Listas

As listas são estruturas de dados que permitem armazenar um coleção de itens de forma dinâmica, ou seja, o número máximo de itens não necessita de estar definido no momento de criação da lista. Os elementos da lista pode ser de diferentes tipos, incluindo outras listas.

As listas são mutáveis sendo guardadas na memória como uma tabela de referências para os elementos que compõem a lista.

Definir uma lista

In [ ]:
```
#lista de números
minha_lista=[1,2,3]
#listas vazias
lista_vazia=[]
lista_vazia2=list()
#lista de strings
lista_nomes=["joaquim","maria","antónio"]
#lista com vários tipos de dados
lista_mista=[1,"joaquim",["outra","lista"]]
print(lista_mista)
lista_de_cinco=list(range(5))
print(lista_de_cinco)
```

In [ ]:
```
print(id(minha_lista))
print(type(minha_lista))
```

As listas são referências para posições de memória

In [ ]:
```
listaA=[1,2,3]
listaB=listaA #não faz uma cópia da lista
listaB[1]=4 #altera o elemento na posição 1 das duas listas
print(listaA)
print(id(listaA))
print(id(listaB))
```

Fazer uma cópia de uma lista

In [ ]:
```
listaA=[1,2,3]
listaB=listaA[:]    #faz uma cópia
listaB[1]=4 #só altera na listaB
print(listaA)
print(listaB)
print(id(listaA))
print(id(listaB))
```

Esta cópia não funciona com todos os tipos de listas pois só copia as referências de primeiro nível. Isto significa que se a lista tiver outras listas isto não funciona. Para estas

situações é necessário fazer uma cópia profunda **(deepcopy)** para isso existe uma função: **copy.deepcopy()**

Listar elementos

In [ ]:
```
print(minha_lista[1])

print(lista_nomes[0])

print(lista_mista[2])

print(lista_mista[2][1])
```

Listar todos

In [ ]:
```
for item in minha_lista:
    print(item)
```

In [ ]:
```
for posicao in range(len(lista_nomes)):
    print(lista_nomes[posicao])
```

Adicionar elementos a uma lista

Método **append()** adiciona um item no final da lista

In [ ]:
```
lista_cores=["vermelho","verde","azul"]

lista_cores.append("amarelo")

print(lista_cores)
```

Método **insert()** insere um item numa posição específica:

In [ ]:
```
lista_cidades=["Viseu","Tondela","Mangualde"]
lista_cidades.insert(0,"Lisboa")
print(lista_cidades)
```

In [ ]:
```
lista_cidades.insert(10,"Porto")
print(lista_cidades)
```

Método **extend()** permite adicionar uma lista no final de outra lista

In [ ]:
```
lista_frutas=["maçã","laranja","morango"]
lista_arvores=["castanheiro","pinheiro"]

lista_frutas.extend(lista_arvores)
print(lista_frutas)
lista_frutas.extend(["pêra","pereira"])
print(lista_frutas)
del lista_arvores
print(lista_arvores)
```

In [ ]:
```
lista = lista_frutas + lista_arvores
print(lista)
```

Remover elementos de uma lista

Método **remove()** remove o primeiro item da lista cujo valor é igual ao indicado

In [ ]:
```
lista_frutas=["maçã","laranja","morango","pêra"]

print(lista_frutas)
lista_frutas.remove("laranja")
print(lista_frutas)
```

Método **pop()** remove e devolve um item de uma posição específica da lista.

In [ ]:
```
lista_marcas=["ford","ferrari","bmw"]
meu_carro=lista_marcas.pop(1)
print(meu_carro)
print(lista_marcas)
```

Instrução **del** pode ser utilizada para remover um item ou vários

In [ ]:
```
lista_marcas=["ford","ferrari","bmw","vw"]
del lista_marcas[1:3] #remove os elementos nas posições 1 e 2
print(lista_marcas)
```

Apagar todos os elementos da lista

In [ ]:
```
lista_marcas=["ford","ferrari","bmw","vw"]
lista_marcas.clear()
print(lista_marcas)
```

In [ ]:
```
lista_marcas=["ford","ferrari","bmw","vw"]
lista_marcas=[]
print(lista_marcas)
```

Repetição

In [ ]:
```
lista_marcas = ["ford","ferrari","bmw","vw"]
print(lista_marcas*2)
```

Pesquisar elementos

Método **index()** devolve a posição do elemento indicado

In [ ]:
```
lista_marcas=["ford","ferrari","bmw","vw"]
meu_carro=lista_marcas.index("vw")
print(meu_carro)
meu_carro=lista_marcas.index("renault") #dá erro porque não existe
print(meu_carro)
```

Testar se um elemento existe na lista com o operador **in**

In [ ]:
```
lista_marcas=["ford","ferrari","bmw","vw"]
meu_carro="renault"

if meu_carro in lista_marcas:
    print("Existe")
else:
    print("Não existe")
```

Contar ocorrências

In [ ]:
```
marcas=["asus","hp","apple","asus","acer"]
print(marcas.count("asus"))
print(marcas.count("samsung"))
```

Escolher um elemento de forma aleatória

In [ ]:
```
import random

lista = ["ford","ferrari","bmw","vw","fiat","renault"]

meu_carro = random.choice(lista)
print(meu_carro)
```

In [ ]:
```
import random

outro_carro = lista[random.randint(0,len(lista)-1)]
print(outro_carro)
```

Fatiamento

In [ ]:
```
lista = ["ford","ferrari","bmw","vw","fiat","renault"]
#posições: 0 e 1
meus_carros=lista[:2]
print(meus_carros)
#posições: 1 até ao final
outros_carros=lista[1:]
print(outros_carros)
#posições: todas por ordem invertida
print(lista[::-1])
```

Igualdade

Para testar se duas listas são iguais é possível utilizar o operador ==, mas os elementos da lista têm de existir na mesma posição.

In [ ]:
```
lista_a = [1,2,3,4]
lista_b = [1,2,3,4]
lista_c = [1,3,2,4]

#True
print(lista_a == lista_b)

#False
print(lista_a == lista_c)
```

```
#Ordenar antes de comparar
print(sorted(lista_a) == sorted(lista_c))
```

Criar uma lista com dados filtrados ou alterados de outra lista

O Python permite criar listas utilizando um método denominado **lista por compreensão** que possibilita escrever programas mais compactos.

A forma mais simples de uma lista por compreensão é: **[<expressão> for in <iterável>]**

In [ ]:
```
import random

#uma função que devolve uma lista de n números aleatórios
def gera_lista(n):
    return [random.randint(1,100) for i in range(n) ]

print(gera_lista(5))
```

In [ ]:
```
numeros=[1,2,3,4,5,6,7,8,9,10]

dobro = [n*2 for n in numeros]

print(dobro)
```

Uma lista por compreensão com filtro

In [ ]:
```
lista_marcas=["ford","ferrari","bmw","vw"]

#lista dos carros cujo nome começa por f
lista_marcas_f=[elemento for elemento in lista_marcas if elemento.startswith("f"

print(lista_marcas_f)
```

In [ ]:
```
numeros=[1,2,3,4,5,6,7,8,9,10]

numeros_pares=[x for x in numeros if x%2==0]
print(numeros_pares)
```

In [ ]:
```
palavras=["maçã", "sol", "lua", "estrela", "céu"]
palavras_longas=[palavra for palavra in palavras if len(palavra)>3]
print(palavras_longas)
```

Listas e funções

As listas são passadas por referência para as funções, isto significa que a função recebe a lista e não uma cópia e todas as alterações realizadas dentro da função afetam a lista que foi passada para chamar a função.

In [ ]:
```
lista=["um","dois","três","quatro"]

def RemoveUmElemento(elementos,elemento_remover):
```

```
    elementos.remove(elemento_remover)

print(lista)
RemoveUmElemento(lista,"três")
print(lista)
```

Converter um dicionário numa lista

In [ ]:
```
meu_dicionario = {'a':1,'b':2,'c':3,'d':4}

#converter o dicionário numa lista de tuplos
#cada elemento da lista é um tuplo com a chave e o valor
minha_lista = list(meu_dicionario.items())
print(minha_lista)
```

Converter duas listas em um dicionário

A função **zip** cria uma lista de tuplos com os elementos de duas listas combinados pela ordem: [(elemento1_listaA,elemento1_listaB),(elemento2_listaA,elemento2_listaB),...] A função **dict** converte a lista de tuplos num dicionário em que o primeiro elemento do tuplo é a chave e o segundo elemento é o valor.

In [ ]:
```
alunos = ['Ana', 'Joaquim', 'Carla']
notas = [8, 9, 7]
idades = dict(zip(alunos, notas))
print(idades)
```

Função zip

In [ ]:
```
lista1 = [1, 2, 3]
lista2 = ['a', 'b', 'c']
resultado = zip(lista1, lista2)
print(list(resultado))
```

In [ ]:
```
for numero, letra in zip(lista1, lista2):
    print(f'{numero}: {letra}')
```

Converter uma string numa lista

In [ ]:
```
meus_numeros = "10 5 12 -4 0 11"
minha_lista=meus_numeros.split(" ")
print(minha_lista)
print(minha_lista[2])
```

Criar uma lista de dicionários

As listas podem conter qualquer tipo de dados, cada elemento pode ser um inteiro, uma string ou até outra lista, array ou dicionário. Com dicionários e listas podemos guardar muitos dados de forma mais estruturada.

In [ ]:
```
lista_alunos=[]
aluno_a={'nome':'joaquim','turma':'10ºT','processo':'12345'}
aluno_b={'nome':'maria','turma':'10ºT','processo':'54321'}
aluno_c={'nome':'antónio','turma':'10ºT','processo':'12300'}
lista_alunos.append(aluno_a)
lista_alunos.append(aluno_b)
lista_alunos.append(aluno_c)

#mostrar o nome do primeiro aluno
print(lista_alunos[0]['nome'])

#adicionar um aluno com dados do utilizador
nome=input("Nome do aluno:")
turma=input("Turma do aluno:")
processo=input("Processo do aluno:")
aluno_novo={'nome':nome,'turma':turma,'processo':processo}
lista_alunos.append(aluno_novo)

for aluno in lista_alunos:
    print(aluno['nome'])
```

Ordenação

In [ ]:
```
lista_numeros=[33,-1,4,5,1]
lista_ordenada=sorted(lista_numeros)
print(lista_ordenada)
```

In [ ]:
```
lista_numeros=[33,-1,4,5,6,1]
lista_numeros.sort()
print(lista_numeros)
```

In [ ]:
```
nomes=["maria","joaquim","ana","carla"]
nomes_ordenados=sorted(nomes)
print(nomes_ordenados)
```

In [ ]:
```
nomes=["maria","joaquim","ana","carla"]
nomes.sort()
print(nomes)
```