

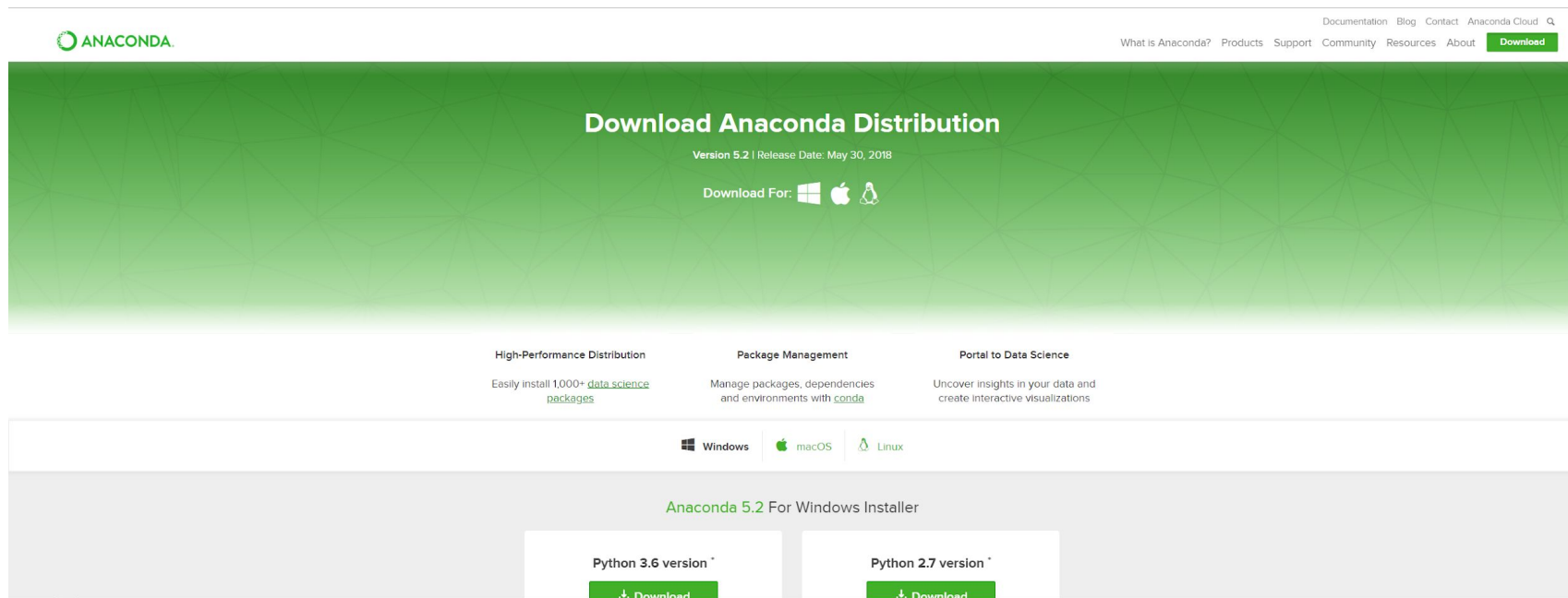
Introdução a Algoritmos

Programando com Python

Instalação

Instalação

Anaconda:



The screenshot shows the Anaconda website's download page. At the top, the Anaconda logo is on the left, and navigation links (Documentation, Blog, Contact, Anaconda Cloud, What is Anaconda?, Products, Support, Community, Resources, About) and a green 'Download' button are on the right. The main section has a green background with a white geometric pattern and the text 'Download Anaconda Distribution' and 'Version 5.2 | Release Date: May 30, 2018'. Below this, it says 'Download For:' followed by icons for Windows, macOS, and Linux. The bottom section is divided into three columns: 'High-Performance Distribution' (with a link to 'data science packages'), 'Package Management' (with a link to 'conda'), and 'Portal to Data Science'. At the very bottom, there's a section for 'Anaconda 5.2 For Windows Installer' with two buttons: 'Python 3.6 version' and 'Python 2.7 version', each with a 'Download' link.




ANACONDA

Documentation Blog Contact Anaconda Cloud

What is Anaconda? Products Support Community Resources About [Download](#)

Download Anaconda Distribution




Version 5.2 | Release Date: May 30, 2018

Download For:   

High-Performance Distribution
Easily install 1,000+ [data science packages](#)

Package Management
Manage packages, dependencies and environments with [conda](#)

Portal to Data Science
Uncover insights in your data and create interactive visualizations

 Windows  macOS  Linux

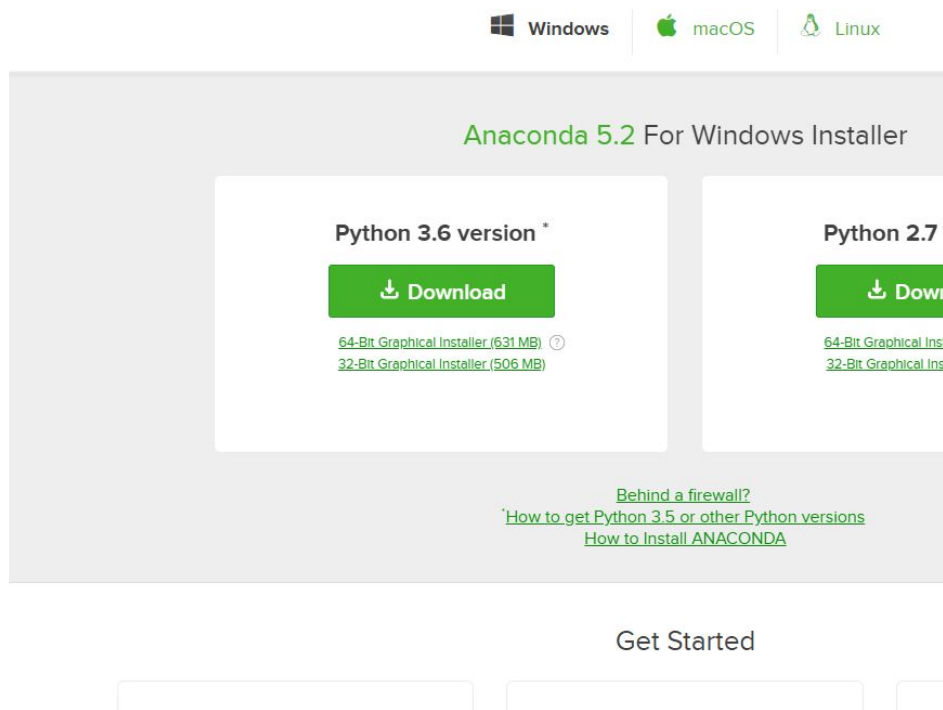
Anaconda 5.2 For Windows Installer

Python 3.6 version ^{*} [Download](#)

Python 2.7 version ^{*} [Download](#)

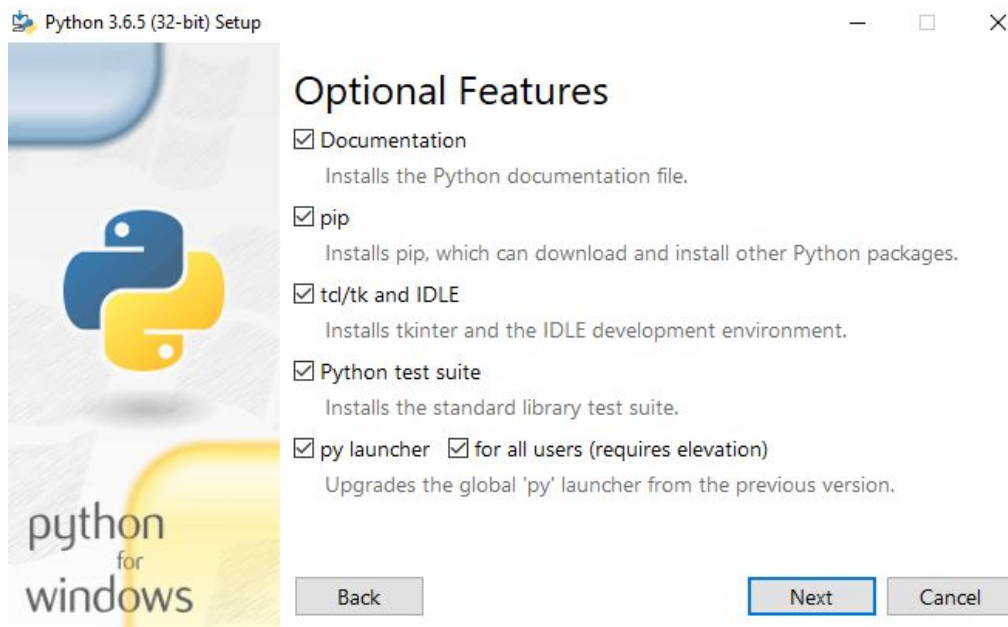
Instalação

Anaconda:



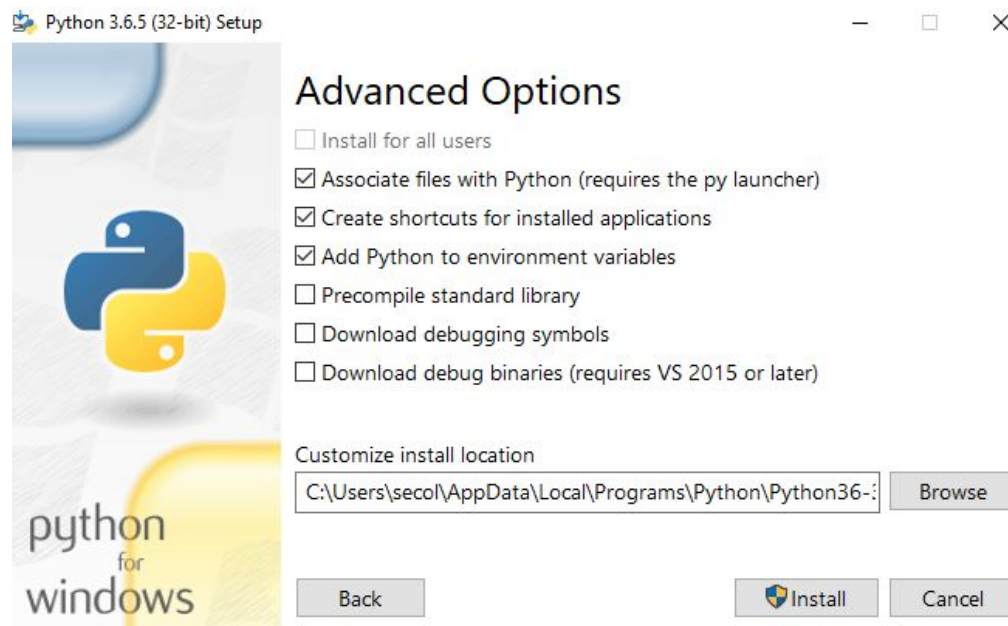
Instalação

Python:

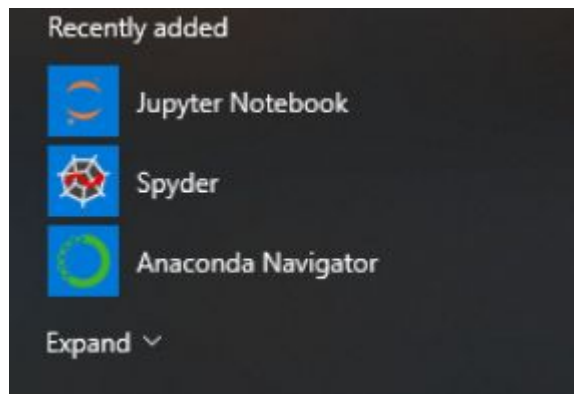


Instalação

Python:



Instalação



Instalação

Anaconda Navigator

File Help

ANACONDA NAVIGATOR

Sign in to Anaconda Cloud

[Home](#)
[Environments](#)
[Learning](#)
[Community](#)

[Documentation](#)
[Developer Blog](#)
[Feedback](#)

Applications on base (root) Channels Refresh

jupyterlab
0.32.1

An extendable environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

Launch

jupyter
notebook
5.5.0

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

Launch

qtconsole
4.3.1

PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

Launch

spyder
3.3.8

Scientific Python Development Environment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features.

Launch

vscode
1.23.1

Streamlined code editor with support for development operations like debugging, task running and version control.

Launch

glueviz
0.13.3

Multidimensional data visualization across files. Explore relationships within and among related datasets.

Install

orange3
3.13.0

Component-based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.

Install

rstudio
1.14.423

A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.

Install

Noções Básicas

Linguagens de Programação

Como falar com um computador?

Linguagens de Programação

Como falar com um computador?

- Qual tarefa um computador realmente faz?

Linguagens de Programação

Como falar com um computador?

- Qual tarefa um computador realmente faz?
- O que é lógica booleana e como operadores lógicos funcionam:



Introduzindo Operações Lógicas

AND, NAND, OR, XOR e NOT.

AND		
A	B	Saída
0	0	0
0	1	0
1	0	0
1	1	1

NAND		
A	B	Saída
0	0	1
0	1	1
1	0	1
1	1	0

OR		
A	B	Saída
0	0	0
0	1	1
1	0	1
1	1	1

XOR		
A	B	Saída
0	0	0
0	1	1
1	0	1
1	1	0

NOT	
A	Saída
0	1
1	0

Introduzindo Operações Lógicas

É verdade se as duas variáveis de entrada forem verdade (1)!

AND		
A	B	Saída
0	0	0
0	1	0
1	0	0
1	1	1

NAND		
A	B	Saída
0	0	1
0	1	1
1	0	1
1	1	0

OR		
A	B	Saída
0	0	0
0	1	1
1	0	1
1	1	1

XOR		
A	B	Saída
0	0	0
0	1	1
1	0	1
1	1	0

NOT	
A	Saída
0	1
1	0

Introduzindo Operações Lógicas

AND, NAND, OR

É verdade se ao menos uma das variáveis de entrada seja falsa (0)!

AND		
A	B	Saída
0	0	0
0	1	0
1	0	0
1	1	1

NAND		
A	B	Saída
0	0	1
0	1	1
1	0	1
1	1	0

OR		
A	B	Saída
0	0	0
0	1	1
1	0	1
1	1	1

XOR		
A	B	Saída
0	0	0
0	1	1
1	0	1
1	1	0

NOT	
A	Saída
0	1
1	0

Introduzindo Operações Lógicas

AND, NAND, OR, XOR e NOT.

É verdade se ao menos uma das variáveis de entrada seja verdadeira (1)!

AND		
A	B	Saída
0	0	0
0	1	0
1	0	0
1	1	1

NAND		
A	B	Saída
0	0	1
0	1	1
1	0	1
1	1	0

OR		
A	B	Saída
0	0	0
0	1	1
1	0	1
1	1	1

XOR		
A	B	Saída
0	0	0
0	1	1
1	0	1
1	1	0

NOT	
A	Saída
0	1
1	0

Introduzindo Operações Lógicas

AND, NAND, OR, XOR e NOT.

É verdade quando as variáveis de entrada tem valores diferentes!

AND		
A	B	Saída
0	0	0
0	1	0
1	0	0
1	1	1

NAND		
A	B	Saída
0	0	1
0	1	1
1	0	1
1	1	0

OR		
A	B	Saída
0	0	0
0	1	1
1	0	1
1	1	1

XOR		
A	B	Saída
0	0	0
0	1	1
1	0	1
1	1	0

NOT	
A	Saída
0	1
1	0

Introduzindo Operações Lógicas

AND, NAND, OR, XOR e NOT.

É verdade quando a variável de entrada é falsa e falsa quando a variável é verdadeira!

AND		
A	B	Saída
0	0	0
0	1	0
1	0	0
1	1	1

NAND		
A	B	Saída
0	0	1
0	1	1
1	0	1
1	1	0

OR		
A	B	Saída
0	0	0
0	1	1
1	0	1
1	1	1

XOR		
A	B	Saída
0	0	0
0	1	1
1	0	1
1	1	0

NOT	
A	Saída
0	1
1	0

Falando Computação

Como então conversar com a máquina?

```
var1 ← Verdadeiro
```

```
var2 ← Falso
```

```
Se var1 e var2, então faça:
```

```
    imprime "Primeira linha."
```

```
ou se var1 ou var2, então faça:
```

```
    imprime "Segunda linha."
```

```
senão, faça:
```

```
    imprime "Terceira linha."
```

Tipos de Dados

Tipo	Descrição	Sintaxe
String	Uma cadeia de caracteres	“Olá mundo!”
Integer	Número inteiro	1; 2; 43; 9000
Float	Número com ponto flutuante	1.2; 3.1416; 9.6
Boolean	Valores booleanos	True; False; 0; 1
Complex	Números complexos	4+5i
Dict	Dicionário	{ 'chave' : valor }
List	Lista mutável	[1, 2, 3, 4, 5]
Tuple	Tupla imutável	(1, 2, 3, 4, 5)

Aritmética

Operadores Aritméticos

Operador	Descrição
+	Adição
-	Subtração
*	Multiplicação
/	Divisão
%	Módulo (resto)
//	Divisão (parte inteira)
**	Exponenciação

Operadores Aritméticos

```
In [1]: print('7 + 2 = ', 7 + 2) # operação de adição  
        print('7 - 2 = ', 7 - 2) # operação de subtração  
        print('7 * 2 = ', 7 * 2) # operação de multiplicação  
        print('7 / 2 = ', 7 / 2) # operação de divisão  
        print('7 % 2 = ', 7 % 2) # resto da divisão (módulo)  
        print('7 // 2 = ', 7 // 2) # divisão absoluta  
        print('7 ** 2 = ', 7 ** 2) # potência (exponenciação)
```

```
7 + 2 = 9  
7 - 2 = 5  
7 * 2 = 14  
7 / 2 = 3.5  
7 % 2 = 1  
7 // 2 = 3  
7 ** 2 = 49
```

Operadores Lógicos

Demonstrando a utilização dos operadores **and**, **or** e **not** em Python

```
In [34]: # Criamos 3 variáveis do tipo Boolean, var_1 e var_3 receberam valor True(1 ou verdadeiro) e
# a variável var_2 recebeu o valor False (0 ou falso)

var_1 = True
var_2 = False
var_3 = True

print(var_1 and var_2) # Retorna False pois 1 and 0 = 0
print(var_1 and var_3) # Retorna True pois 1 and 1 = 1

print(var_1 or var_2) # Retorna True pois 1 or 0 = 1
print(var_1 or var_3) # Retorna True pois 1 or 1 = 1

print(not var_2) # Retorna True pois not 0 = 1
print(not var_1) # Retorna False pois not 1 = 0

False
True
True
True
True
False
```


Operadores Básicos de Comparação

Operador	Descrição	Sintaxe
Igualdade	Compara a igualdade entre dois valores	==
Maior que	Verifica se o primeiro valor é maior que o segundo valor	>
Menor que	Verifica se o primeiro valor é menor que o segundo valor	<
Maior ou igual	Verifica se o primeiro valor é maior ou igual ao segundo valor	>=
Menor ou igual	Verifica se o primeiro valor é menor ou igual ao segundo valor	<=
Diferente	Verifica se o primeiro valor é diferente do segundo valor	!=

Operadores Básicos de Comparação

```
In [35]: print(2 == 2) # Imprime True  
         print(2 > 3) # Imprime False  
         print(2 < 3) # Imprime True  
         print(3 >= 3) # Imprime True  
         print(3 <= 2) # Imprime False  
         print(3 != 3) # Imprime False
```

```
True  
False  
True  
True  
False  
False
```

Aritmética

Exercícios!

Strings

Strings

O que é uma String?

Strings

O que é uma String?

“Eu sou uma String!”

Strings

O que é uma String?

“Eu sou uma String!”

Uma string é qualquer cadeia de caracteres. Pode ser formada de uma única palavra como um nome, uma cor ou pode conter n palavras como uma mensagem, um texto...

Operações em Strings

→ Uppercase e Lowercase

```
In [40]: minha_string = 'Eu sou uma String!'

# Imprime o resultado da função upper
print(minha_string.upper())
# Imprime o resultado da função lower
print(minha_string.lower())
```

```
EU SOU UMA STRING!
eu sou uma string!
```


Operações em Strings

→ Replace

```
In [37]: nova_string = 'Abacaxi, banana, abacate, sorvete e batata!'
```

```
# Imprime o resultado da função replace
```

```
print(nova_string.replace('a', 'o'))
```

```
Abocoxi, bonono, obocote, sorvete e bototo!
```

Operações em Strings

→ Find

```
In [41]: print(nova_string.find('banana'))
```

9

Strings

Exercícios!

Listas e Arrays

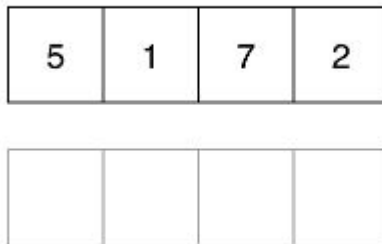
Listas e Arrays

O que é uma lista?

Listas e Arrays

O que é uma lista?

Listas são estruturas que agrupam algum tipo de dado e permitem iterações e acesso direto a cada dado.



5	1	7	2

Listas e Arrays

Acessar um dado específico em uma lista:

```
In [44]: # Declaramos a lista
minha_lista = ['Joaquim', 'Carlos', 'Luciana', 'Marcelo', 'Ellen', 'Márcia']

# Acessamos o dado guardado no índice 1 da lista
minha_lista[1]
```

```
Out[44]: 'Carlos'
```

Listas e Arrays

Acessar um dado específico em uma lista:

```
In [44]: # Declaramos a lista
minha_lista = ['Joaquim', 'Carlos', 'Luciana', 'Marcelo', 'Ellen', 'Márcia']

# Acessamos o dado guardado no índice 1 da lista
minha_lista[1]
```

```
Out[44]: 'Carlos'
```

Não deveria retornar “Joaquim”?

Listas e Arrays

Acessar um dado específico em uma lista:

```
In [44]: # Declaramos a lista
minha_lista = ['Joaquim', 'Carlos', 'Luciana', 'Marcelo', 'Ellen', 'Márcia']

# Acessamos o dado guardado no índice 1 da lista
minha_lista[1]
```

Out[44]: 'Carlos'

Não deveria retornar “Joaquim”?

```
In [45]: minha_lista[0]
```

Out[45]: 'Joaquim'

Operações Básicas em Listas

Modificar um valor:

```
In [46]: minha_lista[0] = 'Amarelo'  
         minha_lista[3] = 33  
  
         minha_lista
```

```
Out[46]: ['Amarelo', 'Carlos', 'Luciana', 33, 'Ellen', 'Márcia']
```

Operações Básicas em Listas

Selecionar valores por *range*:

```
In [47]: print(minha_lista[0:4])  
print(minha_lista[:4])  
print(minha_lista[:])  
print(minha_lista[3:6])  
  
['Amarelo', 'Carlos', 'Luciana', 33]  
['Amarelo', 'Carlos', 'Luciana', 33]  
['Amarelo', 'Carlos', 'Luciana', 33, 'Ellen', 'Márcia']  
[33, 'Ellen', 'Márcia']
```

Listas Bidimensionais

Como declarar uma matriz?

```
In [48]: lista_bidimensional = [[1, 2], [3, 4], [5, 6]]  
  
print(lista_bidimensional[0])  
print(lista_bidimensional[1][1])
```

```
[1, 2]
```

```
4
```

Funções Auxiliares

→ Append e Insert

```
In [49]: lista = ['Abacaxi', 'Banana', 'Laranja']  
         lista.append(3)  
         lista.insert(2, 'Morango')  
  
         lista
```

```
Out[49]: ['Abacaxi', 'Banana', 'Morango', 'Laranja', 3]
```

Funções Auxiliares

→ Remove

```
In [50]: lista.remove('Morango')
```

```
lista
```

```
Out[50]: ['Abacaxi', 'Banana', 'Laranja', 3]
```

Funções Auxiliares

→ Sorted

```
In [51]: lista = [33, 2, 565, 1, 0, -22]  
         sorted(lista)
```

```
Out[51]: [-22, 0, 1, 2, 33, 565]
```

Funções Auxiliares

→ Len

```
In [52]: len(lista)
```

```
Out[52]: 6
```

→ Max

```
In [53]: max(lista)
```

```
Out[53]: 565
```

→ Min

```
In [54]: min(lista)
```

```
Out[54]: -22
```


Listas e Arrays

Exercícios!

Tuplas

Tuplas

Qual a diferença entre tuplas e listas?

Tuplas

Qual a diferença entre tuplas e listas?

Tuplas são imutáveis, ou seja; Uma vez criada, uma tupla mantém seu valor até o final da execução do programa.

```
In [55]: tupla = (2, 4, 22, 'Morango')
```

```
tupla
```

```
Out[55]: (2, 4, 22, 'Morango')
```

Operações com Tuplas

Podemos fazer basicamente tudo o que fazemos com listas:

```
In [29]: print(len(tupla))
```

```
4
```

Acesso aos dados de uma tupla:

```
In [56]: print(tupla[0])  
         print(tupla[3])
```

```
2
```

```
Morango
```

Tuplas

Exercícios!

Dicionários

Dicionários

O que são dicionários?

Dicionários

O que são dicionários?

São estruturas de dados compostas por { “chave” : valor }!

```
novο_dicionario = {  
    "nome" : "Carlos",  
    "idade" : 41,  
    "sexo" : "M",  
    "peso" : 97,  
    "casado" : True  
}
```

Dicionários

Como declarar um dicionário e acessar seus dados?

```
In [57]: novo_dicionario = {  
    "nome" : "Carlos",  
    "idade" : 41,  
    "sexo" : "M",  
    "peso" : 97,  
    "casado" : True  
}  
  
# Imprime o valor referenciado pela chave em questão  
print(novo_dicionario['nome'])  
print(novo_dicionario['idade'])  
print(novo_dicionario['sexo'])  
print(novo_dicionario['peso'])  
print(novo_dicionario['casado'])
```

```
Carlos  
41  
M  
97  
True
```

Dicionários

Como declarar um dicionário e acessar seus dados?

```
In [59]: # Criamos uma instância de dicionário
outro_dicionario = dict()

# Atribuimos seus valores através de suas chaves
outro_dicionario['nome'] = 'Raul'
outro_dicionario['idade'] = 88
outro_dicionario['sexo'] = 'M'
outro_dicionario['peso'] = 80
outro_dicionario['casado'] = False
```

Dicionários

Como declarar um dicionário e acessar seus dados?

```
In [60]: print(novo_dicionario)
```

```
{'nome': 'Carlos', 'idade': 41, 'sexo': 'M', 'peso': 97, 'casado': True}
```

```
In [61]: print(outro_dicionario)
```

```
{'nome': 'Raul', 'idade': 88, 'sexo': 'M', 'peso': 80, 'casado': False}
```

Dicionários

Como verificar a existência de uma chave em um dicionário?

```
In [58]: # Imprime se encontra chave específica no dicionário  
print('nome' in novo_dicionario)  
print('endereco' in novo_dicionario)
```

```
True  
False
```

Dicionários

Exercícios!

Condicionais

Condicionais

O que são condições como são executadas?

Condicionais

O que são condições como são executadas?

Se **condição**, então:

Faça isso.

Ou se **condição**, então:

Faça isso.

Senão:

Faça isso.

Condicionais

O que são condições como são executadas?

```
In [62]: var_a = 32

if var_a < 40:
    print('É menor que 40')
elif var_a == 40:
    print('É igual a 40')
else:
    print('É maior que 40')
```

É menor que 40

Condicionais

Vamos entender o que ocorre!

```
In [63]: numero_1 = 468
          numero_2 = 37

          # Verificando o resto da divisão por 2
          print(numero_1 % 2)
          print(numero_2 % 2)
```

```
0
1
```

Condicionais

Vamos entender o que ocorre!

```
In [63]: numero_1 = 468
          numero_2 = 37

          # Verificando o resto da divisão por 2
          print(numero_1 % 2)
          print(numero_2 % 2)

0
1
```

Quando utilizamos operadores de comparação:

```
In [64]: print(numero_1 % 2 == 0)
          print(numero_2 % 2 == 0)

True
False
```

Condicionais

Um exemplo melhor:

```
In [65]: if numero_1 % 2 == 0:
          print('Primeiro número é par')
        else:
          print('Primeiro número é ímpar')

          if numero_2 % 2 == 0:
            print('Segundo número é par')
          else:
            print('Segundo número é ímpar')
```

Primeiro número é par
Segundo número é ímpar

Condicionais

Exercícios!

Loops

Loops

O que são loops e qual seu propósito?

Loops

O que são loops e qual seu propósito?

→ For Loops

Varrem uma lista/array de informação e executam um bloco de ação para cada item encontrado ou para cada instância de execução.

Para cada **item** em uma **lista de itens**, faça:
Bloco de ação...

Loops

O que são loops e qual seu propósito?

→ For Loops

```
In [66]: lista_itens = ['Raul', 'José', 23, 44, True, False]
```

```
for item in lista_itens:  
    print(item)
```

```
Raul  
José  
23  
44  
True  
False
```

Loops

O que são loops e qual seu propósito?

→ For Loops

```
In [67]: for index, item in enumerate(lista_itens):  
         print('Item : {0} na posição {1} da lista'.format(item, index))
```

```
Item : Raul na posição 0 da lista  
Item : José na posição 1 da lista  
Item : 23 na posição 2 da lista  
Item : 44 na posição 3 da lista  
Item : True na posição 4 da lista  
Item : False na posição 5 da lista
```

Loops

O que são loops e qual seu propósito?

→ While Loops

Executam um bloco de ação enquanto alguma condição é satisfeita.

Enquanto **condição** é satisfeita, faça:
Bloco de ação...

Loops

O que são loops e qual seu propósito?

→ While Loops

```
In [70]: contador = 0  
  
while contador < 10:  
    contador += 1  
  
print(contador)
```

10

Loops

Um exemplo:

Imagine que temos uma lista onde queremos somar seus valores, porém, queremos que caso o valor seja o número 7 a soma não ocorra e caso o número seja 10, a soma dos valores desta lista seja interrompida.

Loops

Um exemplo:

Imagine que temos uma lista onde queremos somar seus valores, porém, queremos que caso o valor seja o número 7 a soma não ocorra e caso o número seja 10, a soma dos valores desta lista seja interrompida.

```
In [71]: lista = [1, 6, 4, 7, 9, 33, 10, 42, 98]

contador = 0

for item in lista:
    if item == 7:
        continue
    elif item == 10:
        break
    else:
        contador += item

print(contador)
```

Loops

Exercícios!

Funções

Funções

O que são funções?

Funções

O que são funções?

Funções são blocos de código capazes de efetuar uma determinada ação toda vez que são chamados.

Funções podem ou não receber valores como parâmetros, vale lembrar que estes valores são apenas visíveis ao bloco da função, ou seja, não tem definição fora deste bloco (escopo).

```
def nome_da_funcao():  
    #bloco de código dentro do escopo da função
```

Funções

Declarando e executando funções:

```
In [73]: def print_bom_dia():  
         print('Bom dia!')  
  
         # Executa a função criada acima  
         print_bom_dia()  
  
         Bom dia!
```

Funções que recebem parâmetros de entrada:

```
In [74]: def print_bom_dia(nome):  
         print('Bom dia, {}'.format(nome))  
  
         # Executa a função criada acima, passando Jorge como parâmetro  
         print_bom_dia('Jorge')  
  
         Bom dia, Jorge!
```

Funções

Funções com parâmetros *default*:

```
In [75]: def print_bom_dia(nome = 'Raul'):
          print('Bom dia, {}'.format(nome))

          # Executa a função criada acima, passando Jorge como parâmetro
          print_bom_dia('Jorge')
          # Executa a função criada acima, utilizando o valor padrão para o parâmetro nome
          print_bom_dia()
```

Bom dia, Jorge!

Bom dia, Raul!

Funções

Passando parâmetros para funções:

```
def funcao(nome, idade, sexo):  
    # código...  
  
funcao('Jorge', 32, 'masculino')
```

```
def funcao(nome, idade, sexo):  
    # código...  
  
funcao(idade = 23, sexo = 'feminino', nome = 'Gabrielle')
```

Funções

Passando parâmetros para funções:

```
In [76]: def funcao(nome, idade, sexo):  
         print('{0} tem {1} anos, sexo {2}'.format(nome, idade, sexo))  
  
         funcao('Jorge', 32, 'masculino')  
         funcao(idade = 23, sexo = 'feminino', nome = 'Gabrielle')  
  
         Jorge tem 32 anos, sexo masculino  
         Gabrielle tem 23 anos, sexo feminino
```

Funções

Funções que retornam valores:

```
def retorna_tupla(numero):  
    return (numero ** 2, numero * 2)
```

```
def retorna_tupla(numero):  
    valor_potencia = numero ** 2  
    valor_multiplicacao = numero * 2  
    return (valor_potencia, valor_multiplicacao)
```


Funções

Funções que retornam valores:

```
In [77]: def retorna_tupla(numero):  
         return (numero ** 2, numero * 2)  
  
         retorno_potencia, retorno_multiplicacao = retorna_tupla(8)  
  
         print(retorno_potencia)  
         print(retorno_multiplicacao)
```

64

16

Funções

Exercícios!