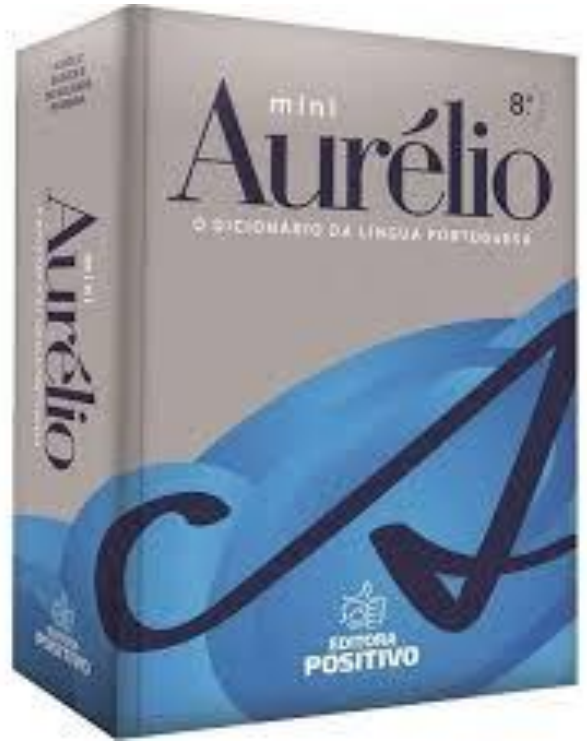
Programação em Python

CEFET-MG

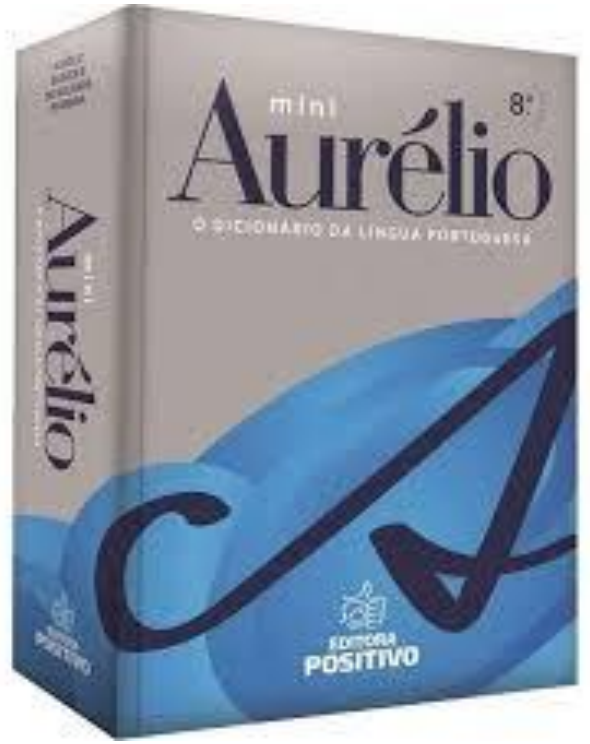
- **Dicionários**
  - Semelhante ao ‘struct’ em C.
  - Permite armazenar valores correlatos.
    - chave-valor.
  - Qualquer objeto Python.
  - Identificado pelo ‘{:}’.
  - Ex.:

```
>>> shows = {‘Lex Luthor’: ’convite’, ‘Dias de Truta’: 50}

>>> print(shows[‘Dias de Truta’])
```



- **Dicionários**
  - Semelhante ao ‘struct’ em C.
  - Permite armazenar valores correlatos.
    - chave-valor.
  - Qualquer objeto Python.
  - Identificado pelo ‘{:}’ ou dict()
  - Ex.: >>> prof_guido = {‘peso’: 100, ‘idade’: 40, ‘filhos’: 2}



Muito útil para guardar informações de um objeto específico!

- **Dicionários: Acessando valores**
  - Nome do dicionário + chave entre colchetes ‘[]’.
  - Ex.:
    - print(prof_guido[‘idade’])

- **Dicionários: Adiconando novos elementos.**
  - Estrutura de memória dinâmica
  - nome[‘*chave*’] = *valor*
  - Ex.:
    - print(prof_guido)
    - prof_guido['estado_civil'] = 'preso‘
    - print(prof_guido)

- **Exercícios**

1. Crie um dicionário vazio chamado ‘cadastro’.
2. Acrescente, a cada iteração, as informações pessoais, CPF, ID, cep, cidade natal e telefone. Crie uma interface para pedir ao usuário inserir as informações desejadas.

- **Dicionários: Modificando valores**
  - nome[‘*chave*’] = *valor*
  - Ex.:
    - print(prof_guido)
    - prof_guido['estado_civil'] = ‘casado‘
    - print(prof_guido)

CEFET-MG

- **Dicionários: Removendo elementos**
  - instrução ‘del’
  - Ex.:
    - print(prof_guido)
    - del prof_guido['estado_civil']
    - print(prof_guido)

- **Dicionários: Fazendo uma cópia**
  - Método ‘copy()’
  - Ex.:

```
>>> estado = dict()
>>> brasil = list()
>>> for c in range(3)
>>>     estado[‘uf’] = str(input(‘Unidade Federativa: ’))
>>>     estado[‘sigla’] = str(input(‘Sigla do Estado: ’))
>>>     brasil.append(estado.copy())
>>> print(brasil)
```

  - **Retire o .copy(), qual o comportamento do algoritmo?**

CEFET-MG

- **Dicionários: Percorrendo um dicionário**
  - Através das chaves e valores.
  - for *chave*, *valor* in nome.items()
  - Ex.:

 # método items(): devolve uma lista de pares chave-valor.

```
>>> user_0 = {'username': ‘miziphi', 'first': ‘Guido', 'last': ‘Pantuza', ‘idade’: 40}

>>> for i, value in user_0.items():

>>>     print(‘\nKey:  {}’.format(i))

>>>     print(‘Value: {}’.format(value))
```

CEFET-MG

- **Dicionários: Percorrendo um dicionário**
  - Através das chaves.
  - Método keys()
  - for *chave* in nome.keys()
  - Ex.:

```
>>> user_0 = {'username': ‘miziphi', 'first': ‘Guido', 'last': ‘Pantuza', ‘idade’: 40}

>>> for i in user_0.keys()):

>>>     print(‘\nKey:  {}’.format(i), end=‘’)
```

CEFET-MG

- **Dicionários: Percorrendo um dicionário**
  - Através dos valores.
  - Método values()
  - for *valor* in nome. values()
  - Ex.:

```
>>> user_0 = {'username': ‘miziphi', 'first': ‘Guido', 'last': ‘Pantuza', ‘idade’: 40}

>>> for i in user_0.values():

>>>     print(‘\nValor:  {}’.format(i))
```

- **Dicionários: Percorrendo um dicionário**
  - Método set()
  - Set: conjuntos de elementos iguais
  - Útil para imprimir um valor repetido, apenas uma vez
  - Ex.: Em uma pesquisa, foi perguntado qual a linguagem favorita. Quais linguagens foram citadas?

```
>>> fav_languages = {'jen': 'python', 'sarah': 'c', 'edward': ‘c', 'phil': 'python', }
>>> print(‘As linguagens citadas foram: ‘, end=‘’)
>>> for language in set(fav_languages.values()): print(language.title())
```

- **Dicionários: Percorrendo um dicionário**
  - De forma ordenada.
  - sorted()
  - Ex.:

```
>>> favorite_languages = {'jen': 'python', 'sarah': 'c', 'edward': 'ruby', 'phil':
'python', }

>>> for name in sorted(favorite_languages.keys()):

>>>     print('{}, obrigado'.format(name.title())

>>>     print(f'{name.title()}, obrigado')
```

CEFET-MG

- **Lista de dicionários:**
  - Cria o dicionário.
  - Em seguida, crie a lista.
  - Ex.:

```
>>> prof_0 = {‘nome’:’guido’, ‘sobrenome’:’pantuza’}

>>> prof_1 ={‘nome’:’fulano’, sobrenome’: ‘beltrano’}

>>> prof_2 = {‘nome’:’paulo’, ‘sobrenome’:’cavegato’}

>>> profs = [prof_0, prof_1, prof_2]
```

  - **E se fizer a ocontrário?**

CEFET-MG

- **Dicionários de listas:**
  - Nome da lista é uma chave.
  - Elementos da lista são os valores.
  - Não aninhe listas e dicionários com muitos níveis.
  - Ex.:

```
>>> pizza = {‘preço’:100, ‘ings’:[‘molho’, ’tomate’, ’muçarela’]}
```

Lista

CEFET-MG

- **Dicionário de dicionários:**
  - Deve ser evitado.
  - Não aninhe dicionários com muitos níveis.
  - Ex.:

```
>>> users = {
        'aeinstein': {'first': 'albert', 'last': 'einstein', 'location': 'princeton', },
        'mcurie': {'first': 'marie', 'last': 'curie', 'location': 'paris', },
    }
```

- **Exercícios**

3. Use um dicionário para armazenar a nota de algumas pessoas. Pense em cinco nomes e use-os como chaves em seu dicionário. Atribua notas aleatórias no intervalo [0,10] e armazene cada um como um valor em seu dicionário. Calcule a média, se a média da turma for menor que 6 imprima, ‘Rever a metodologia.’, se for igual a 6, ‘Por pouco!’, se for entre 6 e 8, ‘Satisfatório’, se superior a 8, ‘Aí SIM!’

Programação em Python

CEFET-MG

- import random

- # Criando o dicionário com 5 nomes e notas aleatórias de 0 a 10
- notas = {
- "Ana": random.randint(0, 10),
- "Bruno": random.randint(0, 10),
- "Carla": random.randint(0, 10),
- "Diego": random.randint(0, 10),
- "Eduarda": random.randint(0, 10)
- }

- print("Notas da turma:", notas)

- # Calculando a média
- media = sum(notas.values()) / len(notas)
- print(f"Média da turma: {media:.2f}")

```
# Verificando a situação
if media < 6:
    print("Rever a metodologia.")
elif media == 6:
    print("Por pouco!")
elif 6 < media <= 8:
    print("Satisfatório")
else:
    print("Aí SIM!")
```

- **Exercícios**

4. Glossário: Um dicionário Python pode ser usado para modelar um dicionário de verdade. No entanto, para evitar confusão, vamos chamá-lo de glossário.
   - Pense em cinco comandos ou funções relacionadas às listas que você conheceu nos capítulos anteriores. Use essas palavras como chaves em seu glossário e armazene seus significados como valores.
   - Mostre cada palavra e seu significado em uma saída formatada de modo elegante. Você pode exibir a palavra seguida de dois-pontos e depois o seu significado, ou apresentar a palavra em uma linha e então exibir seu significado indentado (\t) em uma segunda linha. Utilize o caractere de quebra de linha (\n) para inserir uma linha em branco entre cada par palavra-significado em sua saída.

Programação em Python

CEFET-MG

```
# Criando o glossário com comandos/funções de listas
glossario = {
    "append": "Adiciona um elemento ao final da lista.",
    "insert": "Insere um elemento em uma posição específica da lista.",
    "remove": "Remove a primeira ocorrência de um valor especificado na lista.",
    "pop": "Remove e retorna o último elemento da lista (ou de um índice específico).",
    "sort": "Ordena a lista em ordem crescente por padrão."
}

# Exibindo de forma elegante
for palavra, significado in glossario.items():
    print(f"{palavra}:\n\t{significado}\n")
```

- **Exercícios:**

5) Crie um dicionário que contenha três rios importantes e o país que cada rio corta.

a) Use um laço para exibir uma frase sobre cada rio, por ex., O Nilo corre pelo Egito.

b) Use um laço para exibir o nome de cada rio incluído no dicionário.

c) Use um laço para exibir o nome de cada país incluído no dicionário.

Programação em Python

CEFET-MG

```
# Criando o dicionário com rios e países
rios = {
    "Nilo": "Egito",
    "Amazonas": "Brasil",
    "Danúbio": "Alemanha"
}

# Exibir frases sobre cada rio
for rio, pais in rios.items():
    print(f"O {rio} corre pelo {pais}.")

print("\n--- Lista de rios ---")
# Exibir apenas os nomes dos rios
for rio in rios.keys():
    print(rio)

print("\n--- Lista de países ---")
# Exibir apenas os nomes dos países
for pais in rios.values():
    print(pais)
```

- **Exercícios:**

6) Crie uma lista de pessoas que devam participar de uma enquete sobre linguagem favorita. Inclua alguns nomes que já estejam no dicionário e outros que não estão. Percorra a lista de pessoas que devem participar da enquete. Se elas já tiverem respondido à enquete, mostre uma mensagem agradecendo-lhes por responder. Se ainda não participaram da enquete, apresente uma mensagem convidando-as a responder.

```
# Dicionário com pessoas que já responderam à enquete
respostas = {
    "Ana": "Python",
    "Bruno": "C++",
    "Carla": "Java"
}

# Lista de pessoas que devem participar (algumas já responderam, outras não)
pessoas = ["Ana", "Carla", "Diego", "Eduarda", "Fernando"]

# Percorrer a lista de pessoas
for pessoa in pessoas:
    if pessoa in respostas:
        print(f"Obrigado, {pessoa}, por já ter respondido à enquete!")
    else:
        print(f"{pessoa}, você ainda não respondeu à enquete. Participe!")
```

- **Exercícios:**

7) Animais de estimação: Crie vários dicionários, em que o nome de cada dicionário seja o nome de um animal de estimação. Em cada dicionário, inclua o tipo do animal e o nome do dono. Armazene esses dicionários em uma lista chamada pets. Em seguida, percorra sua lista com um laço e, à medida que fizer isso, apresente tudo que você sabe sobre cada animal de estimação.

```
# Criando os dicionários de animais de estimação
cachorro = {
    "tipo": "Cachorro",
    "dono": "Ana"
}

gato = {
    "tipo": "Gato",
    "dono": "Bruno"
}

papagaio = {
    "tipo": "Papagaio",
    "dono": "Carla"
}

peixe = {
    "tipo": "Peixe",
    "dono": "Diego"
}

coelho = {
    "tipo": "Coelho",
    "dono": "Eduarda"
}

# Armazenando todos os pets em uma lista
pets = [cachorro, gato, papagaio, peixe, coelho]

# Percorrendo a lista e exibindo informações
for pet in pets:
    print(f"Animal: {pet['tipo']}, Dono: {pet['dono']}")
```

CEFET-MG

- **Exercícios:**

Lugares favoritos: Crie um dicionário chamado favorite_places. Pense em três nomes para usar como chaves do dicioná rio e armazene de um a três lugares favoritos para cada pessoa. Para deixar este exercício um pouco mais interessante, peça a alguns amigos que nomeiem alguns de seus lugares favoritos. Percorra o dicionário com um laço e apresente o nome de cada pessoa e seus lugares favoritos.

Programação em Python

CEFET-MG

```
# Criando o dicionário com listas de lugares favoritos
favorite_places = {
    "Ana": ["Praia", "Parque", "Biblioteca"],
    "Carlos": ["Montanha", "Museu"],
    "Mariana": ["Cinema"]
}


# Percorrendo o dicionário e mostrando os lugares
for pessoa, lugares in favorite_places.items():
    print(f"\n{pessoa} gosta dos seguintes lugares:")
    for lugar in lugares:
        print(f"- {lugar}")
```

- **Exercícios:**

Lanchonete: Crie uma lista chamada sandwich_orders e a preencha com os nomes de vários sanduíches. Em seguida, crie uma lista vazia chamada finished_sandwiches. Percorra a lista de pedidos de sanduíches com um laço e mostre uma mensagem para cada pedido, por exemplo, Preparei seu sanduíche de atum. À medida que cada sanduíche for preparado, transfira-o para a lista de sanduíches prontos. Depois que todos os sanduíches estiverem prontos, mostre uma mensagem que liste cada sanduíche preparado.

Programação em Python

CEFET-MG

```
# Lista de pedidos de sanduíches
sandwich_orders = ["atum", "frango", "queijo", "presunto", "vegetariano"]

# Lista vazia para armazenar os sanduíches prontos
finished_sandwiches = []

# Preparando cada sanduíche
while sandwich_orders:
    pedido = sandwich_orders.pop(0)  # Remove o primeiro pedido da lista
    print(f"Preparei seu sanduíche de {pedido}.")
    finished_sandwiches.append(pedido)  # Adiciona na lista de prontos

# Mensagem final com todos os sanduíches preparados
print("\nTodos os sanduíches foram preparados:")
for sandwich in finished_sandwiches:
    print(f"- Sanduíche de {sandwich}")
```

- **Exercícios:**

Sem pastrami: Usando a lista sandwich_orders do Exercício Anterior, garanta que o sanduíche de 'pastrami' apareça na lista pelo menos três vezes.

Acrescente um código próximo ao início de seu programa para exibir uma mensagem informando que a lanchonete está sem pastrami e, então, use um laço while para remover todas as ocorrências de 'pastrami' e sandwich_orders.

Garanta que nenhum sanduíche de pastrami acabe em finished_sandwiches.

Programação em Python

CEFET-MG

- # Lista de pedidos de sanduíches (incluindo pastrami várias vezes)
- sandwich_orders = ["atum", "pastrami", "frango", "pastrami", "queijo", "pastrami", "vegetariano"]

- # Lista vazia para armazenar os sanduíches prontos
- finished_sandwiches = []

- # Aviso de que não há pastrami
- print("Desculpe, estamos sem pastrami hoje!")

- # Removendo todas as ocorrências de 'pastrami'
- while "pastrami" in sandwich_orders:
- sandwich_orders.remove("pastrami")

```
# Preparando cada sanduíche (sem pastrami)
while sandwich_orders:
    pedido = sandwich_orders.pop(0)
    print(f"Preparei seu sanduíche de {pedido}.")
    finished_sandwiches.append(pedido)

# Mensagem final com todos os sanduíches preparados
print("\nTodos os sanduíches foram preparados (sem pastrami):")
for sandwich in finished_sandwiches:
    print(f"- Sanduíche de {sandwich}")
```

- **Exercícios:**

Escreva um programa que leia quatro notas dos alunos de uma turma e armazena as notas em um dicionário, onde a chave é o nome do aluno.

a. O programa deve permitir que se digite um novo aluno até que se entre com uma string ‘quit’ como nome.

b. Escreva uma função que retorna a média do aluno, dado seu nome.

Programação em Python

CEFET-MG

```
# Dicionário para armazenar os alunos e suas notas
alunos = {}

# Leitura dos dados
while True:
    nome = input("Digite o nome do aluno (ou 'quit' para sair): ")
    if nome.lower() == "quit":
        break

    # Lendo 4 notas do aluno
    notas = []
    for i in range(4):
        nota = float(input(f"Digite a nota {i+1} de {nome}: "))
        notas.append(nota)

    # Armazenando no dicionário
    alunos[nome] = notas
```

```
# Função para calcular a média de um aluno
def media_aluno(nome, dicionario):
    if nome in dicionario:
        return sum(dicionario[nome]) / len(dicionario[nome])
    else:
        return None


# Exemplo de uso
print("\n=== Médias dos alunos ===")
for aluno in alunos:
    print(f"{aluno}: {media_aluno(aluno, alunos):.2f}")
```

- **Exercícios:**

  Crie um dicionário que é uma agenda e coloque nele os seguintes dados: chave (cpf), nome, idade, telefone. O programa deve ler um número indeterminado de dados, criar a agenda e imprimir todos os itens do dicionário no formato chave: nome-idade-telefone.

```
# Dicionário da agenda
agenda = {}

while True:
    cpf = input("Digite o CPF (ou 'quit' para sair): ")
    if cpf.lower() == "quit":
        break

    nome = input("Digite o nome: ")
    idade = int(input("Digite a idade: "))
    telefone = input("Digite o telefone: ")

    # Cada CPF terá como valor um outro dicionário
    agenda[cpf] = {
        "nome": nome,
        "idade": idade,
        "telefone": telefone
    }

# Impressão dos dados da agenda
print("\n=== Agenda completa ===")
for cpf, dados in agenda.items():
    print(f"{cpf}: {dados['nome']} - {dados['idade']} - {dados['telefone']}")
```

- **Exercícios:**

  Crie um dicionário d e coloque nele os dados fornecidos pelo usuário: nome, idade, telefone, endereço. Ainda usando d, imprima todos os itens do dicionário no formato chave: valor, ordenado pela chave.

CEFET-MG

```
# Criando o dicionário vazio
d = {}

# Lendo os dados do usuário
d["nome"] = input("Digite o nome: ")
d["idade"] = input("Digite a idade: ")
d["telefone"] = input("Digite o telefone: ")
d["endereco"] = input("Digite o endereço: ")

# Imprimindo os itens do dicionário ordenados pela chave
print("\n=== Dados fornecidos ===")
for chave in sorted(d.keys()):
    print(f"{chave}: {d[chave]}")
```

- **Desafio:Gerenciamento de Estoque**

- Você foi contratado para criar um sistema simples de gerenciamento de estoque. Seu objetivo é desenvolver um programa que utilize dicionários para organizar, buscar e atualizar informações de um estoque de produtos. O estoque será representado por um dicionário. A chave é o nome do produto. O valor é outro dicionário contendo: quantidade: número de unidades disponíveis no estoque e; preco: preço unitário do produto.

CEFET-MG

- **Desafio:Gerenciamento de Estoque**
- O programa deve:
  - Exibir o estoque completo: Mostre todos os produtos com suas quantidades e preços.
  - Adicionar um novo produto: Peça o nome do produto, a quantidade e o preço, e adicione ao estoque.
  - Atualizar um produto existente: Permita ao usuário atualizar a quantidade ou o preço de um produto.

CEFET-MG

- **Desafio:Gerenciamento de Estoque**

- O programa deve:
  - Calcular o valor total do estoque: Multiplique a quantidade pelo preço de cada produto e exiba o valor total.
  - Remover um produto do estoque: Peça o nome do produto e remova-o.
  - Exibir o estoque ordenado por nome do produto ou valor total (quantidade * preço).

```
# Estoque inicial vazio
estoque = {}

while True:
    print("\n=== Sistema de Estoque ===")
    print("1 - Exibir estoque (por nome)")
    print("2 - Exibir estoque (por valor total)")
    print("3 - Adicionar produto")
    print("4 - Atualizar produto")
    print("5 - Remover produto")
    print("6 - Valor total do estoque")
    print("0 - Sair")

    opcao = input("Escolha uma opção: ")
```

```
    if opcao == "1":
        if not estoque:
            print("O estoque está vazio.")
        else:
            print("\n=== Estoque por nome ===")
            for produto in sorted(estoque.keys()):
                qtd = estoque[produto]["quantidade"]
                preco = estoque[produto]["preco"]
                print(f"{produto}: Quantidade = {qtd}, Preço = R${preco:.2f}")

    elif opcao == "2":
        if not estoque:
            print("O estoque está vazio.")
        else:
            print("\n=== Estoque por valor total ===")
            # Ordena pelo valor total (quantidade * preço), decrescente
            for produto, info in sorted(estoque.items(), key=lambda item: item[1]["quantidade"]*item[1]["preco"],
reverse=True):
                qtd = info["quantidade"]
                preco = info["preco"]
                print(f"{produto}: Quantidade = {qtd}, Preço = R${preco:.2f}")
```

```
elif opcao == "3":
    nome = input("Digite o nome do produto: ")
    if nome in estoque:
        print("Produto já existe no estoque. Use atualizar para modificar.")
    else:
        quantidade = int(input("Digite a quantidade: "))
        preco = float(input("Digite o preço unitário: "))
        estoque[nome] = {"quantidade": quantidade, "preco": preco}
        print(f"{nome} adicionado ao estoque.")

elif opcao == "4":
    nome = input("Digite o nome do produto a atualizar: ")
    if nome not in estoque:
        print("Produto não encontrado.")
    else:
        print("Digite os novos valores (pressione Enter para manter o atual):")
        qtd_str = input(f"Quantidade atual ({estoque[nome]['quantidade']}): ")
        preco_str = input(f"Preço atual ({estoque[nome]['preco']:.2f}): ")
        if qtd_str:
            estoque[nome]['quantidade'] = int(qtd_str)
        if preco_str:
            estoque[nome]['preco'] = float(preco_str)
        print(f"{nome} atualizado.")
```

```
elif opcao == "5":
    nome = input("Digite o nome do produto a remover: ")
    if nome in estoque:
        del estoque[nome]
        print(f"{nome} removido do estoque.")
    else:
        print("Produto não encontrado.")

elif opcao == "6":
    total = sum(info["quantidade"]*info["preco"] for info in estoque.values())
    print(f"\nValor total do estoque: R${total:.2f}")

elif opcao == "0":
    print("Encerrando o sistema...")
    break

else:
    print("Opção inválida. Tente novamente.")
```