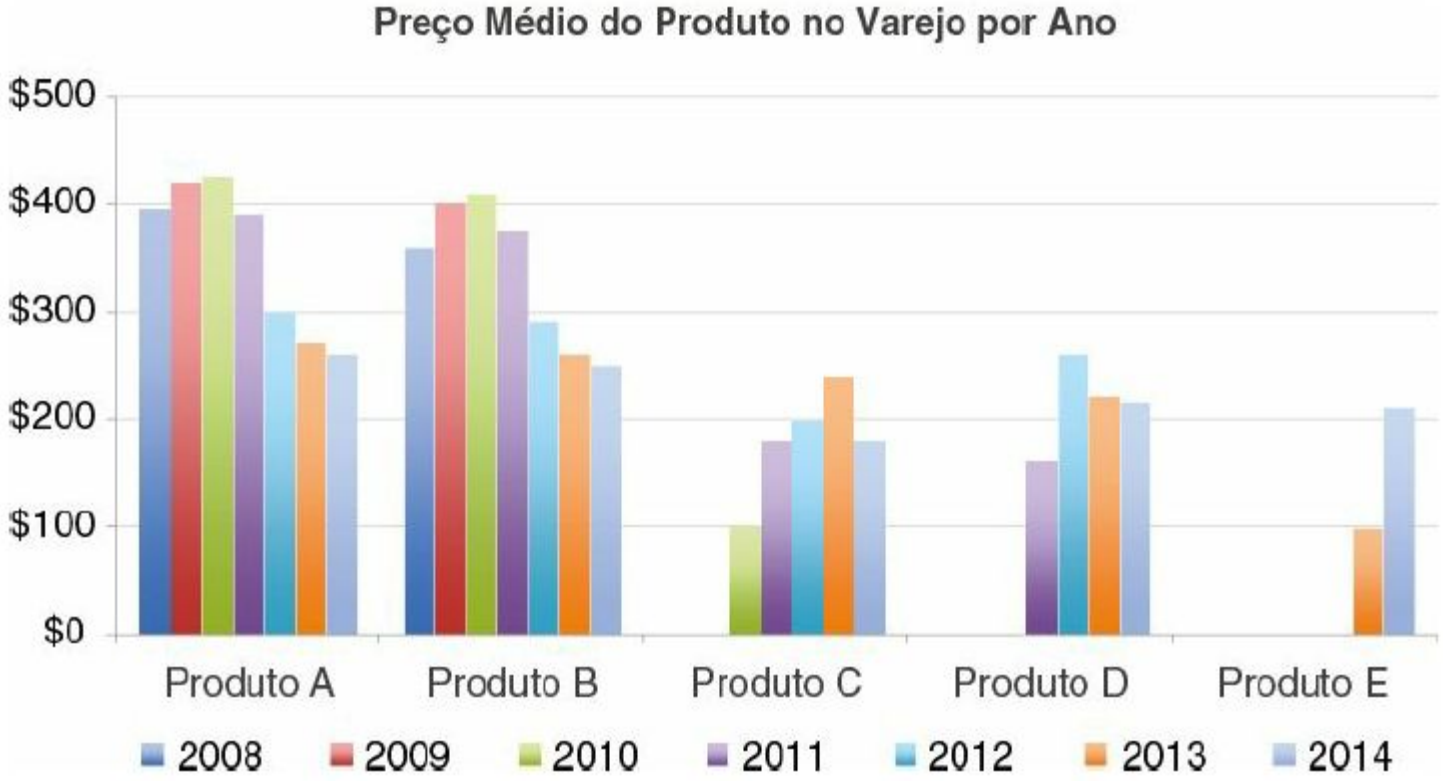
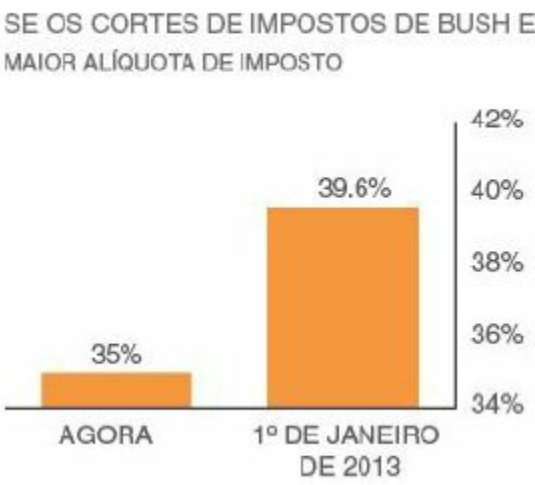


FIGURA 0.6 Exemplo 3 (antes): apresentação de dados

FIGURA 0.7 Exemplo 3 (depois): storytelling com dados

Linha de base diferente de zero:
conforme originalmente representada



Linha de base zero:
conforme deveria ser representada

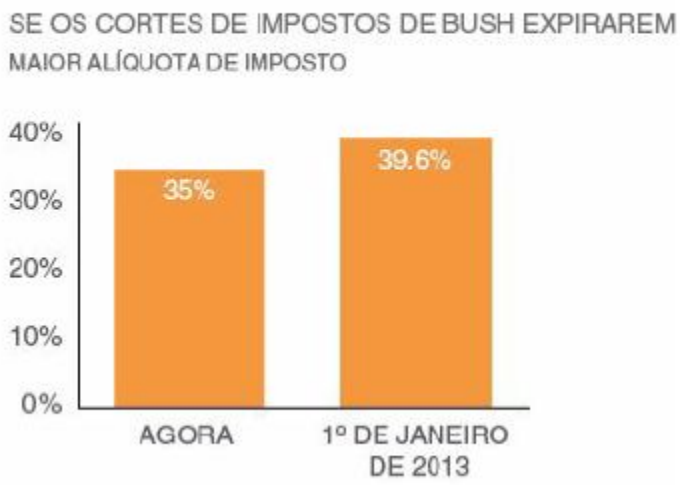
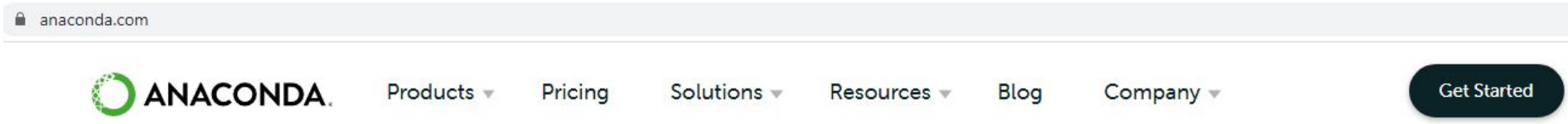


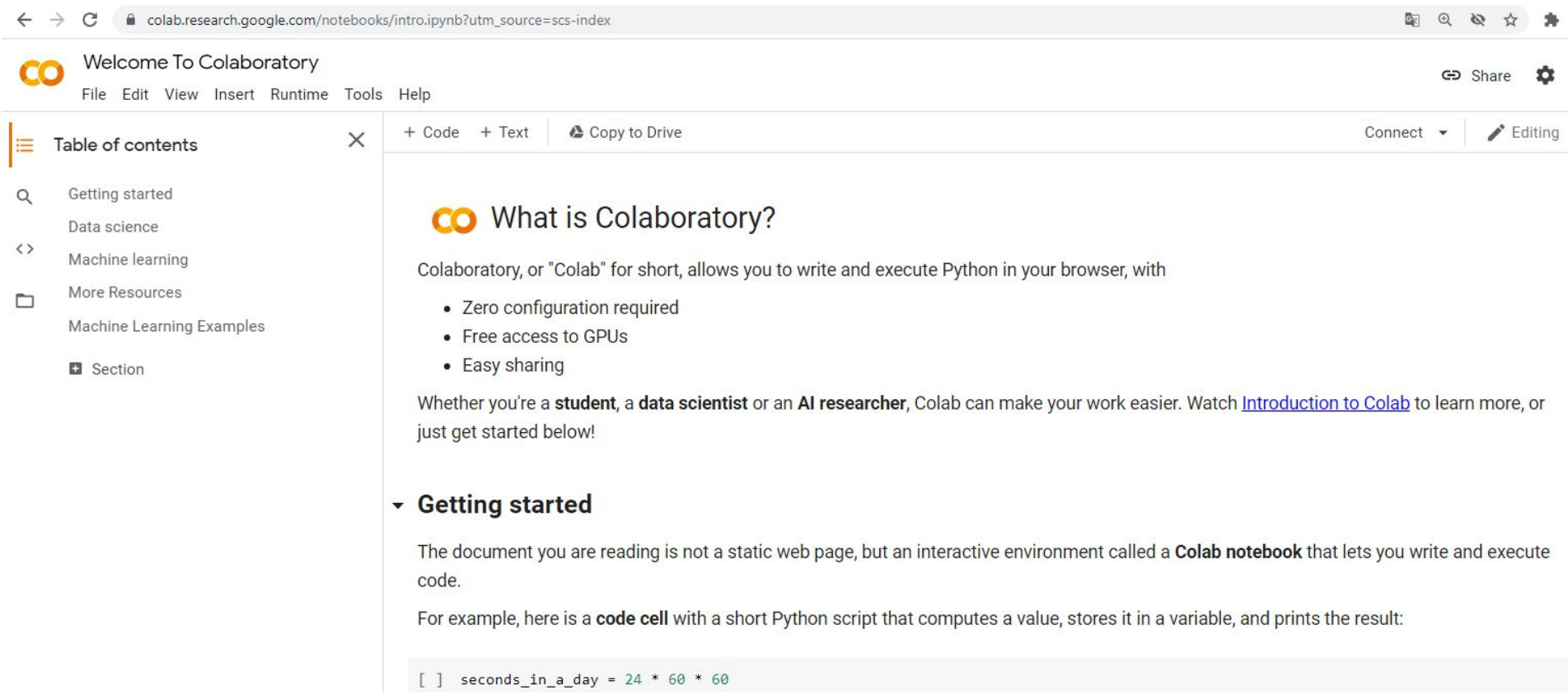
FIGURA 2.13 OS GRÁFICOS DE BARRA DEVEM TER UMA LINHA DE BASE ZERO



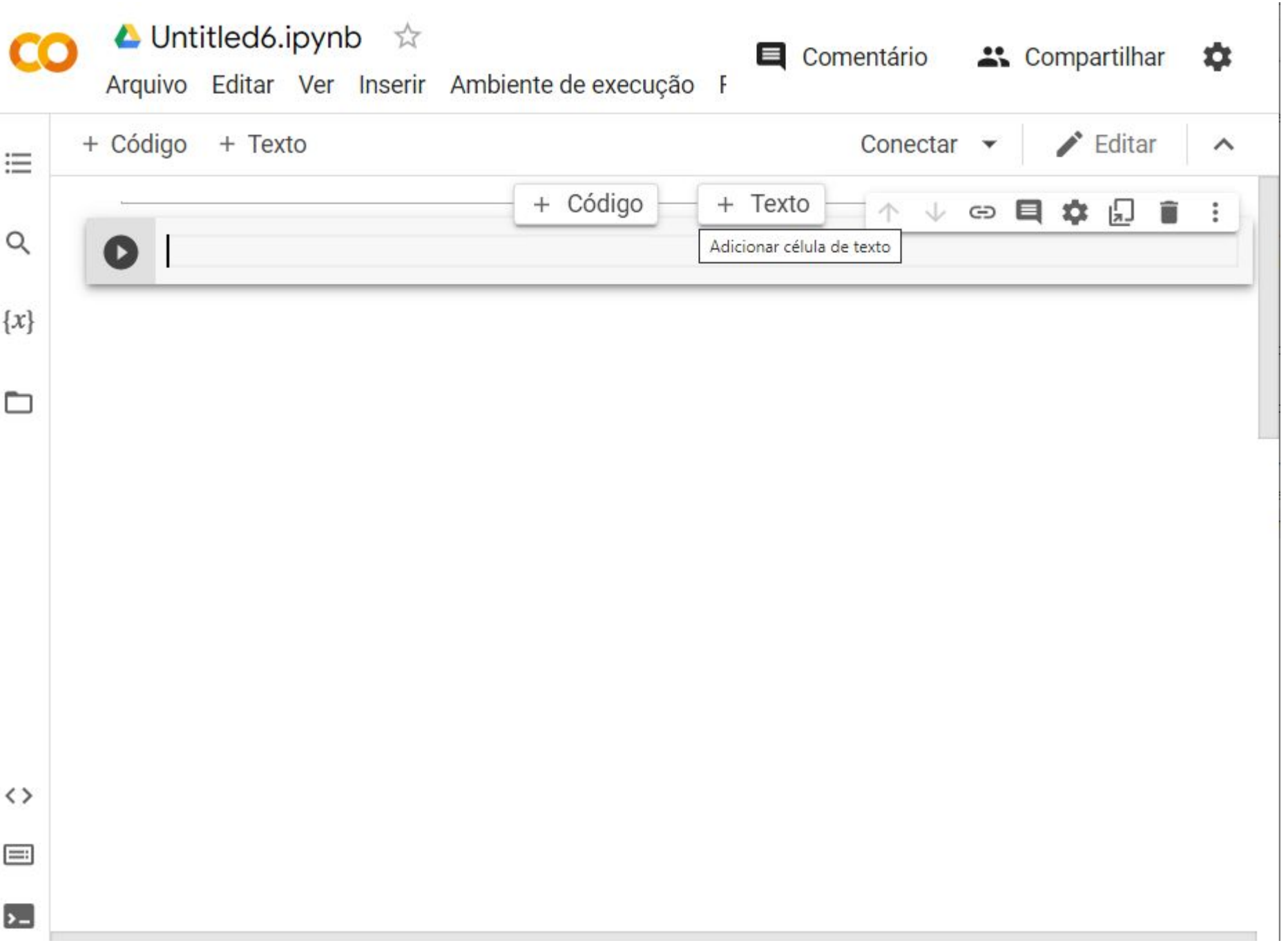
Data science technology for human sensemaking.

A movement that brings together millions of data science practitioners, data-driven enterprises, and the open source community.

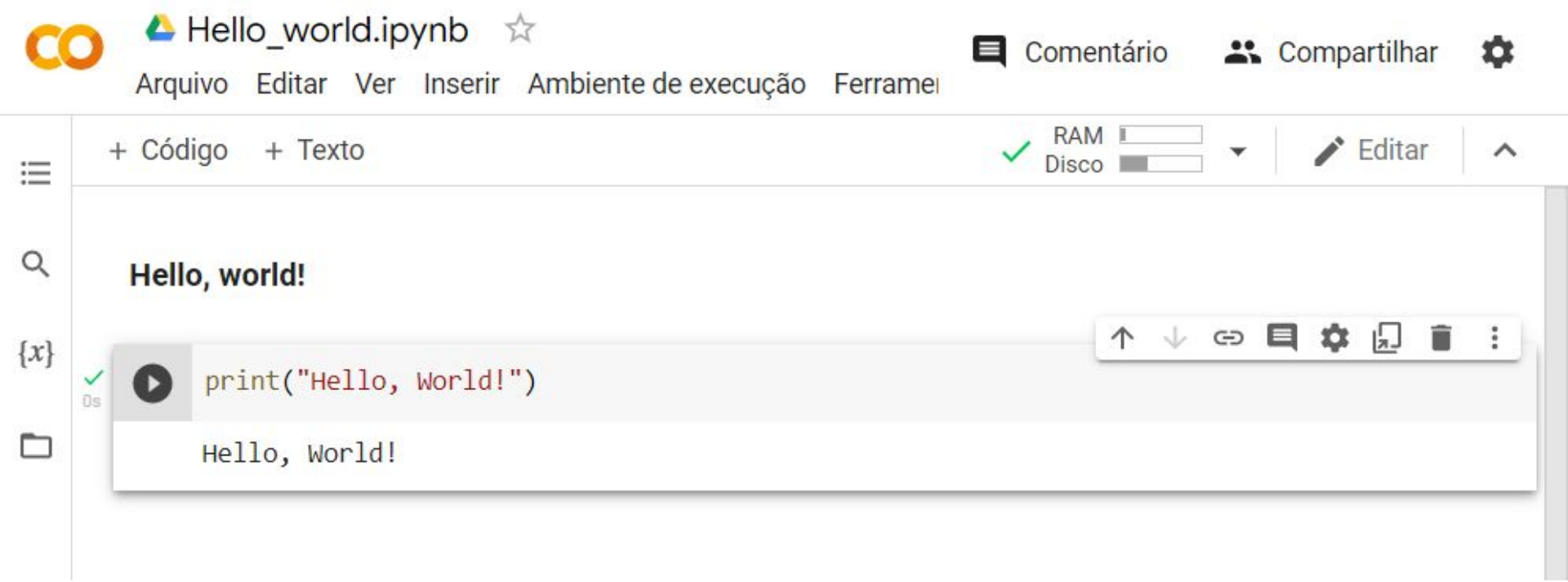
https://www.anaconda.com/



https://colab.research.google.com/notebooks


Untitled6.ipynb
Arquivo Editar Ver Inserir Ambiente de execução
Comentário
Compartilhar
+ Código + Texto
Conectar
Editar
+ Código
+ Texto
Adicionar célula de texto

Linguagem Python


Hello_world.ipynb
Comentário
Compartilhar
Arquivo Editar Ver Inserir Ambiente de execução Ferrame
+ Código + Texto
RAM
Disco
Editar
Hello, world!
print("Hello, World!")
0s
Hello, World!

Operações:

```
[2] # Adição:
    4+2
```

```
[3] # Subtração:
    5-2
```

```
[4] # Multiplicação:
    3*2
```

```
[5] # Divisão:
    8/2
```

```
[6] # Potência:
    2**3
```

```
[7] # Raiz quadrada:
    3**(1/2)
```

```
[8] # Raiz:
    27**(1/3)
```

```
[9] # uso da função de potência pow(a,b)=a**b
    pow(4,0.5)
```

Tipos de variáveis numéricas e funções pow(), round() e abs():

```
[10] # tipo inteira: int()
     int(5)
```

```
[11] # tipo decimal: float()
     float(3.8)
```

```
[12] # tipo decimal: float()
     float(3)  #Podemos converter o número de inteiro para decimal
```

```
[13] # exemplo int
     int(4.5)
```

```
[14] # função: Arredondar Números Decimais Em Python
     round(3.6)
```

```
[15] # exemplo float
     float(3.5)
```

```
[16] # A função abs() retorna o valor absoluto
     abs(-6)
```

As variáveis em Python são utilizadas para armazenar valores.

```
[1]  # Atribuindo um valor à variável var
     var = 3
```

```
[2]  # imprimindo a variável var
     print(var)
```

```
[6]  # Atribuindo um valor à variável var e imprimindo:
     var = 5
     print(var)
```

```
[4]  # Para realizar múltiplas atribuições, podemos utilizamos a seguinte notação:

     var1, var2, var3 = 20, 32, 45
     print("var 1 = ",var1)
     print("var 2 = ",var2)
     print("var 3 = ",var3)
```

```
[5]  #Operações com Variáveis:
     a=2
     b=5
     c=a+b
     print(" c = a+b \n c = ",c)
```

Funções:

[7]
```
# A estrutura para escrever uma função está representada abaixo:

def funcao():
  print("Estou começando a entender Python!")
```

[8]
```
funcao()
```

[10]
```
# Uma função que soma dois valores

def func(a, b):
  """
  Add two numbers together
  Returns the_sum : type of arguments
  """
  c = a+b
  print(c)
```

[11]
```
func(3,2)
```

[13]
```
func?
```

Expressões Lambda:

```
[14] # Vamos definir uma expressão com lambda
     # para elever números à terceira potência (cubo)

     cubo = lambda x: pow(x, 3)
```

```
[15] cubo(2)
```

0s

```
8
```

```
[16] cubo(5)
```

0s

```
125
```

Listas:

```
[ ]  # Criando uma lista

     lista = [10, 25, 20, 35, 40]
     print(lista)
```

```
[ ]  # Verificando o tipo de variável

     type(lista)
```

```
[ ]  # Podemos acessar valores individuais da lista, utilizando os índices
     lista[0]
```

```
[ ]  # Podemos acessar valores individuais da lista, utilizando os índices
     lista[0]
```

```
[ ]  # Podemos adicionar valores individuais da lista
     lista.append(56)
```

```
[ ]  print(lista)
```

```
[ ]  # Podemos deletar valores individuais da lista
     del lista[4]
     print(lista)
```

```
# Podemos, ainda, atualizar os valores das listas
lista[2]=21
print(lista)
```

```
[39] #Podemos concatenar (juntar) duas listas, utilizando o símbolo de +
     lista_1=[1,2,3,4]
     lista_2=[5,1,7]
     lista_3=lista_1+lista_2
     print(lista_3)
```

```
[40] # Para verificar o tamanho da lista utilizamos o método len.
     len(lista_3)
```

```
[41] #Para encontrar o valor máximo, utlizamos max
     max(lista_3)
```

```
[ ]  # mínimo
     min(lista_3)
```

```
[42] #soma
     sum(lista_3)
```

```
[43] # a frequencia de um elemento
     lista_3.count(1)
```

```
[44] #Podemos encontar a posição (índice) do elemento dentro da lista, utilizando o método index
     lista_3.index(1)
```

```
[ ] #Para ordenar uma lista em ordem crescente utilzamos o método sort.
    lista4 = [3, 2, 6, 5, 3]
    lista4
```

```
[ ] lista4.sort()
    lista4
```

```
[ ] lista4.count(3)
```

```
[ ] lista4.index(3)
```

```
# Para deixar de forma descrescente
lista4.sort(reverse = True)
lista4
```

Dicionários:

```
[ ]  # Criando um dicionário

     dict = {"Chevrolet": 10000,
             "Fiat": 15000,
             "Ford": 12000}
```

```
[ ]  dict
```

```
[ ]  # Podemos conhecer o tamanho do dicionário utilizando o método len

     len(dict)
```

```
[ ]  # Para conhecer as chaves utilizamos o método keys

     dict.keys()
```

```
[ ]  # Para conhecer os valores utilizamos o método values

     dict.values()
```

```
[ ]  # Podemos utilizar o método items para verificar os itens presentes em um
     # dicionário

     dict.items()
```

```
# Para adicionar um novo item, utilizamos a seguinte notação:

dict["Pegeout"] = 10000
```

```
dict
```

```
# Para atualizar o valor, seguimos a mesma ideia

dict["Chevrolet"] = 15000
```

```
dict
```

```
# Para excluir determinado valor, utilizamos a chave

del dict["Pegeout"]
```

```
dict
```

```
# Podemos criar um dicionário de listas!!

dictl = {"a": [1, 2, 3], "b": [5, 8, 10]}
```

```
dictl
```

```
# Verificando os valores associados à uma chave

dictl["a"]
```

Tuplas:

Uma tupla é uma sequência de valores, mas esses valores são imutáveis.

```
tupla = ("a", "b", "c")
tupla
```

```
[3] type(tupla)
```

```
[4] # Para acessar determinado valor em uma tupla utilizamos a seguinte notação

    tupla[0]
```

```
[6] # Podemos converter uma lista para uma tupla usando o método list

    lista = list(tupla)
    lista
```

```
[8] # Podemos converter a lista para tupla, utilizando o método tuple
    list_a= ['a','b',2]
    tupla_a = tuple(list_a)
    tupla_a
```

```
[11] #modificando uma lista (atualizando)
     list_a[1]='f'
```

```
[12] list_a
```

```
['a', 'f', 2]
```

```
#exemplo de erro ao tentar modificar uma tuple

tupla_a[1]='f'
```

Condicionais: Condicionais possuem um papel importante na programação de computadores e ajudam, de forma lógica, a automatizar análises e processos.

```
[ ]  # Primeiro exemplo de condicional if:

     if 4>2:
       print("Condicional funcionando perfeitamente!")
```

```
[15] # exemplo if e else:

     idade = 20

     if idade >= 18:
       print("Você é maior de idade.")

     else:
       print("Você é menor de idade.")
```

```
[21] # Vamos usar o elif para ter várias condições

     var = "caso1"

     if var == "caso1":
       print("lógica para o caso1")

     elif var == "caso2":
       print("lógica para o caso2")

     else:
       print("Outro caso")
```

```python
# Vamos ver um processo automático, utilizando Placeholders

idade = int(input("Digite a sua idade (em número): "))

if idade >= 18:
  print("Você é maior de idade.")

else:
  print("Você é menor de idade.")
```

Loop for: O "loop for" executa uma sequência de instruções várias vezes e abrevia o código que gerencia a variável de loop.

```
[31] # Vamos supor que queiramos multiplicar por 2 todos os elementos da lista abaixo

     lista = [1, 2, 3, 4]
```

```
[32] # Vamos utlizar o loop for

     for i in lista:
       a = i*2
       print(a)
```

```
lista*2
```

```
# Vamos supor agora que temos uma lista e queremos multiplicar por três
# cada elemento e depois de multiplicar somar todos os elementos

lista2 = [1, 2, 3]

soma = 0

for x in lista2:
  z = x*3
  soma += z

print(soma)
```

[35]
```
# Contando os itens de uma lista

lista = [10, 15, 20, 25, 40]

contador = 0

for item in lista:
    contador += 1

print(contador)
```

Loop While:

O "loop while" repete uma instrução ou grupo de instruções enquanto uma determinada condição for verdadeira. Ele testa a condição antes de executar o corpo do loop.

[37]
```
# Usando o loop while para imprimir os valores de 10 a 19

counter = 8
while counter < 15:
    print(counter)
    counter = counter + 1
```