ESTRUTURAS DE DADOS

Autoria de Carolina Soares

ESTRUTURAS DE DADOS

O que veremos:

- Listas
- Tuplas
- Sets
- Dicionários

CARACTERÍSTICAS DAS ESTRUTURAS

Estrutura	Ordenado	Mutável	Construtor	Exemplo
`list`	sim	sim	`[]` ou `list()`	`[1, 2.5, 'Pedro', True]`
`tuple`	sim	não	`()` ou `tuple()`	`(1, 2.5, 'Pedro', True)`
`set`	não	sim	`{}` ou `set()`	`{1, 2.5, 'Pedro', True}`
`dict`	sim	sim	`{}` ou `dict()`	`{6: 'jun', 7: 'jul'}`

https://phylos.net/2021-03-08/python-sequencias-e-colecoes

ARMAZENANDO ITENS EM LISTAS

Uma lista é uma coleção ordenada de valores, separados por vírgula e dentro de colchetes [], elas são utilizadas para armazenar diversos itens em uma única variável.

```
list = []
print(type(list))
```

Saída do código:

```
<class 'list'>
```

CRIANDO LISTAS COM O RANGE

```
list(range(10))
```

Saída: `[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]`

CRIANDO LISTAS DE STRINGS

```
lista = ['uva', 'maçã', 'pêra', 'morango']
print(lista)
```

Saída: `['uva', 'maçã', 'pêra', 'morango']`

CRIANDO LISTAS MISTAS

```
lista = ['abc', 0.5, True, 90]
print(lista)
```

Saída: `['abc', 0.5, True, 90]`

VERIFICANDO TAMANHO DA LISTA

```
lista = ['abc', 0.5, True, 90]

print(len(lista))
```

Saída: 4

ACESSANDO UM ITEM NA LISTA

```
lista = ['uva', 'maçã', 'pêra', 'morango']
print(lista[0])
```

Saída: uva

```
print(lista[2])
```

Saída: pêra

```
print(lista[-1])
```

Saída: morango

LISTA DENTRO DA LISTA

```
usuarios = [
    ["Maria", 18],
    ["Roberto", 20],
    ["Laura", 25],
    ["Carlos", 19]
] # lista de usuários que guarda
uma lista de nomes e idades.
```

ACESSANDO UM ITEM DENTRO DE OUTRA LISTA

```
lista = [[2, 3, 5], [1, 2], [9, 6, 9]]

print(lista[1][0])
```

Saída: 1

FATIAMENTO DE LISTA

```
lista = [2, 4, 12, 20, 34, 1, 9]
lista[ inicio : fim : passo ]
```

FATIANDO UMA LISTA

```
lista = [2, 4, 12, 20, 34, 1, 9]
```

```
print(lista[2:5])
```

Saída: [12, 20, 34]

```
print(lista[:5])
```

Saída: [2, 4, 12, 20, 34]

```
print(lista[5:])
```

Saída: [1, 9]

```
print(lista[-3:6])
```

Saída: [34, 1]

```
print(lista[:5:2])
```

Saída: [2, 12, 34]

PERCORRENDO UMA LISTA

```
lista = ["uva", "maçã", "pêra", "morango"]

for l in lista:
    print(l)
```

Saída do código:

```
uva
maçã
pêra
morango
```

VERIFICANDO SE UM ITEM EXISTE

```
lista = ["uva", "maçã", "pêra", "morango"]

"pêra" in lista
```

Saída: True

```
"pêra" not in lista
```

Saída: False

VERIFICANDO SE UM ITEM EXISTE

```
lista = ["uva", "maçã", "pêra", "morango"]

if "pêra" in lista:
    print("Pêra existe na lista!")
else:
    print("Pêra NÃO existe na lista!")
```

Saída: Pêra existe na lista!

MÉTODOS DAS LISTAS

- `lista.append("valor1") # adiciona um valor`
- `lista.clear() # limpar lista`
- `lista.copy()  # copiar lista`
- `lista.count("valor1") # contar quantos elementos tem na lista com esse valor`
- `lista.extend(lista2) # adiciona no final da lista os elementos de outra lista`
- `lista.index("valor1") # retorna o índice onde o valor especificado estar`
- `lista.insert(2, "valor2") # insere elemento em posição especificada`
- `lista.pop() # remove e retorna elemento em posição especificada (se nenhum elemento é passado é removido o último elemento)`
- `lista.remove("valor2") # remove elemento especificado da lista`
- `lista.reverse() # inverte a ordem da lista`
- `lista.sort() # ordena a lista`

ARMAZENANDO ITENS EM TUPLAS

As tuplas funcionam da mesma maneira que as listas, mas uma vez criadas elas não podem ser mais mudadas.

```
tupla = ()
print(type(tupla))
```

Saída do código:

```
<class 'tuple'>
```

CRIANDO TUPLAS

```
pontoXY = (-2, 5)

pontoXYZ = (-1, 0.5, 3)

usuarioId = ("143535", "Maria")

produto = ("Camisa", "P", 56.00, True)
```

ACESSANDO UMA TUPLA

```
tupla = (2, 5, 3)
print(tupla[1])
```

Saída: 5

MÉTODOS DAS TUPLAS

- `tupla.count("valor1") # contar quantos elementos tem na lista com esse valor`
- `tupla.index("valor1") # retorna o índice onde o valor especificado estar`

ARMAZENANDO ITENS EM SETS

Os sets são conjuntos criados para armazenar vários itens, mas diferente das listas e tuplas, os sets são desordenados, parcialmente imutáveis, não possuem itens duplicados e não podem ser indexados.

```
conjunto = set()
print(type(conjunto))
```

Saída do código:

```
<class 'set'>
```

CRIANDO E OPERANDO COM SETS

```
x = {'a', 'f', 'c', 'j'}
y = {'c', 'a', 'f', 'd'}
uniao = x | y # ou x.union(y)
print(uniao)
Saída: {'c', 'd', 'f', 'a', 'j'}
diferenca = x - y  # ou x.symmetric_difference_update(y)
print(diferenca)
Saída: {'j'}
intercessao = x & y # ou x.intersection_update(y)
print(intercessao)
Saída: {'a', 'c', 'f'}
```

MÉTODOS DOS CONJUNTOS

- `conjunto.add("valor1") # insere elemento no set`
- `conjunto.clear() # remove todos os elementos do set`
- `conjunto.copy() # retorna cópia do set`
- `conjunto.difference(conjunto1) # retorna um set com a diferença entre 2 ou mais sets`
- `conjunto.difference_update(conjunto1) # remove elementos incluídos no segundo set`
- `conjunto.discard("valor2") # remove item especificado`
- `conjunto.intersection(conjunto1) # retorna o set interseção de 2 sets`
- `conjunto.intersection_update(conjunto1) # remove itens do set não presentes no segundo set especificado`
- `conjunto.isdisjoint(conjunto1) # retorna True se os 2 sets são disjuntos`
- `conjunto.issubset(conjunto1) # retorna True se o set é subconjunto do segundo set`
- `conjunto.issuperset(conjunto1) # retorna True se o set contém o segundo set`
- `conjunto.pop() # remove (e retorna) um elemento arbitrário do set`
- `conjunto.remove("valor3") # remove o elemento especificado`
- `conjunto.symmetric_difference(conjunto1) # retorna o set com a diferença simétrica de dois sets`
- `conjunto.symmetric_difference_update(conjunto1) # insere a diferença simétrica desse set em outro`
- `conjunto.union(conjunto1) # retorna um set com a união dos sets`
- `conjunto.update("valor1") # atualiza o primeiro set com sua união com um ou mais sets`

ARMAZENANDO ITENS EM DICIONÁRIOS

Os dicionários são usados para armazenar elementos com uma chave única, seguida de um valor, que pode ser do tipo lista, dicionário, inteiro, string e etc.

```
dicionario = {"chave": "valor"}
```

CRIANDO UM DICIONÁRIO

```
dicionario = {
    "nome": "maria",
    "idade": 22,
    "sexo": "feminino"
} print(dicionario)
```

Saída: `{'nome': 'maria', 'idade': 22, 'sexo': 'feminino'}`

ACESSANDO UM ITEM NO DICIONÁRIO

```
dicionario = {"nome": "maria", "idade": 22, "sexo": "feminino"}

print(dicionario["nome"])
```

Saída: 'maria'

VERIFICANDO TAMANHO DO DICIONÁRIO

```
dicionario = {"nome": "maria", "idade": 22, "sexo": "feminino"}

print(len(dicionario))
```

Saída: 3

PERCORRENDO UM DICIONÁRIO

```
livros = {1: ["Dom Casmurro", "Machado de Assis"], 2: ["1984", "George Orwell"], 3: ["O Pequeno Príncipe", "Antoine de Saint-Exupéry"]}


for id in livros:
    print(f"Id: {id} | Livro: {livros[id]}")
```

Saída: Id: 1 | Livro: ['Dom Casmurro', 'Machado de Assis']

Saída: Id: 2 | Livro: ['1984', 'George Orwell']

Saída: Id: 3 | Livro: ['O Pequeno Príncipe', 'Antoine de Saint-Exupéry']

MÉTODOS DOS DICIONÁRIOS

- `dicionario.clear() # remove todos os elementos do dicionário`
- `dicionario.copy() # retorna uma cópia do dicionário`
- `dicionario.fromkeys("chave") # retorna dicionário com chaves e valores especificados`
- `dicionario.get("chave") # retorna o valor relativo à chave dada, ou valor default dado`
- `dicionario.items() # retorna uma lista contendo uma tupla para cada par chave:valor`
- `dicionario.keys() # retorna objeto iterável com as chaves do dicionário`
- `dicionario.pop() # remove o elemento relativo à chave especificada`
- `dicionario.popitem() # remove o último par chave:valor inserido`
- `dicionario.setdefault("chave") # retorna o valor relativo à chave dada. Se a chave não existe insere chave:valor`
- `dicionario.update({"chave": "valor"}) # Atualiza o dicionário com pares chave:valor dados`
- `dicionario.values() # retorna objeto iterável os valores do dicionário`

LISTAS E TUPLAS EM UM DICIONÁRIO

DICIONÁRIO COM TUPLA NA CHAVE

```
livros = {
    ("Dom Casmurro", "Machado de Assis"): 1,
    ("1984", "George Orwell"): 2,
    ("O Pequeno Príncipe", "Antoine de Saint-Exupéry"): 3
} print(livros)
```

Saída: `{('Dom Casmurro', 'Machado de Assis'): 1, ('1984', 'George Orwell'): 2, ('O Pequeno Príncipe', 'Antoine de Saint-Exupéry'): 3}`

DICIONÁRIO COM LISTA NO VALOR

```
livros = {
    1: ["Dom Casmurro", "Machado de Assis"],
    2: ["1984", "George Orwell"],
    3: ["O Pequeno Príncipe", "Antoine de Saint-Exupéry"]
} print(livros)
```

Saída: {1: ['Dom Casmurro', 'Machado de Assis'], 2: ['1984', 'George Orwell'], 3: ['O Pequeno Príncipe'], 4: ['Antoine de Saint-Exupéry']}

EXERCÍCIOS

1. Solicite 3 notas de um aluno e guarde em uma lista. Em seguida, mostre qual foi a maior nota da lista acessando ela e depois exiba a média entre as três notas.

2. Crie um programa que calcula a distância entre dois pontos, para isso crie duas tuplas para o ponto a e b. O ponto a irá receber valores para xa e ya e o ponto b receberá valores para xb e yb, cada valor será solicitado ao usuário. Em seguida, crie uma variável para calcular a distância entre os pontos das tuplas, use a fórmula: $\sqrt{((Xb - Xa)^2 + (Yb - Ya)^2)}$ e ao final mostre o resultado do cálculo. Lembre-se que para obter a raiz quadrática de um número eleve a potência de $1/2$.

3. Crie dois conjuntos e insira no primeiro conjunto números de 50 a 70 e no segundo conjunto insira números de 65 a 80, utilize o método range para simplificar a inserção dos números nos conjuntos. Logo, crie três variáveis, uma que irá mostrar a união entre esses conjuntos, outra que mostra a diferença entre os conjuntos e a última que exibe a interseção dos conjuntos.

4. Faça um programa que cria um dicionário de avaliação de linguagens de programação, onde conterá os elementos Python, JavaScript, C++, Java e PHP como chaves do dicionário. E o valor de cada elemento será solicitado para o usuário digitar uma avaliação de 0 a 5 para as linguagens. Em seguida, mostre qual foi a linguagem com melhor e pior avaliação do dicionário.

EXERCÍCIOS

5. Crie uma lista de pessoas com sub-listas que contenham nome, idade e estado civil de 5 pessoas. Utilize um loop para perguntar essas informações 5 vezes. Depois, pegue cada idade e mostre quem é o mais velho da lista e mostre a média entre todas as idades. Também mostre a quantidade de pessoas com estado civil, solteiro e casado.