26 Reunião 26: 26/NOV/2020

Figure 1: Rei leão

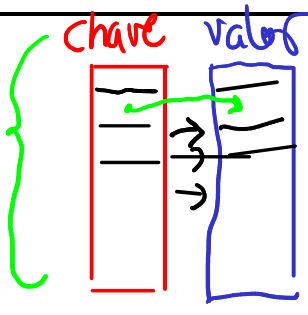
Figure 2: Xiii!

26.1 Reuniões passadas

- tipos nativos `str`, `int`, `float`, `bool`, `NoneType`, `list`, `dict`
- `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**
- operador `in`: `in range()`, `in str`, `in list` e `in dict`

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

Atribuição não cria nada, além de um apelido



26.2 Dicionários

Figure 3: Dicionário imagens-palavras

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
`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`

```
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',
```

26.3 Exercício: mais consultas interativas

Escreva um programa que leia um arquivo com dígitos de um número como π e um inteiro positivo `k`. O programa deve responder interativamente consulta a respeito da frequência dos subnúmeros de `k` dígitos no número.

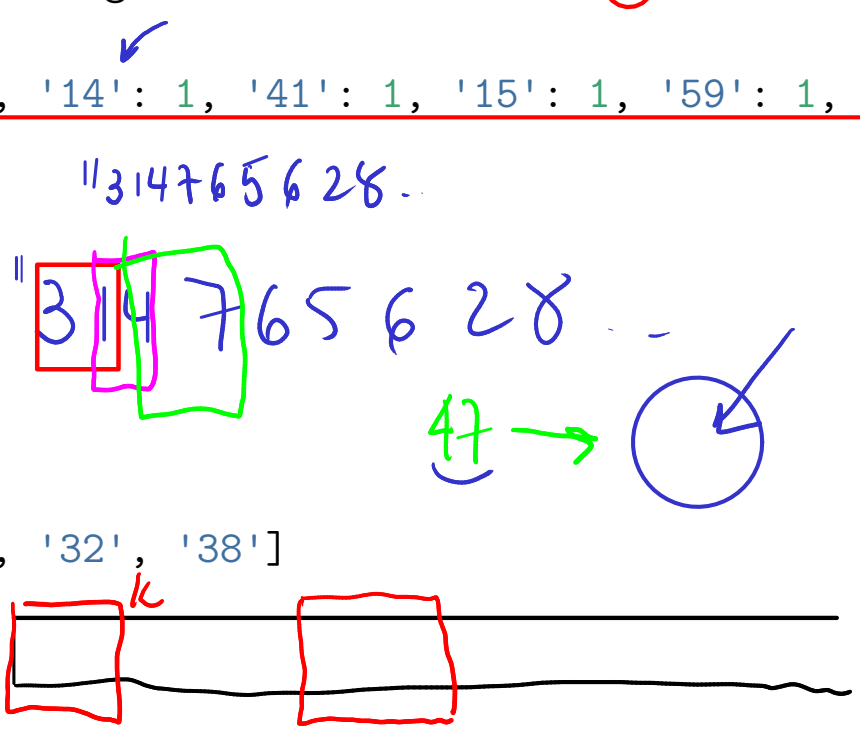
Para você testar seu programa você pode usar os arquivos que estão aqui

Exemplos

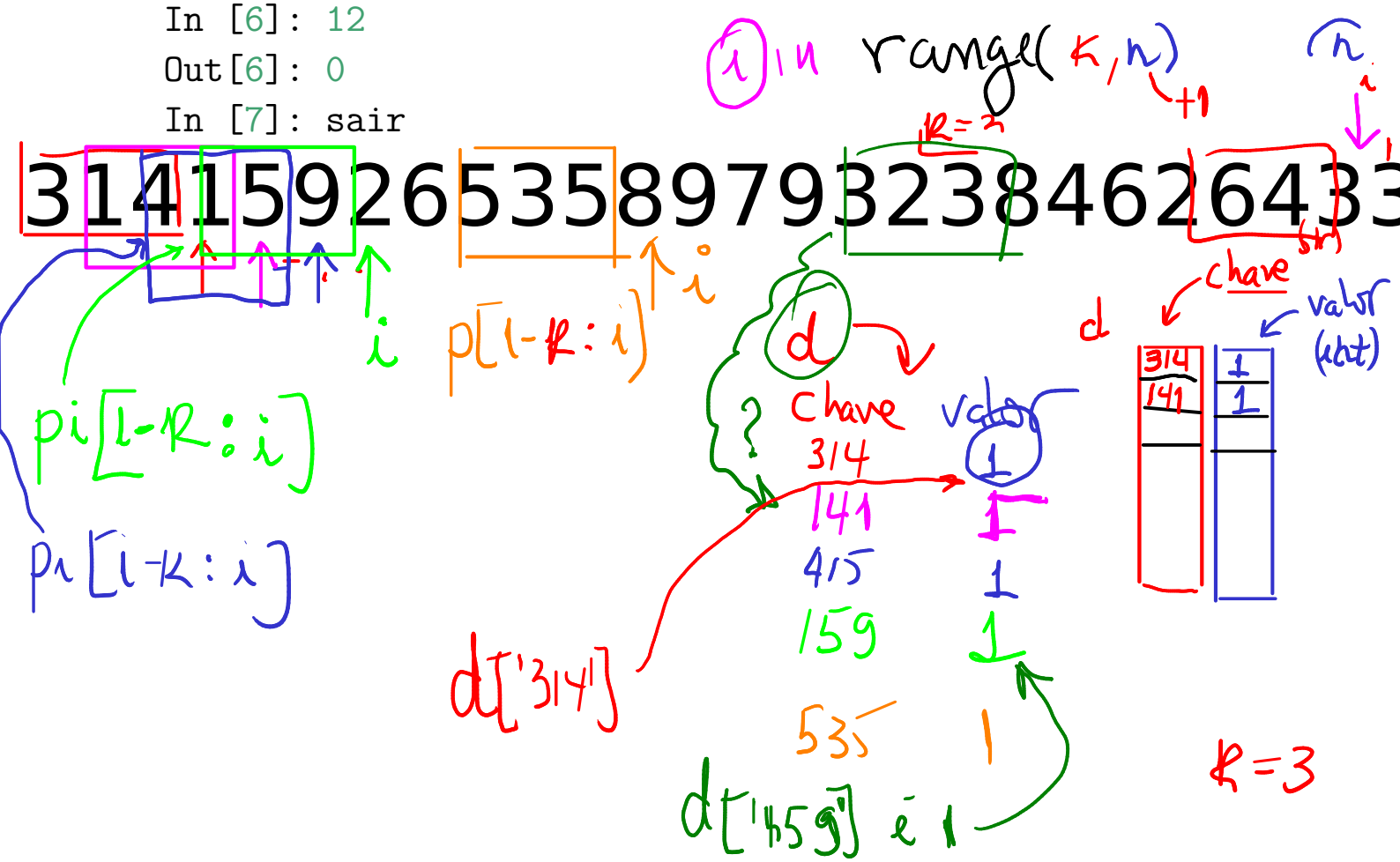
A seguir estão alguns exemplos de consultas:

- `sair`: encerra a execução do programa
- `mostre`: mostra a tabela com todos os subnúmeros de `k` digitos que foram encontrados e sua frequência
- número: mostra a frequência do número
- `max`: exibe a maior frequência de um subnúmero e a lista dos subnúmeros que têm essa frequência.

```
Digite o nome do arquivo: pi-30.txt
arquivo 'pi-30.txt' lido.
Digite o número de digitos dos subnúmeros: 2
In [1]: mostre
Out[1]: {'31': 1, '14': 1, '41': 1, '15': 1, '59': 1, '92': 1
In [2]: 31
Out[2]: 1
In [3]: 59
Out[3]: 1
In [4]: 7
Out[4]: 0
In [5]: max
Out[5]: 2: ['26', '32', '38']
In [6]: len
Out[6]: 26
In [7]: sair
```



```
Digite o nome do arquivo: pi-1million.txt
arquivo 'pi-1million.txt' lido.
Digite o número de digitos dos subnúmeros: 4
In [1]: len
Out[1]: 10000
In [2]: max
Out[2]: 141: ['3391']
In [3]: 1234
Out[3]: 78
In [4]: 2345
Out[4]: 108
In [5]: 3456
Out[5]: 90
In [6]: 12
Out[6]: 0
In [7]: sair
```



Solução

```
SAIR   = 'sair'
LEN    = 'len'
MAX    = 'max'
LIMPE  = 'limpe'
CRIE   = 'crie'

def main():
    # 1 pegue o nome do arquivo
    nome = input("Digite o nome do arquivo com pi: ")
    # 2 abra o arquivo
    arq = open(nome, 'r', encoding='utf-8')
    # 3 leia o conteudo do arquivo
    pi = arq.read()
    # 4 feche o arquivo
    arq.close()
    #
    print(f"arquivo '{nome}' lido.")
    subnumeros = {}

    i = 1
    cmd = input(f"In [{i}]: ").strip()
    while cmd != SAIR:
        if cmd == CRIE:
            k = int(input("Digite o número de digitos da fatia: "
            subnumeros = init_dicionario(pi, k)
            resp = 'dicionário criado'
        elif cmd == LIMPE:
            subnumeros = {}
            resp = 'dicionario limpo'
        elif cmd == LEN:
            resp = len(subnumeros)
```

```
        elif cmd == MAX:
            valor, lst_chaves = maior_valor(subnumeros)
            resp = str(valor) + ": " + str(lst_chaves)
        elif cmd in subnumeros:
            resp = subnumeros[cmd]
        else:
            resp = 0
        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
        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(pi, k):
    '''(str, int) -> dict
```

```
    RECEBE um string `pi` com os digitos pi e um inteiro `k`.
    RETORNA um dicionário em que:
        - as chaves são strings com os subnúmeros de `pi` com `k
        - os valores são o número de ocorrências das respectivos
    '''
    # crie o dicionário
    d = {}
    # percorra todos os subnúmeros de pi de tamanho k
    for i in range(k,len(pi)):
        # pegue o próximo subnúmero
        subnumero = pi[i-k:i]
        # verifique se o subnumero está no dicionário
        if subnumero in d:
            d[subnumero] += 1
        else:
            d[subnumero] = 1
    return d

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