Fatec
Rio Claro

Estrutura de Dados

Prof. Orlando Saraiva Júnior
orlando.nascimento@fatec.sp.gov.br

Estrutura de Dados

Módulo Collections

Fatec
Rio Claro

O módulo de *collections* fornece diferentes tipos de contêineres, que são objetos usados para armazenar objetos diferentes e fornecer uma maneira de acessá -los.

Módulo Collections

Fatec Rio Claro

namedtuple	Cria uma tupla com campos nomeados semelhantes às tuplas comuns.
deque	Listas duplamente ligadas que fornecem adição e remoção eficientes de itens das duas extremidades da lista.
defaultdict	Uma subclasse de dicionário que retorna valores padrão para as teclas ausentes.
ChainMap	Um dicionário que mescla vários dicionários.
Counter	Um dicionário que retorna as contagens correspondentes aos seus objetos/chave.
UserDict UserList UserString	Esses tipos de dados são usados para adicionar mais funcionalidades à sua estrutura de dados de base, como um dicionário, lista e string. E podemos criar subclasses a partir deles para dicto/list/string personalizado.

Named tuples

Fatec
Rio Claro

O namedtuple fornece uma extensão do tipo de dados de tupla embutido. Os objetos nomeados são imutáveis, semelhantes às tuplas padrão.

Portanto, não podemos adicionar novos campos ou modificar os existentes após a criação da instância do nome nomeado.

Eles contêm chaves mapeadas para um valor específico e podemos iterar através de tuplas nomeadas por índice ou chave.

A função NomeTuple é principalmente útil quando várias tuplas são usadas em um aplicativo e é importante acompanhar cada uma das tuplas em termos do que eles representam.

Named tuples

Fatec
Rio Claro

```
from collections import namedtuple

Book = namedtuple ('Book', ['name', 'ISBN', 'quantity'])

Book1 = Book('Hands on Data Structures', '9781788995573', '50')

#Accessing data items

print('Using index ISBN:' + Book1[1])

print('Using key ISBN:' + Book1.ISBN)
```

Deque

Fatec
Rio Claro

Uma *deque* é uma fila de extremidade dupla (deque) que suporta a adição e remoção de elementos de ambos os lados da lista. Deques são implementadas como listas duplamente encadeadas, que são muito eficientes para inserir e excluir elementos em complexidade de tempo O(1).

Deque

Fatec
Rio Claro

```
from collections import deque
fila = deque()
fila.append("João")
fila.append("Maria")
fila.append("Carlos")
fila.appendleft("Atendimento Prioritário")
cliente_atendido = fila.popleft()
cliente_desistiu = fila.pop()
```

Ordered dictionaries

Fatec
Rio Claro

Um dicionário ordenado é um dicionário que preserva a ordem das chaves inseridas.

Se a ordem das chaves for importante para qualquer aplicação, OrderedDict pode ser usado.

No código a seguir, criamos um dicionário, od, usando o módulo *OrderedDict*.

Podemos observar que a ordem das chaves é a mesma de quando criamos a chave.

Ordered dictionaries

Fatec
Rio Claro

```
from collections import OrderedDict

od = OrderedDict({'my': 2, 'name ': 4, 'is': 2, 'Orlando' :5})

od['hello'] = 4

print(od)
```

Default dictionary

Fatec
Rio Claro

O dicionário padrão (defaultdict) é uma subclasse da classe de dicionário interna (dict) que tem os mesmos métodos e operações que a classe de dicionário, com a única diferença de que ele nunca gera um KeyError, como um dicionário normal faria. defaultdict é uma maneira conveniente de inicializar dicionários:

Default dictionary

Fatec
Rio Claro

```
from collections import defaultdict

contador_palavras = defaultdict(int)
texto = "o gato subiu no telhado o gato miou"

for palavra in texto.split():
    contador_palavras[palavra] += 1

print(contador_palavras)
```

Default dictionary

Fatec
Rio Claro

```
from collections import defaultdict


# Valor padrão será uma lista vazia
alunos_por_curso = defaultdict(list)


# Adicionando alunos
alunos_por_curso["Python"].append("João")
alunos_por_curso["Python"].append("Maria")
alunos_por_curso["Java"].append("Carlos")
```

Chain Map

Fatec
Rio Claro

O ChainMap é usado para criar uma lista de dicionários.

A estrutura de dados da Coleções.ChainMap combina vários dicionários em um único mapeamento.

Sempre que uma chave é pesquisada no ChainMap, ela analisa todos os dicionários um por um, até que a chave não seja encontrada:

Chain Map

Fatec
Rio Claro

```
from collections import ChainMap
config_padrao = {
    "resolucao": "1080p",
    "volume": 70,
    "idioma": "Português"
}
config_jogador = {
    "volume": 90,
    "tema": "Escuro"
}
config_final = ChainMap(config_jogador, config_padrao)
```

Counter

Fatec
Rio Claro

um objeto hashável é aquele cujo valor de hash permanecerá o mesmo durante sua existência no programa. O contador é usado para contar o número de objetos hasháveis.

A chave do dicionário é um objeto hashável, enquanto o valor correspondente é a contagem desse objeto.

Em outras palavras, os objetos contadores criam uma tabela de hash na qual os elementos e suas contagens são armazenados como chaves do dicionário e pares de valores.

Counter

Fatec
Rio Claro

```
from collections import Counter
inventory = Counter('hello')
print(inventory)
print(inventory['l'])
print(inventory['e'])
print(inventory['o'])
```

Counter

Fatec
Rio Claro

Objetos de dicionário e contador são semelhantes no sentido de que os dados são armazenados em um par {chave, valor}, mas em objetos de contador, o valor é a contagem da chave, enquanto no caso do dicionário, pode ser qualquer valor.

Assim, quando queremos apenas ver quantas vezes cada palavra única ocorre em uma string, usamos o objeto de contador.

UserDict

Fatec
Rio Claro

Python suporta um contêiner, UserDict, presente no módulo collections, que encapsula os objetos do dicionário. Podemos adicionar funções personalizadas ao dicionário.

Isso é muito útil para aplicações em que queremos adicionar/atualizar/modificar as funcionalidades do dicionário.

UserDict

Fatec
Rio Claro

```
from collections import UserDict
class NotasAlunos(UserDict):
    def inserir_nota(self, aluno, nota):
        if nota < 0 or nota > 10:
            raise ValueError("A nota deve estar entre 0 e 10.")
        self.data[aluno] = nota  # Usamos self.data para acessar o dict interno
    def media_turma(self):
        if not self.data:
            return 0
        return sum(self.data.values()) / len(self.data)
```

UserDict

Fatec
Rio Claro

```
# Criando o dicionário de notas
turma = NotasAlunos()

# Inserindo notas válidas
turma.inserir_nota("João", 8)
turma.inserir_nota("Maria", 9)
turma.inserir_nota("Ana", 10)
```

UserList

Fatec
Rio Claro

Uma UserList é um contêiner que encapsula objetos de lista. Ela pode ser usada para estender a funcionalidade da estrutura de dados da lista.

UserList

Fatec
Rio Claro

```
class ListaDeTarefas(UserList):
    def append(self, item):
        if item in self.data:
            print(f"Tarefa '{item}' já existe na lista. Não será adicionada.")
        else:
            super().append(item)  # Usa o método original para adicionar
    def mostrar_tarefas(self):
        print("Minhas tarefas:")
        for i, tarefa in enumerate(self.data, start=1):
            print(f"{i}. {tarefa}")
```

UserList

Fatec
Rio Claro

```
# Criando a lista de tarefas
minhas_tarefas = ListaDeTarefas()

minhas_tarefas.append("Estudar Python")
minhas_tarefas.append("Fazer exercícios de lógica")
minhas_tarefas.append("Estudar Python")  # Tentativa duplicada

minhas_tarefas.mostrar_tarefas()
```

UserString

Fatec
Rio Claro

Strings podem ser consideradas um array de caracteres.

Em Python, um caractere é uma string de um comprimento e atua como um contêiner que envolve um objeto string.

Ele pode ser usado para criar strings com funcionalidades personalizadas.

UserString

Fatec
Rio Claro

```
from collections import UserString
class MyString(UserString):
    def append(self, value):
        self.data += value

s1 = MyString("data")
print("Original:", s1)
s1.append('h')
print("After append: ", s1)
```

Síntese

Fatec
Rio Claro

Nesta aula, exploramos os containers do módulo *collections* do Python, que oferecem estruturas de dados especializadas e mais eficientes para determinados cenários em comparação com os tipos integrados padrão.

O objetivo principal foi mostrar como esses containers ampliam as possibilidades de manipulação de dados no Python, tornando o código mais simples, expressivo e eficiente em determinadas tarefas e como integrá-las em algoritmos mais elaborados.

Esse conhecimento serve como base para escrever programas mais organizados e performáticos, preparando o terreno para estudos mais avançados em estruturas de dados e algoritmos.

Fatec
Rio Claro

Dúvidas

Prof. Orlando Saraiva Júnior
orlando.nascimento@fatec.sp.gov.br

Prática

Prática

Fatec
Rio Claro

Neste exemplo, o código representa um baralho de cartas no estilo francês.

Fonte: https://pythonfluente.com/2/#data_structures_part

Prática

Fatec
Rio Claro

Crie um sistema simples em Python para gerenciar a matrícula de alunos em disciplinas, aplicando os conceitos de orientação a objetos.

Crie uma estrutura de dados adequada para **Aluno** contendo o nome (string), matricula (string ou número identificador único) e uma lista de disciplinas nas quais o aluno está matriculado.

Crie uma estrutura de dados adequada para **Disciplina** contendo o nome (string), codigo (string ou número identificador único) e uma lista de alunos matriculados nessa disciplina.