

Aula 04

- Módulos e *Packages*
- Arquivos
 - Leitura e Escrita
 - Tipos: Texto e Binário
- Arquivos CSV
- Arquivos JSON
- Arquivos XML



MÓDULOS



- **Definição**
 - Um **módulo** é um pedaço de código que possui uma funcionalidade específica.
- Em Python um módulo é ***simplesmente um arquivo com extensão .py.***
- O nome do módulo é o nome do arquivo.

Módulos

```
calculadora.py ×  
1      #  
2      # Módulo calculadora.py  
3      #  
4  
5      def adicao(a,b):  
6          |      return a+b  
7  
8      def subtracao(a,b):  
9          |      return a-b
```

Módulos

- Importando ***todas*** as funcionalidades do módulo ***calculadora***

```
1  import calculadora
2
3  a = 10
4  b = 20
5
6  print(calculadora.adicao(a,b))
7  print(calculadora.subtracao(a,b))
8
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TE

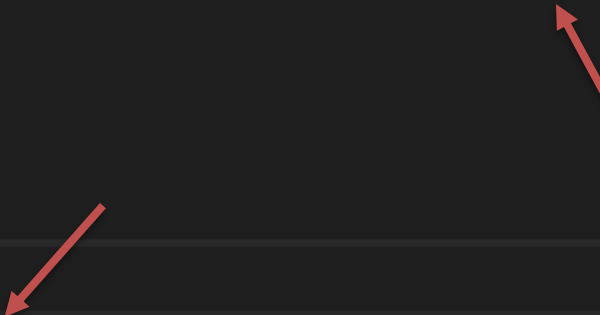
30

-10

Módulos

- Importando ***todas*** as funcionalidades do módulo ***calculadora*** ***com*** um novo preferencial

```
1  import calculadora as calc
2
3  a = 10
4  b = 20
5  |
6  print(calc.adicao(a,b))
7  print(calc.subtracao(a,b))
8
```



PROBLEMS

OUTPUT

DEBUG CONSOLE

30

-10

Módulos

- Importando ***apenas*** uma determinada funcionalidade

```
1  from calculadora import adicao as add
2
3  a = 10
4  b = 20
5
6  print(add(a,b))
7
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

30



PACKAGES



Packages

- Um pacote é simplesmente um diretório que cote um conjunto de módulos e um arquivo chamado `__init__.py`.
- Exemplo do pacote **Graphics**

```
Graphics/  
    __init__.py  
    Bmp.py  
    Jpeg.py  
    Png.py  
    Tiff.py  
    Xpm.py
```

Packages

- Para importar um módulo específico do pacote é utilizado a chamada:

```
import Graphics.Bmp  
image = Graphics.Bmp.load("bashful.bmp")
```

- O arquivo `__init__.py` possui apenas uma lista contendo os nomes dos módulos:

```
__all__ = ["Bmp", "Jpeg", "Png", "Tiff", "Xpm"]
```

ARQUIVOS

- Para manipulação de arquivos em Python não é necessário importar bibliotecas.
- O primeiro passo é utilizar a função ***open*** que retorna um tipo denominado ***file object***.
- Os arquivos podem ser do tipo:
 - ***Text Files***
 - Cada linha termina com um caractere especial chamado ***EOL***
– ***End of Line***
 - ***Binary Files***
 - Pode ser processado apenas por aplicações que conhecem a sua estrutura.

- Função *Open()*

file_object = open("filename", "mode")

Value	Description
'r'	Read mode
'w'	Write mode
'a'	Append mode
'b'	Binary mode (added to other mode)
'+'	Read/write mode (added to other mode)

Leitura de ***todas as linhas*** do arquivo

```
4 f = open("countries.txt", "r")
5
6 #todas as linhas
7 todas = f.read()
8 print(todas)
9 f.close()
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

```
Argentina
Bolívia
Brasil
Chile
Colômbia
Equador
Ilhas Falklands
Argentina
Bolívia
Brasil
Chile
Colômbia
Equador
Ilhas Falklands
Ilhas Geórgia do Sul
Guiana Francesa
Guiana
Paraguai
```

Leitura do arquivo

```
4  f = open("countries.txt", "r")
5
6  #primeira linha
7  linha = f.readline()
8  print(linha.strip("\n"))
9
10 #leitura dos primeiros 3 bytes
11 data = f.read(3)
12 print(data)
13
14 #todas as linhas como lista
15 f.seek(0)      #início do arquivo
16 linhas = f.readlines()
17 print(linhas)
18
19 f.close()
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

Argentina

Bol

['Argentina\n', 'Bolívia\n', 'Brasil\n', 'Chile\n', 'Guiana Francesa\n', 'Guiana\n', 'Paraguai\n', 'Peru\n', 'Uruguai\n', 'Venezuela\n']

Escrita no arquivo

```
27  #escrita
28  f = open("data.txt","w")
29  f.write("Primeira linha de texto \n")
30  f.write("Segunda linha de texto \n")
31  f.close()
32
33  #escrita no final
34  f = open("data.txt","a")
35  f.write("Terceira linha de texto \n")
36  f.close()
37
38  #leitura
39  f = open("data.txt","r")
40  print(f.read())
41  f.close()
42
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

```
Primeira linha de texto
Segunda linha de texto
Terceira linha de texto
```

Escrita no arquivo

```
44 lista = ["Segunda", "Terça", "Quarta", "Quinta",  
45          "Sexta", "Sabado", "Domingo"]  
46  
47 #modo leitura/escrita  
48 f = open("data.txt", "r+")  
49  
50 #escrever conteúdo da lista  
51 f.writelines(lista)  
52  
53 linhas = f.readlines()  
54 print(linhas)  
55 f.close()
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

```
[ 'SegundaTerçaQuartaQuintaSextaSabadoDomingo' ]
```

Leitura e Escrita de *Arquivos binários*

```
57 #módulo para serialização de dados
58 import pickle as pk
59
60 v1 = "Hello Word!"
61 v2 = ["João da Silva", "Ana Maria"]
62
63 #escrever arquivo binário
64 f = open("data.dat","wb")
65 pk.dump(v1,f)
66 pk.dump(v2,f)
67 f.close()
68
69 #leitura arquivo binário
70 f = open("data.dat","rb")
71 v1 = pk.load(f)
72 v2 = pk.load(f)
73 print(v1)
74 print(v2)
75 f.close()
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

```
Hello Word!
['João da Silva', 'Ana Maria']
```

- Manipulação de arquivos **CSV**

- ***Módulo***

- `import csv`

- ***Principais Funções***

- ***`csv.reader`***
 - ***`csv.writer`***

Leitura de um arquivo

```
1  import os
2  import csv
3  os.system("cls")
4
5  #leitura do arquivo
6  f = open("data/sample.csv","r",encoding="iso-8859-1")
7  reader = csv.reader(f)
8  for row in reader:
9      print(row)
10 f.close()
11
```

PROBLEMS TERMINAL ... 1: Python Debu ▾ + 1

```
['1', 'Eldon Base for stackable storage shelf, platinum', 'Muhan
e', '3', '-213.25', '38.94', '35', 'Nunavut', 'Storage & Organiz
']
['2', '1.7 Cubic Foot Compact "Cube" Office Refrigerators', 'Bar
'293', '457.81', '208.16', '68.02', 'Nunavut', 'Appliances', '0.
['3', 'Cardinal Slant-D® Ring Binder, Heavy Gauge Vinyl', 'Barry
93', '46.71', '8.69', '2.99', 'Nunavut', 'Binders and Binder Acc
0.39']
['4', 'R380', 'Clay Rozendal', '483', '1198.97', '195.99', '3.99
'Telephones and Communication', '0.58']
```

Leitura e escrita de uma *lista*

```
12  #escrita no arquivo
13  data = []
14  data.append((1, "João da Silva", "joao@joao.com"))
15  data.append([2, "Ana Maria", "ana@ana.com"])
16
17  f = open("data/output.csv", "w", newline='')
18  writer = csv.writer(f, delimiter=";")
19
20  for linha in data:
21      writer.writerow(linha)
22  f.close()
23
24  #leitura no arquivo
25  f = open("data/output.csv", "r", newline='')
26  reader = csv.reader(f)
27  for row in reader:
28      print(row)
29  f.close()
```

- Manipulação de arquivos **JSON**
 - *JavaScript Object Notation*
 - É uma sintaxe para armazenamento e troca de informações de texto.

```
{  
  "employees": [  
    { "firstName": "John" , "lastName": "Doe" },  
    { "firstName": "Anna" , "lastName": "Smith" },  
    { "firstName": "Peter" , "lastName": "Jones" }  
  ]  
}
```

- Manipulação de arquivos **JSON**
 - JSON é texto simples
 - JSON é "auto descritivo" (legível)
 - JSON é hierárquico (valores dentro dos valores)

Leitura e Escrita JSON – Exemplo 1

```
4  import json
5
6  data = {}
7  data["sp"] = "São Paulo"
8  data["rj"] = "Rio de Janeiro"
9  data["mg"] = "Minas Gerais"
10
11  #escrever formato JSON
12  f = open("data/output.json","w")
13  json.dump(data,f, sort_keys=True,indent=4)
14  f.close()
15
16  #ler formato JSON
17  f = open("data/output.json","r")
18  data = json.load(f)
19  f.close()
20
21  print(data)
```

PROBLEMS

TERMINAL

...


1: Python Debu ▾



```
{'mg': 'Minas Gerais', 'rj': 'Rio de Janeiro', 'sp': 'São Paulo'}
```

Leitura e Escrita JSON – Exemplo 1

```
4 import json
5
6 data = {}
7 data["sp"] = "São Paulo"
8 data["rj"] = "Rio de Janeiro"
9 data["mg"] = "Minas Gerais"
10
11 #escrever formato JSON
12 f = open("data/output.json","w")
13 json.dump(data,f, sort_keys=True,indent=4)
14 f.close()
15
16 #ler formato JSON
17 f = open("data/output.json")
18 data = json.load(f)
19 f.close()
20
21 print(data)
```



```
{
    "mg": "Minas Gerais",
    "rj": "Rio de Janeiro",
    "sp": "S\u00e3o Paulo"
}
```

PROBLEMS

TERMINAL

...

1: Python Debu ▾

+

🗑

^

```
{'mg': 'Minas Gerais', 'rj': 'Rio de Janeiro', 'sp': 'São Paulo'}
```

rique
ática

Leitura e Escrita JSON – Exemplo 2

```
11 data = {}
12 data["vermelho"] = {"nome": "Vermelho", "rgb": "255,0,0", "hex": "#FFFF00"}
13 data["verde"] = {"nome": "Verde", "rgb": "0,255,0", "hex": "#00FF00"}
14 data["azul"] = {"nome": "Azul", "rgb": "0,0,255", "hex": "#0000FF"}
15
16 #escrever formato JSON
17 f = open("data/output.json", "w")
18 json.dump(data, f, sort_keys=True, indent=4)
19 f.close()
20
21 #ler formato JSON
22 f = open("data/output.json", "r")
23 data = json.load(f)
24 f.close()
```

Leitura e Escrita JSON – Exemplo 2

```
11 data = {}
12 data["vermelho"] = {"nome": "Vermelho", "rgb": "255,0,0", "hex": "#FFFF00"}
13 data["verde"] = {"nome": "Verde", "rgb": "0,255,0", "hex": "#00FF00"}
14 data["azul"] = {"nome": "Azul", "rgb": "0,0,255", "hex": "#0000FF"}
15
16 #escrever formato JSON
17 f = open("data/output.json", "w")
18 json.dump(data, f, sort_keys=True, indent=4)
19 f.close()
20
21 #ler formato JSON
22 f = open("data/output.json", "r")
23 data = json.load(f)
24 f.close()
```

```
{
  "azul": {
    "hex": "#0000FF",
    "nome": "Azul",
    "rgb": "0,0,255"
  },
  "verde": {
    "hex": "#00FF00",
    "nome": "Verde",
    "rgb": "0,255,0"
  },
  "vermelho": {
    "hex": "#FFFF00",
    "nome": "Vermelho",
    "rgb": "255,0,0"
  }
}
```

Leitura e Escrita JSON – Exemplo 3



Star 31,161

Requests is an elegant and simple
HTTP library for Python. built for

Requests: HTTP for Humans

Release v2.18.4. ([Installation](#))

license Apache 2.0 wheel yes python 2.6, 2.7, 3.4, 3.5, 3.6 codecov 90% Say Thanks! 🙏

Requests is the only *Non-GMO* HTTP library for Python, safe for human consumption.

Note:

The use of **Python 3** is *highly* preferred over Python 2. Consider upgrading your applications and infrastructure if you find yourself *still* using Python 2 in production today. If you are using Python 3, congratulations — you are indeed a person of excellent taste. —*Kenneth Reitz*

```
pip install requests
```

Leitura e Escrita JSON – Exemplo 3

```
4 import json
5 import requests
6
7
8 url = 'https://fipe.parallelum.com.br/api/v1/carros/marcas'
9
10 data = requests.get(url=url)
11 print(data.json())
12
13 for marca in data.json():
14     print(marca)
15
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

```
{'nome': 'Gurgel', 'codigo': 24}
{'nome': 'HAFEI', 'codigo': 152}
{'nome': 'Honda', 'codigo': 25}
{'nome': 'Hyundai', 'codigo': 26}
{'nome': 'Isuzu', 'codigo': 27}
{'nome': 'JAC', 'codigo': 177}
```

- Manipulação de arquivos ***XML***
 - e**X**tensible **M**arkup **L**anguage
 - Linguagem definida pelo consórcio da W3C
<www.w3c.org/XML/>
 - Originalmente criada como uma linguagem de marcação.
 - Extensível
 - Usuários podem adicionar novas tags

Leitura e Escrita de Arquivos



```
<?xml version="1.0" encoding="UTF-8"?>
<frutas>
  <fruta>
    <id>1</id>
    <nome>Maça</nome>
    <cor>vermelha</cor>
    <preco_por_quilo>4.89</preco_por_quilo>
  </fruta>
  <fruta>
    <id>2</id>
    <nome>Banana</nome>
    <cor>amarelo</cor>
    <preco_por_quilo>2.45</preco_por_quilo>
  </fruta>
</frutas>
```



Leitura de um arquivo XML

```
4 import xml.etree.ElementTree as ET
5 from xml.dom import minidom
6
7 data = minidom.parse("data/sample.xml")
8
9 cds = data.getElementsByTagName("CD")
10 print("Qtd. CD: %d" %len(cds))
11
12 for cd in cds:
13     title = cd.getElementsByTagName("TITLE")[0]
14     artist = cd.getElementsByTagName("ARTIST")[0]
15     print("%-30s %-20s" %(title.firstChild.data,artist.firstChild.data))
16
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

Private Dancer
Midt om natten
Pavarotti Gala Concert
The dock of the bay
Picture book
Red

Tina Turner
Kim Larsen
Luciano Pavarotti
Otis Redding
Simply Red
The Communards

Leitura de um arquivo XML

```
4 import xml.etree.ElementTree as ET
5 from xml.dom import minidom
6
7 data = minidom.parse("data/sample.xml")
8
9 cds = data.getElementsByTagName("CD")
10 print("Qtd. CD: %d" % len(cds))
11
12 for cd in cds:
13     title = cd.getElementsByTagName("TITLE")[0].text
14     artist = cd.getElementsByTagName("ARTIST")[0].text
15     print("%-30s %-20s" % (title, artist))
16
```

<?xml version="1.0" encoding="UTF-8"?>

<CATALOG>

<CD>

<TITLE>Empire Burlesque</TITLE>

<ARTIST>Bob Dylan</ARTIST>

<COUNTRY>USA</COUNTRY>

<COMPANY>Columbia</COMPANY>

<PRICE>10.90</PRICE>

<YEAR>1985</YEAR>

</CD>

<CD>

<TITLE>Hide your heart</TITLE>

<ARTIST>Bonnie Tyler</ARTIST>

<COUNTRY>UK</COUNTRY>

<COMPANY>CBS Records</COMPANY>

<PRICE>9.90</PRICE>

<YEAR>1988</YEAR>

</CD>

PROBLEMS

OUTPUT

DEBUG CON

Private Dancer
Midt om natten
Pavarotti Gala Concert
The dock of the bay
Picture book
Red

Tina
Kim
Lucia
Otis
Simp
The

Escrita de um arquivo XML

```
17 data = {}
18 data["vermelho"] = {"nome": "Vermelho", "rgb": "255,0,0", "hex": "#FFFF00"}
19 data["verde"] = {"nome": "Verde", "rgb": "0,255,0", "hex": "#00FF00"}
20 data["azul"] = {"nome": "Azul", "rgb": "0,0,255", "hex": "#0000FF"}
21
22 cores = ET.Element("cores")
23 for cor in data.items():
24     item = ET.SubElement(cores, "cor", name=cor[0])
25     ET.SubElement(item, "nome").text = cor[1]["nome"]
26     ET.SubElement(item, "rgb").text = cor[1]["rgb"]
27     ET.SubElement(item, "hex").text = cor[1]["hex"]
28
29 f = open("data/output.xml", "wb")
30 f.write(b"<?xml version='1.0' encoding='UTF-8'?>")
31 f.write(ET.tostring(cores))
32 f.close()
```

Escrita de um arquivo XML

```
17 data = {}
18 data["vermelho"] = {"nome": "Vermelho", "rgb": "255,0,0", "hex": "#FFFF00"}
19 data["verde"] = {"nome": "Verde", "rgb": "0,255,0", "hex": "#00FF00"}
20 data["azul"] = {"nome": "Azul", "rgb": "0,0,255", "hex": "#0000FF"}
21
22 cores = ET.Element("cores")
23 for cor in data.items():
24     item = ET.SubElement(cores, "cor", name=cor[0])
25     ET.SubElement(item, "nome").text = cor[1]["nome"]
26     ET.SubElement(item, "rgb").text = cor[1]["rgb"]
27     ET.SubElement(item, "hex").text = cor[1]["hex"]
28
29 f = open("data/output.xml", "wb")
30 f.write(b"<?xml version='1.0' encoding='UTF-8'>")
31 f.write(ET.tostring(cores))
32 f.close()
```

```
<?xml version='1.0' encoding='UTF-8'>
<cores>
  <cor name="vermelho">
    <nome>Vermelho</nome>
    <rgb>255,0,0</rgb>
    <hex>#FFFF00</hex>
  </cor>
  <cor name="verde">
    <nome>Verde</nome>
    <rgb>0,255,0</rgb>
    <hex>#00FF00</hex>
  </cor>
  <cor name="azul">
    <nome>Azul</nome>
    <rgb>0,0,255</rgb>
    <hex>#0000FF</hex>
  </cor>
</cores>
```

ATIVIDADE PRÁTICA

- **Exercício 1:** Escreva um programa em Python capaz de ler as primeiras ***n*** linhas de um arquivo texto. O número de linhas é informado pelo usuário.

- ***Exercício 2:*** Escreva um programa que realiza a leitura um arquivo e armazena o conteúdo em uma lista.

- ***Exercício 3:*** Escreva um programa capaz de contar a frequência de palavras em um arquivo.

- ***Exercício 4:*** Escreva um programa capaz de copiar o conteúdo de um arquivo em outro arquivo.

- **Exercício 5:** Escreva um programa em Python capaz de:
 - Carregar um arquivo no formato CSV em um Lista
 - Gravar o conteúdo da Lista em um arquivo JSON

- **Exercício 6:** Escreva um programa que determina o preço de um veículo a partir da **FIPE API**. O veículo pode ser informado pelo usuário.