17 Reunião 17: 27/OUT/2020

Pensamento

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

17.1 Reuniões passadas

funções → listas

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]

None

17.1.1 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
```

17.1.2 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
```

17.1.3 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
```

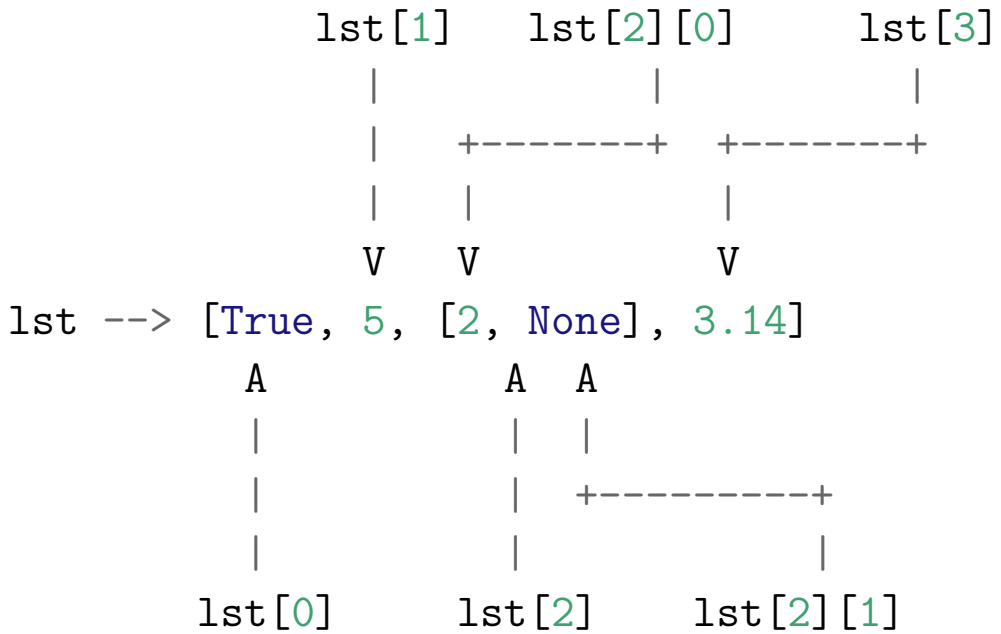
Agora `27` pode ser chamado de `27`, `i` ou `j`.

Podemos usar o operador de indexação `[]` para acessar os elementos de uma lista. Em matemática as pessoas usam índices subscritos $\mathtt{x}_0$, $\mathtt{x}_1$, $\mathtt{x}_{27}$. 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]
```

Ums 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
```

17.1.4 Percorrer uma lista

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

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

17.2 Hoje

- mais listas: exercício para calcular média
- listas com funções: quebrar o exercício com as funções:
  - `leia_notas()`: mostra como criar e retornar lista em uma função
  - `media_notas()`: `(list) -> float`
  - `maiores_notas()`: `(list, float) -> list`
- comando `for ... in range(ini, fim, passo ):`

17.3 Exercício: médias de provas

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

17.3.1 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
```

17.3.2 Solução embaralhada

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 tabulaçao.

Hmm. Olhar a solução de outro muitas vezes não é fácil.

```
def main():
'''
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.
'''
nota = float(input(f"Digite a {i+1}a. nota: "))
soma_notas += nota

i += 1

lista_notas += [nota]

# calcule a soma das notas e crie uma lista com as notas
i = 0
while i < n:

# calcule a média
media_notas = soma_notas/n
print(f"A média das notas é {media_notas:.1f}")

# conte quantas notas são acima da média
cont = 0
i = 0
while i < n:
```

```
print(f"{cont} nota(s) maiores que {media_notas:.1f})")

#-----------------------------------------------------
main()

# inicializacões
soma_notas = 0
lista_notas = []

if lista_notas[i] > media_notas:
cont += 1
i += 1

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

17.3.3 Solução desembaralhada

Há outras versões

```
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: "))

    # inicializacões
    soma_notas = 0
    lista_notas = [] # lista vazia

    # calcule a soma das notas e crie uma lista com as notas
    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
    media_notas = soma_notas/n
    print(f"A média das notas é {meia_notas:.1f}")

    # conte quantas notas são acima da média
    cont = 0
    i = 0
    while i < n:
```

```
        if lista_notas[i] > media_notas:
            cont += 1
        i += 1

    print(f"{cont} notas são maiores que {media_notas:.1f}")
```

```
#---------------------------------------
main()
```

```
i = 5
while i < n:
    soma = soma + i
    i += 1
```

```
j = 7
while j <= k+2:
    print(j)
    j += 3
```

```
for var in range(inicio, fim, passo):
```

```
for i in range(5, n, 1):
    soma = soma + i
```

```
for j in range(7, k+3, 3):
    print(j)
```

Saída: 7 10 13 16

17.4 Comando for ... in range()

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

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)
```

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

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

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

Imprime: 2 3 4 5 6 7 8 9

```
for i in range(10, 2, -1): # (ini, fim, passo)
    print(i)
```

Imprime 10 9 8 7 6 5 4 3

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

Imprime: 2 5 8

Se não especificar início = 0

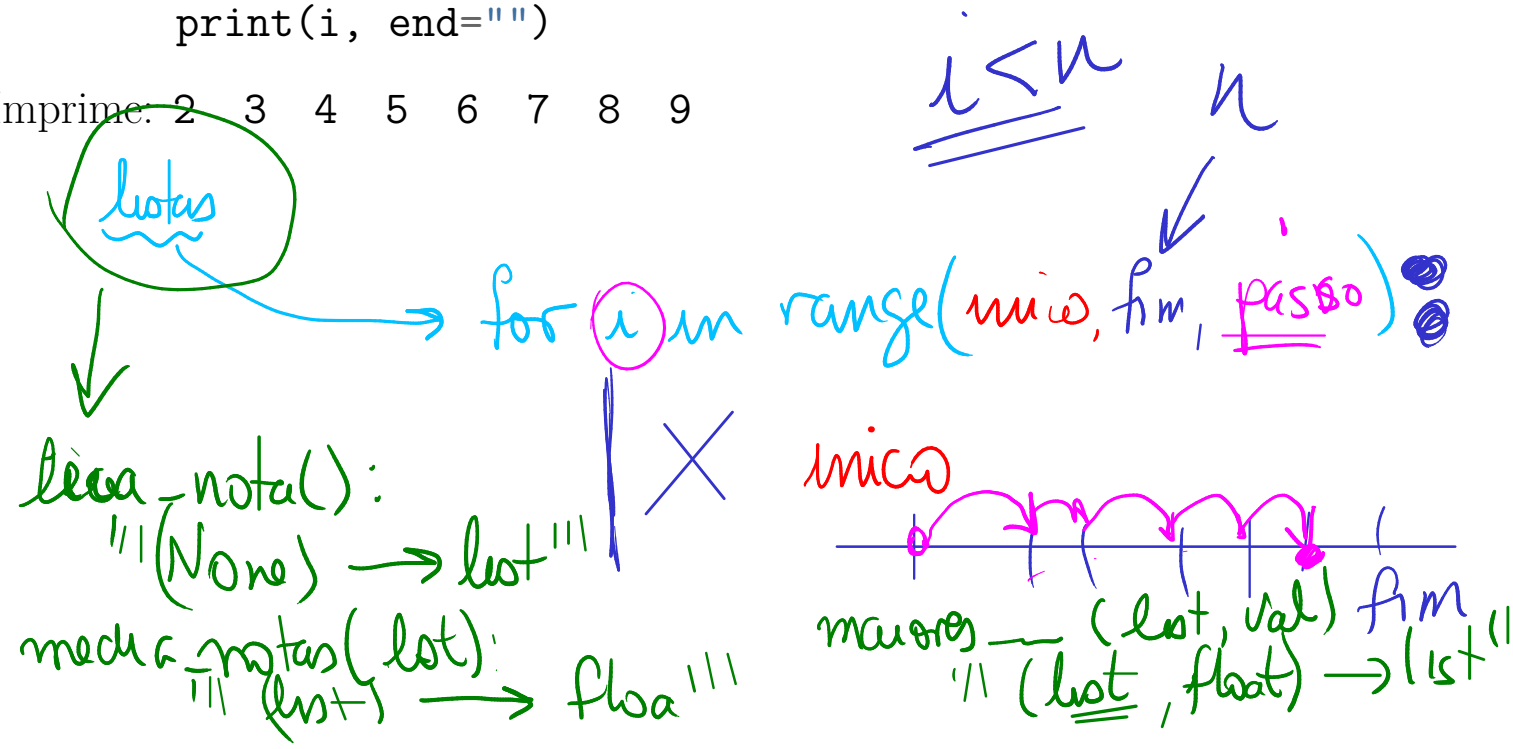
Se não especificar passo = 1

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

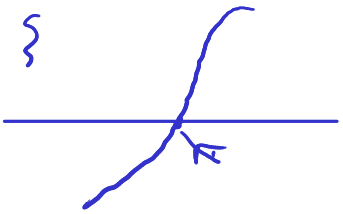
Imprime: 0 1 2 3 4 5 6 7 8 9

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

Imprime: 2 3 4 5 6 7 8 9



17.5 Exercício 2: médias de provas II



O mesmo problema usando funções. :-)

(a) Escreva uma função `leia_notas()` que **recebe** `None` como parâmetro 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 `python 4 5.2 7.9 5.1 4.3` a chamada `leia_notas()` deve retornar retorna `[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`.

(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

```
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
```

17.5.1 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("%d "%(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("Digite a %da. nota: "%(i+1)))
        notas += [nota] # notas = notas + [nota]

    return notas # != 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 # soma/n

#-----------------------------------------------
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

#--------------------------------------------------
main()
```

17.6 Mais concatenação e repetição

O operador `+` concatena listas.

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

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

```
print([0] * 4) # o mesmo que print([0] + [0] + [0] + [0])
print(["ola", "adeus"]*2) # ["ola", "adeus"] + ["ola", "adeu
```