

Introdução à Linguagem Python

Prof. Cláudio Fleury

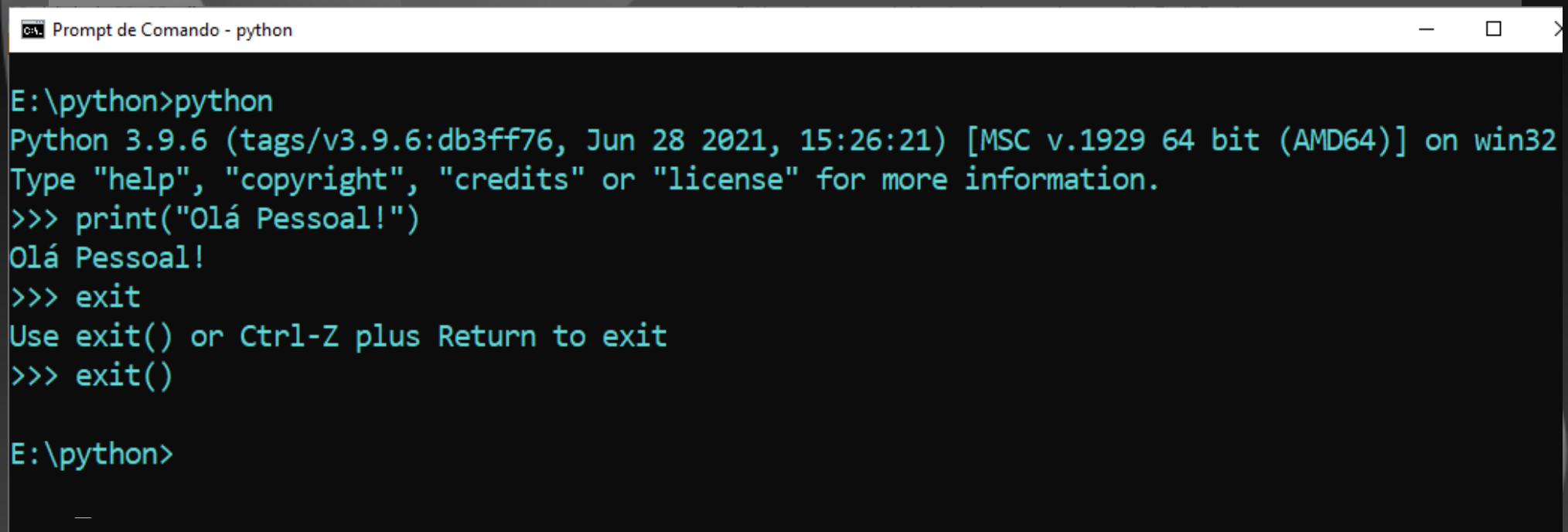
Set-22



Shells

Interpretador

- Do Inglês: concha, casca, aparência, aspecto exterior
- **Interpretador Python** (linha de comando)
 - Sem menus
 - Sem facilidades de edição de arquivos



```
Prompt de Comando - python

E:\python>python
Python 3.9.6 (tags/v3.9.6:db3ff76, Jun 28 2021, 15:26:21) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Olá Pessoal!")
Olá Pessoal!
>>> exit
Use exit() or Ctrl-Z plus Return to exit
>>> exit()

E:\python>
```

Shells

- **IDLE** (do Inglês: inativo, ocioso, desocupado)
 - Disponível em todas as distribuições Python
 - Ferramenta útil para programação diária
 - Tem editor com realce de sintaxe
 - *Prompt* com preenchimento automático (TAB) de nomes de variáveis, nomes de arquivos e comandos

The screenshot shows the IDLE Shell 3.9.5 window. The 'File' menu is circled in red. Below it, the shell prompt shows the Python version and system information. The 'Run' button in the toolbar is also circled in red. The code editor shows a Python script with the following content:

```
print("Olá pessoal!")
print("Vamos programar com Python!")
```

The shell output shows the execution of the script, displaying the messages 'Olá pessoal!' and 'Vamos programar com Python?'.

Shells

- **IPython** - Python Interativo <http://ipython.org/ipython-doc/stable/interactive/qtconsole.html>
 - Versão melhorada da versão Python de linha de comando
 - Ajuda com `help(comando)` ou `comando?`
 - *Prompt* com preenchimento automático (TAB)
 - Acesso a diretórios: `!dir`, `pwd`, `cd`
 - Histórico de edição multilinha
 - Comandos mágicos
 - `%load` carrega arquivo do disco ou formulário URL p/ edição
 - `%timeit` mede tempo de execução de um *script*

Shells

- **IPython Notebook** - documento que executa, armazena, carrega sequência de comandos Python, incluindo texto explicativo, imagens e outras mídias
- Útil para
 - Documento cálculos e processamento de dados
 - Apoia o aprendizado e ensino do Python, Estatística, Engenharia...
 - Documentação de código novo
- Comandos mágicos
 - No início da célula: `%%file nomearq.txt`
salva o conteúdo da célula em que se encontra o comando
 - Sinal de exclamação executa comandos do S.O.
`!python hello.py`

Shells

- **Spyder** - Scientific PYthon Development EnviRonment
- Instalado com o **Anaconda** (plataforma profissional para desenvolvimento com Python)
- Características
 - Poderoso ambiente de desenvolvimento interativo para a linguagem Python
 - Edição avançada, testes interativos, recursos de depuração (display de var.s, explorador de objetos etc.) e introspecção
 - Ambiente de computação numérica graças ao suporte do Ipython e das bibliotecas (pacotes) **Scipy**, **NumPy** e **matplotlib**

Shells

The image shows the Spyder Python IDE interface. The main window is titled "Spyder (Python 3.8)". The menu bar includes File, Edit, Search, Source, Run, Debug, Consoles, Projects, Tools, View, and Help. The toolbar contains various icons for file operations, running, and debugging. The file explorer on the left shows the current file as "untitled0.py*". The code editor displays a Python script with the following content:

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Tue Sep  6 15:59:43 2022
4
5 @author: profc
6 """
7
8
```

The right sidebar contains a "Usage" box with the text: "Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console." Below this are tabs for Help, Variable explorer, Plots, and Files. The console window at the bottom right shows the following output:

```
Python 3.8.3 (default, Jul 2 2020, 17:30:36) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.16.1 -- An enhanced Interactive Python.

In [1]:
```

The status bar at the bottom indicates "LSP Python: ready", "Kite: ready (no index)", "conda: base (Python 3.8.3)", "Line 8, Col 1", "UTF-8", "CRLF", "RW", and "Mem 85%".

Shells

- **Visual Studio Code** (VS Code) - Produto MS gratuito
- Plataforma profissional para desenvolvimento com Python
- Características
 - Poderoso ambiente de desenvolvimento interativo para a linguagem Python
 - Edição avançada, testes interativos, recursos de depuração (display de var.s, explorador de objetos etc.) e introspecção
 - Ambiente de computação numérica graças ao suporte do Ipython e das bibliotecas (pacotes) **Scipy**, **NumPy** e **matplotlib**

Shells

The image shows a Visual Studio Code editor window titled "admin.py - mediall-v3.4 - Visual Studio Code". The editor is displaying a Python script in a file named "admin.py". The script is part of a Django application and is used for processing payments. It includes a Django model named "psm_tmp" and a method named "processar_pagtos" that queries the database and generates a document.

The left sidebar shows the "RUN AND DEBUG" panel with the "Python: Django" interpreter selected. Below it, the "LOCALS" panel shows the current state of the program, including variables like "aliv", "cabec", "cursor", "detalhe", "dto", "fol", "ger", "id_leg", "leg", "lin", "linha", "linhas", "liv", "livro", "op", "prest", and "ps".

The main editor area shows the following code:

```
504 # numeração das folhas/termos: psm_tmp ordenada por top (tipo da oper.), dto (data da oper.), nom (nome dest.)
505 tmp.execute((" SELECT *, (row_number() OVER (ORDER BY length(A.cpf_cnpj) DESC, tac, nom, top, dto) - 1) as nr "
506             "FROM psm_tmp A "
507             "INNER JOIN psm_profissional B ON A.cpf_cnpj = B.cpf_cnpj "
508             "WHERE tac = 'P'"))
509 qsf2 = self.dictfetchall(tmp)
510 prest = ''; folha = {}
511 for op in qsf2:
512     if op['top'] == 'A': # saldo de acoes da tesouraria
513         tesouraria = op['sub']
514     if op["cpf_cnpj"] != prest:
515         prest = op["cpf_cnpj"]
516         cabec.append([op["nome_dest"], op["cpf_cnpj"], d2s(op["data_nasc"]), op["nacion"], op["endereco"].strip(),
517                     fmt_cep(op["cep"]), op["cidade"], op["est_civ"], op["doc_ident"]])
518     if len(op["cpf_cnpj"]) == 11:
519         fol = folha[op["ger"]]
520     else:
521         folha[op["ger"]] = op['nr'] # guarda numeração das folhas nas operações da holding
522         fol = 1
523     # det: [linha, data_registro, data_operacao, saldo/qtde_acao, dife_acao, data_upag, folha]
524     detalhe.append([ op['lin'], d2s(op['dto']), d2s(op['dto']), op['sub'], op['aliv'], op['liv'], fol ])
525     arq = cria_documento(cabec, detalhe, livro)
526     messages.success(request, f'Livro de ações gerado: {arq}.')
527 else:
528     messages.warning(request, f'Nenhuma ação processada para o livro {arq}.')
529
530 @admin.display(description='Processar Pagamentos')
531 def processar_pagtos(self, request, queryset): ...
```

The bottom of the editor shows the "TERMINAL" panel with the following output:

```
grade = Table(lista, colWidths=larguras, rowHeights=alturas, repeatRows=3)
File "e:\python\prjts_django\mediall\mediall-v3.4\avm\lib\site-packages\reportlab\platypus\tables.py", line 308, in __init__
raise ValueError("%s data error - %d rows in data but %d in row heights" % (self.identity(), nrow, len(rowHeights)))
ValueError: <Table@0x132895C61F0 4 rows x 30 cols> with cell(0,0) containing
'Das Ações, Sua Integralização ' data error - 4 rows in data but 22 in row heights
[06/Sep/2022 14:45:37] "POST /admin/psm/livro/ HTTP/1.1" 500 172439
```

The status bar at the bottom shows the current file is "admin.py" at line 524, column 1, with 3 spaces, UTF-8 encoding, CRLF line endings, and Python 3.9.6 (avm: venv) interpreter.

Calculadora

Calculadora

- Python é uma linguagem interpretada
 - Podemos colocar uma sequência de comandos em um arquivo texto, salvá-los como um programa Python (*script*), com o nome do arquivo tendo extensão **.py**, convencionalmente
 - Podemos tbm entrar comandos individuais no prompt do interpretador Python (sinal `>>>`) que serão avaliados imediatamente – interatividade: usuário pode validar um comando a qualquer momento...

Laço: lê, avalia, mostra

(**REPL**-Read, Evaluate, Print, Loop)

- Exemplo:
`>>> 2 + 2`
`4`
`>>>`

Calculadora

- Operações Aritméticas
 - adição (+), subtração (-), multiplicação (*), divisão (/) e potenciação (**)
- Exemplos
 - $10 + 10000 \rightarrow 10010$
 - $42 - 1.5 \rightarrow 40.5$
 - $47 * 11 \rightarrow 517$
 - $10 / 0.5 \rightarrow 20$
 - $2 ** 5 \rightarrow 32$
 - $3 ** 0.5 \rightarrow 1.7320508075688772$
 - ...

- Funções Matemáticas
 - Disponíveis no pacote **math**: `import math`
 - `exp()`, `log()`, `log10()`, `sqrt()`, `sin()`, `cos()`, ...
- Exemplos
 - `math.exp(1.0)` → 2.718281828459045
 - `dir(math)` → ['acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'comb', 'copysign', 'cos', 'cosh', 'degrees', 'dist', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'isqrt', 'lcm', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'nextafter', 'perm', 'pi', 'pow', 'prod', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc', 'ulp']
 - `help(math.exp)` → Help on built-in function exp in module math:
`exp(x)`: Return **e** raised to the power of x.

Calculadora

- Variáveis

- Uma variável é usada para armazenar um determinado valor ou objeto. Em Python, todos os números (e tudo mais, incluindo funções, módulos e arquivos) são objetos
- Uma variável é criada por meio de atribuição: `x = 0.5`

- Exemplos

- `y = 0.7`
`math.sin(y) ** 2 + math.cos(y) ** 2` → 1.0
- `larg = 20`
`alt = 5 * 8`
`larg * alt` → 800
- `x = y = z = 0` # inicia x, y e z com 0
`x, y, z` → (0, 0, 0)
- Notação concisa: `x = x + c` pode ser `x += c`



Tipos de Dados e Estruturas de Dados

Tipos de Dados

- Use o comando `type()` para ver o tipo de dado da variável
- Exemplos
 - `a = 45`
`type(a)` → `int`
 - `b = 'Olá Pessoal!'`
`type(b)` → `str`
 - `c = 3 - 2j`
`type(c)` → `complex`
 - `d = ["João", 17, 'M', false]`
`type(d)` → `list`
 - `e = {'id':343, 'desc':'âmpola', 'mod':22}`
`type(e)` → `dict`
 - `f = 25.3`
`type(f)` → `float`

Estruturas de Dados

- **Strings, listas e tuplas** são estruturas do tipo **sequência**
- Elas podem ser indexadas e fatiadas da mesma maneira
- **Tuplas** e **strings** são “imutáveis” (não se pode alterar elementos dentro da **tupla** e não se pode alterar caracteres de uma **string**), enquanto as **listas** são “mutáveis”
- Operações
 - **a[i]** retorna o i-ésimo elemento de **a**
 - **a[i:j]** retorna elementos **i** até **j-1**
 - **len(a)** retorna o número de elementos na sequência
 - **min(a)/max(a)** retorna o menor/menor valor na sequência
 - **x in a** retorna True se **x** for elemento em **a**
 - **a + b** concatena **a** e **b**
 - **n * a** cria **n** cópias da sequência **a**

Estruturas de Dados - String

- Exemplos
 - `a = 'Olá Pessoal!'`
 - `a = "Olá Pessoal!"`
 - `a = """Olá Pessoal!"""`
 - `len(a) → 12`
 - `"Olá" + "Pessoal!" → "Olá Pessoal!"`
 - `a.upper() → "OLÁ PESSOAL!"`
 - método **split()** converte uma string em uma lista de strings:
`a = "O Flamengo é campeão!"`
`a.split() → ['O', 'Flamengo', 'é', 'campeão!']`
`a = "Cachorro com fome. O gato lento. A cobra atenta."`
`a.split(".") → ['Cachorro com fome', ' O gato lento', ' A cobra atenta', '']`

Fonte(s)

- Fangohr, H.; Python for Computational Science and Engineering; 2018; DOI: 10.5281/zenodo.1411868; Disponível em: <https://github.com/fangohr/introduction-to-python-for-computational-science-and-engineering>