# Big Data & Data Science

*Parsing* de arquivos

# Formatos de arquivos

- Um arquivo nada mais é do que uma representação de algum tipo de dado, por exemplo, um documento, uma imagem, etc.
- Para identificar o dado presente no arquivo:
  - Extensão (.txt, .jpg, .pdf) → pode ser modificada…
  - Tipo do arquivo → cabeçalho
- Cabeçalho define características básicas para interpretação, estruturação, manipulação, visualização do arquivo em questão

# Formatos de arquivos comuns

Muitos dados virão comumente em:

- CSV
- JSON
- XML

Python possui módulos específicos para lidar com eles!

# Arquivos CSV

*Comma Separated Values*

- Organiza a informação em um “registro” por linha
- Cada coluna/campo dos registros é separado por um caracter delimitador, geralmente uma vírgula
- Arquivos CSV são fáceis de gerenciar/ler/processar
- Microsoft Excel e Google Spreadsheets suportam CSV

# O módulo CSV

Ao importar o módulo “csv”, temos classes para manusear arquivos com este formato. Algumas funções:

- csv.reader
- csv.writer
- csv.DictReader
- csv.DictWriter

https://docs.python.org/3/library/csv.html

# csv.reader

Este método obtém dados de um arquivo de dados CSV

- `csv.reader(arquivo, dialect='x', **fmtparams)`
- Parâmetros obrigatórios:
  - arquivo: um objeto do tipo arquivo (aberto)
- Parâmetros opcionais:
  - dialect: o modo como o arquivo deve ser interpretado (excel, unix)
  - fmtparams: parâmetros que sobrescreverão os especificados no dialeto

# Dialetos do módulo CSV

Como vê-los?

- >>> csv.list_dialects()
- ['excel', 'excel-tab', 'unix']

Mais informações:

- https://docs.python.org/3/library/csv.html#csv-fmt-params

# Análise exploratória dos dados

Um exemplo de arquivo:

```
nobody:*:-2:-2:Unprivileged User:/var/empty:/usr/bin/false
root:*:0:0:System Administrator:/var/root:/bin/sh
daemon:*:1:1:System Services:/var/root:/usr/bin/false
_uucp:*:4:4:Unix to Unix Copy Protocol:/var/spool/uucp:/usr/sbin/uucico
_taskgated:*:13:13:Task Gate Daemon:/var/empty:/usr/bin/false
```

# Análise exploratória dos dados

Como os campos são separados?

- `nobody:*:-2:-2:Unprivileged User:/var/empty:/usr/bin/false`

Vamos criar um “dialeto” novo com o separador:

- csv.register_dialect('unixpwd', delimiter=":")

O que temos agora?

- csv.list_dialects()
- ['excel', 'excel-tab', 'unix', 'unixpwd']

# Análise exploratória dos dados

```
...
>>> f=open("passwd")
>>> cf=csv.reader(f, dialect="unixpwd")
>>> for l in cf:
...   print(l)
```

O que aconteceu?

# csv.DictReader & csv.DictWriter

Similares à csv.reader e csv.writer, porém os dados são mapeados para *dicionários:*

```
arqDict = open(sys.argv[1])
csvDict = csv.DictReader(arqDict)
for linha in csvDict:
  print(linha)
```

# Exercício - CSV

Dado o arquivo “Graffiti_Sites.csv”, escreva um programa que:

- Mostre quantas e quais superfícies e tipos são listadas
- Mostre quantas e quais regiões aparecem no arquivo
- Faça o download da imagem e salve o arquivo com a extensão adequada. **Dica**: use o módulo “requests”.

# Arquivos JSON

*JavaScript Object Notation*

- Formato “leve” para troca de dados
- Independente de linguagem, facilmente compreensível
- Construído sobre as seguintes estruturas de dados: (1) coleção de tuplas “chave-valor” e (2) lista ordenada de valores
  - Objetos → { chave1: valor1, chave2: valor2, …}
  - Arrays → [ valor1, valor2, …, valorN]
  - Valor → string, número, objeto, array, true, false, null
- https://www.json.org

# Arquivo JSON simples


```
{
 "as":"AS14868 COPEL Telecomunicações S.A.",
 "city":"Curitiba",
 "countryCode":"BR",
 "lat":-25.4167,
 "lon":-49.25,
 "region":"PR",
 "timezone":"America/Sao_Paulo"
}
```

# Arquivo JSON simples

```
{
 "as":"AS14868 COPEL Telecomunicações S.A.",
 "city":"Curitiba",
...
}
```

```
CHAVE:valor, ...
```

# Exemplo - JSON

Vamos seguir os passos abaixo:

1. No terminal (ou browser), acessar a URL
   - Ex.: `wget http://ip-api.com/json/`
2. Com o arquivo obtido (index.html), faça
   - `mv index.html json1.json`
3. Abra seu interpretador Python favorito
   - Ex.: `python`

# Exemplo - JSON

```
>>> import json
>>> jf = open("json1.json").read()
>>> meuJson = json.loads(jf)
>>> type(meuJson)
<class 'dict'>
>>> meuJson
{'as': 'AS14868 COPEL Telecomunicações S.A.', 'city': 'Curitiba',
'country': 'Brazil', 'countryCode': 'BR', 'isp': 'COPEL Telecom', 'lat':
-25.4167, 'lon': -49.25, 'org': 'COPEL Telecom', 'query':
'138.204.25.252', 'region': 'PR', 'regionName': 'Parana', 'status':
'success', 'timezone': 'America/Sao_Paulo', 'zip': '74056'}
```

# Exemplo - JSON

```
>>> import json
```

A linha acima importa o pacote de processamento de arquivos JSON

```
>>> json.loads(jf)
```

A linha acima interpreta uma *string* em formato JSON

# Exemplo - JSON

Dado que o JSON carregado é um dicionário:

- Imprima as chaves
- Imprima os valores

# Exemplo - JSON

```
for chave in meuJson:
    print(chave)
```

```
as
city
…
lat
lon
...
```

```
for chave in meuJson:
    print(meuJson[chave])
```

```
AS14868 COPEL Telecomunicações S.A.
Curitiba
...
-25.4167
-49.25
...
```

# Alterando dados

```
>>> meuJson['status']
'success'

>>> meuJson['status'] = 'fail'

>>> meuJson['status']
'fail'
```

# Adicionando dados

```
>>> meuJson['bairros']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'bairros'

>>> meuJson['bairros'] = ['Reboucas', 'Centro']
```

# Removendo dados

```
>>> del meuJson['country']

>>> meuJson.pop('query')
'138.204.25.252'

>>> del meuJson['batata']
KeyError: 'batata'

>>> meuJson.pop('batata', None)
```

# Escrevendo um arquivo JSON

```
>>> saida = open("saida.json", "w")
```

```
>>> json.dump(meuJson, saida)
```

```
>>>saida.close()
```

Veja no seu diretório local o que tem dentro do arquivo “saida.json”

# Obtenção de JSON via *request*

Obtenção de dados via Web:

- import requests
- resposta = requests.get("https://jsonplaceholder.typicode.com/todos")
  - O que tem em “resposta”?
- arqJson = json.loads(resposta.text)

# Exercício – JSON

Dado o arquivo “report1613.json”, faça um programa que:

- Mostre o nome do arquivo
- Conte quantos antivírus (AVs) foram executados neste arquivo
- Conte quantos antivírus detectaram o arquivo como vírus
- Verifique se foram atribuídos rótulos iguais
- Crie um dicionário com chave == rótulo e valor == lista de Avs
- Escreva um novo JSON com o nome do arquivo e o dicionário de detecções obtido no item anterior

# Relembre: manipulação de *arrays*

- Vamos obter uma imagem PGM do tipo P2
  - A primeira linha é o tipo, a segunda o número de colunas e linhas, a terceira o maior pixel (ler em variáveis distintas)
  - Verificar se não há comentários (#)
  - O restante é uma matriz que representa a imagem
- Aplicar filtro de limiar para transforma-la em P&B
- Salvar em uma nova imagem .pgm

# Relembre: manipulação de *arrays*

- Vamos obter uma imagem PGM do tipo P2
  - ArqPGM = open(“ballons.pgm”); tipo = ArqPGM.readline(); col,lin = ArqPGM.readline().split(); pixel = int(ArqPGM.readline())...
  - Dados = ArqPGM.read().split(); NPDados = np.array(Dados, dtype=”?”).reshape((col, lin))
- Aplicar filtro de limiar para transforma-la em P&B

```
>>> limiar = 255/2
>>> for i in range(lin):
...   for j in range(col):
...     if narray[i][j] < limiar:
...        narray[i][j] = 0
...     else:
...        narray[i][j] = 255
```

- Salvar em uma nova imagem .pgm: `np.savetxt(out,narray,fmt="%d")`

# Exercício de fixação

- Obtenha imagens em formato PGM (ou converta-as) e aplique filtros de limiar diferentes (variando entre 0 e 1) e veja a diferença
- Caso sua imagem tenha sido convertida ou gerada em algum programam de manipulação (ex.: GIMP), veja se há comentários
- Verifique se seu PGM está em formato P2

# O módulo “os”

- os.listdir(<DIR>)
  - Lista arquivos em um dado diretório
  - ex.: os.listdir(“.”)
- os.path.join(strDir, strFile)
  - Monta um caminho completo de arquivo para, por exemplo, abrir

- EXERCÍCIO:
  - Dado o diretório “imagens”, aplique o filtro de limiar para cada uma das imagens e grave em novos arquivos distintos em um diretório criado por você via módulo “os”.

# Exercício para entrega HOJE

- Dados os JSON fornecidos pelo professor:
  - Faça o *parsing* dos arquivos usando o módulo “os”
  - Obtenha o dicionário de rótulos de antivírus
  - Crie uma estrutura para armazenar (em um arquivo JSON):
    - Os rótulos distintos e os antivírus que atribuíram tal rótulo
    - A frequência de aparecimento de um dado rótulo
    - Para cada arquivo, qual a taxa de detecção
    - Para o total de arquivos, qual o antivírus mais “eficaz”, i.e., que detectou mais arquivos como não-nulos
  - Mostrar os resultados de maneira ordenada