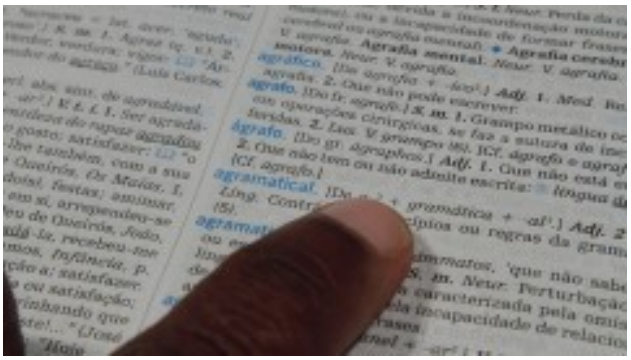
23 Reunião 23: 08/JUL/2021

Figure 1: Fonte: Calvi e Hobbes por Bill Watterson

23.1 Reuniões passadas

- na última terça-feira começamos a andar entre **dicionários** (`dict`)



- operador `in`: `in range()`, `in str`, `in list` e `in dict`

`... in ...`	`... not in ...`	`if ... in ...:`	`for ... in ...`
`i in range(10)`	`i not in range(10)`	`if i in range(10):`	`for i in range(10):`
`c in s`	`c not in s`	`if c in s:`	`for c in s:`
`item in lst`	`item not in lst`	`if item in lst:`	`for item in lst:`
`chave in dicio`	`chave not in dicio`	`if chave in dicio:`	`for chave in dicio:`

- `strip()`: `s.strip()` retorna uma string com `s` sem brancos no inicio e no final
- `split()`: `s.split()` retorna uma lista de strings
- mutabilidade: strings são **imutáveis**, listas e dicionários são **mutáveis**
- tipos nativos `str`, `int`, `float`, `bool`, `NoneType`, `list`, `dict`

23.2 Hoje


Hoje
• criar dicionários
• pertinência em dicionários
chave in dicio # in
• percorrer dicionários
for chave in dicio:
• funções
len(dicio)
dicio.keys()
dicio.get(chave)
• Arquivos
(a) abrir: open()
(b) ler: for linha in arq:
(c) fechar: arq.close()

23.3 Exercício: shell

Neste exercício vocês escreverão um programa que lê um arquivo `.csv` e responde consultas interativas, como um *shell*. No próximo exercício vocês estenderão as funcionalidades deste. A seguir está um roteiro de como proceder.

Baixe os arquivos

Baixem o arquivo `shell.py` e baixem daqui o arquivo `teste.csv` para a mesma pasta em que baixou o programa `shell.py`. Mais tarde vocês podem baixar outros arquivos.

O arquivo `shell.py` tem três funções:

- `main()`: reponsável pela interação. **Está completa.**
- `init_dicio()`: que vocês **deverão escrever.**
- `help()`: usada pela `main()` para apresentar o menu de uso do programa sempre que pedido ou necessário. **Está completa.**

Cada linha dos arquivos `.csv` fornecidos contém **dois campos** separados por uma vírgula. O primeiro campo contém uma **palavra** (`str` sem espaços) e o segundo campo um **número inteiro** que vocês podem imaginar que é a **frequência** da palavra em algum texto. Por exemplo, o conteúdo do arquivo `teste.csv` é o seguinte

```
In [5]: more teste.csv
como , 10
é  ,  20
bom , 30
estudar, 45
MAC0110! , 1000
```

aqui tem um '\n'

espaços

Executem o programa

Executem o programa **como está**. Quando for solicitado forneçam o nome `teste.cvs` do arquivo . A saída que vocês verão será a seguinte.

```
Digite o nome do arquivo: teste.csv
Conteúdo do arquivo 'teste.csv':
linha 0: como , 10
linha 1: é  ,  20
linha 2: bom , 30
linha 3: estudar, 45
linha 4: MAC0110! , 1000

Vixe! O dicionario está vazia... Tchau!
```

Função `init_dicio()`

Aqui começa o trabalho.

Vocês deverão escrever a função `init_dicio()` que **lê o nome de um arquivo** `.csv` com o formato indicado anteriormente e depois lê o conteúdo do arquivo, **uma linha por vez**. A função deve **retornar** um dicionário em que as **chaves** são às palavras e em que os **valores** são as respectivas frequências

A parte da leitura do arquivo linha a linha já está feita e tem 4 passos. Para criar o dicionário a função deverá processar o conteúdo da string `linha`.

Exemplos

Depois de feita a função o programa de vocês terá o seguinte comportamento para o arquivo `teste.cvs`.

```
Digite o nome do arquivo: teste.csv
Conteúdo do arquivo 'teste.csv':
linha 0: como , 10
linha 1: é  ,  20
linha 2: bom , 30
linha 3: estudar, 45
```

```
linha 4: MAC0110! , 1000

Conteúdo do dicionário:
como: 10
é: 20
bom: 30
estudar: 45
MAC0110!: 1000

MAC0110 Shell
In [1]: ?
Digite:
   - uma chave no dicionário para saber o valor associado
   - sair    : sair
   - len     : tamanho do dicionário
   - ?       : exibe este menu

In [2]: como
Out[2]: 10

In [3]: len
Out[3]: 5

In [4]: MAC0110
Out[4]: ERRO
Digite:
   - uma chave no dicionário para saber o valor associado
   - sair    : sair
   - len     : tamanho do dicionário

In [5]: MAC0110!
Out[5]: 1000

In [6]: sair
Out[6]: tchau
```

Solução

```
# para fase de desenvolvimentp
TESTE = True

# comandos aceitos
SAIR    = 'sair'
LEN     = 'len'
AJUDA   = '?'

#-----------------------------------------------------------------
def main():
    ''' () -> None
    USADA para testes
    Não há nada a ser feito.
    '''
    # crie o dicionário
    dicio = init_dicionario()

    # verifique o conteúdo do dicionário criado
    if TESTE:
        # Há alguma chave no dicionario?
        if len(dicio) == 0:
            print("Vixe! O dicionario está vazia... Tchau!")
            return

        # exibe dicionario criado
        print("Conteúdo do dicionário:")
        for chave in dicio:
            print(f"{chave}: {dicio[chave]}")

    # responda a consulta interativamente
    print("\nMAC0110 Shell")
    i = 1
    cmd = input(f"In [{i}]: ")
```

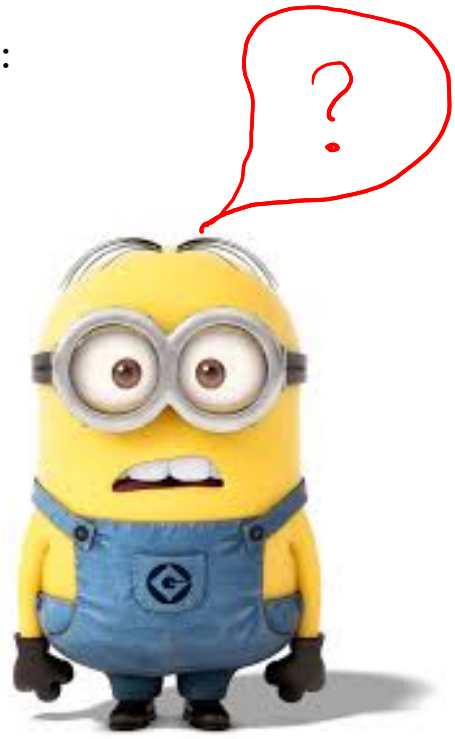
```
while cmd != SAIR:
    if cmd in dicio: # cmd é uma chave do dicionário?
        print(f"Out[{i}]: {dicio[cmd]}")
    elif cmd == LEN: # tamanho do dicionário
        print(f"Out[{i}]: {len(dicio)}")
    elif cmd == AJUDA: # exibe menu
        help()
    else: # ocorreu um erro, exiba menu
        print(f"Out[{i}]: ERRO")
        help()
    i += 1
    cmd = input(f"In [{i}]: ").strip()
print(f"Out[{i}]: tchau")
```

```python
#-----------------------------------------------------------
def init_dicionario():
    '''(None) -> dict
    LÊ o nome de um arquivo csv.
    Cada linha do arquivo possui dois campos
        - chave (str)
        - valor (int)
    RETORNA um dicionário com esses pares chave-valor.
    '''
    # dicionário vazio
    dicio = {}
    # 1 pegue o nome do arquvo
    nome = input("Digite o nome do arquivo: ")
    # 2 abra o arquivo
    # 'r' de leitura (read)
    arq = open(nome, 'r', encoding="utf-8")
    # 3 lê a linhas do arquivo, uma por vez
    if TESTE:
        print(f"Conteúdo do arquivo '{nome}':")
        i = 0
    for linha in arq:
        if TESTE:
            # imprime a string linha lida
            print(f"linha {i}: {str(linha)}", end="")
            i += 1
        lst = linha.split(",")
        if len(lst) == 2:
            chave = lst[0].strip()
            valor = int(lst[1])
            dicio[chave] = valor # insere no dicionário
    if TESTE:
        print("\n") # pula uma linha
    # 4 fechar o arquivo
    arq.close()
    return dicio
```

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

    EXIBE uma mensagem com ajuda para usar o programa.
    '''
    s = "Digite:\n" +\
        "   - uma chave no dicionário para saber o valor associado
       f"   - {SAIR:7} : sair\n" +\
       f"   - {LEN:7} : tamanho do dicionário\n"+\
       f"   - {AJUDA:7} : exibe este menu"
    print(f"{s}")

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



23.4 Exercício: shell *de luxe*

Agora vocês estenderão o conjunto de comando aceitas pelo programa anterior. Para isso vocês deverão escrever algumas funções.

Baixem os arquivos

Baixem o arquivo `shell_de_luxe.py` e baixem daqui o arquivo `teste.csv` para a mesma pasta em que baixou o programa `shell_de_luxe.py`. Mais tarde vocês podem baixar outros arquivos.

O arquivo `shell_de_luxe.py` tem três funções **completamente escritas**:

- `main()`: reponsável pela interação;
- `init_dicio()`: cria o dicionário a partir de um arquivo `.csv` com o formato especificado no exercício anterior; e
- `help()`: usada pela `main()` para apresentar o menu de uso do programa sempre que pedido ou necessário.

Funções a serem escritas

Há quatro funções muito semelhantes a serem escritas:

- `maior_valor(dicio`: **recebe** um dicionário `dicio` e **retorna** *uma* chave de maior valor associado;
- `keys(dicio)`: **recebe** um dicionário `dicio` e **retorna** uma lista com todas as chaves no dicionário;
- `values(dicio)`: **recebe** um dicionário `dicio` e **retorna** uma lista com todos os valores de chaves no dicionário com repetições; e
- `items(dicio)`: **recebe** um dicionário `dicio` e **retorna** uma lista com todos os pares de itens `[chave, valor]` no dicionário, cada par é uma lista.

Façam o maior número de funções que conseguirem, uma por vez, testando uma por vez.

Exemplo

A seguir está um exemplo com o arquivo `teste.csv`.

```
Conteúdo do arquivo 'teste.csv':
linha 0: como , 10
linha 1: é  ,  20
linha 2: bom , 30
linha 3: estudar, 45
linha 4: MAC0110! , 1000

Conteúdo do dicionário:
como: 10
é: 20
bom: 30
estudar: 45
MAC0110!: 1000

MAC0110 Shell

In [1]: ?
Digite:
   - uma chave no dicionário para saber o valor associado
   - sair    : sair
   - chaves  : todas as chaves no dicionário
   - valores : todos os valores no dicionário
   - itens   : todos os itens no dicionário
   - len     : tamanho do dicionário
   - max     : chave de maior valor
   - ?       : exibe este menu

In [2]: chaves
Out[2]: ['como', 'é', 'bom', 'estudar', 'MAC0110!']

In [3]: valores
Out[3]: [10, 20, 30, 45, 1000]
```

```
In [4]: itens
Out[4]: [['como', 10], ['é', 20], ['bom', 30], ['estudar', 45],

In [5]: len
Out[5]: 5

In [6]: max
Out[6]: MAC0110!

In [7]: sair
Out[7]: tchau
```

Solução

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# para fase de desenvolvimento
TESTE = True

# comandos aceitos
MOSTRE  = 'mostre'
SAIR    = 'sair'
CHAVES  = 'chaves'
VALORES = 'valores'
ITENS   = 'itens'
LEN     = 'len'
MAX     = 'max'
AJUDA   = '?'

#-------------------------------------------------------------------
def main():
    # crie o dicionário
    dicio = init_dicionario()

    if TESTE:
        # Há alguma chave no dicionario?
        if len(dicio) == 0:
            print("Vixe! O dicionario está vazia... Tchau!")
            return

        # exibe dicionario criado
        print("Conteúdo do dicionário:")
        for chave in dicio:
            print(f"{chave}: {dicio[chave]}")

    # responda a consulta interativamente
    print("\nMAC0110 Shell")
```

```
    i = 1
    cmd = input(f"In [{i}]: ").strip()
    while cmd != SAIR:
        if cmd in dicio: # cmd é uma chave do dicionário?
            print(f"Out[{i}]: {dicio[cmd]}")
        elif cmd == CHAVES: # exiba as chaves
            print(f"Out[{i}]: {keys(dicio)}")
        elif cmd == VALORES: # exiba os valores
            print(f"Out[{i}]: {values(dicio)}")
        elif cmd == ITENS: # exiba os itens chave:valor
            print(f"Out[{i}]: {items(dicio)}")
        elif cmd == LEN: # tamanho do dicionário
            print(f"Out[{i}]: {len(dicio)}")
        elif cmd == MAX: # chave de maior valor
            print(f"Out[{i}]: {maior_valor(dicio)}")
        elif cmd == AJUDA: # exibe menu
            help()
        else: # erro, exiba menu
            print(f"Out[{i}]: ERRO")
            help()
        i += 1
        cmd = input(f"In [{i}]: ").strip()
    print(f"Out[{i}]: tchau")


#-----------------------------------------------------------
def maior_valor(dicio):
    '''(dict) -> str
    RECEBE um dicionário dicio
    RETORNA uma chave de maior valor.
    PRÉ-CONDIÇÃO: supõe que os valores podem ser comparados
                  com >.
    EXERCÍCIO EXTRA: estender para retornar TODAS as chaves de mai
                     valor.
    '''
```

```
    max_chave = ''
    max_valor = None
    for chave in dicio:
        if max_valor == None or dicio[chave] > max_valor:
            max_chave = chave
            max_valor = dicio[chave]
    return max_chave

#-----------------------------------------------------------
def keys(dicio):
    '''(dict) -> str
    RECEBE um dicionário dicio
    RETORNA uma lista com todas as chaves no dicionario.
    '''
    lst_chaves = []
    for chave in dicio:
        lst_chaves += [chave]
    return lst_chaves

#-----------------------------------------------------------
def values(dicio):
    '''(dict) -> list
    RECEBE um dicionário dicio
    RETORNA uma lista com todos os valores de chaves no
        dicionario, com as repetições.
    '''
    valores = []
    for chave in dicio:
        valores += [dicio[chave]]
    return valores

#-----------------------------------------------------------
def items(dicio):
    '''(dict) -> list
    RECEBE um dicionário dicio
```

```
    RETORNA uma lista com todos os pares chave-valor no
        dicionario. Cada par é uma lista [chave, valor].
    '''
    itens = []
    for chave in dicio:
        itens += [[chave, dicio[chave]]]
    return itens

#-----------------------------------------------------------
def init_dicionario():
    '''(None) -> dict
    LÊ o nome de um arquivo csv.
    Cada linha do arquivo possui dois campos
        - chave (str)
        - valor (int)
    RETORNA um dicionário com esses pares chave-valor.
    '''
    # 1 pegue o nome do arquvo
    dicio = {}
    nome = input("Digite o nome do arquivo: ")
    # 2 abra o arquivo
    # 'r' de leitura (read)
    arq = open(nome, 'r', encoding="utf-8")
    # 3 lê a linhas do arquivo, uma por vez
    if TESTE:
        print(f"Conteúdo do arquivo '{nome}':")
        i = 0
    for linha in arq:
        if TESTE:
            # imprime a string linha lida
            print(f"linha {i}: {str(linha)}", end="")
            i += 1
        lst = linha.split(",")
        if len(lst) == 2:
            chave = lst[0].strip()
```

```
            valor = int(lst[1])
            dicio[chave] = valor
    if TESTE:
        print("\n") # pula uma linha
    # 4 fechar o arquivo
    return dicio
```

arq.close()

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

    EXIBE uma mensagem com ajuda para usar o programa.
    '''
    s = "Digite:\n" +\
        "   - uma chave no dicionário para saber o valor associado
       f"   - {SAIR:7} : sair\n" +\
       f"   - {CHAVES:7} : todas as chaves no dicionário\n" +\
       f"   - {VALORES:7} : todos os valores no dicionário\n" +\
       f"   - {ITENS:7} : todos os itens no dicionário\n"  +\
       f"   - {LEN:7} : tamanho do dicionário\n" +\
       f"   - {MAX:7} : chave de maior valor\n" +\
       f"   - {AJUDA:7} : exibe este menu"
    print(f"{s}")

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

Solução com operações nativas

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
# para fase de desenvolvimentp
TESTE = False

# comandos aceitos
SAIR    = 'sair'
CHAVES  = 'chaves'
VALORES = 'valores'
ITENS   = 'itens'
LEN     = 'len'
MAX     = 'max'
AJUDA   = '?'

#-----------------------------------------------------------------
def main():
    # leia e crie o dicionário
    dicio = init_dicionario()

    if TESTE:
        # Há alguma chave no dicionario?
        if len(dicio) == 0:
            print("Vixe! O dicionario está vazia... Tchau!")
            return

        # exibe dicionario criado
        print("Conteúdo do dicionário:")
        for chave in dicio:
            print(f"{chave}: {dicio[chave]}")
        print("\n") # pula uma linha

    # responda a consulta interativamente
    print("\nMAC0110 Shell")
```

```
    i = 1
    cmd = input(f"In [{i}]: ")
    while cmd != SAIR:
        if cmd in dicio:
            print(f"Out[{i}]: {dicio[cmd]}")
        elif cmd == CHAVES:
            print(f"Out[{i}]: {dicio.keys()}") # NOVO
        elif cmd == VALORES:
            print(f"Out[{i}]: {dicio.values()}") # NOVO
        elif cmd == ITENS:
            print(f"Out[{i}]: {dicio.items()}") # NOVO
        elif cmd == LEN:
            print(f"Out[{i}]: {len(dicio)}")
        elif cmd == MAX:
            print(f"Out[{i}]: {maior_valor(dicio)}")
        elif cmd == AJUDA:
            help()
        else:
            print(f"Out[{i}]: ERRO")
            help()
        i += 1
        cmd = input(f"\nIn [{i}]: ")
    print(f"Out[{i}]: tchau")


#-------------------------------------------------------------
def maior_valor(dicio):
    '''(dict) -> list
    RECEBE um dicionário dicio
    RETORNA a chave de maior valor.
    PRÉ-CONDIÇÃO: supõe que os valores podem ser comparados
                  com >.
    '''
    max_chave = ''
    max_valor = 0
```

```
    for chave in dicio:
        if dicio[chave] > max_valor:
            max_chave = chave
            max_valor = dicio[chave]
    return max_chave

#-------------------------------------------------------------
def init_dicionario():
    '''(None) -> dict
    LÊ o nome de um arquivo csv.
    Cada linha do arquivo possui dois campos
        - chave (str)
        - valor (int)
    RETORNA um dicionário com esses pares chave-valor.
    '''
    # 1 pegue o nome do arquvo
    dicio = {}
    nome = input("Digite o nome do arquivo: ")
    # 2 abra o arquivo
    # 'r' de leitura (read)
    arq = open(nome, 'r', encoding="utf-8")
    # 3 lê a linhas do arquivo, uma por vez
    if TESTE:
        print(f"Conteúdo do arquivo '{nome}':")
        i = 0
    for linha in arq:
        if TESTE:
            # imprime a string linha lida
            print(f"linha {i}: {str(linha)}", end="")
            i += 1
        lst = linha.split(",")
        if len(lst) == 2:
            chave = lst[0].strip()
            valor = int(lst[1])
            dicio[chave] = valor
```

```
    if TESTE:
        print("\n") # pula uma linha
    # 4 feche o arquivo
    return dicio
```

arq.close()

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

    EXIBE uma mensagem com ajuda para usar o programa.
    '''
    s = "Digite:\n" +\
        "   - uma chave no dicionário para saber o valor associado
       f"   - {SAIR:7} : sair\n" +\
       f"   - {CHAVES:7} : todas as chaves no dicionário\n" +\
       f"   - {VALORES:7} : todos os valores no dicionário\n" +\
       f"   - {ITENS:7} : todos os itens no dicionário\n"  +\
       f"   - {LEN:7} : tamanho do dicionário\n" +\
       f"   - {MAX:7} : chave de maior valor\n" +\
       f"   - {AJUDA:7} : exibe este menu"
    print(f"{s}")

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

23.5 Dicionários

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, par `chave:valor` na tabela: `dicio[chave] = valor`
- *busca*: consiste em *obter* o `valor` de um elemento que tenha uma dada `chave`: `valor = dicio[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 recent
<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': 2.71}
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`

23.6 Arquivos

Em Python há pelo menos quatro maneira diferentes de lermos um arquivo. Cada uma delas tem sua utilidade. Até o momento vimos duas delas.

Nos exemplos a seguir utilizaremos o arquivo `jeff.txt`.

```
We hold these truths to be self-evident:
that all men are created equal;
that they are endowed by their Creator
with certain unalienable rights;
that among these are life, liberty,
and the pursuit of happiness.
```

`read()`

Se `arq` é um arquivo, `arq.read()` retorna uma string com todo o conteúdo do arquivo. Os caracteres de *nova linha* `'\n'` no arquivo estão na string.

Exemplo de leitura com `read()`

```
def leitura_com_read():
    ''' (None) -> None

    Exemplo de leitura de um arquivo com read()
    '''
    print("Leitura com read()")
    # 1 pegue o nome do arquivo
    nome_arquivo = input("Digite o nome do arquivo: ")
    # 2 abra o arquivo
    arq = open(nome_arquivo, 'r', encoding='utf-8')
    # 3 leia o conteúdo do arquivo
    txt = arq.read() # retorna uma string
    # 4 feche o arquivo
    arq.close()
    # exiba o conteúdo do arquivo
    print(f"Conteúdo do arquivo '{nome_arquivo}'")
    print(txt)
```

Saída produzida

```
Leitura com read()
Conteúdo do arquivo 'jeff.txt'
We hold these truths to be self-evident:
that all men are created equal;
that they are endowed by their Creator
with certain unalienable rights;
that among these are life, liberty,
and the pursuit of happiness.
```

`for linha in arq:`

Se **`arq`** é um arquivo, **`for linha in arq:`** lê o conteúdo de um arquivo linha a linha. No caso, a variável **`linha`** recebe em cada iteração uma string com o conteúdo de cada linha do arquivo. Os caracteres de *nova linha* **`'\n'`** no arquivos estão nas strings.

Exemplo de leitura com **`for ...`**

```
def leitura_com_for():
    '''(None) -> None

    Exemplo de leitura de um arquivo percorrendo suas
    linhas com for ...
    '''
    print("Leitura com for ..")
    # 1 pegue o nome do arquivo
    nome_arquivo = input("Digite o nome do arquivo: ")
    # 2 abra o arquivo
    arq =open(nome_arquivo, 'r', encoding='utf-8')
    # 3 leia o conteúdo do arquivo linha a linha
    print(f"Conteúdo do arquivo '{nome_arquivo}'")
    i = 0
    for linha in arq:
        print(f"{i}: {linha}")
        i += 1
    # 4 feche o arquivo
    arq.close()
```

Saída produzida

```
Leitura com for ..
Conteúdo do arquivo 'jeff.txt'
0: We hold these truths to be self-evident:

1: that all men are created equal;

2: that they are endowed by their Creator
```

3: with certain unalienable rights;

4: that among these are life, liberty,

5: and the pursuit of happiness.

readline()

Se **arq** é um arquivo, **arq.reasline()** retorna uma string com o conteúdo de uma linha do arquivo. Um novo **arq.readline()** retorna a proxima linha, e mais um novo **arq.readline()** retorna a próxima linha e assim até o final do arquivo se atingido. Os caracteres de *nova linha* **'\n'** no arquivo estão nas strings.

Exemplo de leitura com **readline()**

```
def leitura_com_readline():
    '''(None) -> None

    Exemplo de leitura de um arquivo com readline()
    '''
    print("Leitura com readline()")
    # 1 pegue o nome do arquivo
    nome_arquivo = input("Digite o nome do arquivo: ")
    # 2 abra o arquivo
    arq = open(nome_arquivo, 'r', encoding='utf-8')
    # 3 leia o conteúdo do arquivo linha a linha
    print(f"Conteúdo do arquivo '{nome_arquivo}'")
    i = 0
    linha = arq.readline()
    while linha != '':
        print(f"{i}: {linha}")
        i += 1
        linha = arq.readline()
    # 4 feche o arquivo
    arq.close()
```

Saída produzida

```
Leitura com readline()
Conteúdo do arquivo 'jeff.txt'
0: We hold these truths to be self-evident:

1: that all men are created equal;
```

2: that they are endowed by their Creator

3: with certain unalienable rights;

4: that among these are life, liberty,

5: and the pursuit of happiness.

readlines()

Se **arq** é um arquivo, **arq.reaslines()** retorna uma lista em que cada item é uma string com o conteúdo de uma linha do arquivo. Os caracteres de *nova linha* **\n** no arquivo estão nas strings.

Exemplo de leitura com **readlines()**

```
def leitura_com_readlineS():
    '''(None) -> None

    Exemplo de leitura de um arquivo com readline()
    '''
    print("Leitura com readlines()")
    # 1 pegue o nome do arquivo
    nome_arquivo = input("Digite o nome do arquivo: ")
    # 2 abra o arquivo
    arq = open(nome_arquivo, 'r', encoding='utf-8')
    # 3 leia o conteúdo do arquivo linha a linha
    lst_linhas = arq.readlines()
    # 4 feche o arquivo
    arq.close()
    # exiba o conteúdo do arquivo
    for i in range(len(lst_linhas)):
        linha = lst_linhas[i]
        print(f"{i}: {linha}")
```

Saída produzida

```
Leitura com readlines()
0: We hold these truths to be self-evident:

1: that all men are created equal;

2: that they are endowed by their Creator

3: with certain unalienable rights;
```

4: that among these are life, liberty,

5: and the pursuit of happiness.