Listas, Dicionários e Tuplas em Python











CRONOGRAMA DA AULA

1

Listas (list)

2

Dicionários (dict)

3

Tuplas (tuple)

4

Aplicações na Análise de Dados 5

Quando Usar Cada Estrutura

LISTAS (LIST)

Mutáveis

Acesso

Criação

Métodos



Seu conteúdo pode ser alterado após a criação.



Por índice numérico, começando em 0.



Utiliza colchetes []. Exemplo: minha_lista = [1, 'texto', 3.14]



append(), remove(), pop() para modificar elementos.

Dicionários (dict)

```
cn acnalue pairs
tiong key value
ey warlen catting;
dictionary /dictionary
 curly base(;
 (en key-value nein " curly-disticing) ( curly brane
    hen(pralixes = ())
      tutt key, fpalurs ()
    and (hà);
   <t cury farses)) {
     tant ((a;
    line key, value, " fenection), "
       curtiic stalunctientary
       ditticantiones data dictionay
       cfanily actictianchy ***
compdaine functients:
```

Estrutura de Pares

Armazenam dados em pares de chave-valor. Cada chave deve ser única e imutável.

Criação

Utiliza chaves {}. Exemplo: meu_dicionario = {'nome': 'Ana', 'idade': 30}

Acesso

Valores são acessados usando suas chaves: meu_dicionario['nome']

Métodos Úteis

keys(), values(), items() para iterar sobre elementos.

TUPLAS (TUPLE)

Imutáveis

Uma vez criadas, não podem ser modificadas.





Criação

Utiliza parênteses (). Exemplo: minha_tupla = (1, 'texto', 3.14)

Segurança

Ideais para dados que não devem mudar.





Acesso

Por índice numérico, como nas listas.

APLICAÇÕES NA ANÁLISE DE DADOS

Listas

Dicionários

Tuplas

Coletar dados de diferentes fontes

Representar registros com campos nomeados

Armazenar coordenadas (latitude, longitude)

Armazenar resultados de loops

Armazenar configurações

Representar datas

Preparar dados para processamento

Calcular frequências de itens

Retornar múltiplos valores de funções

QUANDO USAR CADA ESTRUTURA



Listas

Para coleções ordenadas que precisam ser modificadas.



Dicionários

Para associar valores a chaves únicas com acesso rápido.



Tuplas

Para coleções que não devem ser alteradas após criação.

RESUMO

Listas

Versáteis e mutáveis. Ideais para sequências que mudam.

Ordem de Avaliação

Eficientes para mapeamentos.
Perfeitos para dados estruturados por chaves.

Execução Seletiva

Seguras e imutáveis. Excelentes para dados fixos.

```
lists,
lists
list list
 fisit tumple;
 tupples {
     nut, int futhers:
   prticon, a vites
     lists "diectoriy" (tage)
   list 'fimers; Newt ding's
    (diectoryiry = by, on take
     list, imn comale;
  con top thertions")
        = tuple;
          tuplles; ⇒ byplar m.
   tuples (ist"sturtion
     firliss , list", anr 'tn @
```

Vamos avaliar o encontro?