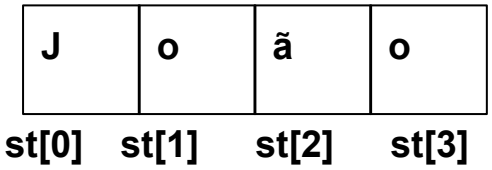
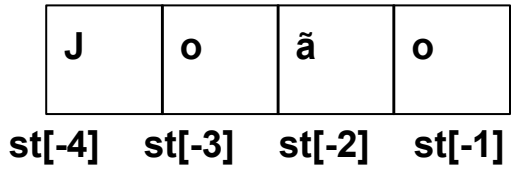
Tópico 7 - Estrutura de Dados

Tuplas e Listas

Sequências

- Valores contíguos (adjacentes) e normalmente relacionados.
  - String é um exemplo de estrutura com valores contíguos: Cadeia de caracteres
- Sequências podem ser acessadas a partir do fim.



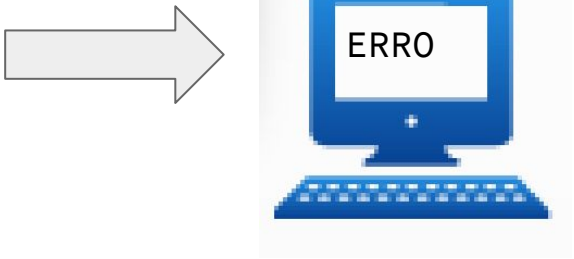


Sequências

- Conhecidas como vetores (arrays) em outras linguagens de programação.
- Há três tipos de sequências básicas em Python:
  - String (Tópico 5)
  - Tuplas
  - Listas

Tuplas

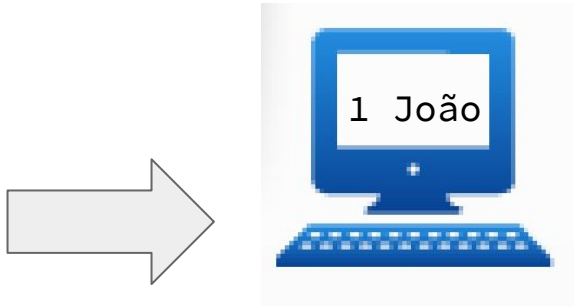
- Estrutura representada com ().
  - t=() - Tupla vazia
  - t=(1,2.5,"Joao")
- Pode relacionar diferentes tipos.
  - t=(**1**,**2.5**,**"Joao"**)
- Imutável: os valores na tupla não podem ser alterados
  - t[0]=3



Tuplas

- Valores podem ser acessados pelos índices como no caso das Strings.

```
x=t[0]
y=t[2]
print(x,y)
```



Exemplo 1
Exemplo 2
Exemplo 3

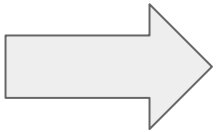
- Outros acessos possíveis por índices:

```
t[1:2]  ⟹  (2.5,) - Tupla com 1 único valor.
t[1:3]   ⟹  (2.5,"Joao")
t[0:2]  ⟹  (1,2.5)
```

Tuplas

- Pode-se concatenar tuplas

```
t=()
t=t+(1,2)
t=t+('a',)
t=t+(4.5,'!',6)
print(t)
```



```
(1,2,'a',4.5,'!',6)
```

Tuplas

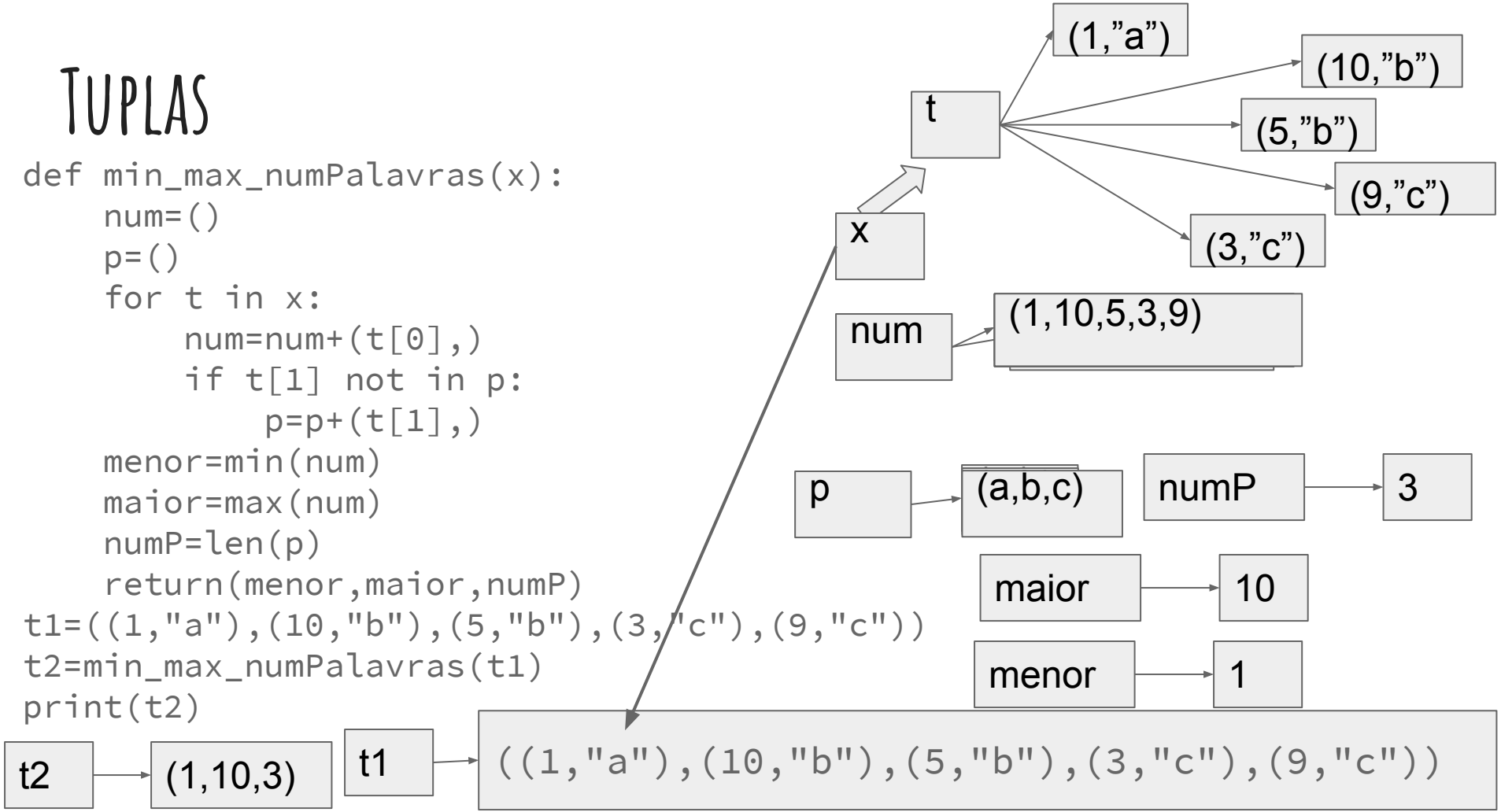
- Facilita a troca de valores

  (x,y)=(y,x)

- Permite que funções retornem uma sequência de valores
- Permite realizar repetições para a coleção de valores na tupla

Tuplas

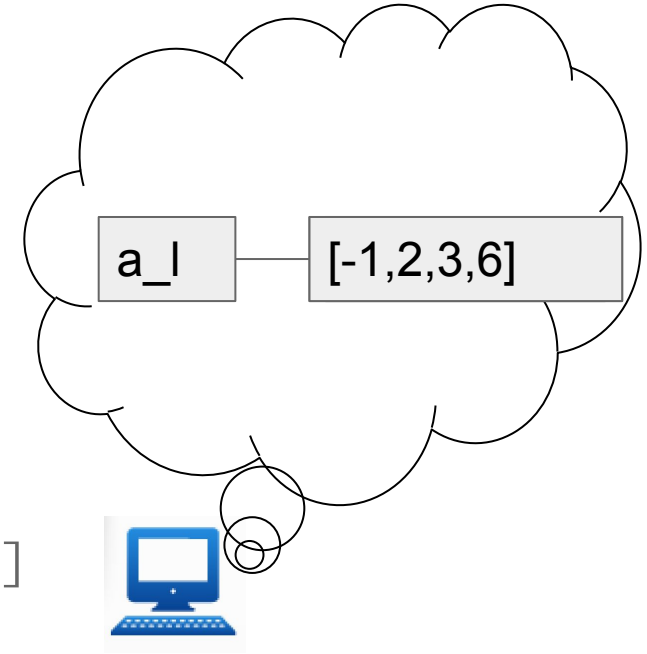
```
def min_max_numPalavras(x):
    num=()
    p=()
    for t in x:
        num=num+(t[0],)
        if t[1] not in p:
            p=p+(t[1],)
    menor=min(num)
    maior=max(num)
    numP=len(p)
    return(menor,maior,numP)
t1=((1,"a"),(10,"b"),(5,"b"),(3,"c"),(9,"c"))
t2=min_max_numPalavras(t1)
print(t2)
```



Tuplas

Listas

- Estrutura representada com [].
  - a_l=[] - Lista vazia
  - a_l=[1,2,3,[4,5]]
  - b_l=[**1**,**2.5**,**"Joao"**]
- Mutável: podem ser alteradas
  - a_l[0]=-1 ⟹ a_l=[-1,2,3,[4,5]]
  - a_l[3]=6 ⟹ a_l=[-1,2,3,6]



Exemplo 8

Listas

- Normalmente armazenam o mesmo tipo de dado
- Motivo:
  - Aplicar estruturas de repetição a partir dos valores da lista.

```
a_l=[10, 20, 30, 40, 50]
soma=0
for i in range(len(a_l)):
    soma = soma + a_l[i]
print(soma/len(a_l))
```

```
a_l=[10, 20, 30, 40, 50]
soma=0
for i in a_l:
    soma = soma + i
print(soma/len(a_l))
```

Listas

- Métodos em listas.
  - Listas são objetos em Python
  - Objetos possuem atributos e métodos
  - Atributos podem ser entendidos como parâmetros do objeto.
  - Métodos podem ser entendidos como funções associadas ao objeto.
  - Acesso ocorre como

```
<objeto>.<nomeMétodo>(<argumentos>)
```

Listas

- Métodos para incluir elementos
  - <lista>**.append(**<elemento>**)**

    a_l=[1,2,3]

    a_l.**append(**4**)** ⟹ a_l[1,2,3,4]

    Altera a lista adicionado um elemento ao final

Listas

- Métodos para incluir elementos de uma lista
  - <lista>**.extend([**e1,e2,...,eN**])**

    a_l=[1,2,3]

    a_l.**extend(**[4,5]**)** ⟹ a_l[1,2,3,4,5]

    Altera a lista adicionado elemento da lista de entrada como elementos na lista de destino

  - Não confundir com concatenar listas

    a_l1=[1,2,3] a_l2=[4,5]
    a_l3=a_l1+a_l2 ⟹ a_l3=[1,2,3,4,5]
    a_l1.extend(a_l2)⟹ a_l1=[1,2,3,4,5]

Listas

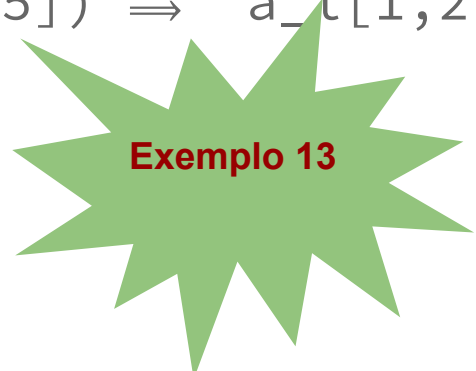
- Diferença obj.append() e obj.extend():

```
a_l=[1,2,3]

a_l.append([4,5]) ⟹  a_l[1,2,3,[4,5]]

a_l.append(4,5) ⟹  ERRO!! Apenas 1 valor!!

a_l.extend([4,5]) ⟹  a_l[1,2,3,[4,5],4,5]
```



Listas

- Métodos e funções para remover elementos de uma lista
  - **del(**<lista>**[**índice**]) - função**

    a_l=[1,2,3] del(a_l[1]) $\Longrightarrow$ a_l=[1,3]

    Remove elemento dado o seu índice na lista.
  - <lista>.pop() - método

    a_l=[1,2,3] x=a_l.pop() $\Longrightarrow$ a_l=[1,2] x=3

    Remove último elemento e retorna este elemento

Listas

- Métodos e funções para remover elementos de uma lista
  - <lista>.remove(elemento)

```
a_l=[81,72,23] x=a_l.remove(72)
```

$\Longrightarrow$ a_l=[81,23] x=72

Remove elemento

Remove a primeira ocorrência do elemento

Retorna erro se o elemento não está na lista

Listas

- Pode-se converter listas em strings e vice-versa.
- list(<string>)
  - Transforma uma string em lista.

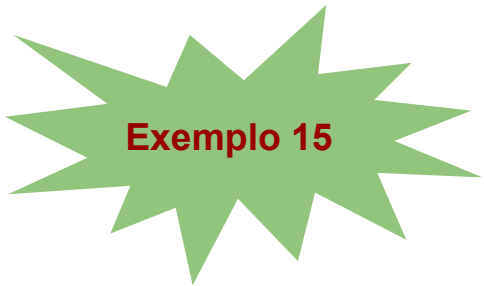
    a_l=list(‘João’)⟹ a_l=[‘J’,’o’,’ã’,’o’]

- <objString>.split()
  - Retorna uma lista com a separação dastring.
  - Por default, separa pelo caracter espaço, mas outros podem ser utilizados
    str = “Joao e Maria”
    x=str.split() ⟹ x=[‘Joao’,’e’,’Maria’]
    x=str.split(‘e’)⟹ x=[‘Joao ’,’ Maria’]

Listas

- <caracter>.join(<[lista]>)
  - Converte uma lista de caracteres em uma string.
  - Adiciona ‘<elementos da lista>‘ entre os elementos da lista.
  - Pode adiconar elementos específicos entre os elementos da lista de caracteres.

```
a_l = [‘@’,’#’,’!’,’x’]
x=’’.join(a_l)⟹ x=’@#!x’
x=’_’.join(a_l)⟹ x=’@_#_!_x’
x=’:’.join(a_l)⟹ x=’@:#:!:x’
```



Listas

- Outras funções e métodos
  - sorted(<lista>)

    a_l=[22,15,13,20] x=sorted(a_l) ⟹ x=[13,15,20,22]

    Retorna uma lista ordenada sem alterar a lista original

  - <lista>.sort()

    a_l=[22,15,13,20] a_l.sort() ⟹ a_l=[13,15,20,22]

    Altera a lista para ficar ordenada.

Listas

- Outras funções e métodos
  - <lista>.reverse()

a_l=[13,15,20,22] a_l.reverse()⟹a_l=[22,20,15,13]

Altera a lista invertendo a ordem.

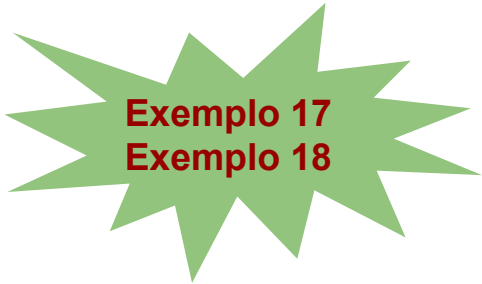
Listas

- Alias - “Aliás”- Pseudônimo
  - Uma lista é um objeto na memória
  - Uma variável aponta para um objeto, sendo referenciado (acessado) pelo nome da variável.

    print([1,2,3]) $\Longrightarrow$ o objeto [1,2,3] existe na memória.

    x=[1,2,3] e print(x) $\Longrightarrow$ o objeto [1,2,3] existe na memória e posso acessá-lo através da variável x.

Listas



- Alias - “Aliás”- Pseudônimo

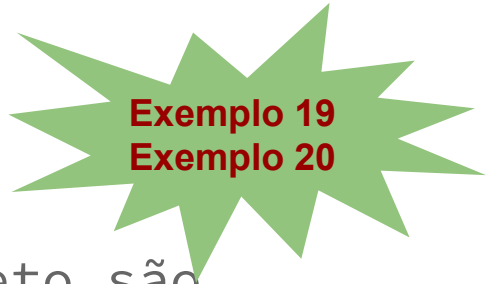
  x=[1,2,3], y=x e print(x,y) ⟹ o objeto [1,2,3] existe na memória e posso acessá-lo através da variável x e y.

  x=[1,2,3] y=x

  y.append(4) print(x,y) ⟹ o objeto muda para [1,2,3,4], temos x=[1,2,3,4] e y=[1,2,3,4]

  Mudança em y afeta x pois ocorre no objeto.

Listas



- Alias - “Aliás”- Pseudônimo
  - Variáveis apontando para o mesmo objeto são afetadas pelas mudanças naquele objeto.
  - Solução em alguns casos:
    - Clonar: copiar todos os elementos da lista

`x=[1,2,3] e y=x[:]` $\Longrightarrow$ `x=[1,2,3] e y=[1,2,3]`

`x.append(4) e y.append(5)` $\Longrightarrow$ `x=[1,2,3,4] e y=[1,2,3,5]`