SENAI

Ciência de Dados (CIDA)

Aula 2 –Estruturas de Dados

Prof.: Hugo S. Idagawa

Estruturas de Dados 1 - Listas

- Para facilitar a manipulação dos dados coletados durante uma análise estatística, é comum armazená-las em estruturas de dados.

- Até o momento, utilizamos a estrutura conhecida como lista ou vetor ou “array”, a qual possui a seguinte representação:

```
lista_dados = [1.8, 2.6, 7.8, 9.8]
```

- A estrutura acima permite que façamos algumas operações como a indexação individual de alguns elementos:

```
x = lista_dados[0]  # coloca em x o valor 1.8
y = lista_dados[-1] # coloca em y o valor 9.8
z = x + y     # soma x com y
print(z)      # imprime o valor 11.6
```

Estruturas de Dados 1 - Listas

- Algumas operações úteis sobre as listas:

```
append(valor)          # adiciona valor ao fim da lista
sort()                 # ordena a lista
len(lista_dados)       # retorna o tamanho da lista
range(início, fim, passo)
   # cria uma sequência de números, o valor fim é
   # obrigatório, início=0 (padrão), passo=1 (padrão)
```

- Ex: Para a lista de números abaixo, implemente um programa em python que responda os itens a seguir:

```
lista_dados = [-0.5, 1.8, 43.8, 52, 2.6, 7.8, 9.8, -2.5]
```

a) Imprima o número de elementos total da lista
b) Imprima a lista ordenada
c) Imprima o menor elemento da lista
d) Imprima os valores maiores do que zero da lista
e) Adicione o elemento 4.5 na lista na posição ordenada correta

Estruturas de Dados 1 - Listas

- Ex: Para a lista de números abaixo, implemente um programa em python que responda os itens a seguir:

```
dados = [45, 56, -89.0, 23.4, 1.5, 2.5, 5.5, 10.0, -50.0, 1.0]
```

a) Adicione o elemento -12.5 na lista
b) Calcule os valores dos limites superior e inferior de um boxplot desses dados
c) Crie uma nova lista onde os outliers foram removidos dos dados originais
d) Apresente o boxplot de ambas as listas

- Uma outra estrutura de dados bastante flexível e que permite armazenar mais informações por vez é o dicionário (ou “dict”).

Estruturas de Dados 1 - Dict

- Um dicionário é uma estrutura que associa valores com chaves e permite que você recupere o valor correspondente de uma dada chave rapidamente. Ele é definido no código utilizando as chaves “{” e “}”.

```
dict_vazio = {}      # dicionário vazio
notas = {"João" : 80, "Maria" : 95}      # dict de 2 chaves
```

- Para acessar o valor de uma chave do dicionário, fazemos a indexação de forma semelhante à lista:

```
valor = notas["João"]       # faz valor = 80
valor2 = notas["Maria"]     # faz valor = 95
```

- Assim como uma lista, podemos atribuir novos valores a um dicionário realizando a indexação da chave:

```
Notas["João"] = 100 # agora a nova nota de João é 100
```

Estruturas de Dados 1 - Dict

- Em estatística, é comum organizarmos os dados em diferentes chaves para melhor estruturar as informações:

```
dados = {"nomes" : ["João", "José", "Maria"],
         "idades" : [25, 29, 15],
         "notas" : [50, 20, 86]}
```

- Nesse exemplo, podemos acessar todos a lista de todos os nomes do dicionário da seguinte forme:

```
alunos = dados["nomes"]     # armazena a lista de alunos
print(alunos)               # imprime os nomes dos alunos
```

- Se quisermos acessar individualmente um único elemento, podemos acessar a chave como uma lista:

```
valor = dados["idades"][2] # valor é igual a 15
```

Estruturas de Dados 1

- Para modificar um determinado valor do dicionário, podemos indexar o elemento diretamente como uma lista:

```
# As linhas de código abaixo, modificam os valores do dict
dados["nomes"][1] = "ABC"
dados["idade"][0] = 60
print(dados)
```

- De forma semelhante à lista, podemos adicionar um elemento ao dicionário utilizando a função “append” das listas:

```
# As linhas de código abaixo, adicionam 3 valores ao dict
dados["nomes"].append("Carlos")
dados["idade"].append(20)
dados["notas"].append(5.5)
Print(dados)
```

Estruturas de Dados 1

- Ex: para a tabela de dados de seguros abaixo, implemente-a como um dicionário, desenhe o seu boxplot e imprima qual é o ramo de seguro que corresponde ao “outlier”.

RAMO	%
Automóvel	33,6
Saúde	14,0
Incêndio	12,9
Vida	12,2
Riscos Diversos	5,5
Habitação	5,3
Transporte	3,1
Acidentes Pessoais	2,9
Obrigatório Veículos	1,7
Riscos de Engenharia	1,0
Responsabilidade Civil *	0,9

Fonte (Fenaseg, in Exame, Fev / 93)

Estruturas de Dados 1 - Arquivos

- Até o momento, temos inserido os dados manualmente nos nossos programas, porém isso se torna inviável e sujeito a erros quando a quantidade de informações é muito grande.

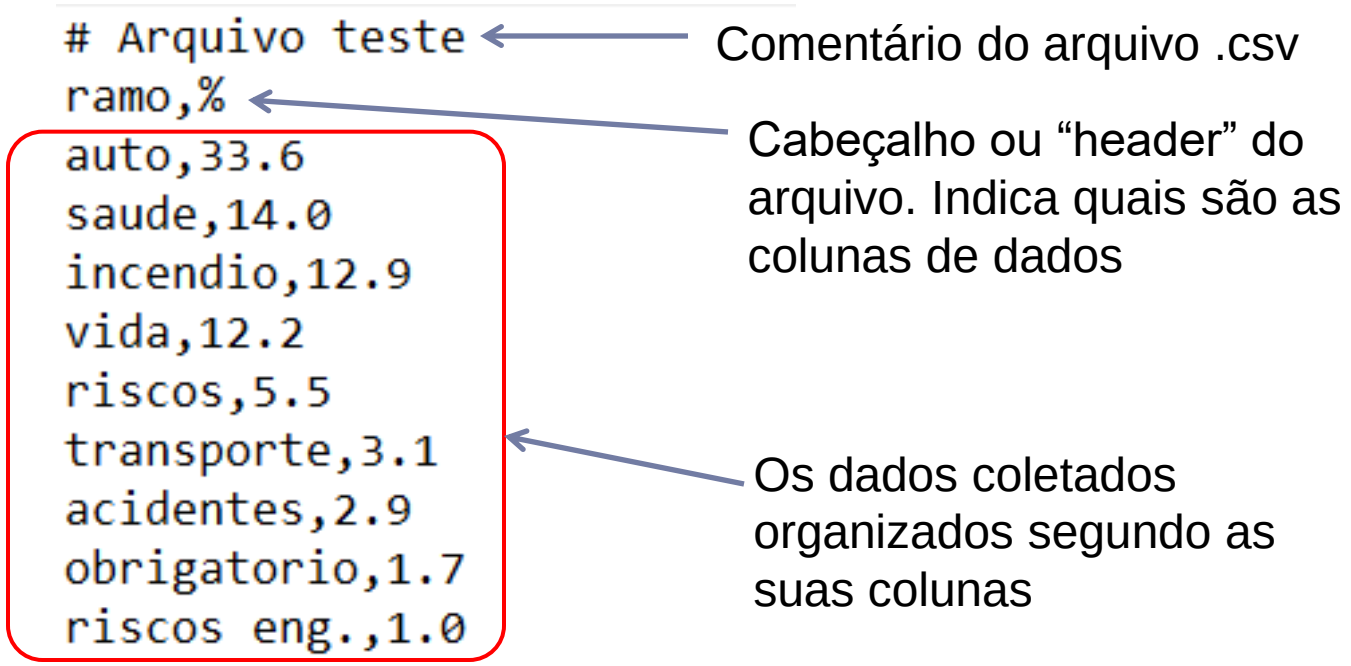
- Assim, uma solução é carregar esses dados diretamente de um banco de dados ou de um arquivo genérico. Em geral, essas informações podem ter sido alimentadas por um usuário, por um equipamento, ou mesmo, baixadas da internet.

- Um tipo de arquivo bastante comum para análise de dados é o csv ("**c**omma **s**eparatted **v**alues" ou valores separados por vírgula).

- Esses arquivos têm como a vantagem de serem facilmente visualizados e editados em um editor de textos simples, como o bloco de notas.

Estruturas de Dados 1 - Arquivos

- Abaixo temos um exemplo de arquivo .csv



Estruturas de Dados 1 - Arquivos

- Para sermos capazes de operar esses arquivos, podemos utilizar a biblioteca “pandas” do python que já possui as seguintes funções de manipulação de arquivos:

  - `read_csv(arquivo)`: função que lê um arquivo .csv
  - `read_excel(arquivo)`: função que lê um arquivo do Excel
  - `to_csv(arquivo)`: função que permite a escrita de um arquivo .csv
  - `to_dict()`: função que transforma os dados lidos de um arquivo para o formato de um dicionário.

- Assim, utilizando o arquivo “dados_seguros.csv” (localizado no repositório do github), realize a leitura desse arquivo e implemente o programa do último exemplo.