Python - Aula 2

Brenda Solari
Andrei Inacio
Heitor Lopes

CONTEÚDO

- O que é Modularização?
- Biblioteca de acesso à diretórios e arquivos;
- Manipulação de Arquivos texto;
- Arquivos CSV e JSON;
- Persistência de dados (arquivos binários);
- Trabalhando com Imagens (opencv);
- Biblioteca numpy;
- Biblioteca Argparse;

O que é Modularização?

- Divisão do sistema em módulos (procedimentos e funções)
- Vantagens:
  - Facilita manutenção do código.
  - Possibilita o desenvolvimento em equipe;
  - Permite a reutilização do código;

- Exemplo de função:

```
def calcularIdade(data_nascimento):
        // Comandos
        return idade

def obterDados():
        // Comandos
        return imagens_data, imagens_nome
```

O que é Modularização?

- Funções mesmo arquivo

```
def calcularIdade(data_nascimento):

        // Comandos
        return idade

idade1 = calcularIdade('18/01/2010')
idade2 = calcularIdade('12/05/2000')
```

- Funções em arquivos diferentes (mesmo diretório)

```
#arquivo funcoes.py na pasta dir
def calcularIdade(data_nascimento):
        // Comandos
    return idade
```

```
#arquivo executar.py
from dir.funcoes import calcularIdade

idade1 = calcularIdade('18/01/2010')
```

```
#arquivo executar.py
import dir.funcoes as f  # alias é opcional

idade1 = f.calcularIdade('18/01/2010')
```

- Criar arquivo __init__.py dentro dos diretórios (<= python 3.3)

O que é Modularização?

Modules in Python 3

```
sound/                          Top-level package
      __init__.py               Initialize the sound package
      formats/                  Subpackage for file format conversions
              __init__.py
              wavread.py
              wavwrite.py
              aiffread.py
              aiffwrite.py
              auread.py
              auwrite.py
              ...
      effects/                  Subpackage for sound effects
              __init__.py
              echo.py
              surround.py
              reverse.py
              ...
      filters/                  Subpackage for filters
              __init__.py
              equalizer.py
              vocoder.py
              karaoke.py
              ...
```

Bibliotecas de acesso à diretórios e arquivos

- Bibliotecas que permitem listar, criar, copiar mover e deletar arquivos e diretórios.
  - os, os.path, glob, shutil, io

- Necessária quando precisamos ler ou gravar dados do disco rígido.

```
import os
nomes = os.listdir(“./diretorio/”)

import glob
path_completo = glob.glob(“./diretorio/”)
```

Bibliotecas de acesso à diretórios e arquivos

- os
  - os.listdir (lista os arquivos/diretórios)
  - os.getcwd (retorna o diretório atual)
  - os.mkdir (cria um diretório)
  - os.remove (remove um arquivo ou diretório)
  - os.rename (renomeia um diretório)
  - os.link (cria link simbólico - atalho)
- os.path
  - os.path.exists (verifica se caminho existe)
  - os.path.isdir (verifica se é um diretório)
  - os.path.isfile (verifica se é um arquivo)
  - os.path.getsize (retorna o tamanho de um arquivo)
  - os.path.join (criação de path)

Bibliotecas de acesso à diretórios e arquivos

- glob
  - glob.glob (lista todos os arquivos/diretórios de um diretório)
  - Possibilita o uso de filtros.
- shutil
  - shutil.copyfile (copia um arquivo)
  - shutil.copy (copia arquivo ou diretório)
  - shutil.copytree (copia recursivamente um diretório)
  - shutil.move (move um arquivo ou diretório)
- io
  - io.open

Manipulação de arquivos texto

- Arquivo Texto
  - Utilizamos o comando **open**

arq = open(“nome.extensao”, “mode”)

mode	significado
‘r’	abre arquivo para leitura (default)
‘w’	cria arquivo para escrita
‘x’	exclusivo para criação (erro se arquivo existe)
‘a’	abre para escrita (adiciona ao final do arquivo se já existe)
‘b’	modo binário
‘t’	modo texto (default)
‘+’	atualização (escrita e leitura)

Manipulação de arquivos texto

- Escrita de texto

```
arq = open("teste.txt", “w”)
arq.write(“linha 1 \n”)
arq.write(“linha 2 \n”)
arq.close()
```

- Leitura de texto

```
arq = open("teste.txt", "r")
for linha in arq:
    print(linha)
arq.close()
```

- método readlines() lê todo o conteúdo do arquivo.

Arquivos CSV

Arquivo de texto, formato regulamentado pelo RFC 4180, usado para transferência de informações entre aplicações.

Fulano	10	Avenida Central
Beltrano	58	Av. das Américas
Ciclano		Rua 7 de setembro, 55

```
Fulano,10,"Avenida Central"
Beltrano,58,"Av. das Américas"
Ciclano,,"Rua 7 de setembro, 55"
```

```
import csv
csvfile = open('arquivo.csv')
data = csv.reader(csvfile, delimiter=',', quotechar='"')
for row in data:
    print(row[0]+" - "+row[1])
```

Arquivos JSON (JavaScript Object Notation - Notação de Objetos JavaScript)

Representação baseada em pares chave-valor usado para transferência de informações entre aplicações.

```
{
  “firstName”: “Jonathan”,
  “lastName”: “Freeman”,
  “loginCount”: 4,
  “isWriter”: true,
  “worksWith”: [“Spantree Technology Group”, “InfoWorld”],
  “pets”: [
    {
      “name”: “Lilly”,
      “type”: “Raccoon”
    }
  ]
}
```

```
import json
dict = {'nome': 'Fulano', 'idade': 22}
json.dump(dict, open('arquivo.json', 'w'))
```

```
import json
dict = json.load(open('arquivo.json', 'r'))
```

Manipulação de arquivos binário

- Escrita de arquivo binário

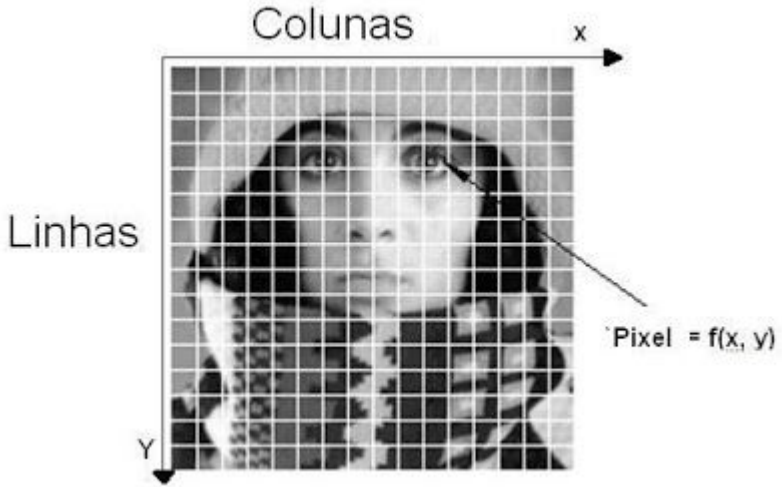
```
import pickle
dict = {"nome": "Fulano", "idade": 18}
arq = open("teste.pickle", "wb")
pickle.dump(dict, arq)
arq.close()
```

- Leitura de arquivo binário

```
import pickle
dict = pickle.load(open("teste.pickle", "rb"))
print(dict)
```

Trabalhando com Imagens

- Imagem digital: Matriz de *pixels* e cada elemento dessa matriz identifica intensidade (brilho) da imagem naquele ponto.
    - Tons de cinza: 1 canal (0-255)
    - RGB (colorida): 3 canais (0-255)
- opencv: Biblioteca para o desenvolvimento de aplicações na área de Visão Computacional.



Trabalhando com Imagens

Instalação: pip install opencv-python

import cv2

- Ler imagem:
  cv2.imread(`path, flag`)
  Flags: https://docs.opencv.org/3.4/d4/da8/group__imgcodecs.html
- Gravar imagem:
  cv2.imwrite(`path, img`)
- Mostrar imagem:
  cv2.imshow(`window_name, img`)
- Redimensionar imagem:
  cv2.resize(img, `dsize) #dsize->tupla com nova dim.`

```
* img (numpy array), path (string), dsize(tupla),
  flag(inteiro/constante)
```

https://docs.opencv.org/3.4/

Biblioteca Numpy

Biblioteca que suporta os tipos de dados arrays e matrizes multidimensionais e disponibiliza um grande conjunto de funções para trabalhar com essas estruturas.

As dimensões das matrizes são chamadas de eixo.

- Instalação da biblioteca:
  - pip install numpy

- Exemplo de uso:

```
import numpy as np
array_np = np.array( [ [1, 5, 8, 6, 0], [1, 2, 3, 4, 5] ] )
array_zeros = np.zeros( (8) )
array_ones = np.ones( (220,220) )
array_ones = np.ones( (220,220,3) )
```

Biblioteca Numpy

```
dados = np.ones((5,5))
```

- Alguns atributos do objeto numpy array :
  - ndarray.ndim (5)
  - ndarray.size (25)
  - ndarray.shape (5,5)
  - ndarray.dtype (float64)

- Acessar dados pelo índice
  - print(dados[0]) # Primeira linha
  - print(dados[0][2]) # Primeira linha, terceira coluna
  - print(dados[0:2]) # Primeira e segunda linha
  - print(dados[: , 1:2]) # Segunda coluna
  - print(dados[: , 2:]) # Da terceira até a última coluna
  - print(dados[: , -1]) # Última coluna

Biblioteca Numpy - Funções

- np.max(matriz) #maior valor da variável
- np.min(matriz)   #menor valor da variável
- np.max(matriz, axis=0) #maior valor no eixo 0 (colunas)
- np.max(matriz, axis=1) #maior valor no eixo 1 (linhas)
- np.mean(matriz) #média dos valores
- np.std(matriz) #Desvio padrão
- np.var(matriz) #Variância
- dados.reshape(3,6) # Altera a configuração da matriz
- dados.flatten() # Converte em uma matriz de 1 dimensão
- dados.T # matriz transposta
- np.add(mat1, mat2) #Soma de duas matrizes
- np.subtract(mat1, mat2) #Subtração de duas matrizes
- np.matmul(mat1, mat2) #Multiplica duas matrizes
- np.dot(mat1, mat2) #Multiplica dois vetores
- mat1 * mat2 #Multiplicação dos elementos de duas matrizes
- np.save(open('test.npy', 'wb'), mat1) # Salva matriz
- np.load(open('test.npy', 'rb')) # Salva matriz

Módulo argparse

- Parâmetros enviados via linha de comando são armazenados em sys.argv.
- argparse é um módulo que facilita a captura dos parâmetro recebidos via linha de comando.

```
import argparse
parser = argparse.ArgumentParser(description='Process some integers.')
parser.add_argument('--f', dest='file', action='store', required=True, help='file name ' )
parser.add_argument('--n', dest='num', action='store', type=int, default=1, help='Number of files (default: 1)')

args = parser.parse_args()
print(args.file)
print(args.num)
```

Exercício 1

Crie um sistema que recebe na linha de comando quatro parâmetros: um diretório de imagens (ou arquivo zip contendo as imagens compactadas); um diretório destino, onde as imagens serão armazenadas; o tamanho que as imagens serão redimensionadas, e um tamanho mínimo da imagem. O sistema deverá ler as imagens do diretório de origem, redimensioná-las de acordo com as orientações abaixo, e gravá-las no diretório destino:

- Serão redimensionadas apenas as imagens com tamanho (largura ou altura) maior que o tamanho mínimo da imagem recebido por parâmetro. Imagens que não atendem esse a critério devem ser ignoradas.
- O redimensionamento da imagem deverá ser proporcional ao maior tamanho (largura ou altura) da imagem. Por exemplo: dado uma imagem de tamanho 180x120 e o tamanho mínimo recebido é 150, a imagem final será 150x100.
- Um arquivo imagens.txt deverá ser criado para armazenar o nome e a classe das imagens redimensionadas. Cada linha do arquivo deverá ter a seguinte estrutura: nome_imagem,classe
- Um arquivo imagens_desconsideradas.txt deverá ser criado para armazenar o nome e o tamanho original das imagens não processadas. Cada linha do arquivo deverá ter a seguinte estrutura: nome_imagem,largura,altura.

Obs1: Crie um arquivo separado para implementar as funções necessárias.
Obs2: Se o primeiro parâmetro for um arquivo zip, usar biblioteca zipfile.

Exercício 2

No seguinte quadro tem as vendas de uma loja de roupas, correspondentes ao ano 2019 (**clothes_data.csv**):

month	pants	skirt	coat	jersey	hoodie	scarf	total_units	total_profit
1	250	150	520	920	120	150	2110	21100
2	263	120	510	610	210	120	1833	18330
3	214	134	455	955	355	134	2247	22470
4	340	113	587	887	187	113	2227	22270
5	360	174	456	776	156	174	2096	20960
6	276	155	489	749	189	155	2013	20130
7	298	112	478	898	178	112	2076	20760
8	370	140	586	996	286	140	2518	25180
9	354	178	610	810	210	178	2340	23400
10	199	189	830	1030	230	189	2667	26670
11	234	210	730	1330	240	210	2954	29540
12	290	176	740	1440	180	176	3002	30020

- Leia os dados desde um arquivo csv usando a biblioteca de sua preferência.

Exercício 2

- Leia o lucro total de todos os meses e mostre eles usando um gráfico de linhas (X: #mes e Y: lucro)
- Leia todos os dados de vendas do produto e mostre eles usando um gráfico de múltiplas linhas(X: #mes e Y: unidades vendidas em números) usando cores diferentes e lenda dos dados.
- Leia os dados de vendas de *pants* e *skirt* mostre eles usando o gráfico de barras (X: #mes e Y: unidades vendidas em números)

RECURSOS PARA O EXERCÍCIO

- pip install matplotlib
- Documentação
- Video

Recursos

- https://docs.python.org/3/reference/import.html
- https://docs.python.org/3/library/filesys.html
- https://docs.python.org/3/library/pickle.html
- https://docs.python.org/3/library/argparse.html
- https://docs.python.org/3/library/csv.html
- https://docs.python.org/3/library/json.html
- https://numpy.org/doc/stable/reference/arrays.indexing.html