|  |  |
| --- | --- |
|  | **Dicionário** |
| Mutável | Sim |
| Definir | Dicionário = {chave:”valor”} EX: aluno = {“nome”:”Joaquim”,”idade”:15} |
| Mostrar um | Dicionário[“chave”]  EX: Aluno[“nome”] – dá erro caso a chave não exista  Aluno.get(“morada”,”Não tem morada defenida) – assim não dá erro caso a chave não exista |
| Adicionar | Dicionário[“nova chave”] = “valor”  EX: Aluno[“morada”] = “Viseu” |
| Remover | Del dicionário[“chave”]  EX: Del aluno[“morada”] – dá erro se não existir  Valor = dicionário.pop(“morada”,None)  Morada = aluno.pop(“morada”None) – se não existir a chave morada a variável morada fica com None |
| Editar | Dicionario[“chave”] = novo valor  EX: Aluno[“idade”] = 16 |
| Listar | For chave, valor **in** Dicionario.itens():  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 morada”)  (valor)  Valores = aluno.values()  If “viseu” **in** valores:  Print(“O dicionário tem o valor viseu”) |
| Igualdade | Operador ==  A ordem dos elementos não interessa |
| Funções | Dict() = cria/converte um dicionário  Keys() = lista das chaves do dicionário  Values() = lista dos valores do dicionário  Update() = junta dois dicionários |
| Notas | Para juntar dois dicionários com o operador |  A chave tem de ser um valor imutável(string,bool,int) |

|  |  |
| --- | --- |
|  | **Conjunto (Set)** |
| Mutável | Sim |
| Definir | Conjunto = {1,2,3,4}  EX: Nomes = {“Joaquim”,”maria”,”antonio”} |
| Mostrar um | Não é possivel indexar  EX: Mostrar o 3 elemento  For pos, valor **in** enumerate(conjunto,0):  If pos == 2  Print(valor)  break |
| Adicionar | Conjunto.add(10)  EX: Nomes.add(“mais um”) |
| Remover | Conjunto.remove(3) = remove o valor 3  Conjunto.remove(“Joaquim”) – remove o nome Joaquim 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 importa |
| Funções | União: resultado = conjunto.union(outro\_conjunto)  Interseção: resultado = conjunto.intersection(outro\_conjunto)  Diferença: resultado = conjunto.difference(outro\_conjunto)  Diferença simétrica: resultado = conjunto.symmetric\_difference(outro\_conjunto) |
| Notas | Coleção não ordenada, que não garante as posições dos elementos, todos os elementos têm de ser imutáveis e não repetidos(ignorados) |

|  |  |
| --- | --- |
|  | **Tuplo** |
| Mutável | Não |
| Definir | Meu\_tuple = 1,2,3  Outro\_tuple = (1,2,3,4) |
| Mostrar um | Print(meu\_tuple[0]) |
| Adicionar | Não se aplica |
| Remover | Não se aplica |
| Editar | Não se aplica |
| Listar | For valor **in** Meu\_tuple:  Print(valor)  Ou  For i **in** range(len(Meu\_tuple)):  Print(Meu\_tuple[i] |
| Testar se existe | If 2 **in** Meu\_tuple:  Print(“Meu tuple tem um 2”) |
| Igualdade | Operador ==  A ordem dos elementos interessa |
| Funções | Max – devolver o maior – Max(Meu\_tuple)  Min – devolver o menor – Min(Meu\_tuple)  Count() – contar o nº de ocorrências – Meu\_tuple.count(2)  Sum – devolve a soma de todos os valores – sum(Meu\_tuple)  Operador + para juntar dois tuples – resultados = tupleA + tupleB |
| Notas | São imutáveis, ou seja, depois de criado não pode ser alterado  Poder ter valores repetidos |

|  |  |
| --- | --- |
|  | **Lista** |
| Mutável | Sim |
| Definir | Minha\_lista = [1,2,3,4,5,6] |
| Mostrar um | Print(Minha\_lista[2]) |
| Adicionar | Minha\_lista.append(10) |
| Remover | Minha\_lista.remove(10) |
| Editar | Minha\_lista[2] = 10 |
| Listar | For valor **in** Minha\_lista:  Print(valor)  Ou  For i in range(len(Minha\_lista)):  Print(Minha\_lista[i]) |
| Testar se existe | Operador in  If 2 in Minha\_lista:  Print(“Existe o 2”) |
| Igualdade | Operador ==  Ordem dos elementos interessa |
| Funções | As mesmas dos tuples  slicing |
| Notas | Podem ter elementos mutáveis (dicionários,tuples,listas,arrays) |