Fundamentos do *Python*

Python for Data Analysis - Cap. 2

Interpretador *Python* (2.1)

- Linguagem interpretada;
- Executa instruções linha a linha;
- Prompt padrão (>>>);
- Uso no terminal (*python*);
- Execução de *scripts* (python arquivo.py).

```
$ python
Python 3.10.4 | packaged by conda-forge | (main, Mar 24 2022, 17:38:57)
[GCC 10.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> a = 5
>>> print(a)
5
```

Figura 1: Representação do interpretador *Python* sendo iniciado

IPython (2.2)

- *Shell* interativo melhorado;
- *Prompt* diferenciado (In [ ]);
- Execução de comandos *Python*;
- Resultados exibidos automaticamente.

```
$ ipython
Python 3.10.4 | packaged by conda-forge | (main, Mar 24 2022, 17:38:57)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.31.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]: a = 5

In [2]: a
Out[2]: 5
```

Figura 2: Representação do interpretador *IPython* sendo iniciado

Recursos *IPython* (2.2)

- ***Tab Completion*:** autocompletar comandos e variáveis;
- ***Introspection*:** ? e ?? mostram informações sobre objetos;
- ***%run*:** executar *scripts* diretamente.

```
In [3]: b = [1, 2, 3]

In [4]: b.<Tab>
append()  count()   insert()  reverse()
clear()   extend()  pop()     sort()
copy()    index()   remove()
```

```
In [1]: b = [1, 2, 3]

In [2]: b?
Type:        list
String form: [1, 2, 3]
Length:      3
Docstring:
Built-in mutable sequence.
```

Figuras 3 e 4: Representação dos recursos de *Tab Completion* e *Introspection* do interpretador *IPython*

```
$ ipython
Python 3.10.4 | packaged by conda-forge | (main, Mar 24 2022, 17:38:57)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.31.1 -- An enhanced Interactive Python. Type '?' for help.

In [1]: %run hello_world.py
Hello world

In [2]:
```

Figura 5: Representação do recurso de *%run* do interpretador *IPython*

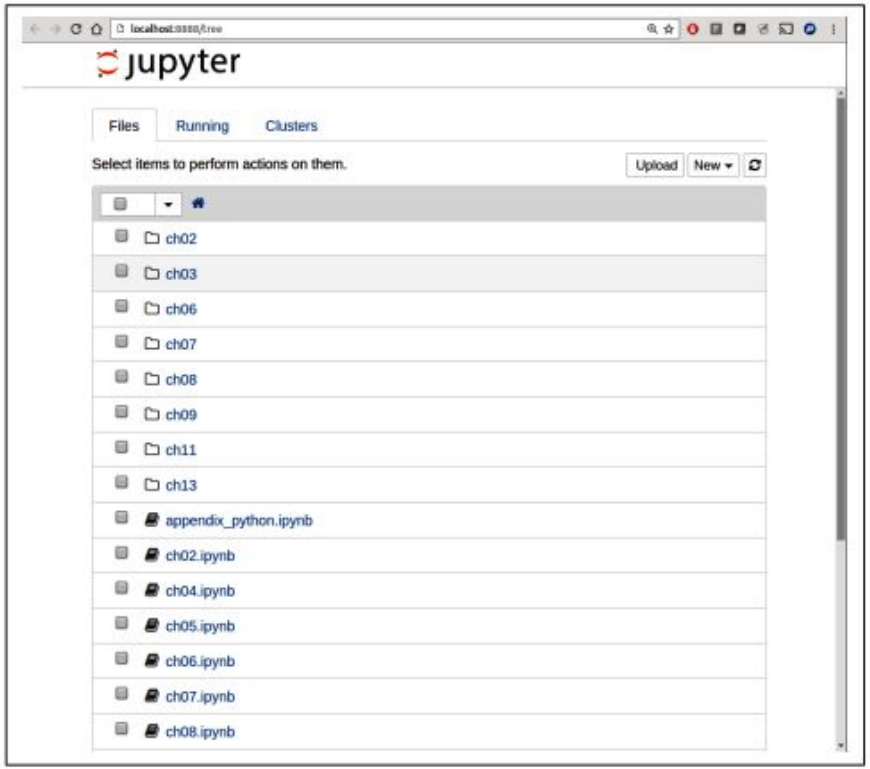
Jupyter Notebook

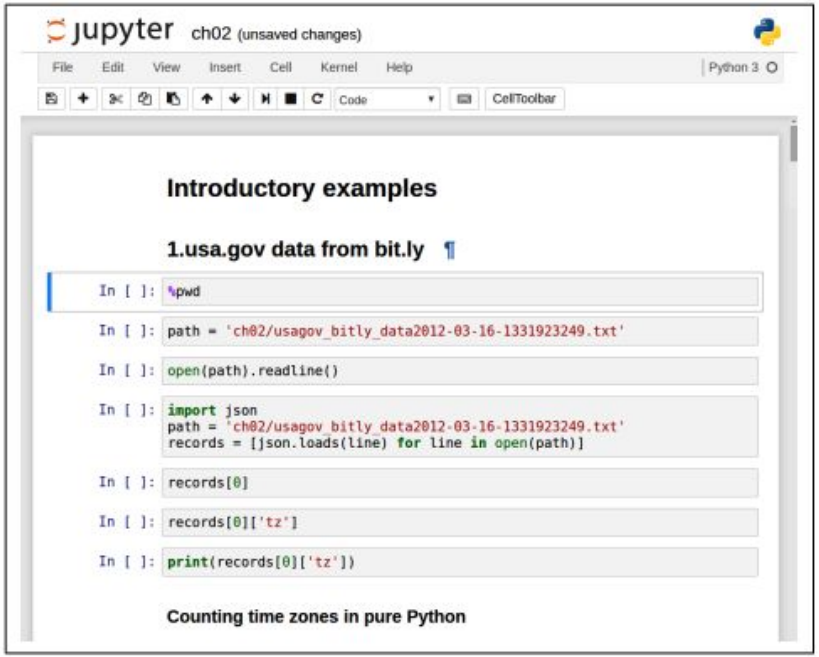
- Documentos interativos;
- Código + texto + visualizações;
- Arquivos .ipynb;
- Fácil execução e compartilhamento.

```
$ jupyter notebook
[I 15:20:52.739 NotebookApp] Serving notebooks from local directory:
/home/wesm/code/pydata-book
[I 15:20:52.739 NotebookApp] 0 active kernels
[I 15:20:52.739 NotebookApp] The Jupyter Notebook is running at:
http://localhost:8888/?token=0a77b52fefe52ab83e3c35dff8de121e4bb443a63f2d...
[I 15:20:52.740 NotebookApp] Use Control-C to stop this server and shut down
all kernels (twice to skip confirmation).
Created new window in existing browser session.
```

Figura 6: Representação do *Jupyter Notebook* sendo iniciado

Jupyter Notebook - localhost





Figuras 7 e 8: *Jupyter Notebook* aberto localmente via navegador *web*.

Fundamentos do *Python*

- Ênfase em legibilidade e simplicidade;
- Indentação obrigatória;
- Tudo é objeto;
- Comentários (#).

```
results = []
for line in file_handle:
    # keep the empty lines for now
    # if len(line) == 0:
    #   continue
    results.append(line.replace("foo", "bar"))
```

Figura 9: Exemplo de código *Python* com comentários

Controle de fluxo

- Condicionais: if, elif, else;
- Laços: for, while;
- break, continue, pass;
- Função range().

```
In [17]: a = 5

In [18]: type(a)
Out[18]: int

In [19]: a = "foo"

In [20]: type(a)
Out[20]: str
```

```
for x in array:
    if x < pivot:
        less.append(x)
    else:
        greater.append(x)
```

```
In [21]: "5" + 5
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-21-7fe5aa79f268> in <module>
----> 1 "5" + 5
TypeError: can only concatenate str (not "int") to str
```

Figuras 10, 11 e 12: Exemplos de códigos *Python* com o uso do laço de repetição *for*, função pronta *type()* e erro de tipo de dado (*TypeError*)

Conclusão

- Foco nos fundamentos do *Python*, *IPython* e *Jupyter*;
- Base para capítulos seguintes;
- Essencial para análise de dados.