Listas, Dicionários, Tuplas e Conjuntos

Lista

Listas são coleções heterogêneas de objetos, que podem ser de qualquer tipo, inclusive outras listas.

As listas no Python são **mutáveis**, podendo ser alteradas a qualquer momento. Listas podem ser fatiadas da mesma forma que as ***strings***, mas como as listas são mutáveis, é possível fazer atribuições a itens da lista.

```
lista = [a, b, ..., z]
```

```
#exemplo de listas
lista1=[1,2,3,4]
lista2=['python','java','c#']
lista3 = [1,2,'python', 3.5, 'java']
lista4 = list('python')
```

Vejamos um exemplo em que um aluno tem cinco notas e desejamos calcular a média aritmética dele:

```
#calculo da média
notas = [6,7,5,8,9]
soma=0
x=0
while x<5:
    soma += notas[x]
    x += 1
printf(f"Média: {soma/x:5.2f}")
```

```
#Cálculo da média com notas digitadas
notas = [0,0,0,0,0]
soma = 0
x=0
while x<5:
    notas[x] = float(input(f"Nota {x}: "))
    soma += notas[x]
    x +=1
x=0
while x<5:
    prinf(f"Nota{x}: {notas[x]:6.2f}")
    x += 1
printf(f"Média: {soma/x:5.2f}")
```

```
#Apresentação de números
numeros = [0,0,0,0,0]

x=0
while x<5:
    numeros[x] = int(input(f"Número {x+1}: "))
    x +=1
```

```
while True
    escolhido = int(input("Que posição você quer imprimir (0
    if escolhido == 0:
        break;
    print(f"Voce escolheu o número: {numeros[escolhido-1]}")
```

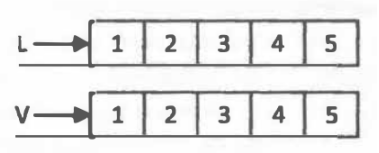
Uma lista em Python é um **objeto** e, quando atribuímos um objeto a outro, estamos apenas copiando a mesma referência da lista e, não seus dados em si. No caso abaixo, `V` funciona como um apelido de `L`, ou seja, `V` e `L` são a mesma lista.

```
L = [1,2,3,4,5]
V = L
V[0]
V
L
```

Quando modificamos `V[0]`, estamos modificando o mesmo valor de `L[0]`, pois ambos são **referências**, ou **apelidos** para a mesma lista na memória.

Para criar uma cópia **independente** de uma lista, utilizaremos outra sintaxe.

```
L = [1,2,3,4,5]
V = L[:]
V[0]=6
V
L
```



Podemos fatiar uma lista, da mesma forma que fizemos com strings.

```
L = [1,2,3,4,5]
L[0:5]
L[:5]
L[:-1]
L[1:3]
L[1:4]
L[3:]
L[-2]
```

Podemos usar a função `len()` com listas.

```
#Repetição do tamanho da lista usando len()
L=[1,2,3]
x=0
while x<len(L):
    print(L[x])
    x +=1
```

Adição de elementos da lista

Para adicionar um elemento ao fim da lista, utilizaremos o método `append()`. Em Python, chamamos um método escrevendo o nome dele após o nome do objeto.

```
#Adição de elementos à Lista
L=[]
while True:
    n=int(input("Digite um número (0 sai): "))
    if n==0:
        break
    L.append(n)
x=0
while x<len(L)
    print(L[x])
    x +=1
```

Outra forma de adicionarmos a lista;

```
L=[]
L = L+[1]
L = L + [3,4,5] # o compilador executa automaticamente o
# método extend(lista[])
L.extend([6,7,8])
```

Remoção de elementos da lista

Podemos retirar alguns elementos da lista ou todos eles. Utilizaremos a instrução **del**.

```
L=["a","b","c","d","e"]
del L[1]
L
del L[1:3]
L
```

Exemplo:

```
# uma lista de Bandas
bandas = ['Ivete Sangalo', 'Wesley Safadão', 'Pink Floyd', 'A

#varrendo a lista inteira
for banda in bandas:
    print banda

# trocando o último elemento
bandas[-1] = 'GG da Bahia'

# incluindo
bandas.append('AC/DF')

#removendo
bandas.remove('Welsey Safadão')

#ordena a lista
bandas.sort()

#ordena sem alterar seus elementos
sorted(bandas)

#inverte a lista
bandas.reverse()

#Imprime numerado
print(list(enumerate(bandas)))
for i, banda in enumerate(bandas):
    print(i+1, '=>', banda)

#Imprime do segundo item em diante
print bandas[1:]
```

Enumerate

A função **enumerate()** gera uma tupla em que o primeiro valor é o índice e o segundo é o elemento da lista sendo enumerada.

```
L=[5,9,13]
x=0
for e in L:
    print(f"[{x}] {e}")
    x+=1

L = [5,9,13]
for x,e in enumerate(L):
    print(f"[{x}] {e}")

#enumerate(L) vai gera tupla (0,5), (1,9), (2,13)
```

💡 A função `enumerate()` retorna uma **tupla** de dois elementos a cada iteração: um número sequencial e um item da sequência correspondente.

💡 As operações de ordenação (***sort***) e inversão (***reverse***) são realizadas na própria lista, sendo assim, não geram novas listas.

```
#Usando listas como filas
#O primeiro a chegar é o primeiro a sair (FIFO - First In Fir
lista = ['A', 'B', 'C']
print('lista:', lista)
```

```
# A lista vazia é avaliada como falsa
while lista:

    #em filas, o primeiro item é o primeiro a sair
    # pop(0) remove e retorna o primeiro item
    print('Saiu', lista.pop(0), ', faltam', len(lista))

#Usando listas como pilhas
#O último elemento a chegar é o primeiro a sair (LIFO - Last
#Mais itens na lista
lista +=['D', 'E', 'F']
print('lista: ', lista)

while lista:

    # em pilhas, o primeiro item é o último a sair
    # pop() remove e retorna o último item
    print('Saiu', lista.pop(), ', faltam', len(lista))
```

Exemplo de fila

```
#Simulação de uma fila de banco
ultimo = 10
fila= list(range(l, ultimo + 1))
while True:
    print(f"\nExistem {len(fila)} clientes na fila")
    print(f"Fila atual: {fila}")
    print( "Digite F para adicionar um cliente ao fim da fila
    print( "ou A para realizar o atendimento. S para sair.")
    operação= input("Operação (F, A ou S):")
    if operação== "A":
        if len(fila) > 0:
            atendido= fila.pop(0)
            print(f"Cliente {atendido} atendido")
```

```
        else:
            print("Fila vazia! Ninguém para atender.")
    elif operação== "F":
        ultimo += 1 # Incrementa o ticket do novo cliente
        fila.append(ultimo)
    elif operação == "S":
        break
    else:
        print( "Operação inválida! Digite apenas F, A ou S!")
```

Exemplo de pilha

```
#Pilha de pratos
prato= 5
pilha= list(range(l, prato+ 1))
while True:
    print(f"\nExistem {len(pilha)} pratos na pilha")
    print(f"Pilha atual: {pilha}")
    print("Digite Epara empilhar um novo prato,")
    print( "ou D para desempilhar. S para sair.")
    operação= input( "Operação (E, D ou S): ")
    if operação== "D":
        if len(pilha) > 0:
            lavado= pilha.pop(-1)
            print(f"Prato {lavado} lavado")
        else:
            print("Pilha vazia! Nada para lavar.")
    elif operação== "E":
        prato+= 1 # Novo prato
        pilha.append(prato)
    elif operação == "S":
        break
    else:
        print("Operação invãlida! Digite apenas E, D ou S!")
```

Exemplo: Fazendo uma procura

```
L=[7,9,10,12]
p = int(input("Digite um número a pesquisa:"))
for e in L:
    if e==p:
        print("Elemento encontrado!")
        break
else: #parecido com o da instrução else, a instrução while po
    print("Elemento não encontrado")
```

Listas dentro de listas

Vimos que strings podem ser indexadas ou acessadas letra por letra. Um fator interessante é que podemos acessar as strings dentro da lista, letra por letra, usando um segundo índice:

```
S = ["maçãs", "peras", "kiwis"]
print(S[0][0])
print(S[0][1])
print(S[1][1])
```

Isso nos leva a outra vantagem das listas em Python: listas dentro de listas. Temos também que os elementos de uma lista não precisam ser do mesmo tipo.

```
#Listas com elementos de tipos diferentes
produto1 = ["maçã", 10, 0.30]
produto2 = ["pera", 5, 0.75]
```

```
produto3 = ["kiwi", 4, 0.98]
compras = [produto1, produto2,produto3]
print(compras)
for e in compras:
    print(f"Produto: {e[0]} ")
    print(f"Quantidade: {e[1]}")
    print(f"Preço: {e[2]:5.2f}")
```

```
#Criação e impressão da lista de compras
compras=[]
while True:
    produto = input("Produto: ")
    if produto == "fim":
        break
    quantidade = int(input("Quantidade: "))
    preco = float(input("Preço: "))
    compras.append([produto, quantidade, preço])
soma=0.0
for e in compras:
    print(f"e[0]:20s x {e[1]:5d} {e[2]:5.2f}{e[1]*e[2]:6.2f}"
    soma += e[1]*e[2]
print(f"Total: {soma:7.2f}")
```

```
#exemplo de matriz

dim = 6,12
mat={}

mat[3, 7] = 3
mat[4, 6] = 5
mat[6, 3] = 7
mat[5, 4] = 6
mat[2, 9] = 4
mat[1, 0] = 9
```

```
for lin in range(dim[0]):
    for col in range(dim[1]):
        print(mat.get((lin, col),0))
    print
```

```
# Matriz em forma de string
matriz = '''
    0 0 0 0 0 0 0 0 0 0 0 0
    9 0 0 0 0 0 0 0 0 0 0 0
    0 0 0 0 0 0 0 0 0 4 0 0
    0 0 0 0 0 0 0 3 0 0 0 0
    0 0 0 0 0 0 5 0 0 0 0 0
    0 0 0 0 6 0 0 0 0 0 0 0
        '''

mat = {}

# Quebra a matriz em linhas
for lin, linha in enumerate(matriz.splitlines()):

    # Quebra a linha em colunas
    for col, coluna in enumerate(linha.split()):

        coluna = int(coluna)
        # Coloca a coluna no resultado,
        # se for diferente de zero
        if coluna:
            mat[lin, col] = coluna
            print(mat)

# Some um nas dimensões pois a contagem começa em zero
print('Tamanho da matriz completa:', (lin + 1) * (col + 1))
print('Tamanho da matriz esparsa:', len(mat))
```

Exercícios da Aula

1. Vamos tentar resolver alguns desafios. Dada a lista = [12, -2, 4, 8, 29, 45, 78, 36, -17, 2, 12, 8, 3, 3,-52] faça um programa que:
   a. imprima o maior elemento;
   b. imprima o menor elemento;
   c. imprima os números pares;
   d. imprima o número de ocorrências do primeiro elemento da lista;
   e. imprima a média dos elementos;
   f. imprima a soma dos elementos de valor negativo
2. Faça um jogo da Forca utilizando listas. Dada uma palavra, dê algumas chances para o usuário acertar.
3. Faça um programa que leia uma expressão com parênteses. Usando pilhas, verifique se os parênteses foram abertos e fechados na ordem correta.
   Exemplo:

   `(())` **OK**

   `()()(()())` **OK**

   `( ) )` Erro

   Você pode adicionar elementos à pilha sempre que encontrar abre parênteses e desempilhá-la a cada fecha parênteses. Ao desempilhar, verifique se o topo da pilha é um abre parênteses. Se a expressão estiver correta, sua pilha estará vazia no final.
4. A lista de temperaturas de Mons, na Bélgica, foi armazenada na lista T= [-10, -8, 0, 1, 2, 5, -2,-4]. Faça um programa que imprima a menor e a maior temperatura, assim como a temperatura média.
5. Escreva um programa que copie os valores pares para uma lista e os valores ímpares para outra lista. A lista inicialmente de valores é `V=[9,8,7,12,0,13,21]` .
6. Escreva um programa que controla a utilização das salas de um cinema. Imagine que a lista lugares_vagos=[10,2,1,3,0] contenha o número de lugares vagos nas salas 1,2,3,4 e 5, respectivamente. Esse programa lerá o número da sala e a quantidade de lugares solicitados. Ele deve informar se

é possível vender o número de lugares solicitados, ou seja, se ainda há lugares livres. Caso seja, possível vender os bilhetes, atualizará o número de lugares livres. A saída ocorre quando se digita 0 no número da sala.

7. Escreva um **jogo da velha** para dois jogadores. O jogo deve perguntar onde você quer jogar e alternar entre os jogadores. A cada jogada, verifique se a posição está livre. Verifique também quando um jogador venceu a partida. Um jogo da velha pode ser visto como uma lista de 3 elementos, na qual cada elemento é outra lista também com três elementos.