

# Desenvolvimento Aberto



**Documentação de API + testes**

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

# Atividade - objetivos

Transformar o projeto DesSoft-Desafios em um projeto *profissional*

1. Documentação de usuário (aluno e professor)
2. Documentação de desenvolvimento
3. Traduções e internacionalização (datas)

# Atividade - objetivos

Transformar o projeto DesSoft-Desafios em um projeto *profissional*

1. Documentação de usuário (aluno e professor)
2. Documentação de desenvolvimento
  - Estrutura de projeto
  - **API**
3. Traduções e internacionalização (datas)

# Hoje

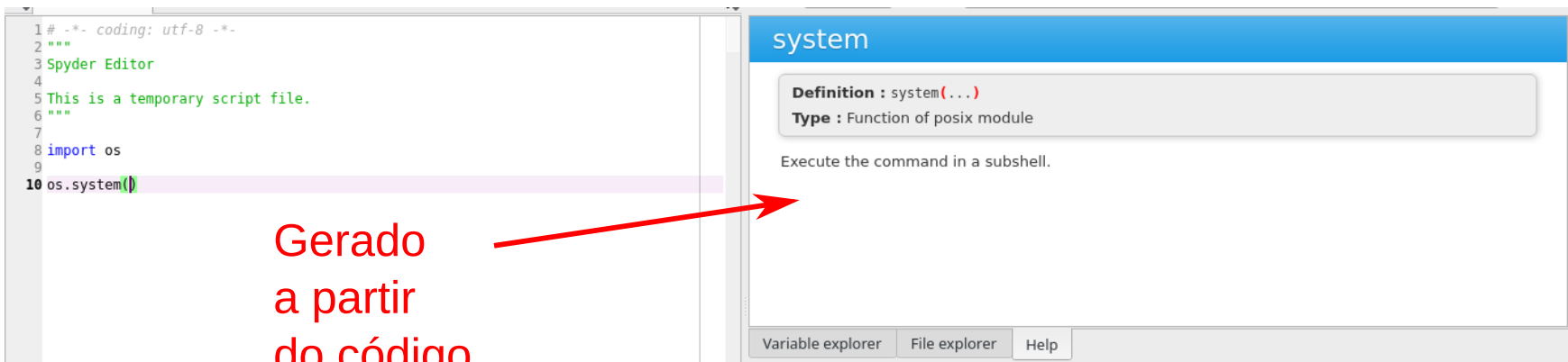
- Demonstração das traduções e documentações de vocês
- Documentação de API usando
  - pydoc
  - sphinx-autodoc
- Padrões de formatação de código
  - linters
  - PEP8

# 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

# Documentação de API



The image shows a code editor window on the left and a documentation panel on the right. The code editor contains a Python script with the following content:

```
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()
```

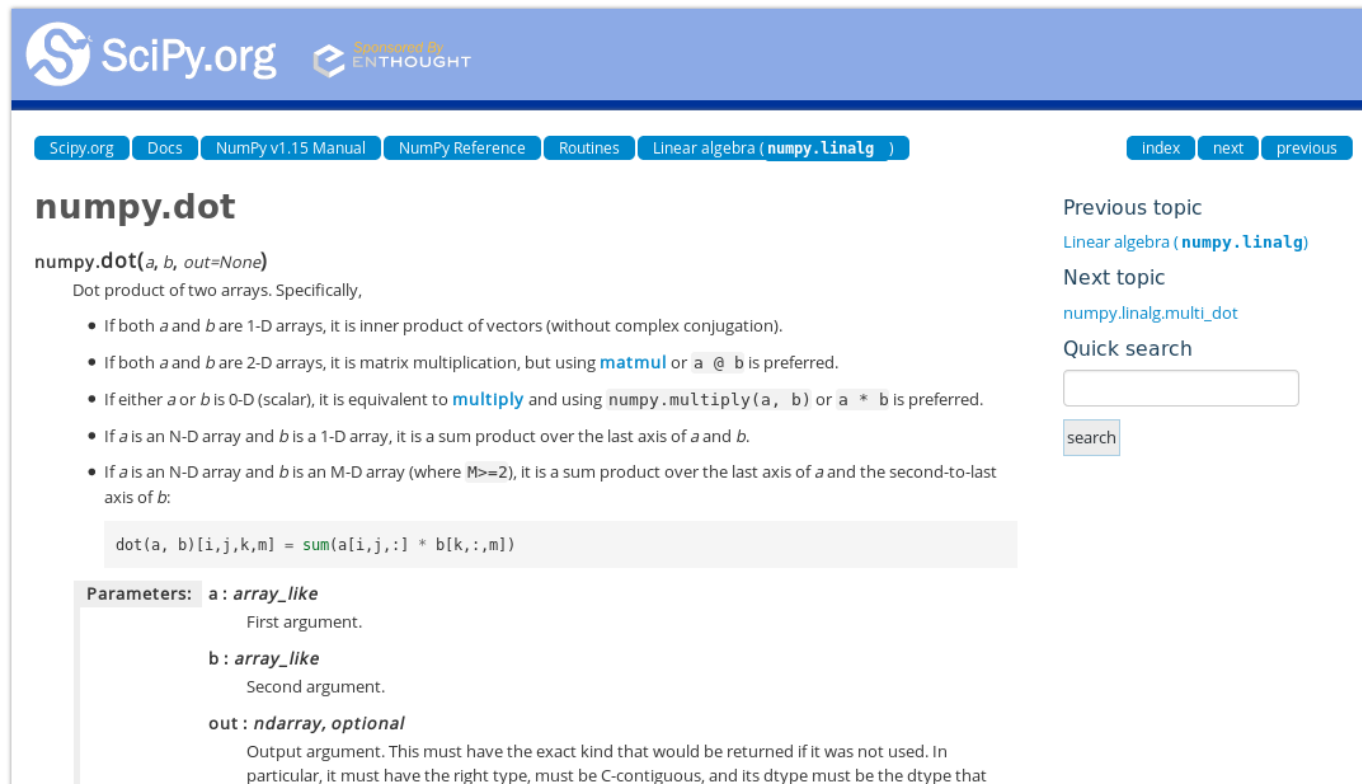
The documentation panel on the right is titled "system" and contains the following information:

- Definition :** `system(...)`
- Type :** Function of posix module
- Execute the command in a subshell.

A red arrow points from the text "Gerado a partir do código" to the documentation panel.

Gerado a partir do código

# Documentação de API



The screenshot shows the SciPy.org website header with the SciPy logo and "Sponsored by ENTHOUGHT". Below the header is a navigation bar with links for "SciPy.org", "Docs", "NumPy v1.15 Manual", "NumPy Reference", "Routines", and "Linear algebra (numpy.linalg)". On the right side of the navigation bar are "index", "next", and "previous" buttons.

## 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

On the right side of the page, there are navigation links: "Previous topic" (Linear algebra (numpy.linalg)), "Next topic" (numpy.linalg.multi\_dot), and a "Quick search" section with an input field and a "search" button.

ref

# Ferramentas

- Python:
  - pydoc, **sphinx-apidoc**
- C/C++
  - Doxygen
- Java
  - Javadoc



# 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
- 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/IDE favorito.

# Testes automatizados

**Ideia:** escrever um programa que verifica se um outro programa responde como esperado

- Definir situações a serem testadas ...
- e o resultado esperado em cada situação

# Testes automatizados

## Não ajudam:

- a revelar novos bugs
- a garantir que um software é livre de bugs

## Ajudam

- a evitar que bugs descobertos voltem
- a evitar que mudanças não intencionais quebrem código que estava funcionando.
- a documentar em quais situações o software funciona.

# Testes unitários

**Ideia:** dada uma função, verificar se ela devolve o valor esperado para um certo conjunto de parâmetros.

- Testa as funções de maneira **isolada**
- **Cobertura:** porcentagem das linhas de código que é executada durante os testes de unidade.
- Serve como documentação da função

# Testes unitários - pytest

```
# content of test_sample.py
def func(x):
    return x + 1

def test_answer():
    assert func(3) == 5
```

That's it. You can now execute the test function:

```
$ pytest
===== test session starts =====
platform linux -- Python 3.x.y, pytest-3.x.y, py-1.x.y, pluggy-0.x.y
rootdir: $REGENDOC_TMPDIR, inifile:
collected 1 item

test_sample.py F [100%]

===== FAILURES =====
----- test_answer -----

    def test_answer():
>         assert func(3) == 5
E         assert 4 == 5
E         + where 4 = func(3)

test_sample.py:5: AssertionError
===== 1 failed in 0.12 seconds =====
```

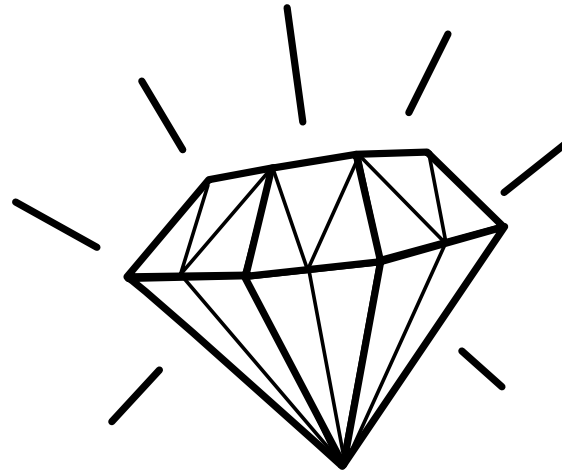


**O quê eu preciso testar?**

Ninguém sabe de verdade....

# Atividade prática

Vamos finalizar nosso projeto de DesSoft hoje:



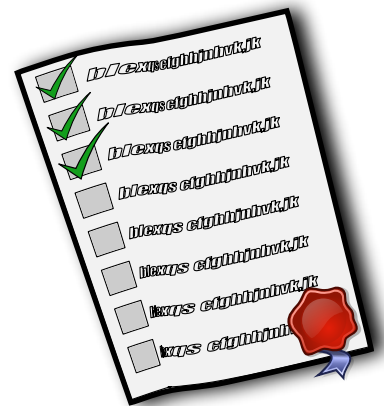
**Skill:** Shiny Code - documentou API do sistema e passou *linter*

**Proof:** url do fork de vocês com tudo funcionando.

Vocês deverão usar **sphinx-apidoc** e fazer a documentação usando *docstrings*.

# Atividade prática

Opcional: criar um conjunto de testes usando



**Skill:** Testado e aprovado - criou conjunto de testes para o sistema de DesSoft

**Proof:** url do arquivo de testes no Github

# Desenvolvimento Aberto



**Documentação de API + testes**

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