DTECH

Módulo 8

Tuplas, Dicionários e Conjuntos

Algoritmos e Estruturas de Dados I

Python

(Rone Ilídio)

Conteúdo

- Tuplas
- Dicionários
- Conjuntos
- Passagem por valor e por referência
- Tratamento de exceção com tipos composto

Tipos Compostos (Revisão)

- Lista
  - Sequência de elementos acessados por índices
  - A ordem é importante
  - Declarada com [] ou list()
- Tupla
  - Semelhante a lista, mas não pode ser alterada após sua criação
  - Declarada com () ou uma sequência separada por ','

Tipos Compostos (Revisão)

- Conjunto
  - A ordem não faz diferença, não possui índices
  - Não possui elementos repetidos
  - Declarados por {} ou set()
- Dicionário
  - Mapeamentos no padrão chave:valor
  - A chave faz a função do índice nas listas

Tuplas

Tuplas

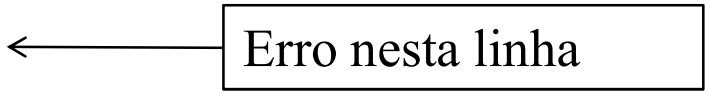
- Semelhantes a listas, mas são imutáveis
- Exemplo 1 (acessa e exibe cada elemento de uma tupla)

```
tupla = 1,2,3,4,5  #ou tupla = (1,2,3,4,5)
for i in range(5):
    print(tupla[i], end=' ')
print('\n',tupla)
```

Tupla

- O exemplo a seguir gera um erro
- Tuplas não podem ser alteradas
- Exemplo 2 (erro)

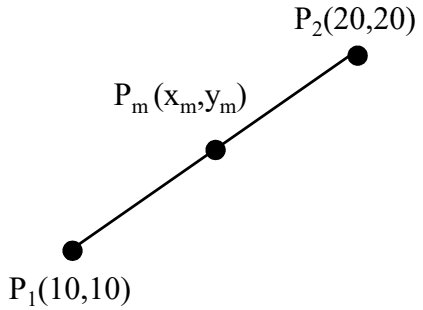
```
tupla = (1,2,3,4,5)
tupla[0] = 10
for i in range(5):
    print(tupla[i], end=' ')
```



Tupla como retorno de função

- Ocorre em funções que retornam mais de um valor
- Exemplo: ponto médio do seguimento de reta $P_1(10,10)$ / $P_2(20,20)$

```
def pontomedio(x1,y1,x2,y2):
    xm = (x1+x2)/2
    ym = (y1+y2)/2
    return xm,ym
pmedio = pontomedio(10,10,20,20)
print('Ponto médio:',pmedio[0],',',pmedio[1])
```



Dicionários

Dicionários

- São estruturas de dados cujos elementos são formados por chave:valor
- Semelhante a uma lista, mas o índice não existe, ele é substituído por um valor
- Pode ser criado com a utilização de {} ou pelo construtor dict()

- Obs: trataremos construtores posteriormente

Dicionários

- Exemplo: cria um dicionário simples e exibe na tela

```
populacao = {"Belo Horizonte":2000000,
  "Ouro Branco": 60000, "Lafaiete":120000}
print(populacao)
```

Dicionários

- Exemplo: exibe apenas um elemento

```
populacao = {"Belo Horizonte":2000000,
  "Ouro Branco": 60000, "Lafaiete":120000}
print(populacao["Lafaiete"])
```

Dicionários

- Exemplo: exibe todas as chave e valores separadamente

```
populacao = {"Belo Horizonte":2000000,
  "Ouro Branco": 60000, "Lafaiete":120000}
for c,v in populacao.items():
  print(c, "->", v)
```

Dicionários

- Exemplo: exibe somente as chaves

```
populacao = {"Belo Horizonte":2000000, "Ouro
  Branco": 60000, "Lafaiete":120000}
for c in populacao.keys():
  print(c)
```

Mude para populacao.values() para exibir somente os valores

Dicionários

- Exemplo 5: insere cada elemento separadamente

```
turma = dict()
for i in range(5):
  nome = input("Nome do aluno:")
  nota = float(input("Nota:"))
  turma[nome]=nota
print(turma)
```

dict() pode ser substituído por {}

Dicionários

- O comando *del* apaga um elemento do dicionário
- Exemplo

```
populacao = {"Belo Horizonte":2000000, "Ouro
  Branco": 60000, "Lafaiete":120000}
print(populacao)
del populacao["Ouro Branco"]
print(populacao)
```

Lista e Dicionário

- Para armazenar informações mais complexas, pode-se criar uma lista cujos elementos são dicionários
- Cada elemento possui mais de um valor.
- No exemplo a seguir, faremos uma lista de contatos

Lista e Dicionário 1/2

```
lista = list()
while True:
  contato = dict()
  contato["nome"] = input("Digite o nome:")
  contato["telefone"] = input("Digite o telefone:")
  contato["email"] = input("Digite o e-mail:")
  lista.append(contato)
  op = input("Deseja cadastrar outro [s ou n]:")
  if op == 'N' or op == 'n':
    break
```

Lista e Dicionário 2/2

```
print()
for c in lista:
  for k, v in c.items():
    print(k,":",v)
  print()
```

Retorna um item, k recebe a chave e v o valor.

Conjuntos

Conjuntos

- A ordem dos elementos não importa
- Não possuem elementos repetidos
- Declaração com set() ou {}
- As vantagem da utilização de conjuntos sobre listas são:
  - Operações não implementadas em listas: união, interseção, diferença
  - Verificação de existem (in) muito mais rápida

Conjunto

- Exemplo

```
conjunto = set(('arroz', 'feijão', 'banana',
  'angu', 'torresmo'))
print(conjunto)
```

A exibição não ocorre na mesma ordem.

Conjunto

- Exemplo

```
conjunto = set(('arroz',
  'feijão','banana','angu','torresmo'))
conjunto.add("farofa")
conjunto.add("arroz")
print(conjunto)
```

Insere a farofa, mas o arroz não repete

Conjunto

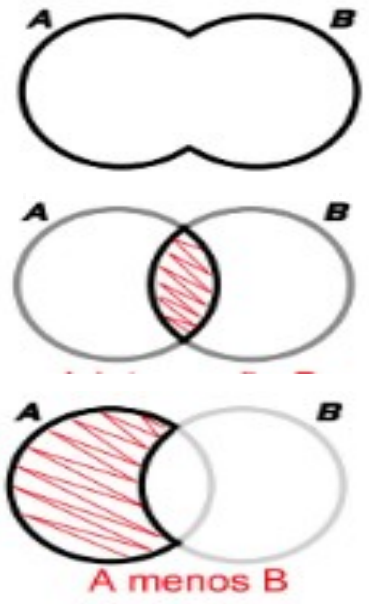
- Exemplo : verifica se um elemento está presente em um conjunto

```
conjunto = set(('arroz','feijão',’pera','angu',‘uva'))
comida = input('Digite o nome de uma comida:')
if comida in conjunto:
  print('Está presente')
else:
  print('Não está presente')
```

O comando in retorna True se o elemento estiver no conjunto

Conjunto

- Operações
  - União
  - Interseção
  - Diferença



Conjunto

- Exemplo: união de conjuntos

```
a = {1,3,5,7}
b = {1,2,3,4}
c = a.union(b)
print(c)
```

Resultado: {1,2,3,4,5,7}

Conjunto

- Exemplo: interseção entre conjuntos

```
a = {1,3,5,7}
b = {1,2,3,4}
c = a.intersection(b)
print(c)
```

Resultado: {1,3}

Conjunto

- Exemplo 3 (diferença)

```
a = {1,3,5,7}
b = {1,2,3,4}
c = a.difference(b)
print(c)
```

Resultado: {5,7}

Conjunto

- Exemplo 3: diferença

```
a = {1,3,5,7}
b = {1,2,3,4}
c = a.difference(b)
print(c)
```

Resultado: {5,7}

Passagem por valor e por referência

Passagem por valor e por referência

- Variáveis simples são passadas por valor

- Variáveis complexas (como listas, tuplas, dicionários, conjuntos e objetos) são passadas por referência

Passagem por valor

- Exemplo: passagem por valor

```
def dobra(x):
    x *= 2
x = 10
print(x)
dobra(x)
print(x)
```

Resultado:
10
10

Passagem por referência

- Exemplo (passagem por referência)

```
def dobralista(lst):
    for i in range(0,5):
        lst[i] *= 2
lista = [10,20,30,40,50]
print(lista)
dobralista(lista)
print(lista)
```

Resultado:
[10, 20, 30, 40, 50]
[20, 40, 60, 80, 100]

Tratamento de Exceção

Tratamento de Exceção

- Exemplo

```
dados = (10,20,30)
try:
  a = int(input('Qual posição você quer
  acessar?'))
  print(dados[a])
except:
  print('Posição inválida!')
```

Atenção: o tratamento de Exceção pode ser utilizado para qualquer tipo composto.

Exercícios

Exercícios

1. Crie uma função que recebe a, b, c de uma equação do segundo grau e retorne 3 (tupla) valores: x1, x2 e delta. Receba do usuário os valores de a, b e c, chame a função exiba o resultado.
2. Crie uma lista de contados, cada um possuindo nome, telefone e email. Crie um menu onde o usuário pode inserir um novo contato, apagar um contato, procurar pelo nome ou sair. Crie funções para cada operação.
3. Faça o usuário preencher dois conjuntos de número. Exiba a interseção, a união e a diferença entre eles.

Exercício 1

1/2

```
import math
def equacao(a,b,c):
    delta = b**2 - 4 * a * c
    if delta < 0:
        return 0,0,delta
    x1 = (-b + math.sqrt(delta))/(2*a)
    x2 = (-b - math.sqrt(delta))/(2*a)
    return x1,x2,delta
```

Exercício 1

2/2

```
a = float(input('Informe o valor de a:'))
b = float(input('Informe o valor de b:'))
c = float(input('Informe o valor de c:'))
r = equacao(a,b,c)
if r[2]<0:
    print('Essa equação não possui raizes reais.')
else:
    print('X1=',r[0])
    print('X2=',r[1])
```

Exercício 2

1/4

```
def insere():
    nome = input('Digite o nome:')
    telefone = input('Digite o telefone:')
    email = input('Digite o email:')
    contato = dict()
    contato['nome'] = nome
    contato['telefone'] = telefone
    contato['email'] = email
    lista.append(contato)
```

Exercício 2

2/4

```
def apagar():
    n = input('Digite o nome a ser apagado:')
    encontrou = False
    for cont in lista:
        if cont['nome']==n:
            del cont
            encontrou = True
            break
    if not(encontrou):
        print('Nome não encontrado')
```

Exercício 2

3/4

```
def busca():
    n = input('Digite o nome procurado:')
    encontrou = False
    for cont in lista:
        if cont['nome']==n:
            print('Nome:',cont['nome'])
            print('Telefone:',cont['telefone'])
            print('Email:',cont['email'])
            encontrou = True
            break
    if not(encontrou):
        print('Contato não encontrado.')
```

Exercício 2

4/4

```
lista = []
while True:
    print('\nDigite:')
    print('1-Criar um novo contato')
    print('2-Apagar um contato')
    print('3-Procurar contato pelo nome')
    print('4-Sair')
    op = input()
    if op=='1':
        insere()
    if op =='2':
        apagar()
    if op == '3':
        busca()
    if op=='4':
        break
```

Exercício 3

1/2

```
a = set()
b = set()
print('Preencha o primeiro conjunto.')
while True:
    n = input('Digite um número(sair para finalizar):')
    if n == 'sair':
        break
    num = int(n)
    a.add(num)
```

```
print('Preencha o segundo conjunto.')
while True:
    n = input('Digite um número(sair para finalizar):')
    if n == 'sair':
        break
    num = int(n)
    b.add(num)
uniao = a.union(b)
intersecao = a.intersection(b)
diferenca = a.difference(b)
print('União:',uniao)
print('Interseção:',intersecao)
print('Diferença:',diferenca)
```


Exercício 3

2/2