

# Desenvolvimento Aberto



**Documentação de API e linters**

Igor dos Santos Montagner ( [igorsm1@insper.edu.br](mailto:igorsm1@insper.edu.br) )

# Dinâmicas online

- Todo calendário foi adiado em uma semana
  - Datas limite de entregas de skills também
- Aulas continuam iguais (expositiva pequena + prática)
  - Dúvidas pelo chat
  - Pedir atendimento usando Forms no canal *General*
  - Ligação direta para dúvidas
- Recados gerais dados na aba *General*

# Código vs software profissional

Os seguintes pontos transformam um código que fiz para mim em algo útil para outras pessoas

1. Traduções e internacionalização (datas)
2. Documentação de usuário e de desenvolvimento
3. Algum processo de qualidade de software
  - testes automatizados
  - formatação de código e estrutura de repo
4. pacotes de instalação

# Hoje

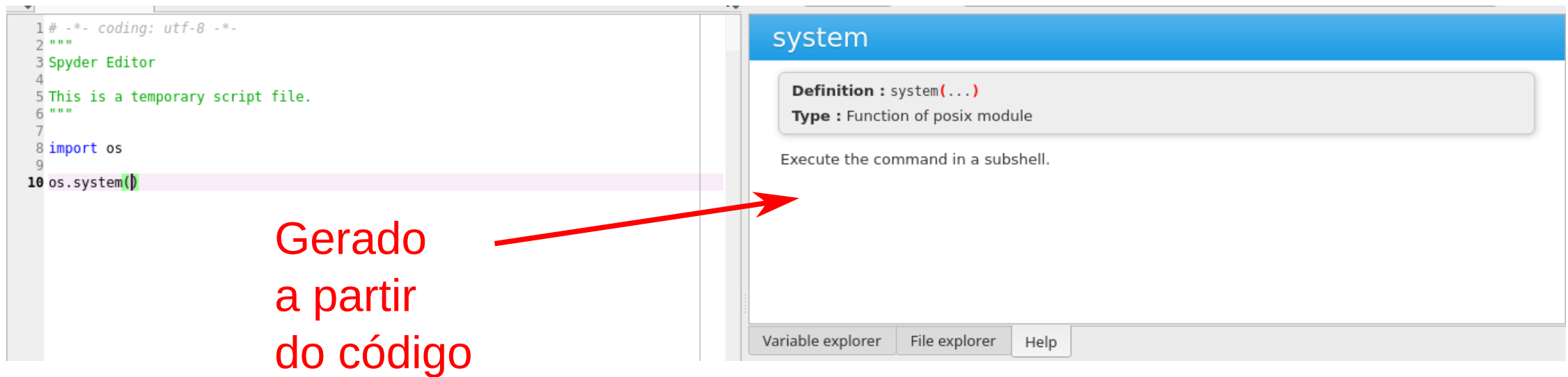
- Ferramentas de documentação
- Documentação de API usando
- Padrões de formatação de código
  - linters
  - PEP8

# Documentação de API

**Objetivo:** explicar o funcionamento das funções, classes e módulos de um programa.

- Focado em detalhes
- Documenta os argumentos esperados e em quais situações a função funciona
- Tipicamente obtida direto do código
- Não detalha como as funções são usadas em conjunto

# Documentação de API



The image shows a Spyder IDE window with a code editor on the left and a help pane on the right. The code editor contains a Python script with a comment and a call to `os.system()`. The help pane displays the documentation for the `system` function, including its definition, type, and description. A red arrow points from the `os.system()` call in the code to the help pane.

```
1 # -*- coding: utf-8 -*-  
2 """  
3 Spyder Editor  
4  
5 This is a temporary script file.  
6 """  
7  
8 import os  
9  
10 os.system()
```

**system**

**Definition :** `system(...)`



**Type :** Function of posix module

Execute the command in a subshell.

Variable explorer   File explorer   Help

Gerado  
a partir  
do código

# Documentação de API

SciPy.org Docs NumPy v1.15 Manual NumPy Reference Routines Linear algebra ( `numpy.linalg` )

index next previous

## numpy.dot

`numpy.dot(a, b, out=None)`

Dot product of two arrays. Specifically,

- If both *a* and *b* are 1-D arrays, it is inner product of vectors (without complex conjugation).
- If both *a* and *b* are 2-D arrays, it is matrix multiplication, but using `matmul` or `a @ b` is preferred.
- If either *a* or *b* is 0-D (scalar), it is equivalent to `multiply` and using `numpy.multiply(a, b)` or `a * b` is preferred.
- If *a* is an N-D array and *b* is a 1-D array, it is a sum product over the last axis of *a* and *b*.
- If *a* is an N-D array and *b* is an M-D array (where  $M \geq 2$ ), it is a sum product over the last axis of *a* and the second-to-last axis of *b*:

```
dot(a, b)[i,j,k,m] = sum(a[i,j,:]* b[k,:,m])
```

**Parameters:**

- a : array\_like**  
First argument.
- b : array\_like**  
Second argument.
- out : ndarray, optional**  
Output argument. This must have the exact kind that would be returned if it was not used. In particular, it must have the right type, must be C-contiguous, and its dtype must be the dtype that

Previous topic  
[Linear algebra \( `numpy.linalg` \)](#)

Next topic  
[numpy.linalg.multi\\_dot](#)

Quick search

# Ferramentas

- Python:
  - pydoc, numpydoc
- C/C++
  - Doxygen
- Java
  - Javadoc

Em geral podem ser plugadas em alguma ferramenta de documentação de projetos.



# Padrões de codificação

```
def funcaoQueFazAlgo(a, b):  
    print('Algo!!')  
  
def outra_funcao(arg1, arg2):  
    print("Outra funcao!", arg1+arg2)  
  
funcaoQueFazAlgo(1, 2)  
outra_funcao(3, 4)
```

# Padrões de codificação

```
[igor@haute-normandie 09-api-padroes-de-codigo]$ pylint porco.py
No config file found, using default configuration
***** Module porco
W:  4, 0: Bad indentation. Found 2 spaces, expected 4 (bad-indentation)
C:  7, 0: Trailing whitespace (trailing-whitespace)
C:  1, 0: Missing module docstring (missing-docstring)
C:  3, 0: Function name "funcaoQueFazAlgo" doesn't conform to snake_case naming style (invalid-name)
C:  3, 0: Argument name "a" doesn't conform to snake_case naming style (invalid-name)
C:  3, 0: Argument name "b" doesn't conform to snake_case naming style (invalid-name)
C:  3, 0: Missing function docstring (missing-docstring)
W:  3,21: Unused argument 'a' (unused-argument)
W:  3,24: Unused argument 'b' (unused-argument)
C:  6, 0: Missing function docstring (missing-docstring)

-----
Your code has been rated at -6.67/10 (previous run: -5.00/10, -1.67)
```

# Padrões de codificação

- Cada projeto tem o seu
- Algumas linguagens tem um estilo padrão
  - Python - PEP8
- Ferramentas ajudam a conferir (forçar) um estilo específico

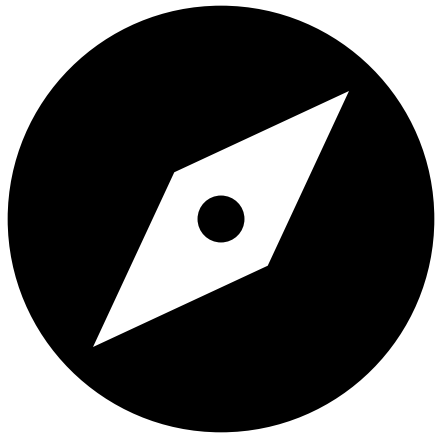
# Ferramentas

- Python: pylint, black
- C/C++: splint, cppchecker, gcc (opções -Wall, -Wextra)
- Java: flag `-Xlint`
- Javascript: ESLint, TSLint (typescript)

Ajudam a manter código limpo e legível. Podem ser plugadas no seu editor favorito.

## Execução obrigatória para muitos projetos grandes

# Atividade prática: Projeto profissional



**Objetivo:** Transformar um código perdido em um projeto "completo"

```
"metadata": {"url": "github pages criado", "group": ["ate três", "alunos"]}
```

# Desenvolvimento Aberto



**Documentação de API + testes**

Igor dos Santos Montagner ( [igorsm1@insper.edu.br](mailto:igorsm1@insper.edu.br) )