17 Reunião 17: 17/JUN/2021

Figure 1: Fonte: Mafalda por Joaquín Salvador Lavado Tejón (Quino)

17.1 Reuniões passadas

Tratamos de listas, *coisas* do tipo `list`. Listas em computação são coisas que em matemática as pessoas chamariam de sequências finitas. Até agora vimos em Python *coisas* dos tipos abaixo

tipo	exemplos de valores
`str`	`"Bom dia!"`, `'Como é bom estudar MAC0110!'`
`int`	..., `-3`, `-2`, `-1`, `0`, `1`, `2`, `3`,...
`float`	`3.1415926`, `2.718281828459045`, `1e-6`
`bool`	`True` e `False`
`NoneType`	`None`
`list`	`[-1, 2, 14]`, `[True, None, 'oi', 3.14, 14]`

Para memorizar

Duas coisas (bestas?) que serão absolutamnete fundamentais:

- **Variáveis são apelidos** que damos a valores, objetos, coisas que vamos usar mais tarde.
- **Atribuições** criam apenas objetos.

  ```
  chefe = Alice
  ```

  diz que `chefe` é um (outro) apelido para `Alice`.
  Não faz aparecer uma outra `Alice`.

17.2 Hoje

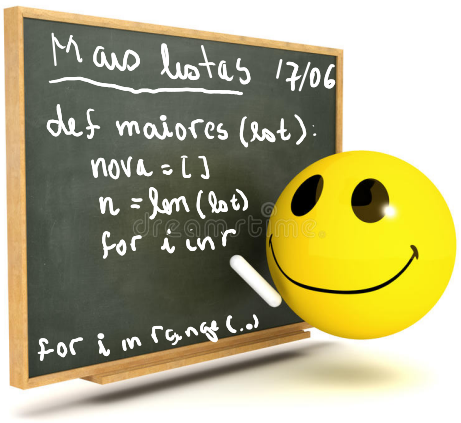


Figure 2: Fonte: Dreamstime.com

Mais listas:

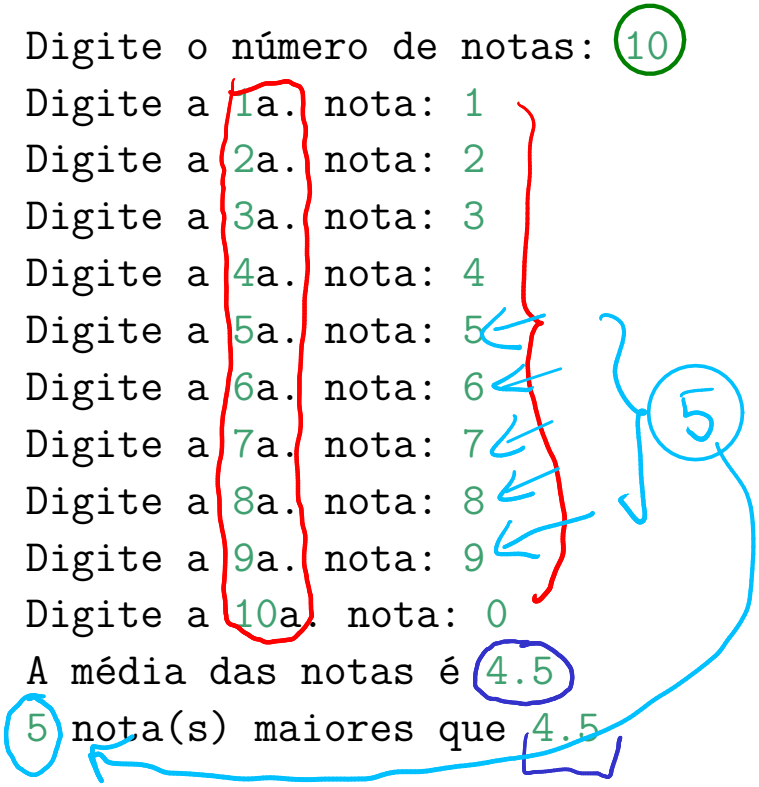
- len(lst)
- listas com funções como parâmetros
- funções que constroem e retornam listas

for i in range(início, fim, passo):
Repete bloco de comandos
i = início, início+passo, início+2 passo, ..., ≤ fim-1

17.3 Exercício: médias de provas

Dadas `n` notas de provas, calcular a média das notas e o número de notas acima dessa média.

Exemplos



```
Digite o número de notas: 10
Digite a 1a. nota: 1
Digite a 2a. nota: 2
Digite a 3a. nota: 3
Digite a 4a. nota: 4
Digite a 5a. nota: 5
Digite a 6a. nota: 6
Digite a 7a. nota: 7
Digite a 8a. nota: 8
Digite a 9a. nota: 9
Digite a 10a. nota: 0
A média das notas é 4.5
5 nota(s) maiores que 4.5
```

```
Digite o número de notas: 4
Digite a 1a. nota: 5.2
Digite a 2a. nota: 7.9
Digite a 3a. nota: 5.1
Digite a 4a. nota: 4.3
A média das notas é 5.6
1 nota(s) maiores que 5.6
```

Programa bagunçado

A seguir está uma solução para este exercício. Infelizmente o programa caiu no chão e as tabulações foram perdidas e os comandos foram embaralhados. A sua tarefa será consertar o programa colocando os comandos na ordem certa e inserindo a tabulação.

```
def main():
'''
Programa que LÊ n notas de provas, CALCULA a média
das notas e IMPRIME o número de notas maiores que
a média calculada.
'''
nota = float(input(f"Digite a {i+1}a. nota: "))
soma_notas += nota

# leia o número de notas
n = int(input("Digite o número de notas: "))

i += 1
cont += 1
i += 1

if lista_notas[i] > media_notas:

# crie apelidos para soma e lista de notas

lista_notas = []

# leia, calcule a soma das notas e coloque-as nota em uma lista

i = 0
cont = 0

while i < n:
while i < n:
```

```
lista_notas += [nota]

soma_notas = 0

# calcule a média das notas
media_notas = soma_notas/n

# conte quantas notas são maiores que a média

i = 0

main()

# imprima o número de notas maiores que a media
print(f"{cont} nota(s) maior(es) que {media_notas:.1f})")

print(f"A média das notas é {media_notas:.1f}")

#---------------------------------------------------
if __name__ == '__main__':
```

Uma solução

```
def main():
    '''
    Programa que LÊ n notas de provas, CALCULA a média
    das notas e IMPRIME o número de notas maiores que
    a média calculada.
    '''
    # leia o número de notas
    n = int(input("Digite o número de notas: "))
    # crie apelidos para soma e lista de notas
    soma_notas = 0
    lista_notas = []
    # leia, calcule a soma das notas e coloque-as nota em uma list
    i = 0
    while i < n:
        nota = float(input(f"Digite a {i+1}a. nota: "))
        soma_notas += nota
        lista_notas += [nota]
        i += 1
    # calcule a média das notas
    media_notas = soma_notas/n
    print(f"A média das notas é {media_notas:.1f}")
    # conte quantas notas são maiores que a média
    cont = 0
    i = 0
    while i < n:
        if lista_notas[i] > media_notas:
            cont += 1
        i += 1
    # imprima o número de notas maiores que a media
    print(f"{cont} nota(s) maior(es) que {media_notas:.1f})")
#-----------------------------------------------
if __name__ == '__main__':
    main()
```

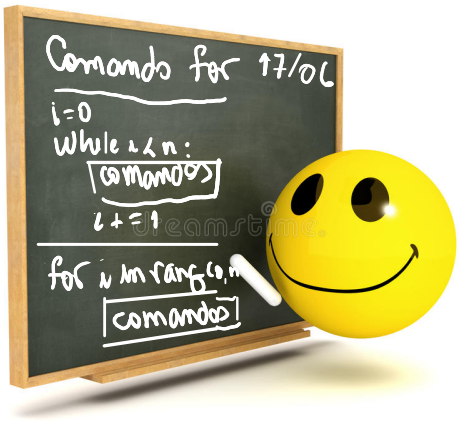


Figure 3: Fonte: Dreamstime.com

17.4 Repetições com `for ... in range(...)`

```
var = inicio
while var < fim:
    |
    |  executa bloco com var = início, início+passo
    |       início+2*passo,...,...  até
    |       fim (EXCLUSIVE)
    |
    var += passo
```

para ANTES de fim

tem o mesmo efeito que

```
for var in range(início, fim, passo):
    |
    |  executa bloco com var = início, início+passo
    |       início+2*passo,...,...  até
    |       fim (EXCLUSIVE)
```

Exemplos

```
for i in range(0, 10, 1): # (ini, fim, passo)
    print(f"{i}", end="")
```

Imprime: `0 1 2 3 4 5 6 7 8 9`

```
for i in range(2, 10, 1): # (ini, fim, passo)
    print(f"{i}", end="")
```

Imprime: `2 3 4 5 6 7 8 9`

```
for i in range(10, 2, -1): # (ini, fim, passo)
    print(f"{i}", end="")
```

Imprime `10 9 8 7 6 5 4 3`

```
for i in range(2, 10, 3): # (ini, fim, passo)
    print(f"{i}", end="")
```

Imprime: `2 5 8`

Tem mais ...

Se não especificar **início**, então **início = 0**

Se não especificar **passo**, então **passo = 1**

```
for i in range(10):  # (fim)
    print(f"{i}", end="")
```

Imprime: `0 1 2 3 4 5 6 7 8 9`

```
for i in range(2,10): # (ini, fim)
    print(f"{i}", end="")
```

Imprime: `2 3 4 5 6 7 8 9`

17.5 Exercício 2: mais médias de provas

Cada função abaixo tem um "sabor" diferente.

O mesmo problema usando funções.

main()

FEITO

(a) Escreva uma função `leia_notas()` que **recebe** que não recebe argumentos e lê `n` e uma sequência de `n` números reais. A função cria e **retorna** uma lista com os números lidos. Por exemplo, para os valores

```
4    5.2   7.9   5.1   4.3
```

a chamada `leia_notas()` deve retornar `[5.2, 7.9, 5.1, 4.3]`.

(b) Escreva uma função `media_notas()` que **recebe** uma lista `notas` e calcula e **retorna** a média dos números nesta lista. Por exemplo, para

```
notas = [5.2, 7.9, 5.1, 4.3]
```

a chamada `media_notas(notas)` deve retornar `5.6`.

len(notas)

(c) Escreva uma função `maiores_notas()` que **recebe** uma lista `notas` de números reais e um número real `valor`. A função cria e **retorna** uma lista com os números da lista maiores que `valor`. Por exemplo, para

```
notas = [5.2, 7.9, 5.1, 4.3]
```

e `valor = 5.6` a chamada `maiores_notas(notas, valor)` deve retornar a lista `[7.9]`.

(d) Usando as funções dos itens anteriores, escreva uma função `main()` que lê `n` notas de provas, calcula a média das notas da prova e imprime o número de notas maiores que a média calculada e quais são essas notas. Por exemplo

FEITO

```
Digite o número de notas: 10
Digite a 1a nota: 1
Digite a 2a nota: 2
Digite a 3a nota: 3
Digite a 4a nota: 4
Digite a 5a nota: 5
Digite a 6a nota: 6
```

```
Digite a 7a nota: 7
Digite a 8a nota: 8
Digite a 9a nota: 9
Digite a 10a nota: 0
A média das notas é 4.5
5 nota(s) maior(es) que 4.5
Notas maiores que 4.5:
5.0
6.0
7.0
8.0
9.0
```

Solução

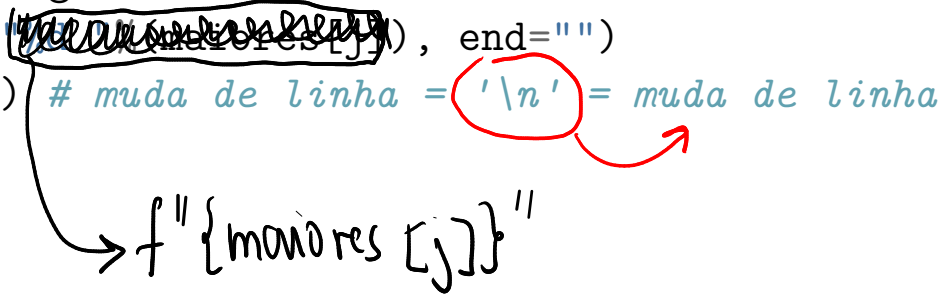
Versão com `for ... in range():` para percorrer uma lista.

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

    Programa que lê n notas de provas, calcula a média
    das notas da prova e imprime o número de notas
    maiores que a média calculada e quais são essas notas.
    '''
    # 1. leia notas
    notas = leia_notas()
    # print("main: notas=", notas)

    # 2. calcule média
    media = media_notas(notas)
    print("A média das notas é %.1f"%(media))

    # 3. imprima valores acima da média
    maiores = maiores_notas(notas, media)
    print("Notas maiores que %.1f: "%(media), end="")
    for j in range(0, len(maiores), 1):
        print(f"{maiores[j]}", end="")
    print("\n") # muda de linha = '\n' = muda de linha
```



```
#-------------------------------------------------
def leia_notas():
    '''(None) -> list

    Cria é retorna uma lista com uma sequência de notas
    lidas.
    Para a entrada: 4    5.2   7.9   5.1   4.3
    retorna [5.2,  7.9,  5.1,  4.3]
    '''
    notas = []
    # leia o número de notas
    n = int(input("Digite o número de notas: "))
    for i in range(0, n, 1):
        nota = float(input(f"Digite a {i+1}a. nota: "))
        notas += [nota] # notas = notas + [nota]

    return notas # return não é print()

#-------------------------------------------------
def media_notas(notas):
    '''(list) -> float

    Recebe ma lista de notas e retorna a sua média.
    Para notas = [5.2,  7.9,  5.1,  4.3]
    Retorna 5.6
    Pré-condicão: notas != []
    '''
    n = len(notas)
    soma = 0
    # percorra a lista somandos as notas
    for i in range(0, n, 1):
        soma += notas[i]
    media = soma / n
    return media # return não é print()
```

```python
#-----------------------------------------------------
def maiores_notas(notas, valor):
    '''(list, float) -> list

    Recebe uma lista `notas` de números reais e um
    número real `valor`. Crie e retorna uma lista
    com as notas maiores que valor.
    Para: notas = [5.2,  7.9,  5.1,  4.3] e valor = 5.6
    Retorna: [7.9]
    '''
    maiores = []
    n = len(notas)

    # percorra a lista
    for i in range(0, n, 1):
        if notas[i] > valor:
            maiores = maiores + [notas[i]]

    return maiores # return não é print()

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

17.6 Mais lista

Aqui vai um resumo do que temos visto.

Criar uma lista

```
# criar uma lista com  2, 3, 5, 7, 11, 13
alguns_primos =   [2, 3, 5, 7, 11, 13]

# criar a lista [0, 1, 2, ..., n-1] com operador +
crescente = [] # lista vazia
i = 0
while i < n:
    crescente = crescente + [i] # operador concatenação
    i += 1

# criar uma lista com n uns usando o operador *
uns = n * [1] # o mesmo que [1] + [1] + ... + [1] n vezes

# criar uma lista com k zeros
zeros = [0] * k # o mesmo que [0] + [0] + ... + [0] k vezes

# criar a lista [n-1, n-2, n-3, ..., 1, 0] usando o operador +
decrescente = []
i = 0
while i < n:
    decrescente = [i] + descrescente # operador concatenação
    i += 1
```

Ver visualização

Acessar os elementos de uma lista

Lembre-se que variáveis são *nomes* ou *apelidos* que damos a valores. Um mesmo valor pode ter vários apelidos:

```
i = 27
j = i
```

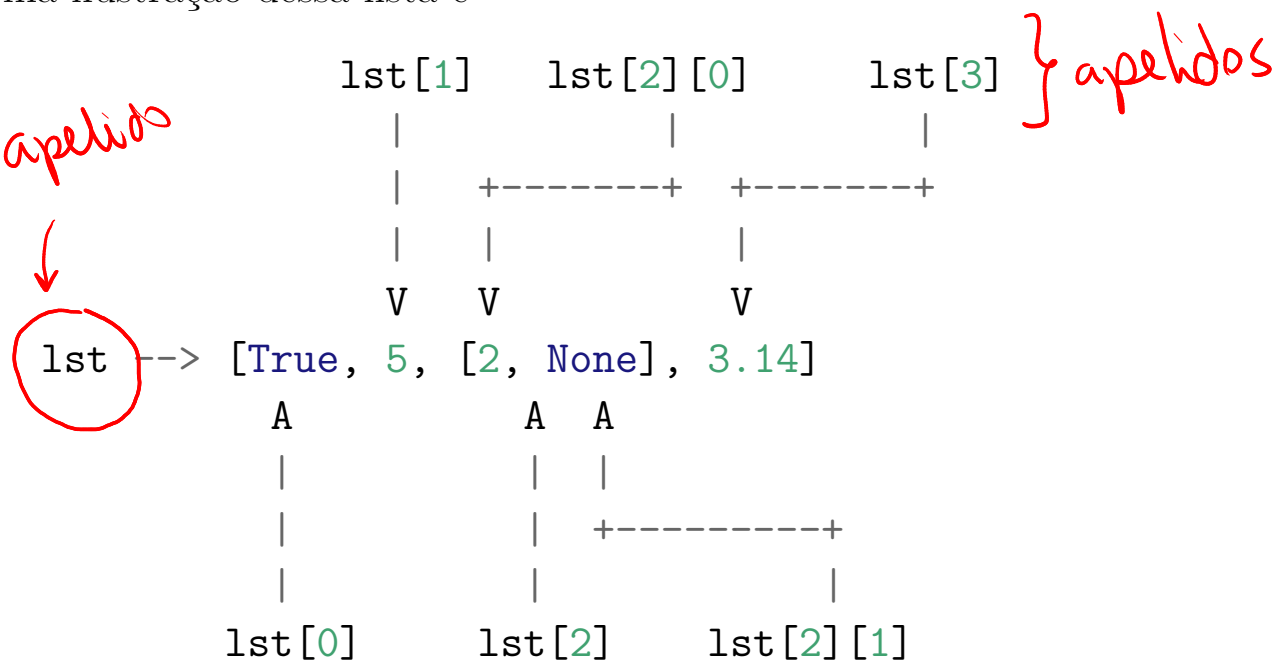
Agora 27 pode ser chamado de 27, `i` ou `j`.

Podemos usar o operador de indexação `[]` para acessar os elementos de uma lista. Em matemática as pessoas usam índices subscritos x_0, x_1, x_{27}. Em Python escrevemos `x[0]`, `x[1]`, `x[27]`. O **índice do primeiro elemento** de uma lista é zero.

Considere a lista

```
lst = [True, 5, [2, None], 3.14]
```

Uma ilustração dessa lista é




```
len(lista)    = 4
len(lista[2]) = 2
```

Temos que

```
lst[0]    # é True
lst[1]    # é  5
lst[2]    # é  [2, None]
lst[3]    # é 3.14
lst[4]    # ERRO
lst[2][0] # é 2
lst[2][1] # é None
lst[3][0] # ERRO
```

Comprimento de uma lista

A função `len()` nos fornece o número de elementos de uma lista. Usando as listas criadas anteriormente.

```
x = len([])             # x é 0
a = len(alguns_primo) # a é 6
b = len(crescente)    # b é n
c = len(uns)            # c é n
d = len(zeros)          # d é k
e = len(decrescente)  # e é n
```

Percorrer uma lista

Sabor `while`

```
# percorre do início para o fim
n = len(lst)
i = 0
while i < n:
    print(f"{i}: {lst[i]}")
    i += 1

# percorre do fim para o início
n = len(lst)
i = n-1
while i >= 0:
    print(f"{i}: {lst[i]}")
    i -= 1
```

Sabor `for _ in range(inicio, fim, passo)`

```
# percorre do início para o fim
n = len(lst)
for i in range(0, n, 1):
    print(f"{i}: {lst[i]}")

# percorre do fim para o início
n = len(lst)
for i in range(n-1, -1, -1):
    print(f"{i}: {lst[i]}")
```

Concatenação e repetição

O operador `+` concatena listas.

```
frutas = ["maca", "laranja", "banana", "cereja"]
print(f"{[1, 2] + [3, 4]}")
print(f"{frutas + [6, 7, 8, 9]}")
```

O operador `*` repete `+` um certo número de vezes.

```
print(f"{[0] * 4}") # o mesmo que print(f"{[0] + [0] + [0] + [0
print(f"{['ola', 'adeus']*2}") # ['ola', 'adeus'] + ['ola', 'ad
```