

# 24 Reunião 24: 13/JUL/2021



## 24.1 Reuniões passadas

- tipos nativos `str`, `int`, `float`, `bool`, `NoneType`, `list`, `dict`
  - `strip()`: `s.strip()` retorna uma string com `s` sem brancos no início e no final
  - `split()`: `s.split()` retorna uma lista de strings
  - mutabilidade: strings são **imutáveis** listas e dicionários são **mutáveis**
  - operador `in`: `in range()`, `in str`, `in list` e `in dict`
- componentes podem ser atualizados*  
↓

tipo	<code>... in ...</code>	<code>if ... in ...:</code>	<code>for ... in ..."</code>
<code>int</code>	<code>i in range(10)</code>	<code>if i in range(10):</code>	<code>for i in range(10):</code>
<code>str</code>	<code>c in s</code>	<code>if c in s:</code>	<code>for c in s:</code>
<code>list</code>	<code>item in lst</code>	<code>if item in lst:</code>	<code>for item in lst:</code>
<code>dict</code>	<code>chave in dicio</code>	<code>if chave in dicio:</code>	<code>for chave in dicio:</code>

## Mais dicionários



Um **dicionário** (tipo `dict`) é um conjunto de objetos ou itens cada um dotado de uma **chave** e de um **valor**.

Dicionários possuem vários métodos nativos úteis. A seguinte tabela fornece um resumo e mais detalhes podem ser encontrados em [Python Documentation](#).

---

Método	Parâmetros	Descrição
<code>keys()</code>	nenhum	Retorna uma vista das chaves no dicionário
<code>values()</code>	nenhum	Retorna uma vista dos valores no dicionário
<code>items()</code>	nenhum	Retorna uma vista dos pares chave-valor no dicionário
<code>get()</code>	key	Retorna o valor associado com a chave; ou <b>None</b>
<code>get()</code>	key,alt	Retorna o valor associado com a chave; ou <b>alt</b>

---

```
In [1]: def limpa( d ):
...:     for chave in d:
...:         print(chave, ':', d[chave])
...:         d[chave] = 0
...:
```

```
In [2]: quitanda = {'banana': 121, 'caqui': 55, 'kiwi': 32}
```

```
In [3]: print(quitanda)
{'banana': 121, 'caqui': 55, 'kiwi': 32}
```

```
In [4]: quitanda['caju']
```

---

```
KeyError Traceback (most recent call last)
<ipython-input-4-255e0fa26b02> in <module>
----> 1 quitanda['caju']
```

```
KeyError: 'caju'
```

```
In [5]: 'caju' in quitanda
```

```
Out[5]: False
```

```
In [6]: 'caqui' in quitanda
```

```
Out[6]: True
```

```
In [7]: quitanda['caqui']
```

```
Out[7]: 55
```

```
In [9]: quitanda.keys()
```

```
Out[9]: dict_keys(['banana', 'caqui', 'kiwi'])
```

```
In [10]: quitanda.values()
```

```
Out[10]: dict_values([121, 55, 32])
```

```
In [11]: quitanda.items()
```

```
Out[11]: dict_items([('banana', 121), ('caqui', 55), ('kiwi', 32)])
```

## 24.2 Hoje



## 24.3 Exercício: shell2

Neste exercício vocês escreverão um programa que

- lê um arquivo `.txt` como uma string;
- cria um dicionário em que as chaves são fatias de tamanho `k` dessa string e o valor associado de cada chave é o número de ocorrências da fatia na string;
- responde consultas sobre o conteúdo do dicionário.

A seguir está um roteiro de como proceder. Vocês terão que escrever apenas uma função.



### Baixe os arquivos

Baixem o arquivo `shell2.py` e baixem [daqui](#) os arquivos `pi-30.txt` e `virus-30.txt` para a mesma pasta em que baixou o programa `shell2.py`.

O arquivo `shell2.py` tem três funções:

- `main()`: responsável pela interação. **Está completa.**
- `init_dicionario()`: que vocês **deverão escrever.**
- `leia_arquivo()`: lê um arquivo e retorna uma string com seu conteúdo. **Está completa.**
- `help()`: usada pela `main()` para apresentar o menu de uso do programa sempre que pedido. **Está completa.**

## `init_dicionario()`: exemplos

A seguir estão exemplos de execução da função `init_dicionario()` no IPython:

```
In [4]: s = "31415926"
```

```
In [5]: d = init_dicionario(s)
```

```
In [6]: d
```

```
Out[6]: {'3': 1, '1': 2, '4': 1, '5': 1, '9': 1, '2': 1}
```

```
In [7]: d = init_dicionario(s,2)
```

```
In [8]: d
```

```
Out[8]: {'31': 1, '14': 1, '41': 1, '15': 1, '59': 1, '92': 1}
```

```
In [9]: s = "GAATTGCTAGC"
```

```
In [10]: d = init_dicionario(s,3)
```

```
In [11]: d
```

```
Out[11]:
```

```
{'GAA': 1,  
  'AAT': 1,  
  'ATT': 1,  
  'TTG': 1,  
  'TGC': 1,  
  'GCT': 1,  
  'CTA': 1,  
  'TAG': 1}
```

## programa: exemplos

Depois de escrever e testar a função `init_dicionario()` execute o programa para ver

```
In [1]: leia
```

```
Digite o nome do arquivo com a string: virus-30.txt
```

```
Out[1]: arquivo lido
```

```
In [2]: mostre
```

```
Out[2]: dict_items([])
```

```
In [3]: crie
```

```
Digite o tamanho da fatia: 3
```

```
Out[3]: dicionário criado
```

```
In [4]: mostre
```

```
Out[4]: dict_items([('GAA', 1), ('AAT', 3), ('ATT', 3), ('TTG', 3)])
```

```
In [5]: AAT
```

```
Out[5]: 3
```

```
In [6]: limpe
```

```
Out[6]: dicionario limpo
```

```
In [7]: mostre
```

```
Out[7]: dict_items([])
```

```
In [8]: crie
```

```
Digite o tamanho da fatia: 5
```

```
Out[8]: dicionário criado
```

In [9]: mostre

Out[9]: dict\_items([('GAATT', 1), ('AATTG', 3), ('ATTGC', 3)],

In [10]: AATTG

Out[10]: 3

In [11]: ? *← ajuda*

Out[11]: Digite:

- uma chave no dicionário para saber o valor associado
- sair : sair do programa
- leia : lê o conteúdo de um arquivo
- crie : cria um dicionário com fatias da string lida
- mostre : exhibe os itens no dicionário
- len : tamanho do dicionário
- limpe : limpa o dicionário
- ? : exhibe este menu

In [12]: sair



## Solução

```
def init_dicionario(s, k=1):  
    '''(str, int) -> dict  
    RECEBE um string s e um inteiro `k`.  
    RETORNA um dicionário em que:  
        - as chaves as fatias de `s` com `k` caracteres  
        - os valores são o número de ocorrências das respectivos  
    '''  
  
    # crie o dicionário  
    d = {}  
    # percorra todos os subnúmeros de pi de tamanho k  
    n = len(s)  
    for i in range(k, n):  
        # pegue a próxima fatia  
        fatia = s[i-k:i]  
        # verifique se a fatia está no dicionário  
        if fatia in d:  
            d[fatia] += 1 ← atualiza  
        else:  
            d[fatia] = 1 ← novo par chave: valor  
                               fatia : 1  
    return d
```

## 24.4 Exercício: shell2 *plus*

Agora vocês estenderão o programa do exercício anterior acrescentando a função `maior_valor()` que **recebe** um dicionário e **retorna** o maior valor de uma chave e a lista com todas as chaves com esse valor.

Baixem o arquivo `shell2plus.py` para a mesma pasta que estão os arquivos `pi-30.txt` e `virus-30.txt`



### `maior_valor()`: exemplos

A seguir estão exemplos de execução da função `maior_valor()` no IPython:

```
In [15]: s = "GAATTGCTAGC"
```

```
In [16]: d = init_dicionario(s,2)
```

```
In [17]: d
```

```
Out[17]:
```

```
{'GA': 1,  
 'AA': 1,  
 'AT': 1,  
 'TT': 1,  
 'TG': 1,  
 'GC': 1,  
 'CT': 1,  
 'TA': 1,  
 'AG': 1}
```

```
In [18]: maior_valor(d)
```

```
Out[18]: (1, ['GA', 'AA', 'AT', 'TT', 'TG', 'GC', 'CT', 'TA',
```

```
In [19]: d = init_dicionario(s,3)
```

```
In [20]: d
```

```
Out[20]:
```

```
{'GAA': 1,  
 'AAT': 1,  
 'ATT': 1,  
 'TTG': 1,  
 'TGC': 1,  
 'GCT': 1,  
 'CTA': 1,  
 'TAG': 1}
```

```
In [21]: d = init_dicionario(s,1)
```

```
In [22]: d
```

```
Out[22]: {'G': 3, 'A': 3, 'T': 3, 'C': 1}
```

```
In [23]: maior_valor(d)
```

```
Out[23]: (3, ['G', 'A', 'T'])
```

```
In [24]: s = "31415926"
```

```
In [25]: d = init_dicionario(s,1)
```

```
In [26]: d
```

```
Out[26]: {'3': 1, '1': 2, '4': 1, '5': 1, '9': 1, '2': 1}
```

```
In [27]: maior_valor(d)
```

```
Out[27]: (2, ['1'])
```

## programa: exemplos

Depois de escrever e testar a função `maior_valor()` execute o programa para ver

```
In [1]: ?
```

```
Out[1]: Digite:
```

- uma chave no dicionário para saber o valor associado
- sair : sair do programa
- leia : lê o conteúdo de um arquivo
- crie : cria um dicionário com fatias da string lida
- mostre : exhibe os itens no dicionário
- len : tamanho do dicionário
- limpe : limpa o dicionário
- max : lista de chaves de maior valor
- ? : exhibe este menu

```
In [2]: leia
```

```
Digite o nome do arquivo com a string: virus-30.txt
```

```
Out[2]: arquivo lido
```

```
In [3]: crie
```

```
Digite o tamanho da fatia: 2
```

```
Out[3]: dicionário criado
```

```
In [4]: mostre
```

```
Out[4]: dict_items([('GA', 1), ('AA', 3), ('AT', 3), ('TT', 3)])
```

```
In [5]: max
```

```
Out[5]: 5: ['GC']
```

```
In [6]: limpe
```

```
Out[6]: dicionario limpo
```

```
In [7]: crie
```

```
Digite o tamanho da fatia: 3
```

```
Out[7]: dicionário criado
```

```
In [8]: maior
```

```
Out[8]: comando desconhecido
```

```
In [9]: max
```

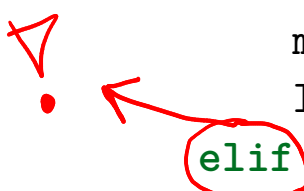
```
Out[9]: 3: ['AAT', 'ATT', 'TTG', 'TGC', 'GCT', 'CTA', 'TAG']
```

```
In [10]: mostre
```

```
Out[10]: dict_items([('GAA', 1), ('AAT', 3), ('ATT', 3), ('TTG', 3)])
```

## Solução

```
def maior_valor(dicio):  
    '''(dict) -> int, list  
    RECEBE um dicionario `dicio`.  
    RETORNA o maior valor de uma chave e uma lista com  
        todas as chaves de maior valor  
    '''  
    maior = 0  
    lst_chaves = []  
    for chave in dicio:  
        if dicio[chave] > maior:  
            maior = dicio[chave]  
            lst_chaves = [chave]  
            elif dicio[chave] == maior:  
                lst_chaves += [chave]  
    return maior, lst_chaves
```



## 24.5 Exercício: shell2 de luxe

Agora vocês estenderão ainda mais o programa escrevendo a função `csv_para_dict()` que lê o nome de um arquivo `csv` com a representação de um dicionário em que a **chave** é uma string e o **valor** é um número inteiro; o 1o. campo é a chave e o 2o. é o valor associado a chave.

Baixem o arquivo `shell2_de_luxe.py` para a mesma pasta que estão os arquivos `pi-30.txt` e `virus-30.txt`. Baixe ainda daqui os arquivos `pi-30.csv` e `virus-30.csv`.



### `csv_para_dict()`: exemplos

A seguir estão exemplos de execução da função `csv_para_dict()` no IPython:

```
In [30]: d = csv_para_dict()
Digite o nome do arquivo csv: pi-30.csv
```

```
In [31]: d
```

```
Out[31]:
```

```
{'314': 1,
 '141': 1,
 '415': 1,
 '159': 1,
 '592': 1,
 '926': 1,
 '265': 1,
 '653': 1,
 '535': 1,
 '358': 1,
 '589': 1,
 '897': 1,
 '979': 1,
```

```
'793': 1,  
'932': 1,  
'323': 1,  
'238': 1,  
'384': 1,  
'846': 1,  
'462': 1,  
'626': 1,  
'264': 1,  
'643': 1,  
'433': 1,  
'338': 1,  
'383': 1,  
'832': 1,  
'327': 1}
```

In [32]: d = csv\_para\_dict()

Digite o nome do arquivo csv: virus-30.csv

In [33]: d

Out[33]:

```
{'GAA': 1,  
'AAT': 3,  
'ATT': 3,  
'TTG': 3,  
'TGC': 3,  
'GCT': 3,  
'CTA': 3,  
'TAG': 3,  
'AGC': 2,  
'GCA': 2,  
'CAA': 2}
```



## programa: exemplos

Depois de escrever e testar a função `csv_to_dict()` execute o programa para ver

```
In [1]: ?
```

```
Out[1]: Digite:
```

- uma chave no dicionário para saber o valor associado
- sair : sair do programa
- leia : lê o conteúdo de um arquivo
- crie : cria um dicionário com fatias da string lida
- mostre : exhibe os itens no dicionário
- len : tamanho do dicionário
- limpe : limpa o dicionário
- max : lista de chaves de maior valor
- grave : grave arquivo csv com conteúdo do dicionário
- carregue : cria um dicionário com o conteúdo de um arquivo
- ? : exhibe este menu

```
In [2]: carregue
```

```
Digite o nome do arquivo csv: virus-30.csv
```

```
Out[2]: dicionário criado
```

```
In [3]: mostre
```

```
Out[3]: dict_items([('GAA', 1), ('AAT', 3), ('ATT', 3), ('TTG', 3)])
```

```
In [4]: max
```

```
Out[4]: 3: ['AAT', 'ATT', 'TTG', 'TGC', 'GCT', 'CTA', 'TAG']
```

```
In [5]: limpe
```

```
Out[5]: dicionario limpo
```

```
In [6]: mostre
```

```
Out[6]: dict_items([])
```

```
In [7]: carregue
```

```
Digite o nome do arquivo csv: pi-30.csv
```

```
Out[7]: dicionário criado
```

```
In [8]: mostre
```

```
Out[8]: dict_items([('314', 1), ('141', 1), ('415', 1), ('159', 1)])
```

## Solução

```
def csv_para_dict():  
    '''() -> dict  
    LÊ o nome de um arquivo csv com a representação de um dicionário  
    em que a chave é uma string e o valor é um número inteiro  
    o 1o. campo é a chave e o 2o. é o valor associado a chave.  
    RETORNA o dicionário contido no arquivo.  
    '''  
  
    dicio = {}  
    # 1 pegue o nome do arquivo  
    nome = input("Digite o nome do arquivo csv: ")  
    # 2 abra o arquivo para leitura 'r' = read  
    arq = open(nome, 'r', encoding='utf-8')  
    # 3 leia o conteúdo do arquivo  
    # print("csv_para_dict(): lendo arquivo e criando dicionário")  
    for linha in arq:  
        lst = linha.split(',')  
        if len(lst) == 2:  
            chave = lst[0].strip()  
            valor = int(lst[1])  
            dicio[chave] = valor  
    # 4 feche o arquivo  
    arq.close()  
    # print("csv_para_dict(): arquivo lido e dicionário criado.")  
    return dicio
```

## 24.6 Programa completo

```
#-----  
SAIR      = 'sair'  
LEIA      = 'leia'  
CRIE      = 'crie'  
MOSTRE    = 'mostre'  
LIMPE     = 'limpe'  
LEN       = 'len'  
MAX       = 'max'  
CARREGUE  = 'carregue'  
GRAVE     = 'grave'  
AJUDE     = '?'  
  
def main():  
    # string con conteudo de um arquivo  
    s = ''  
    # dicionário vazio  
    dicio_fatias = {}  
    i = 1  
    cmd = input(f"In [{i}]: ").strip()  
    while cmd != SAIR:  
        if cmd in dicio_fatias: # valor da chave cmd  
            resp = f"{dicio_fatias[cmd]}"  
        elif cmd == LEIA:  
            s = leia_arquivo()  
            resp = 'arquivo lido'  
        elif cmd == CRIE:  
            if s == '':  
                resp = 'string vazia, arquivo precisa ser lido'  
            else:  
                k = int(input("Digite o tamanho da fatia: "))  
                dicio_fatias = init_dicionario(s, k)
```

```

        resp = 'dicionário criado'
elif cmd == MOSTRE:
    resp = f"{dicio_fatias.items()}"
elif cmd == LIMPE:
    dicio_fatias = {}
    resp = 'dicionario limpo'
elif cmd == LEN:
    resp = f"{len(dicio_fatias)}"
elif cmd == MAX:
    if len(dicio_fatias) == 0:
        resp = 'dicionário está vazio'
    else:
        valor, lst_chaves = maior_valor(dicio_fatias)
        resp = f"{valor}: {lst_chaves}"
elif cmd == CARREGUE:
    dicio_fatias = csv_para_dict()
    resp = 'dicionário criado'
elif cmd == GRAVE:
    if len(dicio_fatias) == 0:
        resp = 'dicionário está vazio'
    else:
        dict_para_csv(dicio_fatias)
        resp = 'dicionário gravado'
elif cmd == AJUDE:
    resp = help()
else:
    resp = "comando desconhecido"

print(f"Out [{i}]: {resp}")
i += 1
cmd = input(f"In [{i}]: ").strip()

```

#-----

```

def maior_valor(dicio):
    '''(dict) -> int, list
    RECEBE um dicionario `dicio`.
    RETORNA o maior valor de uma chave e uma lista com
        todas as chaves de maior valor
    '''
    maior = 0
    lst_chaves = []
    for chave in dicio:
        if dicio[chave] > maior:
            maior = dicio[chave]
            lst_chaves = [chave]
        elif dicio[chave] == maior:
            lst_chaves += [chave]
    return maior, lst_chaves

```

#-----

```

def init_dicionario(s, k=1):
    '''(str, int) -> dict
    RECEBE um string s e um inteiro `k`.
    RETORNA um dicionário em que:
        - as chaves as fatias de `s` com `k` caracteres
        - os valores são o número de ocorrências das respectivos
    '''
    # crie o dicionário
    d = {}
    # percorra todos os subnúmeros de pi de tamanho k
    n = len(s)
    for i in range(k,n):
        # pegue a próxima fatia
        fatia = s[i-k:i]
        # verifique se a fatia está no dicionário
        if fatia in d:

```

```
        d[fatia] += 1
    else:
        d[fatia] = 1
return d
```

#-----

```
def leia_arquivo():
    '''() -> str
    Lê o nome de um arquivo.
    RETORNA uma string com todo o conteúdo do arquivo
    '''
    # 1 pegue o nome do arquivo
    nome = input("Digite o nome do arquivo com a string: ")
    # 2 abra o arquivo para leitura 'r' = read
    arq = open(nome, 'r', encoding='utf-8')
    # 3 leia o conteúdo do arquivo
    s = arq.read()
    # 4 feche o arquivo
    arq.close()
    # print(f"arquivo '{nome}' lido.")
    return s
```

#-----

```
def csv_para_dict():
    '''() -> dict
    Lê o nome de um arquivo csv com a representação de um dicionário
    em que a chave é uma string e o valor é um número inteiro
    o 1o. campo é a chave e o 2o. é o valor associado a chave.
    RETORNA o dicionário contido no arquivo.
    '''
    dicio = {}
    # 1 pegue o nome do arquivo
    nome = input("Digite o nome do arquivo csv: ")
```

```

# 2 abra o arquivo para leitura 'r' = read
arq = open(nome, 'r', encoding='utf-8')
# 3 leia o conteúdo do arquivo
# print("csv_para_dict(): lendo arquivo e criando dicionário")
for linha in arq:
    lst = linha.split(',')
    if len(lst) == 2:
        chave = lst[0].strip()
        valor = int(lst[1])
        dicio[chave] = valor
# 4 feche o arquivo
arq.close()
# print("csv_para_dict(): arquivo lido e dicionário criado.")
return dicio

```

#-----

```

def dict_para_csv(dicio):
    '''(dict) -> None
    RECEBE um dicionario dicio.
    GRAVA o conteúdo do dicionário em arquivo csv em que
        cada linha possui dois campos:
        - chave do dicionário
        - valor da chave
    '''
    # 1 pegue o nome do arquivo
    nome = input("Digite o nome do arquivo: ")
    # 2 abra o arquivo para escrita 'w' = write
    arq = open(nome, 'w', encoding='utf-8')
    # 3 percorra o dicionário gravando cada para chave:valor em
    # print("dict_para_csv(): dicionário sendo gravado...")
    for chave in dicio:
        arq.write(f"{chave},{dicio[chave]}\n") # notar o "\n"
    # 4 feche o arquivo

```



```

    arq.close()
    # print(f"dict_para_csv(): arquivo '{nome}' criado...")

#-----
def help():
    '''(None) -> str

    RETORNA string com mensagem de ajuda para usar o programa.
    '''
    s = "Digite:\n" +\
        "    - uma chave no dicionário para saber o valor associa\n"
    f"    - {SAIR:8}: sair do programa\n" +\
    f"    - {LEIA:8}: lê o conteúdo de um arquivo\n" +\
    f"    - {CRIE:8}: cria um dicionário com fatias da string l\n"
    f"    - {MOSTRE:8}: exibe os itens no dicionário\n" +\
    f"    - {LEN:8}: tamanho do dicionário\n" +\
    f"    - {LIMPE:8}: limpa o dicionário\n"+\
    f"    - {MAX:8}: lista de chaves de maior valor\n" +\
    f"    - {GRAVE:8}: grave arquivo csv com conteúdo do dicion\n"
    f"    - {CARREGUE:8} : cria um dicionário com o conteúdo de\n"
    f"    - {AJUDE:8} : exibe este menu"
    return s

#-----
if __name__ == "__main__":
    main()

```