22 Reunião 22: 06/JUL/2021

Figure 1: Fonte: Dilbert por Scott Adams

22.1 Reuniões passadas

- listas são **mutáveis**
- strings são **imutáveis**
- `strip()`: `s.strip()` retorna uma string como `s` sem brancos no inicio e no final
- `split()`: `s.split()` retorna uma lista com as strings de `s` separadas por `" \n\t\r"`
- operador `in`

`s.strip()` e `s.split()`

```
In [5]: s = "   Como é  bom estudar MAC0110!   "

In [6]: s.strip()
Out[6]: 'Como é  bom estudar MAC0110!'

In [7]: s.split()
Out[7]: ['Como', 'é', 'bom', 'estudar', 'MAC0110!']
```

i in range()

```
print("i in range(n)")
# True se i é um valor entre 0 e n-1, False em caso contrári
i = 2
n = 5
print(f"{i} in range({n}) é {i in range(n)}")

# alternativa
if i in range(n):
    print(f"{i} é um número inteiro entre 0 e {n}")
else:
    print(f"{i} não é um número inteiro entre 0 e {n}")

# percorre/itera sobre todos os inteiros entre 0 e n-1
print(f"range({n})=", end=" ")
for i in range(n):
    print(f"{i}", end=" ")
print("\n----------------") # para mudar de linha
```

item in list

```
print("item in lst")
# True  se item é um elemento da lista lst, False em caso co
item = None
lst  = [True, 3.14, None, 'oi', 5, [2, 3]]
print(f"{item} in {lst} é {item in lst}")

# alternativa
if item in lst:
    print(f"{item} é elemento de {lst}")
else:
    print(f"{item} não é elemento de {lst}")
```

```
    # percorre/itera sobre todos os itens de lst
    print(f"lst=", end=" ")
    for item in lst:
        print(f"{item}", end=" ")
    print("\n--------------------") # para mudar de linha
```

t in str

```
    print("t in s")
    # True se t é substring da string s, False em caso contrário
    s = "MAC0110 é da hora!"
    t = "hora"
    print(f"'{t}' in '{s}' é {t in s}")

    # alternativa
    if t in s:
        print(f"'{t}' é substring de '{s}'")
    else:
        print(f"'{t}' não é substring de '{s}'")

    # percorre/itera sobre todos os caracteres
    print(f"s='", end="")
    for c in s:
        print(f"{c}", end="")
    print("'")
```

22.2 Hoje


Dicionários
Pares (chave, valor)
Busca
inserção
atualização
Dicionários em Python
dicio vazio: d = {}
busca: chave in d
inserir: d[chave] = valor
alterar: d[chave] = valor
Type dict
dicionários são
mutáveis

22.3 Exercício: posição de um item

Escreva um função de nome `indice()` que **recebe** uma elemento/coisa/objeto de nome `item` e uma lista `lst` e **retorna** o menor índice de um posição de `lst` que é igual a `item` Se `item` não ocorre em `lst` a função deve **retornar** `None`.

Exemplos

```
In [3]: palavras = ['Fracassei', 'em', 'tudo', 'o', 'que',\
                    'tentei', 'na', 'vida']

In [4]: indice('em',palavras)
Out[4]: 1

In [5]: indice('x',palavras)
In [6]: i = indice('x',palavras)
In [7]: i == None
Out[7]: True

In [8]: i = indice('tudo', palavras)
In [9]: i
Out[9]: 2

In [10]: lst = [23, 34, 'oi', True, None, 3.14]
In [11]: indice(None, lst)
Out[11]: 4

In [12]: indice(23, lst)
Out[12]: 0

In [13]: indice(3.14, lst)
Out[13]: 5
```

Solução

```
def main():
    '''Unidade para testes'''
    # altere, teste, experimente
    palavras = ['Fracassei', 'em', 'tudo', 'o', 'que',\
       'tentei', 'na', 'vida']
    pal = 'que'
    i = indice(pal, palavras)
    if i == None:
        print(f"'{pal}' não ocorre na lista")
    else:
        print(f"'{pal}' ocorre na posição {i}")

#-----------------------------------------------------------------
def indice(item, lst):
    '''(list, obj) -> int ou None

    RECEBE um valor item e uma lista lst.
    RETORNA o índice da primeira posição de lst
        que é igual ao item. Se item não está em
        lst a função RETORNA None.
    '''
    n = len(lst)
    for i in range(n):
        if lst[i] == item:
            return i
    return None
#-----------------------------------------------------------------
if __name__ == "__main__":
    main()
```

22.4 Exercício: frequências de palavras

Neste exercício por **palavra** entenda-se um string contendo apenas letras como, por exemplo, `'abcdd'`, `'palavra'` ou `'aaa'`.

O **resultado final** deste exercício será um programa lê o conteúdo de um arquivo que contém palavras separadas pelos caracteres `' '` e `'\n'` e **determina** a frequência de cada palavra no arquivo. Para isso vocês deverão seguir os dois passos a seguir. O verdadeiro trabalho e a parte importante é o primeiro passo, fazer a função `init_dicio_palavras()`

Passo 1: `init_dicio_palavras()`

Escreva uma função de nome `init_dicio_palavras()` que recebe como parâmetro uma string `txt` contendo palavras separadas pelos caracteres espaço `' '` ou `'\n'` e retorna duas listas:

- uma lista de strings com todas as palavras que ocorrem no texto, **sem repeticões**; e
- uma lista de inteiros com a respectiva frequência de cada palavra no texto.

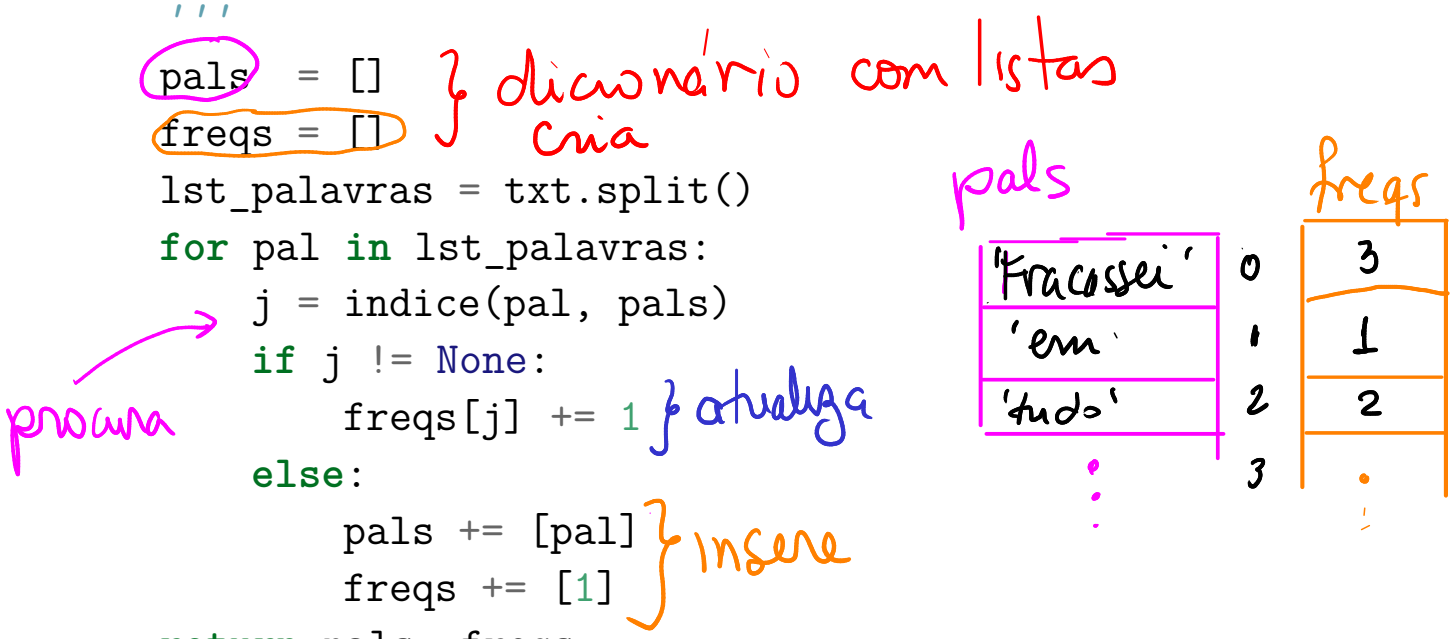
Exemplos

```
In [15]: txt=" aa bb\n c aa dd bb ee cc bb aa "

In [16]: init_dicio_palavras(txt)
Out[16]: (['aa', 'bb', 'c', 'dd', 'ee', 'cc'], [3, 3, 1, 1, 1

In [17]: pals, freq = init_dicio_palavras(txt)

In [18]: pals[0]
Out[18]: 'aa'

In [19]: freq[0]
```

```
Out[19]: 3

In [20]: pals[1]
Out[20]: 'bb'

In [21]: freq[1]
Out[21]: 3

In [22]: pals[3]
Out[22]: 'dd'

In [23]: freq[3]
Out[23]: 1

In [24]: txt = 'a b a b c x x y x z'

In [25]: pals, freq = init_dicio_palavras(txt)

In [26]: pals
Out[26]: ['a', 'b', 'c', 'x', 'y', 'z']

In [27]: freq
Out[27]: [2, 2, 1, 3, 1, 1]
```

Solução

```
def init_dicio_palavras(txt):
    '''(str) -> list, list

    RECEBE uma string `txt` com palavras separadas por caractere
         espaço e ``\n``.
    RETORNA duas listas de mesmo comprimento:

       * uma lista de strings com todas as palavras que ocorrem
         no texto, __sem repetições__;
       * uma lista de inteiros com a respectiva frequência de co
         palavra no texto.
    '''
    pals  = []
    freqs = []
    lst_palavras = txt.split()
    for pal in lst_palavras:
        j = indice(pal, pals)
        if j != None:
            freqs[j] += 1
        else:
            pals += [pal]
            freqs += [1]
    return pals, freqs
```



Passo 2: leitura de arquivo

Baixe daqui o arquivo `darci.txt` para a mesma pasta que está o seu programa. No seu programa troque a linha

```
LEIA_ARQUIVO = False
```

por

```
LEIA_ARQUIVO = True
```

Agora execute a função `main()` e forneça ao programa o (nome do) arquivo `darci.txt`.

Depois disso você pode baixar mais arquivos daqui.

Exemplo

```
Digite o nome do arquivo: darci.txt
Conteudo do arquivo:
Fracassei em tudo o que tentei na vida
Tentei alfabetizar as crianças brasileiras  não consegui
Tentei salvar os índios  não consegui
Tentei fazer uma universidade séria e fracassei
Tentei fazer o Brasil desenvolver se autonomamente
e fracassei
Mas os fracassos são minhas vitórias
Eu detestaria estar no lugar de quem me venceu
Darci Ribeiro

Lista de palavras:
0: 'Fracassei' : 1
1: 'em' : 1
2: 'tudo' : 1
3: 'o' : 2
4: 'que' : 1
```

5: 'tentei' : 1
6: 'na' : 1
7: 'vida' : 1
8: 'Tentei' : 4
9: 'alfabetizar' : 1
10: 'as' : 1
11: 'crianças' : 1
12: 'brasileiras' : 1
13: 'não' : 2
14: 'consegui' : 2
15: 'salvar' : 1
16: 'os' : 2
17: 'índios' : 1
18: 'fazer' : 2
19: 'uma' : 1
20: 'universidade' : 1
21: 'séria' : 1
22: 'e' : 2
23: 'fracassei' : 2
24: 'Brasil' : 1
25: 'desenvolver' : 1
26: 'se' : 1
27: 'autonomamente' : 1
28: 'Mas' : 1
29: 'fracassos' : 1
30: 'são' : 1
31: 'minhas' : 1
32: 'vitórias' : 1
33: 'Eu' : 1
34: 'detestaria' : 1
35: 'estar' : 1
36: 'no' : 1
37: 'lugar' : 1

```
    38: 'de' : 1
    39: 'quem' : 1
    40: 'me' : 1
    41: 'venceu' : 1
    42: 'Darci' : 1
    43: 'Ribeiro' : 1
```

Solução

```
'''
   Programa que lê um arquivo em que as palavras
   estão separadas por espaços e '\n' e conta a
   frequência de cada palavra.
'''
MOSTRE       = True
LEIA_ARQUIVO = False

def main():
    if LEIA_ARQUIVO:
        txt = leia_arquivo()
    else:
        txt = input("Digite palavras separadas por espaços: ")

    # crie as listas de palavras e frequências
    lst_pals, lst_freqs = init_dicio_palavras(txt)

    # mostre lista de palavras e frequências
    print(f"Lista de palavras:")
    for i in range(len(lst_pals)):
        print(f"{i}: '{lst_pals[i]}' : {lst_freqs[i]}")

#-----------------------------------------------------------
def init_dicio_palavras(txt):
```

```
    '''(str) -> list, list

    RECEBE uma string `txt` com palavras separadas por caractere
         espaço e ``'\n'``.
    RETORNA duas listas de mesmo comprimento:

       * uma lista de strings com todas as palavras que ocorrem
         no texto, __sem repeticões__;
       * uma lista de inteiros com a respectiva frequência de co
         palavra no texto.
    '''
    pals  = []
    freqs = []
    lst_palavras = txt.split()
    for pal in lst_palavras:
        j = indice(pal, pals)
        if j != None:
            freqs[j] += 1
        else:
            pals += [pal]
            freqs += [1]
    return pals, freqs

#-----------------------------------------------------------------
def indice(item, lst):
    '''(list, obj) -> int ou None

    RECEBE um valor item e uma lista lst.
    RETORNA o índice da primeira posição de lst
        que é igual ao item. Se item não está em
        lst a função RETORNA None.
    '''
    n = len(lst)
```

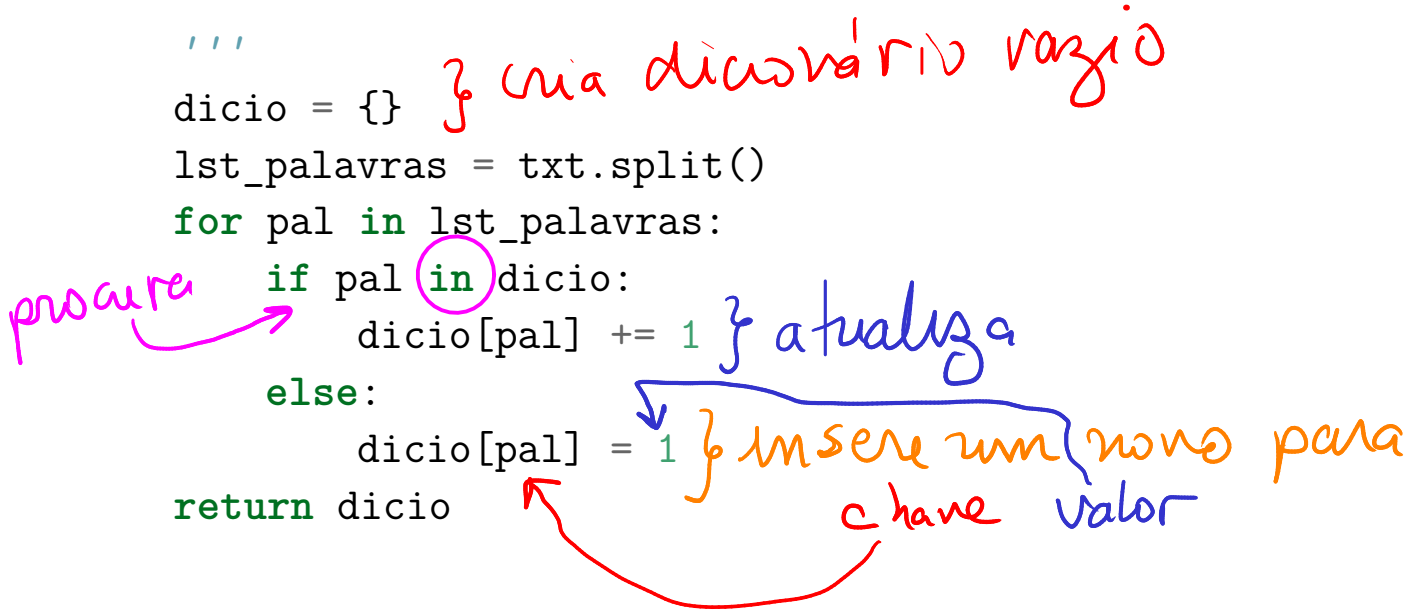
```
    for i in range(n):
        if lst[i] == item:
            return i
    return None

#-------------------------------------------------------------
def leia_arquivo():
    '''() -> str
    LÊ o nome de um arquivo e
    RETORNA uma string com o conteúdo do arquivo.
    '''
    # leitura do texto
    # 1 pegue o nome do arquivo
    nome_in = input("Digite o nome do arquivo: ")

    # 2 abra o arquivo para leitura: 'r'= read
    arq_in = open(nome_in, "r", encoding="utf-8")

    # 3 leia o conteudo do arquivo
    txt = arq_in.read()

    # 4 feche o arquivo
    arq_in.close()

    # exiba o conteúdo do arquivo
    if MOSTRE:
        print(f"Conteudo do arquivo '{nome_in}'")
        print(f"{txt}")

    # retorne a string com o conteúdo do arquivo
    return txt

#-------------------------------------------------------------
```

```python
def crie_arquivo(campo1, campo2):
    '''(str, list, list) -> None
    RECEBE uma duas listas campo1 e campo2.
    CRIA um arquivo csv tendo em cada linha campo1[i] e campo2[i
    '''
    # criação de um arquivo
    # 1 pegue o nome do arquivo
    nome_out = input("Digite o nome do arquivo: ")

    # 2 abra o arquivo para escrita: 'w' = write
    arq_out = open(nome_out, "w", encoding="utf-8" ) # write

    # 3 escreva no arquivo: write()
    for i in range(len(campo1)):
        # '\n' é necessário para mudar de linha
        arq_out.write(f"{campo1[i]},{campo2[i]}\n"  )

    # 4 feche o arquivo
    arq_out.close()

    # avisa que arquivo foi criado
    print(f"arquivo '{nome_out}' foi criado")


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

22.5 Frequência de palavras com dicionários

```
'''
   Programa que lê um arquivo em que as palavras
   estão separadas por espaços e '\n' e conta a
   frequência de cada palavra.
'''
MOSTRE       = True
LEIA_ARQUIVO = True

def main():
    if LEIA_ARQUIVO:
        txt = leia_arquivo()
    else:
        txt = input("Digite palavras separadas por espaços: ")

    # crie o dicicionário de palavras x frequencias
    dicio_palavras = init_dicio_palavras(txt)

    # mostre lista de palavras e frequências
    print(f"Lista de palavras:")
    i = 0
    for pal in dicio_palavras:
        print(f"{i}: '{pal}' : {dicio_palavras[pal]}")
        i += 1

#-----------------------------------------------------------------
def init_dicio_palavras(txt):
    '''(str) -> dict

    RECEBE uma string txt.
    RETORNA um dicionário em que a CHAVE e cada palavra em txt
       e o VALOR é a sua frequência em txt.
```

```
    '''
    dicio = {}
    lst_palavras = txt.split()
    for pal in lst_palavras:
        if pal in dicio:
            dicio[pal] += 1
        else:
            dicio[pal] = 1
    return dicio
```




```
#-----------------------------------------------------------------
def leia_arquivo():
    '''() -> str
    LÊ o nome de um arquivo e
    RETORNA uma string com o conteúdo do arquivo.
    '''
    # leitura do texto
    # 1 pegue o nome do arquivo
    nome_in = input("Digite o nome do arquivo: ")

    # 2 abra o arquivo para leitura: 'r'= read
    arq_in = open(nome_in, "r", encoding="utf-8")

    # 3 leia o conteudo do arquivo
    txt = arq_in.read()

    # 4 feche o arquivo
    arq_in.close()

    # exiba o conteúdo do arquivo
    if MOSTRE:
        print(f"Conteudo do arquivo '{nome_in}'")
```

```
        print(f"{txt}")

    # retorne a string com o conteúdo do arquivo
    return txt

#-----------------------------------------------------------------
def crie_arquivo(campo1, campo2):
    '''(str, list, list) -> None
    RECEBE uma duas listas campo1 e campo2.
    CRIA um arquivo csv tendo em cada linha campo1[i] e campo2[i
    '''
    # criação de um arquivo
    # 1 pegue o nome do arquivo
    nome_out = input("Digite o nome do arquivo: ")

    # 2 abra o arquivo para escrita: 'w' = write
    arq_out = open(nome_out, "w", encoding="utf-8" ) # write

    # 3 escreva no arquivo: write()
    for i in range(len(campo1)):
        # '\n' é necessário para mudar de linha
        arq_out.write(f"{campo1[i]},{campo2[i]}\n"  )

    # 4 feche o arquivo
    arq_out.close()

    # avisa que arquivo foi criado
    print(f"arquivo '{nome_out}' foi criado")


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

22.6 Dicionários

Figure 2: Que fofo!

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

Um dicionário está sujeito a dois tipos de operações:

- *inserção*: consiste em *introduzir* um objeto na tabela
- *busca*: consiste em *encontrar* um elemento que tenha uma dada chave.

Dicionário em Python

Python possui dicionários como um tipo nativo.

As **chaves** podem ser números inteiros ou strings ou outros tipos de dados **imutáveis**.

Uma maneira de criar um dicionário é começar com o dicionário vazio e adicionar pares chave-valor. O dicionário vazio é denotado por `{}`

```
In [1]: d = {}  # dicionário vazio
```

Para inserirmos um par `chave-valor` no dicionário ou alterar o valor associado a uma chave, fazemos simplemente

```
d[chave] = valor
```

Exemplos de uso

```
In [3]: eng2port = {}  # dicionário vazio
In [4]: eng2port['one'] = 'um'
In [5]: eng2port['two'] = 'dois'
In [6]: eng2port['three'] = 'treiss'
In [8]: eng2port
Out[8]: {'one': 'um', 'two': 'dois', 'three': 'treiss'}
In [9]: eng2port['um']
---------------------------------------------------------------
KeyError                                  Traceback (most rec
<ipython-input-9-69ed764ca20e> in <module>
----> 1 eng2port['um']

KeyError: 'um'

In [10]: eng2port['one']
Out[10]: 'um'
In [11]: eng2port['three']
Out[11]: 'treiss'
In [12]: eng2port['three'] = 'três' # são mutáveis
In [13]: eng2port
Out[13]: {'one': 'um', 'two': 'dois', 'three': 'três'}
```

Percorrer dicionários

Para **percorrermos todas as chaves** de um dicionário usamos:

```
In [2]: d = {False: 0, True: 1, 'Bom': 'dia', 'pi': 3.14, 'e'
In [3]: for chave in d:
   ...:     print(f"{chave}: {d[chave]}")
   ...:
False: 0
True: 1
Bom: dia
pi: 3.14
e: 2.71
```

Métodos de dicionários

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

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

22.7 Arquivos

O trecho de código a seguir **lê** o conteúdo de um arquivo e apelida esse conteúdo de `txt`.

```
# 1. pegue nome do arquivo
nome = input("Digite o nome do arquivo: ")
# 2. abra o aquivo para leitura: 'r' de read
arq = open(nome, "r", encoding="utf-8")
# 3. leia seu conteúdo
txt = arq_entrada.read()
# 4. feche o arquivo
arq.close()
print("Conteudo do arquivo: ")
print(f"{txt}")
```

O próximo trecho de código **cria** um arquivo de nome `exemplo_um.txt` e grava nele uma lista de números, um número por linha:

```
numeros = [123, 456, 7890]
# 1. pegue nome do arquivo
nome = "exemplo_um.txt"
# 2. abra o aquivo para escrita: 'w' de write
arq_saida = open(nome, 'w', encoding='utf-8')
# 3. escreva no arquivo
for num in numeros:
    arq_saida.write(f"{num}\n")
    print(f"gravei {num} no arquivo")
# 4. feche o arquivo
arq_saida.close()
```

O código abaixo **lê** do arquivo que foi criado pelo trecho anterior e exibe seus valores:

```
# 1. pegue nome do arquivo
nome = "exemplo_um.txt"
```

```
# 2. abra o arquivo para leitura: 'r' de read
arquivo_entrada = open(nome, 'r', encoding='utf-8')
# 3. leia todo o conteúdo do arquivo
txt = arquivo_entrada.read()
# 4. feche o arquivo
arquivo_entrada.close()

print("Conteúdo do arquivo:")
print(f"{txt}")

print("\nCaracteres do arquivo: \n")
for c in txt:
    print(f"'{c}'")
```

A função `open()` utilizada nestes trechos abre um arquivo para para **leitura** (indicado por `r` – do inglês *read*) ou **escrita** (indicado por `w` – do inglês *write*).

Para saber mais sobre arquivos em Python, consulte o capítulo Trabalhando com Arquivos do livro Como Pensar Como um Cientista da Computação.

Podemos considerar que os elementos básicos em um arquivo texto são caracteres, ou seja, strings de comprimento um. O argumento `encoding='utf-8'` utilizados como argumento da função `open()` indica o uso da codificação conhecida por `utf-8`. Essa codificação permite a representação de caracteres acentuados e ideogramas utilizados em diferentes línguas.

Copie e execute esses trechos de código para ver o resultado. Um problema do primeiro trecho de código é que nele os caracteres que representam os números são gravados todos juntos, sem o uso de um caractere de separação, como espaço ou vírgula. Isso pode ser notado pelo resultado do segundo trecho de código.

Uma solução simples para corrigir esse problema seria incluir um espaço após gravar com `write()` cada número da lista, como mostrado abaixo:

```
numeros = [123, 456, 7890]
# 1. nome do arquivo
```

```
nome = "exemplo_um.txt"
# 2. abra o arquivo
arquivo_saida = open(nome, 'w', encoding='utf-8')
# 3. grave os números no arquivo
for num in numeros:
    arquivo_saida.write(f"{num} ")
    print(f"gravei o {num}")
```

Agora que os números foram "separados", como agrupar os caracteres do texto para que o texto possa ser reconvertido para números?