

Desvendando programação com Python

O conteúdo deste material é licenciado sob a
Licença Atribuição Creative Commons 3.0 Brasil
(CC BY 3.0 BR)

<https://creativecommons.org/licenses/by/3.0/br/>



Copyright 2021

Carlos ROLAND



python™

PARA APRENDER E TREINAR

Insane



PROGRAMMING

70+ Python Projects For Beginners, Intermediate And Experienced Developers

🕒 JUNE 02, 2021 💬 0

PARA APRENDER E TREINAR

CISCO Networking Academy



[Cursos](#) [Carreira](#) [Suporte](#) [Mais](#)



[Português](#)

[Entrar](#)

Capacitação de todos com possibilidades de carreira

A Cisco Networking Academy transforma a vida de alunos, educadores e comunidades com o poder da tecnologia, da educação e das oportunidades de carreira. Disponível para qualquer pessoa, em qualquer lugar.

No momento, estamos oferecendo assistência para que você ensine e aprenda remotamente.

[Saber mais](#)



Uma oportunidade incrível está esperando por você. A tecnologia está mudando o mundo ao conectar bilhões de dispositivos e melhorar a maneira como vivemos, trabalhamos, nos divertimos e tratamos nosso planeta. Nenhum setor está imune. Está pronto para mudar sua vida e, possivelmente, fazer do mundo um lugar melhor?



Redes



SO e TI



Programação



Internet das Coisas



Automação da infraestrutura



Segurança cibernética



Packet Tracer

PARA APRENDER E TREINAR

PCAP - Programming Essentials in Python



My NetAcad ▾ Resources ▾ Courses ▾ Careers ▾ More ▾



Carlos ... ▾

[Home](#) / I'm Learning

Last login on 06/19/2021 at 17:41 PM

I'm Learning

[Refresh Status](#)

[Browse Course Catalog](#)

Courses I've Enrolled In

Status

Search by Course name or ID



All Statuses



In Progress

-acm-

PCAP - Programming Essentials in Python English 0621...

Global Academy




Partner: PCAP - Programming Essentials in Python

Please finish by 30 Dec 2021

[Un-enroll](#)

VAMOS PROGRAMAR!


Para treinar: URI Online Judge - Problems & Contests

ENGLISH

LOGINREGISTERFORUMCONTESTSPROBLEMRANKSFOR PROFESSORSFOR COMPANIES

URI ONLINE JUDGE


THE URI ONLINE JUDGE



The URI Online Judge is a project that is being developed by the Computer Science Department of URI University. The main goal of the project is to provide programming practice and knowledge sharing.

CREDITS


PROBLEM REPOSITORY



The URI Online Judge contains more than 1000 problems divided in 8 big categories. This division help the users to focus on specific programming topics. All problems are available in Portuguese and English.

REPOSITORY

URI ONLINE JUDGE FORUM



The URI Online Judge Forum is the right place for you to get help and to help other users. Share your knowledge and experience in algorithms and

URI ONLINE JUDGE PROBLEMS & CONTESTS

SIGN IN

EMAIL

PASSWORD

☐ REMEMBER ME (7 DAYS)

SIGN IN

FACEBOOKGOOGLEGITLAB


GITHUBBITBUCKETTWITCH

§ By signing up with a social platform you AGREE with the URI Online Judge Terms & Conditions.


You need to allow your EMAIL to be provided by the social sign in. Therefore, allow the default permissions when prompted.

FIRST TIME HERE?
SIGN UP today to view materials, solve problems, ask questions and much more.

RESET PASSWORDACTIVATE

 **PRÊMIO GUIA DO ESTUDANTE DESTAQUES DO ANO**


COMPETITION AND RANKING



Solve the available problems using 11 programming languages, and compete with other users. As a challengee, improve your ranking, solving as many problems as possible and tuning your source code to run faster.

CHECK THE RANK


URI ONLINE JUDGE ACADEMIC



The URI Online Judge Academic is an unique module for professors and team coaches. Here you can create disciplines and lists of exercises. You can also track the progress of your students giving feedback in real time.

ACCESS ACADEMIC

URI ONLINE JUDGE CONTESTS



The URI Online Judge website also has public contests on a regular basis. Get in touch with us to host your contest at URI Online Judge for free. Create new

Estruturas de dados - Listas/Dicionários

```
# script file: wordsCount.py
```

```
name = input('Enter file:')
```

```
handle = open(name, 'r')
```

```
counts = dict()
```

```
for line in handle:
```

```
    words = line.split()
```

```
    for word in words:
```

```
        counts[word] = counts.get(word, 0) + 1
```

```
bigcount = None
```

```
bigword = None
```

```
for word, count in list(counts.items()):
```

```
    if bigcount is None or count > bigcount:
```

```
        bigword = word
```

```
        bigcount = count
```

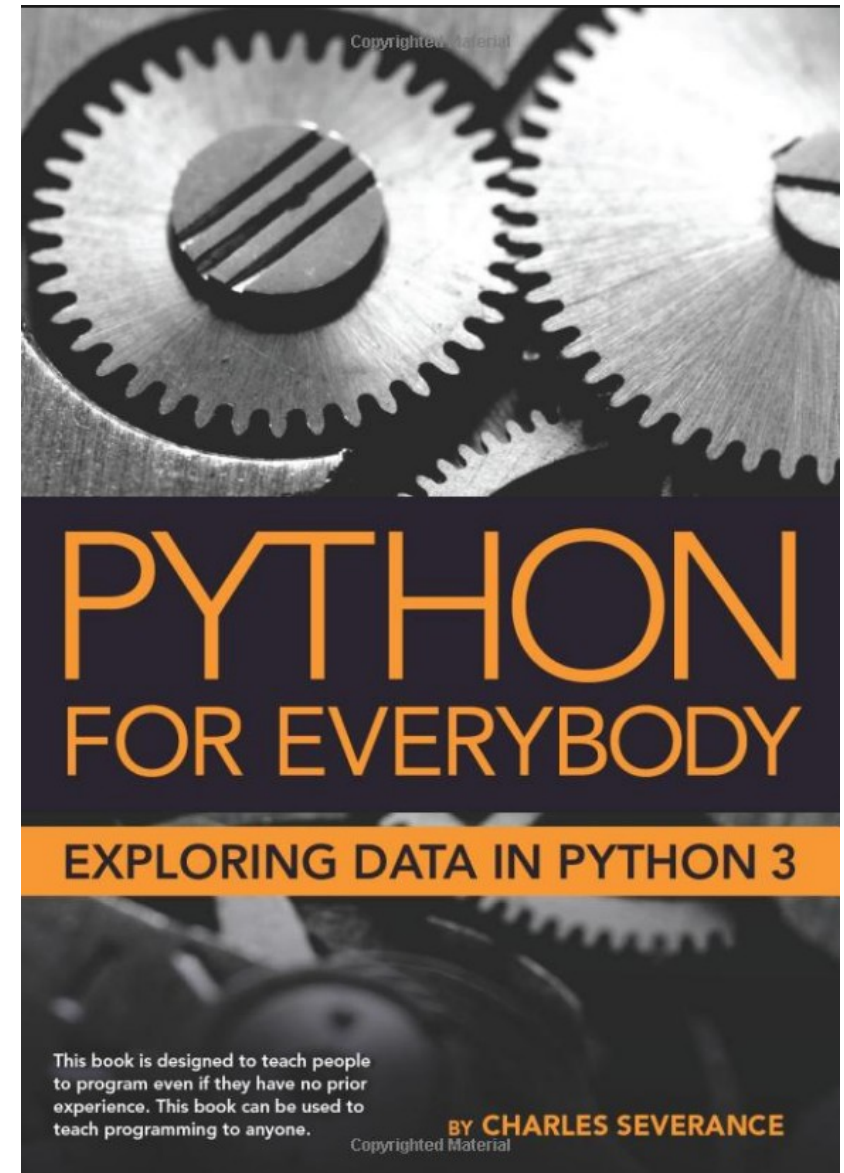
```
print(bigword, bigcount)
```

Estruturas de dados - Listas/Dicionários

Python for Everybody

eBook (PDF)

eBook PT-BR (PDF)



Estruturas de dados - Listas

Listas são sequências

- . Valores de quaisquer tipos
 - . strings também são sequências (caracteres)
 - . compostas por **elementos** ou **itens**
- . Criadas com [...]
 - . [10, 20, 30, 40]
 - . ['terra', 'marte', 'júpiter', 'saturno']
 - . ['spam', 2.0, 5, [10, 20]]

Estruturas de dados - Listas

. Atribuídas a variáveis

```
>>> cheeses = ['Cheddar', 'Edam', 'Gouda']
```

```
>>> numbers = [17, 123]
```

```
>>> empty = []
```

```
>>> print(cheeses, numbers, empty)  
['Cheddar', 'Edam', 'Gouda'] [17, 123] []
```

Estruturas de dados - Listas

- . Para acesso aos elementos (como *strings*)

```
>>> print(cheeses[0])  
Cheddar
```

- . Índices iniciam sempre em **ZERO**

Estruturas de dados - Listas

Listas são **Mutáveis**

. (ao contrário de *strings*)

```
>>> numbers = [17, 123]
```

```
>>> numbers[1] = 5
```

```
>>> print(numbers)  
[17, 5]
```

Estruturas de dados - Listas

Listas são relacionamentos

- . índices -> elementos
- . chamados *mapeamentos*
índice mapeia um elemento

Estruturas de dados - Listas

- . índices em **listas** funcionam como em ***strings***

expressões inteiras podem ser índices

índices inexistentes provocam erros

índices negativos mapeiam do fim -> começo

Estruturas de dados - Listas

. o operador **in** também funciona

```
>>> cheeses = ['Cheddar', 'Edam', 'Gouda']
```

```
>>> 'Edam' in cheeses  
True
```

```
>>> 'Brie' in cheeses  
False
```

Estruturas de dados - Listas

Percorrendo uma lista (apenas leitura)

```
>>> cheeses = ['Cheddar', 'Edam', 'Gouda']
```

```
>>> for cheese in cheeses:  
    print(cheese)
```

Inclusão ou alteração (com índice)

```
>>> for i in range(len(numbers)):  
    numbers[i] = numbers[i] * 2
```

range(n) – retorna **lista** de 0 a n-1

Estruturas de dados - Listas

Listas aninhadas

```
>>> example = ['spam', 1, ['Brie', 'Roquefort', 'Pol le Veq'], [1, 2, 3]]
```

```
>>> print(len(example))
```

Estruturas de dados - Listas

Operações com listas (+ e *)

```
>>> a = [1, 2, 3]
```

```
>>> b = [4, 5, 6]
```

```
>>> c = a + b
```

```
>>> print(c)
```

```
[1, 2, 3, 4, 5, 6]
```

```
>>> [0] * 4
```

```
[0, 0, 0, 0]
```

```
>>> [1, 2, 3] * 3
```

```
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

Estruturas de dados - Listas

Fatiamento de listas

```
>>> t = ['a', 'b', 'c', 'd', 'e', 'f']
```

```
>>> t[1:3]  
['b', 'c']
```

```
>>> t[:4]  
['a', 'b', 'c', 'd']
```

```
>>> t[3:]  
['d', 'e', 'f']
```


Estruturas de dados - Listas

Fatiamento de listas para atualizações

```
>>> t = ['a', 'b', 'c', 'd', 'e', 'f']
```

```
>>> t[1:3] = ['x', 'y']
```

```
>>> print(t)  
['a', 'x', 'y', 'd', 'e', 'f']
```

Estruturas de dados - Listas

Métodos para operação com listas

```
>>> t = ['a', 'b', 'c']
```

```
>>> t.append('d')
```

```
>>> print(t)
['a', 'b', 'c', 'd']
```

```
>>> t1 = ['a', 'b', 'c']
```

```
>>> t2 = ['d', 'e']
```

```
>>> t1.extend(t2)
```

```
>>> print(t1)
['a', 'b', 'c', 'd', 'e']
```

Estruturas de dados - Listas

Métodos para operação com listas

```
>>> t = ['d', 'c', 'e', 'b', 'a']
```

```
>>> t.sort()
```

```
>>> print(t)
['a', 'b', 'c', 'd', 'e']
```

O que acontece com isso?????

```
>>> t = t.sort()
```

Estruturas de dados - Listas

Deletando elemento (índice e salvando dado)

```
>>> t = ['a', 'b', 'c']
```

```
>>> x = t.pop(1)
```

```
>>> print(t)  
['a', 'c']
```

```
>>> print(x)  
b
```

Estruturas de dados - Listas

Deletando elemento (último e salvando dado)

```
>>> t = ['a', 'b', 'c']
```

```
>>> x = t.pop()
```

```
>>> print(t)  
['a', 'b']
```

```
>>> print(x)  
c
```


Estruturas de dados - Listas

Deletando elemento (índice e NÃO salvando)

```
>>> t = ['a', 'b', 'c']
```

```
>>> del t[1]
```

```
>>> print(t)  
['a', 'c']
```

Estruturas de dados - Listas

Deletando elementos (faixa e NÃO salvando)

```
>>> t = ['a', 'b', 'c', 'd', 'e', 'f']
```

```
>>> del t[1:5]
```

```
>>> print(t)  
['a', 'f']
```

Estruturas de dados - Listas

Deletando elementos (NÃO salvando)

```
>>> t = ['a', 'b', 'c']
```

```
>>> t.remove('b')
```

```
>>> print(t)  
['a', 'c']
```

Estruturas de dados - Listas

Listas e funções

```
>>> nums = [3, 41, 12, 9, 74, 15]
```

```
>>> print(len(nums))
```

```
6
```

```
>>> print(max(nums))
```

```
74
```

```
>>> print(min(nums))
```

```
3
```

```
>>> print(sum(nums))
```

```
154
```

```
>>> print(sum(nums)/len(nums))
```

```
25
```

Estruturas de dados - Listas

Listas e funções

`sum()` só para listas numéricas

`len()`, `max()` e `min()` para listas numéricas, *strings*, e outras

Mãos na massa

Calcular média de números

```
total = 0
count = 0
while (True):
    inp = input('Enter a number: ')
    if inp == 'done': break
    value = float(inp)
    total = total + value
    count = count + 1
```

```
average = total / count
print('Average:', average)
```

```
total = 0
count = 0
while (True):
    inp = input('Enter a number: ')
    if inp == 'done': break
    value = float(inp)
    total += value
    count += 1
```

```
average = total / count
print('Average:', average)
```

Time for
a Break

Mãos na massa

Calcular média de números com lista

```
numlist = list()
while (True):
    inp = input('Enter a number: ')
    if inp == 'done': break
    value = float(inp)
    numlist.append(value)

average = sum(numlist) / len(numlist)
print('Average:', average)
```

Mãos na massa

Calcular média de números com lista

```
numlist = list()
while (True):
    inp = input('Enter a number: ')
    if inp == 'done': break
    value = float(inp)
    numlist.append(value)

average = sum(numlist) / len(numlist)
print('Average:', average)
```

```
total = 0
count = 0
while (True):
    inp = input('Enter a number: ')
    if inp == 'done': break
    value = float(inp)
    total += value
    count += 1

average = total / count
print('Average:', average)
```

Estruturas de dados - Listas

Listas e *strings* – separar elementos

```
>>> s = 'spam'
```

```
>>> t = list(s)
```

```
>>> print(t)  
['s', 'p', 'a', 'm']
```

Estruturas de dados - Listas

Listas e *strings* – separar elementos

```
>>> s = 'pining for the fjords'
```

```
>>> t = s.split()
```

```
>>> print(t)
['pining', 'for', 'the', 'fjords']
```

```
>>> print(t[2])
the
```

```
>>> s = 'spam-spam-spam'
>>> delimiter = '-'
>>> s.split(delimiter)
['spam', 'spam', 'spam']
```

Estruturas de dados - Listas

Listas e *strings* – juntar elementos - **join()**

join() é um método e então tem que ser invocado por um **objeto** com a **lista** como parâmetro

```
>>> t = ['pining', 'for', 'the', 'fjords']
```

```
>>> delimiter = ' '
```

```
>>> delimiter.join(t)
'pining for the fjords'
```


Analizando arquivos - Dicionários

```
# script file: wordsCount.py
```

```
name = input('Enter file:')
```

```
handle = open(name, 'r')
```

```
counts = dict()
```

```
for line in handle:    # percorrer linhas
```

```
    words = line.split() # separar palavras
```

```
    for word in words:    # percorrer palavras
```

```
        counts[word] = counts.get(word, 0) + 1
```

```
bigcount = None
```

```
bigword = None
```

```
for word, count in list(counts.items()):
```

```
    if bigcount is None or count > bigcount:
```

```
        bigword = word
```

```
        bigcount = count
```

```
print(bigword, bigcount)
```

Estruturas de dados – Dicionários

Dicionários - listas de pares **{chave: valor}**

Listas – índices são inteiros

Dicionários – índices (quase) qualquer tipo

Para criar – **dict()**

```
>>> eng2pt = dict()
```

```
>>> print(eng2pt)  
{
```

Estruturas de dados – Dicionários

Para inserir itens

```
>>> eng2pt['one'] = 'um'
```

```
>>> print(eng2pt)
{'one': 'um'}
```

```
>>> eng2pt = {'one': 'um', 'two': 'dois', 'three': 'tres'}
```

```
>>> print(eng2pt)
{'one': 'um', 'three': 'tres', 'two': 'dois'}
```

(ordem dos elementos é imprevisível)

Estruturas de dados – Dicionários

Para acessar itens – pela **chave**

```
>>> print(eng2pt['two'])  
'dois'
```

```
>>> print(eng2pt['four'])  
KeyError: 'four'
```

Estruturas de dados – Dicionários

Para acessar itens – operador **in** com **chave**

```
>>> 'one' in eng2pt  
True
```

```
>>> 'um' in eng2pt  
False
```

Estruturas de dados – Dicionários

Para acessar itens – por **valor**

```
>>> vals = list(eng2pt.values())
```

```
>>> 'um' in vals
```

```
True
```

Estruturas de dados – Dicionários

Para tamanho – número de pares (chave:valor)

```
>>> len(eng2pt)  
3
```

Estruturas de dados – Dicionários

Exemplo – contar letras em *strings*

```
word = 'brontosaurus'
d = dict()
for c in word:
    if c not in d:
        d[c] = 1
    else:
        d[c] = d[c] + 1

print(d)
```


Estruturas de dados – Dicionários

Dicionários têm método **get(chave, default)**
se chave existe – retorna valor correspondente
senão – retorna valor default

```
>>> eng2pt = {'one': 'um', 'two': 'dois', 'three': 'tres'}
```

```
>>> print(eng2pt.get('two', 0))  
'dois'
```

```
>>> print(eng2pt.get('four', 0))  
0
```

Estruturas de dados – Dicionários

Exemplo – contar letras em *strings* com **get()**

```
word = 'brontosaurus'
d = dict()
for c in word:
    if c not in d:
        d[c] = 1
    else:
        d[c] = d[c] + 1

print(d)
```

```
word = 'brontosaurus'
d = dict()
for c in word:
    d[c] = d.get(c,0) + 1

print(d)
```

Estruturas de dados – Dicionários

Exemplo – contar palavras em textos (romeo.txt)

But soft what light through yonder window breaks
It is the east and Juliet is the sun
Arise fair sun and kill the envious moon
Who is already sick and pale with grief

Estruturas de dados – Dicionários

Exemplo – contar palavras em textos (prevenindo erros com arquivos)

```
fname = input('Enter the file name: ')
try:
    fhand = open(fname)
except:
    print('File cannot be opened:', fname)
    exit()

counts = dict()
for line in fhand:          # loop externo
    words = line.split()
    for word in words:       # loop interno
        if word not in counts:
            counts[word] = 1
        else:
            counts[word] += 1

print(counts)
```

Estruturas de dados – Dicionários

Exemplo – contar palavras em textos (prevenindo erros com arquivos)

Enter the file name: romeo.txt

```
{'and': 3, 'envious': 1, 'already': 1, 'fair': 1,  
'is': 3, 'through': 1, 'pale': 1, 'yonder': 1,  
'what': 1, 'sun': 2, 'Who': 1, 'But': 1, 'moon': 1,  
'window': 1, 'sick': 1, 'east': 1, 'breaks': 1,  
'grief': 1, 'with': 1, 'light': 1, 'It': 1, 'Arise': 1,  
'kill': 1, 'the': 3, 'soft': 1, 'Juliet': 1}
```

Estruturas de dados – Dicionários

Para percorrer itens – **for** sobre as chaves

```
eng2pt = {'one': 'um', 'two': 'dois', 'three': 'tres'}
```

```
for chave in eng2pt:  
    print(chave, eng2pt[chave])
```

(ordem dos elementos é imprevisível)

Estruturas de dados – Dicionários

Para percorrer itens – ordenando

```
eng2pt = {'one': 'um', 'two': 'dois', 'three': 'tres'}
```

```
# extrai chaves para lista e ordena
```

```
list_keys = list(eng2pt.keys())
```

```
print(list_keys)
```

```
list_keys.sort()
```

```
for key in list_keys:
```

```
    print(key, eng2pt[key])
```

Estruturas de dados – Dicionários

Exemplo – contar palavras em textos pontuados
e *caps* (romeo2.txt)

But, soft! what light through yonder window breaks?
It is the east, and Juliet is the sun.
Arise, fair sun, and kill the envious moon,
Who is already sick and pale with grief,

Usar métodos de *strings* **lower()**, **punctuation()**, **translate()**

`line.translate(str.maketrans(fromstr, tostr, deletestr))`

*Replace the characters in **fromstr** with the character in the same position in **tostr** and delete all characters that are in **deletestr**. The **fromstr** and **tostr** can be **empty** strings and the **deletestr** parameter can be omitted.*

Estruturas de dados – Dicionários

Exemplo – contar palavras em textos pontuados
e *caps* (romeo2.txt)

Python sabe quais caracteres são as pontuações:

```
>>> import string  
>>> string.punctuation  
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
```

Estruturas de dados – Dicionários

Exemplo – contar palavras em **textos pontuados**
e com **caps** (romeo2.txt)

```
import string
fname = input('Enter the file name: ')
try:
    fhand = open(fname)
except:
    print('File cannot be opened:', fname)
    exit()

counts = dict()
for line in fhand:
    line = line.rstrip()
    line = line.translate(line.maketrans("", "", string.punctuation))
    line = line.lower()
    words = line.split()
    for word in words:
        if word not in counts:
            counts[word] = 1
        else:
            counts[word] += 1

print(counts)
```

Estruturas de dados – Dicionários

Exemplo – processar arquivos de msgs email

mbox-short.txt

mbox-short2.txt

mbox.txt

mbox2.txt

Estruturas de dados – Dicionários

Exemplo – processar arquivos de msgs e-mail

Leia as linhas do arquivo e identifique, nas com “From”, os endereços dos remetentes. Conte o número de mensagens de cada pessoa.

Após terminada a leitura de todos os registros do arquivo, imprima o remetente com a maior quantidade de mensagens.

Crie uma lista com tuplas (qtde, e-mail) a partir do dicionário de remetentes e então ordene em ordem reversa para imprimir a pessoa com maior quantidade de mensagens.

Estruturas de dados – Dicionários

Exemplo – processar arquivos de msgs email

```
fname = input('Enter a file name: ')
if len(fname) < 1: fname = "mbox-short.txt"

try:
    fhandle = open(fname)
except:
    print 'Error opening file.'
    exit()

address = dict()
for line in fhandle:
    line = line.rstrip()

    if len(line) == 0 or 'From ' not in line: continue
    wrds = line.split()

    # the second term of the line is the e-mail address
    address[wrds[1]] = address.get(wrds[1], 0) + 1

lst = list()
for key, val in address.items():
    lst.append( (val, key) )

lst.sort(reverse=True)

count, email = lst[0]
print(email, count)
```



That's all Folks!