Listas:
Sequências Mutáveis

Operações em sequências

string: sequência imutável,
lista: sequência mutável

Sequência é uma coleção ordenada e iterável de itens!

Operações válidas em sequências:

- ✓ concatenação
- ✓ replicação
- ✓ pertinência
- ✓ fatiamento

Concatenação: +

lista1+lista2

cria uma nova lista com os elementos da lista1 seguidos dos elementos da lista2

Exemplo:

```
L1= [1, 2, 3, 4]
L2= [-2, -1, 0, 1, 2, 3, 4, 5]
L3 = L1 + L2
print(L3)
```

```
[1, 2, 3, 4, -2, -1, 0, 1, 2, 3, 4, 5]
```

3

Replicação: *

lista * n

cria uma nova lista com *n* cópias de *lista,* concatenadas

Exemplos:

```
L1= [1, 2, 3, 4]
L2= 3* L1
print(L2)
```

```
[1, 2, 3, 4,  1, 2, 3, 4, 1, 2, 3, 4]
```

```
Lzeros= [0]*12
print(Lzeros)
```

Cria uma lista com 12 zeros

```
[0, 0, 0, 0,  0, 0, 0, 0, 0, 0, 0, 0]
```

4

Pertinência: in

el* in *lista

retorna True, se *el* ∈ à *lista* e False, caso contrário

el* not in *lista

retorna True, se *el* ∉ à *lista* e False, caso contrário

Exemplos:

```
L1= [1, 2, 3, 4]
print(3 in L1)           True
print(3 not in L1)       False
print([1,2] in L1)       False
print([1,2] not in L1)   True
```

```
L2 = [ [2,3], [1,2] ]
print(3 in L2)           False
print(3 not in L2)       True
print([1,2] in L2)       True
Print([1,2] not in L2)   False
```


Slice ou Fatiamento: [a:b:n]

Lista[a:b:n]

cria uma nova lista com os elementos da fatia selecionada.

Fatia inicia em a (inclusive) até b (exclusive) de n em n

Exemplos:

```
print(L[2:5])
```

```
[0, 1, 2]
```

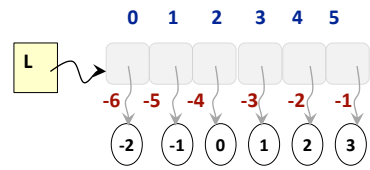
```
print(L[-6:-3])
```

```
[-2, -1, 0]
```

```
print(L[-1::-1])
```

```
[3,2,1,0,-1,-2]
```

```
L= [-2,-1,0,1,2,3]
```



Possibilidades de Fatiamento

Lista[a : b] - cria uma cópia de **a** (inclusive) até **b** (exclusive)

Lista[a :] - cria uma cópia a partir de **a** (inclusive)

Lista[: b] - cria uma cópia até **b** (exclusive)

Lista[:] - cria uma cópia de todos os elementos

Lista[a : b: n] - cria uma cópia de **a** (inclusive) até **b** (exclusive) de **n** em **n** elementos

7

Exemplos com fatiamento

pontos = [-2,-1,0,1,2,3]

Lista[a : b] – valor dos índices [**a,b)**

Lista[a :] – valor dos índices[a,...]

Lista[: b] – valor dos índices [....,b)

Lista[:] - valor dos índices [0,len(Lista]-1

Lista[a : b: n] - valor dos índices [**a,a+n,a+2*n,...b)**

Escreva a fatia p/copiar:

a. do 2º ao 5º elemento
b. do 2º elemento até o último
c. até o 3º
d. do todos os elementos
e. do 1º, 3º,5º

8

Soluções com fatiamento

Escreva a fatia p/copiar:

a. do 2º ao 5º elemento: pontos[1:5] → [-1,0,1,2]

b. do 4º elemento ao último: pontos[3:] → [1,2,3]

c. até o 3º: pontos[: 2] → [-2,-1,0]

d. do todos os elementos: pontos[:] → [-2,-1,0,1,2,3]

e. do 1º, 3º,5º : pontos[0 : 6: 2] ou
pontos[: 6: 2] ou
pontos[: : 2] → [-2,0,2]

Atribuição e modificação de fatias

Suponha que, no teclado do computador de um programador, esteja faltando o número 1 e ele usou a letra *q* para substituí-lo. Assim, o programador criou a seguinte lista de quantidade de copos de leite que tomou no café da manhã em uma semana:

['q','q','q', 2, 3,'q','q','q']

Quando chega na faculdade, o programador quer **substituir** as ocorrências da letra *q* pelo número 1.

1. Quais as fatias devem ser acessadas?
2. Ele não deseja uma cópia destas fatias, mas alterar seus valores. **Como modificar uma fatia de uma lista?**

Substituição de letra por número

Suponha que, no teclado do computador de um programador, esteja faltando o número 1 e ele usou a letra *q* para substituí-lo. Assim, o programador criou a seguinte lista de quantidade de copos de leite que tomou no café da manhã em uma semana:

['q','q','q', 2, 3,'q','q','q']

Quando chega na faculdade, o programador quer **substituir** as ocorrências da letra *q* pelo número 1.

1. Quais as fatias devem ser acessadas?

   Supondo a variável l associada à lista: l[0:3] e l[-1:-4:-1]

1. Ele não deseja uma cópia destas fatias, mas alterar seus valores. **Como modificar uma fatia de uma lista?**

   Pelo comando de atribuição!

11

Fatiamento x Atribuição

$$Lista[a : b:n] = [el_1,..,el_n]$$

$$Lista[a : b:n] = lista2$$

Exemplos: **print(lista)**

```
lista = [-2,-1,0,1,2,3]
listinha = ['p','q']
lista[0:2] = [-3,-2]                [-3,-2,0,1,2,3]
lista[0:4:2] = ['a','b']            ['a',-2,'b',1,2,3]
lista[-1:-3:-1] = listinha          ['a',-2,'b',1,'q','p']
```

12

Solução dos copos de leite

```
l=['q','q','q', 2, 3,'q','q','q']
print(l)
l[0:3] = 3*[1]
l[-1:-4:-1] = 3*[1]
print(l)
```

```
['q', 'q', 'q', 2, 3, 'q', 'q', 'q']
[1, 1, 1, 2, 3, 1, 1, 1]
```


Plotando pontos: listas e sublistas

Utilizando o módulo turtle, construa uma função que receba uma tartaruga, uma cor e uma lista de pontos, onde cada ponto é representado por [x,y]. Esta função deve plotar os pontos na cor. Lembre-se de levantar a tartaruga para colocá-la no ponto. Utilize o método *tart*.dot(5), para desenhar o ponto.

Faça uma função que receba uma lista com 6 sublistas, cada uma representando as coordenadas de um ponto (x,y) e desenhe-os, utilizando a função do item anterior, nas seguintes ordens:

a) último ponto, penúltimo ponto, antepenúltimo ponto,... . Se o ponto(0,0) estiver na lista recebida, desenhá-los em vermelho, senão desenhá-los em azul

b) criar uma lista com os elementos dos índices pares. Cada ponto desta nova lista deve ser modificado para que x=x+160 e y=y-160. Deve ser plotada usando a função do item 1 em amarelo

c) criar uma lista com os elementos dos índices ímpares. Cada ponto desta nova lista deve ser modificado para que x=x/2+10 e y=y*2+10.. Plotar a lista resultante em verde

d) Teste para a lista [[100,100],[-30,30],[0,0],[20,-90],[80,80],[-40,120]]

Ideia da solução

1. *Copia de trás p/frente, determina cor, exibe*
2. *Copia de 2 em 2, iniciando no 0, modifica valores, exibe*
3. *Copia de 2 em 2, iniciando no 1, modifica valores, exibe*

Solução

```
import turtle
def desenhaPol(t,l,cor):
    """Percorrer a lista, elemento a elemento, desenhando o ponto (x,y)"""
    t.color(cor)
    for pto in l:
        t.up()
        t.goto(pto[0],pto[1])
        t.down()
        t.dot(6)

    return
```

Uma Solução em Python

```
t=turtle.Turtle()
pontos = [[100,100],[-30,30],[0,0],[20,-90],[80,80],[-40,120]]

ltras=pontos[-1::-1]
if [0,0] in ltras:
    desenhaPol(t,ltras,"blue")
else:
    desenhaPol(t,ltras,"red")

lPar=pontos[0::2]
lImpar = pontos[1::2]

for i in range(len(lPar)):
    lPar[i][0]+=160
    lPar[i][1]-=160
desenhaPol(t,lPar,"yellow")

for i in range(len(lImpar)):
    lImpar[i][0]=lImpar[i][0]*2+10
    lImpar[i][1]=lImpar[i][0]/2 +10
desenhaPol(t,lPar,"green")
```


Soma por acumulação

Faça uma função *somaCumulativa* que receba uma lista e retorne uma nova lista onde na posição **i** tem-se a soma dos elementos da lista recebida entre as posições inicial e a posição **i** (inclusive)

Exemplo:

l = [1,2,3,4,5]

lsoma = [1,3,6,10,15]

Soma por acúmulo: ideia

l=[1,2,3,4,5]	lsoma = [1]
l=[1,2,3,4,5]	lsoma = [1,3]
l=[1,2,3,4,5]	lsoma = [1,3,6]
l=[1,2,3,4,5]	lsoma = [1,3,6,10]
l=[1,2,3,4,5]	lsoma = [1,3,6,10,15]

somaAcum = ∅
Para cada índice da lista
    soma fatia que termina no índice (inclusive)
    adiciona soma à somaAcum

19

SomaCumulativa: Solução 1

```
def somaPrefixo(l):
    soma=0
    for el in l:
        soma+=el
    return soma
```

```
sum(lista[:i+1])
```

```
def somaCumulativa(lista):
    acum = []
    for i in range(len(lista)):
        somalPrefixo= somaPrefixo(lista[:i+1])
        acum = acum + [somalPrefixo]
    return acum

l=[1,2,3,4,5]
print(somaCumulativa(l))
```

20

SomaCumulativa: Solução 2

```
def somaCumulativa(lista):
    acum = []
    for i in range(len(lista)):
        somalPrefixo= sum(lista[:i+1])
        acum = acum + [somalPrefixo]
    return acum

l=[1,2,3,4,5]
print(somaCumulativa(l))
```

O que muda entre um elemento da lista e seu sucessor?

l=[1,2,3,4,5] → [1,3,6,10,15]

21

SomaCumulativa: Solução 3

Há uma relação simples entre duas somas cumulativas consecutivas:
soma dos ***i* elementos = soma dos *i-1* elementos + elemento *i***

```
def somaCumulativa(lista):
    acum = [lista[0]]
    for el in lista[1:]):
        somalPrefixo= acum[i-1]+ el
        acum = acum + [somalPrefixo]
    return acum

l=[1,2,3,4,5]
print(somaCumulativa(l))
```

22

Controle de presença de alunos (1/9)

Um curso livre de 5 encontros (ou 5 aulas), sem matrícula prévia, fornece certificado aos alunos que comparecem a todas as aulas.

No primeiro dia, o professor passou uma lista de presença para registrar os alunos presentes.

Para facilitar o controle de frequência, o professor deseja montar uma lista com estes nomes para fazer a chamada nas demais aulas.

Controle de presença de alunos (2/9)

Faça um programa que permita o professor digitar os nomes dos alunos (termina digitando a palavra "fim") que compareceram na primeira aula, exibindo-os no seguinte formato:

Nome |_____|_____|_____|_____|_____|

Nome |_____|_____|_____|_____|_____|

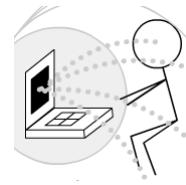
Nome |_____|_____|_____|_____|_____|

... ...

Nome |_____|_____|_____|_____|_____|

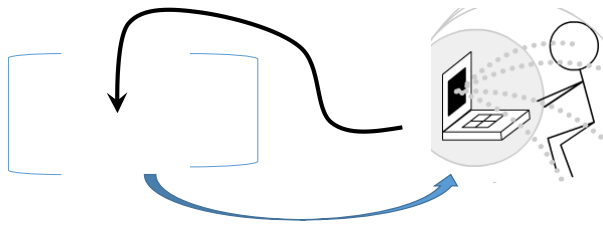
Controle de presença de alunos (3/9)

Como preencher uma lista com valores digitados pelo usuário?



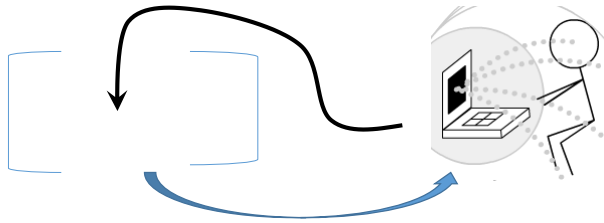
Controle de presença de alunos (4/9)

Como preencher uma lista com valores digitados pelo usuário?



Controle de presença de alunos (5/9)

Como preencher uma lista com valores digitados pelo usuário?

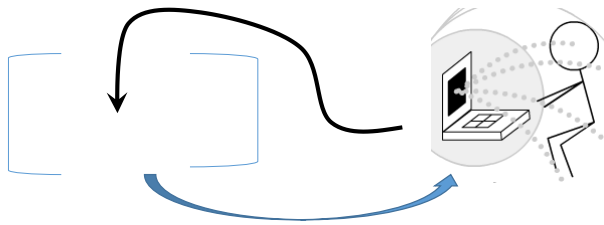


```
lista = lista + [valor lido]
```

27

Controle de presença de alunos (6/9)

Como preencher uma lista com valores digitados pelo usuário?



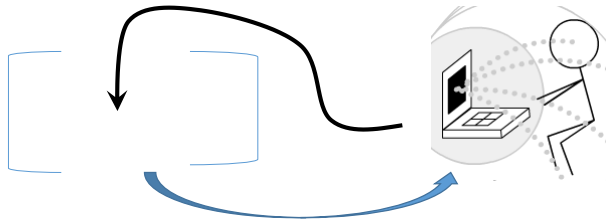
```
lista = lista + [valor lido]
```

Quando parar?

28

Controle de presença de alunos (7/9)

Como preencher uma lista com valores digitados pelo usuário?



lista = lista + [valor lido]

Quando parar? Quando o usuário digitar "fim"

29

Controle de presença de alunos (8/9)

```
def preenche():
    """ coloca na lista nomes digitados"""
    lista=[]
    nome = input("Nome do aluno? -fim para finalizar- ")
    while nome !="fim" :
        lista=lista+[nome]  #cria uma cópia da lista com o nome no final
        nome = input("Nome do aluno? -fim para finalizar- ")
    return lista

def exibe(lista):
    for nome in lista:
        print("%-10s |_____|_____|_____|_____|_____|"%nome)
    return


lista=preenche()
exibe(lista)
```

30

Controle de presença de alunos (9/9)

```
def preenche():
    """ coloca na lista nomes digitados"""
    lista=[]
    nome = input("Nome do aluno? -fim para finalizar- ")
    while nome !="fim" :
        lista=lista+[nome]  #cria uma cópia da lista com o nome no final
        nome = input("Nome do aluno? -fim para finalizar- ")
    return lista

def exibe(lista):
    for nome in lista:
        print("%-10s |_____|_____|_____|_____|_____|"%nome)
    return

lista=preenche()
exibe(lista)
```

DESAFIO 1: Colocar todos os nomes (concatenando brancos) com a quantidade de caracteres do nome mais comprido

DESAFIO 2: cada elemento da lista de chamada deve conter: nome, lista com 5 zeros (total de faltas até o momento

31

Reajustes da Conta Poupança

Faça um programa que capture o índice de reajuste de cada mês do ano anterior de uma caderneta de poupança.

A seguir, para cada investidor, capture do teclado, o número da conta, o valor aplicado e o mês em que aplicou, mostrando quanto possui no final do ano.

A entrada de dados é finalizada quando for digitado o valor 0 como o número de conta.

Observação: *Lembre-se que o montante aplicado no segundo mês é equivalente ao montante inicial acrescido do reajuste do primeiro mês.*

Exemplo: Reajustes

Jan	Fev	Mar	Abr	Mai	Jun	Jul	Ago	Set	Out	Nov	Dez
0.1	0.1	0.1	0.2	0.2	0.2	0.3	0.2	0.3	0.2	0.1	0.3

Valor aplicado: 100,00

Mês Aplicação: 11

100,00 *1.1 → 110,00 * 1.3 → 143,00

32

Reajuste da poupança: ideia

Algoritmo:

1. *Preencher uma lista com os reajustes*
2. *Le conta do usuário*
3. *Enquanto a conta ≠0*

   Le valor e mês da aplicação

   Calcula e exibe valor final

Criar uma lista vazia e adicionar as taxas de reajustes lidas.

Percorrer a lista a partir do mês (índice: mês-1) até o final

Reajuste de poupança em Python (1/2)

```
def leReajuste():
     meses=['jan','fev','mar','abr','mai','jun','jul','ago','set',
            'out','nov','dez']
     reajustes=[]
     for mes in meses:
          tx=float(input('Taxa do mês de '+mes+': '))
          reajustes = reajustes + [tx]
     return reajustes

def calculaValor(l,mes,valor):
    tot=valor
    for i in range(mes-1,12):
        tot=tot *(1+l[i])
    return tot
```

Reajuste de poupança em Python (2/2)

```
lreajustes=leReajuste()
conta =  int(input('Qual a conta? 0 p/finalizar'))
while conta != 0:
  valor = float(input('Qual o valor aplicado? '))
  mes = int(input('Em que mes foi aplicado?'))
  tot=calculaValor(lreajustes,mes,valor)
  print('Valor final: R$%.2f'%tot)
  conta = int(input('Qual a conta? 0 p/finalizar'))
```