Módulo V: Laços, Listas, Filas

Prof. Marília Nestor: nestor.marilia@gmail.com

Dica: Udemy

Coursera, EdX, OpenCourseWare

Revisão de uma linha

- Transforme 3.0 em int
- Transforme 4 em float
- Pergunte ao computador o tipo de "Bebi água"
- Imprima "Eae"
- Imprima "Como estamos?", leia a resposta e a imprima
- Defina a variável A como "Tudo bem?"
- Imprima a variável e leia a resposta
- Escreva um comentário

Revisão de mais linhas

- Escreva um comentário de 5 linhas
- Faça uma função utilizando if, elif e else
- O que essas linhas vão fazer? Idente para ficar semanticamente correto.

```
A=2
B=3
if (A>B):
print ("A é maior que B")
elif (A<B):
print ("B é maior que A")
else:
print ("A e B têm o mesmo valor")
```

Revisão de mais linhas

- URI 1006

```
A = float(input("Qual é o primeiro número?"))
B = float(input("Qual é o segundo número?"))
C = float(input("Qual é o terceiro número?"))
soma = ((A * 2) + (B * 3) + (C * 5)) / 10
print("MEDIA = %0.1f" %soma)
```

Revisão de mais linhas

- Qual a saída?

```
conta = 0
while(conta <= 10):
    conta += 1
    print(conta)
print("Valor da variável conta é: ", conta)
```

- E se eu quiser que dê 10?
- E se eu não quiser número por número?

Laço, lista e fila

- Laço: Também conhecido como looping, em um laço se dá um comando com uma condição. Esse comando é repetido enquanto a condição for atendida.
- Lista: Lista é uma série de elementos ligados ao anterior, de maneira a facilitar ações como colocar elementos, tirar, mudar de posição, por exemplo, apenas com o 'endereço' (ponteiro) do primeiro elemento e um comando.

Tipo: list

- Fila: É uma lista em que possui duas regras: quando é inserido um elemento, ele vai para o fim da fila, quando é retirado um elemento, é sempre o primeiro da fila (mais antigo).

Laços

- Função while
- Função for...in

Repete o comando de acordo com a quantidade de vezes da condição de iteração

for **variável** in **iteração**

    Instrução identada

- Digite no computador:

```
for x in [1,2,3]:
    print (x)
```

- Digite no computador:

```
soma = 20
while (soma<=20):
    soma +=2
else:
    print ("Valor da soma é", soma)
```

Laços

- Função range

Sequência numérica com início e fim

Muito utilizada em conjunto com for...in

- Digite no computador:

list (range (5)):

- Digite no computador:

list (range (5,20)):

- Digite no computador:

list (range (5,50,2)):

- Digite no computador:

```
for i in range(5):
    print(i)
else:
    print('após iteração')
```

Exercícios

- Use while

Leia 3 números e os some

- Use for...in e range

Leia o número inicial, o final e a distância em range e imprima oi nessa iteração

- Use if, for...in e range

Brinque de tim! Imprima números de 1 a 100, mas quando for múltiplo de 3 ou terminar com 3, imprima tim!

Resoluções

- Use for...in e range

Leia o número inicial, o final e a distância em range e imprima oi nessa iteração

```
A=int (input ("inicial"))
B=int (input ("final"))
C=int (input ("espaço"))
for i in range(A,B,C):
    print ("oi")
```

Resoluções

- Use if, for...in e range

Brinque de tim! Imprima números de 1 a 100, mas quando for múltiplo de 3 ou terminar com 3, imprima tim!

```
for i in range(1,101):
    if i%3==0:
        print ("tim")
    elif ((i+7)%10==0):
        print ("tim")
    else:
        print (i)
```

Listas

- Estrutura list
- Elementos contidos na lista se chamam objetos
- Para atribuição de valor, objetos devem ser limitados por colchetes e separados por víngulas:

```
A=[1,2,3,4,5]
```

- Ou explicita-se lista, de maneira que só recebe um objeto não numeral:

```
A = list(('oi'))
```

- A lista pode conter 0 objetos: A = [] ou A= list()
- A lista pode conter qualquer objeto em python:

```
A=['oi', 1, 2, True]
```

Listas, acesso aos elementos

- Cada objeto da lista será associado a um número inteiro não negativo
- O primeiro elemento é associado ao número [0], o segundo ao [1], até o enésimo elemento ser associado a n-1, sendo associado apenas a [-1]
- Para acessar um elemento, se usa a referência da lista+sua posição em colchetes:

```
A = ['123', 'como', 'voce', 'esta?']
print (A[0])
print (A[1])
print (A[-2])
print (A[-1])
```

```
A = list('oi, tudo bem?')
print (A[0])
print (A[1])
print (A[-2])
print (A[-1])
```

Listas, acesso aos elementos

Digite no computador:

```
A=[1,2,3]
B=[A,'B','C']
C=[A,B]
print (C)
```

Depois troque C por

```
C=A+B
```

Listas, funções [].append(), del[[]] e len()

- A função append adiciona um elemento ao fim da lista na estrutura <lista>.append(objeto)
- A função del exclui elementos da lista na estrutura del<lista><elemento>
- A função len conta a quantidade de elementos da lista na estrutura
- len(<lista>)

Digite no compuador:

```
A=[1,2,3]
B=[A,'B','C']
C=[A,B]
A.append(4)
del A[2]
del C[-1]
print (C)
print("C tem", len(C), "elementos")
```

```
A=[1,2,3]
B=[A,'B','C']
C=[A,B]
print("A tem", len(A), "elementos")
A.append(4)
print (C)
print("Agora, A tem", len(A), "elementos")
print("C tem", len(C), "elementos")
```

Exercício

- Crie uma lista com 3 strings
- Adicione uma string ao final da lista do último exercício e imprima
- Apague o primeiro elemento e imprima a nova lista e o seu tamanho

Listas com laços

- Usa-se laços para percorrer uma lista

Digite no computador:

```
A = [1,2,3,4,5]
x = len(A)
i = 0
while(i < x):
    print(A[i])
    i+=1
```

```
A = [1,2,3,4,5]
x = len(A)
for i in A:
    print(i)
    i+=1
```

```
for i in [1,3,5,7,9]:
    print (i*10)
```

```
for i in range(1,9,3):
    print(i*10)
```

Listas, fatiamento

- Fatia-se uma lista quando se quer apenas certos elementos em determinado intervalo

Pode ser escolhida:

A=[1,2,3,4,5,6,7,8,9]

- Apenas certa parte da lista, indicando o local do primeiro e do último elemento:

A[0:2] ou A[-3:-1]

- Essa parte da lista, mas acrescentando intervalos entre os elementos:

A[0:7:3], A[2:02:2]

- Ou apenas o intervalo desejado:

A[::2], A[::4]

Listas, funções insert(), pop() e alteração de objetos

- A função insert inclui um objeto a determinado local da lista com a estrutura: <lista>.insert(<local>,<objeto>):

```
A=[sim, ou]
A.insert(2,nao)
```

- A função pop() retorna e exclui um elemento da lista ao mesmo tempo com a estrutura: <lista>.pop(<local>). Caso não seja especificado o local, o último elemento será retirado:

```
A.pop()
```

- Para alterar um elemento, basta especificar o local e o objeto para substituição na estrutura: <lista><[local]>=<objeto>:

```
A[2]= 'mesmo'
```

Exercícios

1. Tendo a lista: A=[1, 2, 'oi', True, 7, 3.5, 222, 'nao', 'sim'], faça programas que imprimam:

- Quantos elementos há na lista até chegar ao inteiro 222
- Quantas strings há na lista e quais as posições delas

2. Tendo a lista: B=[100, 500, 'claro', 'escuro', 7.6], faça programas que imprimam:

- A soma dos números int e float nas listas A e B

3. Altere os elementos 0,2 e 4 da lista A pelos números 8, 9 e 9.5 e imprima.
4. Delete os elementos 2 e 3 da lista B, adicione os 0 e 5 da lista A no lugar e imprima.

Resoluções

1. Tendo a lista: A=[1, 2, 'oi', True, 7, 3.5, 222, 'nao', 'sim'], faça programas que imprimam:

- Quantos elementos há na lista até chegar ao inteiro 222

```
A=[1, 2, 'oi', True, 7, 3.5, 222, 'nao', 'sim']
i, j = 0, 0
for i in A:
    if i==222:
        print (j)
    else:
        j+=1
```

Resoluções

1. Tendo a lista: A=[1, 2, 'oi', True, 7, 3.5, 222, 'nao', 'sim'], faça programas que imprimam:

- Quantas strings há na lista e quais as posições delas

```
A=[1, 2, 'oi', True, 7, 3.5, 222, 'nao', 'sim']
j, k = 0,0
for i in A:
        if type (i)==str:
                print ('Na posição', j)
                j+=1
                k+=1
        else:
                j+=1
print ('Então, há ', k, ' strings na lista')
```

```
A=[1, 2, 'oi', True, 7, 3.5, 222, 'nao', 'sim']
j, k = 0,0
for i in A:
        while type (i)==str:
                k+=1
                break
print ('Há ', k, ' strings na lista')
for i in A:
        if type (i)==str:
                print ('Na posição', j)
                j+=1
        else:
                j+=1
```

Resoluções

2. Tendo a lista: B=[100, 500, 'claro', 'escuro', 7.6], faça programas que imprimam:

- A soma dos números int e float nas listas A e B

```
A=[1, 2, 'oi', True, 7, 3.5, 222, 'nao', 'sim']
B=[100, 500, 'claro', 'escuro', 7.6]
j, k = 0,0
for i in A:
        if type (i)==int:
                j+=i
        elif type (i)==float:
                j+=i
for i in B:
        if type (i)==int:
                j+=i
        elif type (i)==float:
                j+=i
print (j)
```

Resoluções

3. Altere os elementos 0,2 e 4 da lista A pelos números 8, 9 e 9.5 e imprima.

```
A=[1, 2, 'oi', True, 7, 3.5, 222, 'nao', 'sim']
print ('Antes da alteração')
print (A)
A[0]= 8
A[2]=9
A[4]=9.5
print ('Depois da alteração')
print (A)
```

Resolução

4. Delete os elementos 2 e 3 da lista B, adicione os 0 e 5 da lista A no lugar e imprima.

```
A=[1, 2, 'oi', True, 7, 3.5, 222, 'nao', 'sim']
B=[100, 500, 'claro', 'escuro', 7.6]
print ('Antes:')
print (A)
print (B)
B[2]= A[0]
B[3]= A[5]
print ('Depois:')
print (A)
print (B)
```

```
A=[1, 2, 'oi', True, 7, 3.5, 222, 'nao', 'sim']
B=[100, 500, 'claro', 'escuro', 7.6]
print ('Antes:')
print (A)
print (B)
B.pop(2)
B.pop(3)
print ('Durante:')
print (A)
print (B)
print ('Depois:')
B.insert(2,A[0])
B.insert(3,A[5])
print (A)
print (B)
```

Tuplas

- Listas imutáveis
- Utilidade diferente da lista
- Diferença da declaração é sutil:

Declaração de lista: A=['oi,', 'tudo', 'bem?'] / A = list('oi, tudo bem?')

Declaração de tupla: A=('tudo,', 'e', 'você?') / A = 'tudo,', 'e', 'você?'

Pilhas

- Mesma ideia do mundo físico: último a entrar, primeiro a sair
- Funciona com funções que já vimos, mas a regra é que o último que entra ganha prioridade sobre os que já estão na lista

Ex:

```
A=[]
A.append(1)
A.append(2)
A.append(3)
print (A)
A.pop()
print (A)
```

Filas

- Mesma ideia do mundo físico: primeiro que entra é o primeiro que sai
- Vamos usar o **deque**, do **collections,** com **deque**, **appendleft** e **popleft**
- Sem OO, usamos as funções de uma fila regular

Ex:

```
from collections import deque
A=deque([1,2,3,4,5])
print (A)
A.popleft()
print (A)
A.appendleft(100)
print (A)
```