|  |  |
| --- | --- |
|  | **Dicionário** |
| Mutável | Sim |
| Definir | Dicionário = {‘chave’:’ valor’}, aluno = {“nome”: “Joaquim”,” idade”:”15”} |
| Mostrar um | Dicionário[‘chave’]  aluno[‘nome’] – dá erro se a chave não existir  aluno.get (‘morada’, ‘Não tem morada definida’) – não dá erro se não existir a chave |
| Adicionar | Dicionário [‘nova chave’] = ‘valor pretendido’  aluno[‘morada’] = ‘Viseu’ |
| Remover | del [dicionário[‘chave’]]  del [aluno]’morada’] – dá erro se a chave não existir  valor = dicionário.pop (‘chave’, None)  valor = alunos.pop (‘morada’, None) – se a chave morada não existir a variável morada fica com None |
| Editar | Dicionário = [‘chave’] = ‘novo valor’  Aluno = [‘idade’] = 16 |
| Listar | for chave, valor in dicionário:  print (f” {chave}: {valor}  **ou**  for chave in dicionário:  print (f” {chave}: {dicionário[chave]} |
| Testar se existe  (chave, valor) | If ‘morada’ in aluno:  print (“O aluno tem o campo morada”)  **ou**  (valor)  Valores = alunos.values()  If ‘vise’ in valores:  print (“O dicionário tem o valor Viseu”) |
| Igualdade | Operador ==  A ordem dos elementos não precisa de ser a mesma |
| Funções | dict () – cria/converte um dicionário  keys () – lista de chaves do dicionário  values () - lista de valores do dicionário |
| Notas | Dá para juntar dois dicionários com o operador |  A chave tem que ser um valor imutável |

|  |  |
| --- | --- |
|  | **Conjunto (Set)** |
| Mutável | Sim |
| Definir | Conjunto = {1,2,3,4,5}  Nomes = {“Maria”,” Joaquim”, “António”} |
| Mostrar um | Não é possível indexar  Mostrar o 3º elemento  For pos in enumerate(conjunto,0)  If pos == 2:  Print(valor) |
| Adicionar | Conjunto.add (10)  Nomes.add (“marco”) |
| Remover | Conjunto.remove(3) – remove o número do conjunto  Nomes.remove(‘Maria’) – remove o nome do conjunto |
| Editar | Primeiro remover o elemento e depois adicionar o novo |
| Listar | for pos, valor in enumerate(nomes,0):  print (f” {pos} – {valor} |
| Testar se existe | Operador in  If ‘Joaquim’ in nomes:  print (‘O conjunto tem um Joaquim’) |
| Igualdade | Operador ==  A ordem dos elementos não interessa |
| Funções | União – resultado = conjunto.union(outro\_conjunto)  Interseção – resultado= conjunto.intersection(outro\_conjunto)  Diferença – resultado = conjunto.difference(outro\_conjunto)  Diferença simétrica -resultado = conjunto.symetric(outro\_conjunto) |
| Notas | Coleção não ordenada, que não garante a posição dos elementos, todos os elementos têm de ser imutáveis e não repetidos(ignorado) |

|  |  |
| --- | --- |
|  | **Tuplo** |
| Mutável | Não |
| Definir | Tuple = (1,2,3,4)  Tuple\_2 = 1,2,3,4 |
| Mostrar um | print (tuple [0]) |
| Adicionar | Não é possível |
| Remover | Não é possível |
| Editar | Não é possível |
| Listar | for valor in tuple:  print(valor)  **ou**  for i in range (tuple):  print(tuple[i]) |
| Testar se existe | If 2 in tuple:  print (“O valor existe”) |
| Igualdade | Operador ==  A ordem dos elementos interessa |
| Funções | max – devolve o valor máximo – max(tuple)  min – devolve o valor mínimo – min(tuple)  sum – devolve a soma de todos os valores – sum(tuple)  count () – devolve o nr de ocorrências – tuple.count(2)  operador + para juntar dois tuples – tuple + tuple\_2 |
| Notas | São imutáveis, ou seja, depois de criado não pode ser alterado  Pode ter valores repetidos |

|  |  |
| --- | --- |
|  | **Lista** |
| Mutável | Sim |
| Definir | Lista = [1,2,3,4,5,6,7] |
| Mostrar um | print (lista2[]) |
| Adicionar | Lista.append (10) |
| Remover | Lsiat.remove (3) |
| Editar | Lista [2] = 10 |
| Listar | For valor in lista:  Print(valor)  **Ou**  For i in range(len(lista)):  Print(lista[i]) |
| Testar se existe | If 2 in lista:  print (“O valor pretendido está na lista”) |
| Igualdade | Operador ==  Ordem dos elementos interessa |
| Funções | As mesmas funções usados nos tuples  Slicing |
| Notas | Podem ter elementos mutáveis(dicionários, listas, tuples, arryas) |