RACIOCÍNIO ALGORÍTMICO

# TUPLAS E DICIONÁRIOS

# ALGORITMO

## TUPLAS

Tuplas possuem estrutura muito parecida com listas, sendo a principal diferença o fato das tuplas serem estruturas imutáveis, ou seja, uma vez definido o seu conteúdo o mesmo não pode ser modificado.

```
endereço_puc = (“Rua Imaculada Conceição”, 1555, “Prado Velho”, “Curitiba”, “PR”)
documentos_joao = (“5321456-9”, “123456789-00”)
tupla_vazia = tuple()
```

# ALGORITMO

## TUPLAS

Exemplo de aplicação 1: *Elaborar um programa que solicita ao usuário o cadastro de endereços para entrega de produtos de uma loja.*

```
01. enderecos = []
02. print("Cadastro de Endereços de Entrega")
03. while True:
04.     logradouro = input("Digite o logradouro: ")
05.     numero = int(input("Digite o número: "))
06.     bairro = input("Digite o bairro: ")
07.     cidade = input("Digite a cidade: ")
08.     estado = input("Digite o estado: ")
09.     novo_endereco = (logradouro, numero, bairro, cidade, estado)
10.     enderecos.append(novo_endereco)
11.     if input("Deseja cadastrar um novo endereço (s/n): ") == "n":
12.         break
13. print("Os endereços cadastrados são:")
14. for i in range(0, len(enderecos)):
15.     endereco = enderecos[i]
16.     print(f"{i}. {endereco[0]}, {endereco[1]}, {endereco[2]} - {endereco[3]}/{endereco[4]}")
```

# ALGORITMO

## TUPLAS

```
/Users/user/untitled1/bin/python /Users/user/projects/untitled1/ExAplic01.py
Cadastro de Endereços de Entrega
Digite o logradouro: Rua Imaculada Conceição
Digite o número: 1555
Digite o bairro: Prado Velho
Digite a cidade: Curitiba
Digite o estado: PR
Deseja cadastrar um novo endereço (s/n): s
Digite o logradouro: Rua Professor Eurico Rabelo
Digite o número: 0
Digite o bairro: Norte
Digite a cidade: Rio de Janeiro
Digite o estado: RJ
Deseja cadastrar um novo endereço (s/n): n
Os endereços cadastrados são:
0. Rua Imaculada Conceição, 1555, Prado Velho - Curitiba/PR
1. Rua Professor Eurico Rabelo, 0, Norte - Rio de Janeiro/RJ

Process finished with exit code 0
```

Neste exemplo, caso seja feita efetuada a tentativa de alteração de algum dados da tupla vai ocorrer um erro de tipo:

```
14. for i in range(0, len(enderecos)):
15.     endereco = enderecos[i]
16.     print(f"{i}. {endereco[0]}, {endereco[1]}, {endereco[2]} - {endereco[3]}/{endereco[4]}")
17.     endereco[0] = "Novo endereço"
```

```
/Users/user/untitled1/bin/python /Users/user/projects/untitled1/ExAplic01.py

Traceback (most recent call last):
  File "/Users/user/projects/untitled1/ExAplic01.py", line 18, in <module>
    endereco[0] = "Novo endereço"
TypeError: 'tuple' object does not support item assignment

Process finished with exit code 1
```

# ALGORITMO

## TUPLAS

Exemplo de aplicação 2: *Elaborar um programa que cria uma tupla contendo duas listas como dados. Alterar os dados da primeira lista e verificar se ocorre mudança de dados da tupla.*

```
01. tupla_com_lista = ([1, 2, 3], [4, 5, 6])
02. print(id(tupla_com_lista[0]))
03. tupla_com_lista[0].append(9)
04. print(tupla_com_lista)
05. print(id(tupla_com_lista[0]))
```

```
/Users/user/untitled1/bin/python /Users/user/projects/untitled1/ExAplic02.py
4380125824
([1, 2, 3, 9], [4, 5, 6])
4380125824

Process finished with exit code 0
```

# ALGORITMO

## TUPLAS

Exemplo de aplicação 3: *Elaborar um programa que concatene tuplas.*

```
01. endereco_puc = ("Rua Imaculada Conceição", 1555, "Prado Velho", "Curitiba", "PR")
02. print(id(endereco_puc))
03. endereco_puc += ("Brazil",)
04. print(endereco_puc)
05. print(id(endereco_puc))
```

```
/Users/user/untitled1/bin/python /Users/user/projects/untitled1/ExAplic03.py
4430577968
('Rua Imaculada Conceição', 1555, 'Prado Velho', 'Curitiba', 'PR', 'Brazil')
4431654976

Process finished with exit code 0
```

# ALGORITMO

## TUPLAS

Exemplo de aplicação 4: *Elaborar um aplicativo que demonstre a necessidade de finalizar uma tupla de um único elemento com uma vírgula ao final.*

```
01. tupla1 = ("Brasil")
02. print(type(tupla1))
03. tupla2 = ("Brasil",)
04. print(type(tupla2))
```

```
/Users/user/untitled1/bin/python /Users/user/projects/untitled1/ExAplic04.py
<class 'str'>
<class 'tuple'>

Process finished with exit code 0
```

# ALGORITMO

## DUPLAS NOMEADAS

Exemplo de aplicação 5: *Elaborar um programa que faça uso de uma tupla chamada Endereco, contendo dados nomeados.*

```
01. from collections import namedtuple
02.
03. Endereco = namedtuple("Endereco", ["logradouro", "numero", "bairro", "cidade", "estado"])
04. endereco_puc = Endereco(logradouro="Rua Imaculada Conceição", numero=1555, bairro="Prado Velho",
    cidade="Curitiba", estado="PR")
05. print(f"Endereço: {endereco_puc[0]}")
06. print(f"  Número: {endereco_puc.numero}")
07. print(f"  Bairro: {endereco_puc.bairro}")
08. print(f"  Cidade: {endereco_puc.cidade}")
09. print(f"  Estado: {endereco_puc.estado}")
```

```
/Users/user/untitled1/bin/python /Users/user/projects/untitled1/ExAplic05.py
Endereço: Rua Imaculada Conceição
  Número: 1555
  Bairro: Prado Velho
  Cidade: Curitiba
  Estado: PR

Process finished with exit code 0
```

# ALGORITMO

## DICIONÁRIOS

Dicionários são estruturas de dados heterogêneos não ordenados, as quais permitem o acesso aos seus dados a partir de uma estrura de pares chave-valor. Uma estrutura de par chave-valor cria uma referência de par ordenado onde uma chave é cadastrada e, quando solicitada, endereça um determinado valor.

Para criar um dicionário é preciso usar colocar seus dados entre chaves, como segue:

```
faturamento = {“janeiro”: 2000, “fevereiro”: 3500, “março”: 2800}
```

O primeiro dado antes do símbolo de : é a chave, seguido do valor. Para acessar o valor cadastrado usa-se a chave. Por exemplo:

```
faturamento_janeiro = faturamento[“janeiro”]
```

E, de forma análoga, a chave é usada para alterar um valor:

```
faturamento[“janeiro”] = 5000
```

# ALGORITMO

## DICIONÁRIOS

Exemplo de aplicação 6: *Elaborar um programa que simule o cadastro de telefones com dicionário como uma agenda. Ao final, exiba o dicionário.*

```
01. agenda = {}
02. print("*** Cadastro de telefones ***")
03. while True:
04.     contato = input("Digite o nome do contato: ")
05.     telefone = input("Digite o telefone do contato: ")
06.     agenda[contato] = telefone
07.     if input("Deseja cadastrar um novo contato (s/n): ") == "n":
08.         break
09. print(agenda)
```

```
/Users/user/untitled1/bin/python /Users/user/projects/untitled1/ExAplic06.py
*** Cadastro de telefones ***
Digite o nome do contato: Maria
Digite o telefone do contato: (41) 98765-4321
Deseja cadastrar um novo contato (s/n): s
Digite o nome do contato: João
Digite o telefone do contato: (11) 12345-6789
Deseja cadastrar um novo contato (s/n): s
Digite o nome do contato: Rosana
Digite o telefone do contato: (21) 91827-3645
Deseja cadastrar um novo contato (s/n): n
{'Maria': '(41) 98765-4321', 'João': '(11) 12345-6789', 'Rosana': '(21) 91827-3645'}

Process finished with exit code 0
```

# ALGORITMO

## DICIONÁRIOS - EXISTÊNCIA DE CHAVE OU VALOR

Exemplo de aplicação 7: *Elaborar um programa que simule o cadastro de telefones com dicionário como uma agenda. Caso seja de um nome já existente, perguntar se deseja alterar dados existente. Caso seja um telefone já existente, informar que este telefone já está cadastrado em outro contato, não podendo ser efetuada a inclusão. Ao final, exiba o dicionário.*

```
01. agenda = {}
02. print("*** Cadastro de telefones ***")
03. while True:
04.     contato = input("Digite o nome do contato: ")
05.     telefone = input("Digite o telefone do contato: ")
06.     if contato in agenda:
07.         if input(f"Contato já cadastrado com o número {agenda[contato]}. Deseja alterar? (s/n)
    ") == "n":
08.             continue
09.     if telefone in agenda.values():
10.         print("Telefone já cadastrado para outro contato")
11.         continue
12.     agenda[contato] = telefone
13.     if input("Deseja cadastrar um novo contato (s/n): ") == "n":
14.         break
15. print(agenda)
```

# ALGORITMO

## DICIONÁRIOS - REMOÇÃO DE CHAVE

Existem duas formas de remover um item de dentro de um dicionário: usando a palavra reservada ***del*** e usando o método ***pop*** do dicionário. A remoção com ***del*** pode ser feita da seguinte forma:

```
del agenda[“Maria”]
```

Embora funcional, caso a chave “Maria” não exista, vai gerar um erro de execução:

```
/Users/user/untitled1/bin/python /Users/user/projects/untitled1/ExAplic07.py

Traceback (most recent call last):
  File "/Users/user/projects/untitled1/ExAplic07.py", line 3, in <module>
    del agenda["maria"]
KeyError: 'maria'

Process finished with exit code 1
```

Desta forma, para usar ***del*** é necessário primeiro verificar se a chave existe. A segunda forma usa o método ***pop()*** do dicionário, onde é retornado o dado removido em caso de sucesso ou um valor padrão caso tenha falhado a remoção, porém sem gerar erro no sistema.

# ALGORITMO

## DICIONÁRIOS - REMOÇÃO DE CHAVE

Exemplo de aplicação 08. *Elaborar um programa que exemplifique a remoção de itens de um dicionário usando o método pop().*

```
01. agenda = {"Maria": "(41) 98765-4321", "João": "(11) 12345-6789"}
02. print(agenda.pop("Maria", "Contato com nome Maria não localizado"))
03. print(agenda.pop("José", "Contato com nome José não localizado"))
04. print(agenda)
```

```
/Users/user/untitled1/bin/python /Users/user/projects/untitled1/ExAplic08.py
(41) 98765-4321
Contato com nome José não localizado
{'João': '(11) 12345-6789'}

Process finished with exit code 0
```

# PYTHON

## LISTAS - MÉTODOS

Um dicionário é um objeto em Python, ou seja, não faz parte dos tipos primitivos. Em sendo um objeto, possui métodos, que são ações que pode realizar. Segue a lista de métodos que podem ser aplicados sobre dicionários em Python:

| Listas - Métodos | |
|---|---|
| **Método** | **Descrição** |
| clear() | Remove todos os elementos de um dicionário |
| copy() | Retorna a cópia de um dicionário |
| fromkeys() | Retorna um dicionário com chaves e valores específicos |
| get() | Retorna o valor de uma chave específica |
| items() | Retorna um lista contendo uma tupla para cada par chave-valor |
| keys() | Retorna uma lista contendo as chaves de um dicionário |
| pop() | Remove um elemento de uma chave específica |
| popitem() | Remove o último elemento inserido no dicionário |
| setdefault() | Retorna o valor de uma chave específica. Caso não existe, insere este valor |
| update() | Adiciona um conjunto de pares chave-valor a um dicionário |
| values() | Retorna a lista de valores de um dicionário |

# ALGORITMO

## DICIONÁRIOS - MÉTODOS

```
01. agenda = {"Maria": "(41) 98765-4321", "João": "(11) 12345-6789", "Rosana": "(21) 91827-3645"}
02. print(agenda)
03. # adiciona um novo elemento no vetor
04. agenda["José"] = "(19) 98877-1122"
05. print(agenda)
06. # cria um dicionário com chaves e valores específicos
07. chaves = ["chave1", "chave2", "chave3"]
08. valores = "valor"
09. novo_dicionario = dict.fromkeys(chaves, valores)
10. print(novo_dicionario)
11. # mostra a lista de itens do dicionário
12. print(agenda.items())
13. # mostra a lista de chaves do dicionário
14. print(agenda.keys())
15. # mostra a lista de valores do dicionário
16. print(agenda.values())
17. # remove um item de chave específica
18. agenda.pop("Maria")
19. print(agenda)
20. # remove o último item inserido
21. agenda.popitem()
22. print(agenda)
23. # retorna ou insere o valor de uma chave em específico
24. print(agenda.setdefault("Rosana", "(21) 91827-3645"))
25. print(agenda.setdefault("Maria", "(41) 98765-4321"))
26. print(agenda)
27. # adiciona o conteúdo de um dicionário a outro
28. agenda.update(novo_dicionario)
29. print(agenda)
30. # limpa o conteúdo de um dicionário
31. agenda.clear()
32. print(agenda)
```

```
/Users/user/untitled1/bin/python /Users/user/projects/untitled1/listas_metodos.py
{'Maria': '(41) 98765-4321', 'João': '(11) 12345-6789', 'Rosana': '(21) 91827-3645'}
{'Maria': '(41) 98765-4321', 'João': '(11) 12345-6789', 'Rosana': '(21) 91827-3645', 'José': '(19) 98877-1122'}
{'chave1': 'valor', 'chave2': 'valor', 'chave3': 'valor'}
dict_items([('Maria', '(41) 98765-4321'), ('João', '(11) 12345-6789'), ('Rosana', '(21) 91827-3645'), ('José', '(19) 98877-1122')])
dict_keys(['Maria', 'João', 'Rosana', 'José'])
dict_values(['(41) 98765-4321', '(11) 12345-6789', '(21) 91827-3645', '(19) 98877-1122'])
{'João': '(11) 12345-6789', 'Rosana': '(21) 91827-3645', 'José': '(19) 98877-1122'}
{'João': '(11) 12345-6789', 'Rosana': '(21) 91827-3645'}
(21) 91827-3645
(41) 98765-4321
{'João': '(11) 12345-6789', 'Rosana': '(21) 91827-3645', 'Maria': '(41) 98765-4321'}
{'João': '(11) 12345-6789', 'Rosana': '(21) 91827-3645', 'Maria': '(41) 98765-4321', 'chave1': 'valor', 'chave2': 'valor', 'chave3': 'valor'}
{}

Process finished with exit code 0
```

# TUPLAS E DICIONÁRIOS

## EXERCÍCIOS

### Tuplas

1. Criar um programa que efetua o cadastro de pessoas com nome, rg e cpf por meio de tuplas, adicionando elas a uma lista. Imprimir a lista ao final do programa.

2. Criar um programa que cadastre funcionário de um empresa e seus dependentes. O funcionário deve ser cadastrado com matrícula, nome e dependentes. Os dependentes devem ser inseridos dinamicamente em uma tupla. Dica: usar o operador +=.

3. Criar um programa que cadastre locais históricos do mundo com suas coordenadas fazendo uso de tuplas com parâmetros nomeados. Dica: usar a função namedtuple().

# TUPLAS E DICIONÁRIOS

## EXERCÍCIOS

### Dicionários

4. Criar um programa que efetua cadastro de produtos e preços. Caso o produto já existe, pergunta se o usuário pretende atualizar o valor. Imprimir o dicionário ao final do programa em formato de lista.

# TUPLAS E DICIONÁRIOS

## EXERCÍCIOS

### Desafio

5. Criar um programa que solicita valor de vendas e o mês onde cada venda ocorreu. Independente de repetição de meses, o aplicativo deve totalizar por mês todas as vendas cadastradas. Ao final deve informar o valor de vendas de todos os meses do ano. Obs.: se for digitado errado o nome do mês, informar que o mês é inválido.