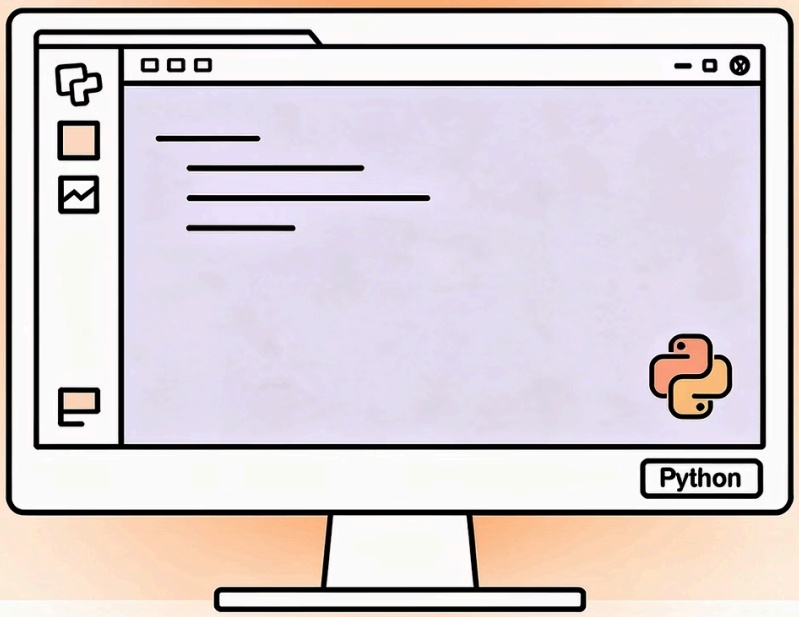
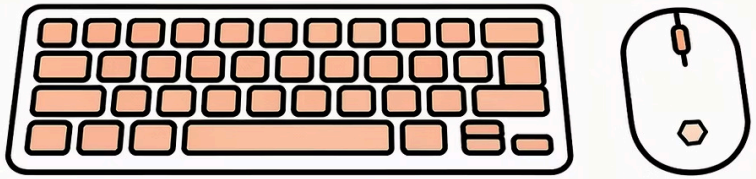




CEFET/RJ

Pacotes e Ajuda em Python (VS Code)

Esta aula cobre instalação de pacotes, uso de funções e recursos de ajuda no Python, focando em práticas essenciais para ciência de dados.

Eduardo Ogasawara

eduardo.ogasawara@cefet-rj.br

https://eic.cefet-rj.br/~eogasawara

CONCEITOS FUNDAMENTAIS

O que é uma Função?

Uma função é um bloco de código reutilizável que encapsula uma tarefa específica. Ela recebe entradas (parâmetros), executa um processamento e produz uma saída (retorno).

Funções ajudam a organizar o código, evitar repetição e facilitar manutenção. A ideia central é encapsular uma tarefa para usar sempre que necessário.

```
def cubo(x):
 return x**3

print(cubo(2)) # Saída: 8
```

SINTAXE

Chamando Funções e Argumentos

Argumentos Posicionais

Passados pela ordem definida na função. A posição determina qual parâmetro recebe cada valor.

Argumentos Nomeados

Especificados pelo nome do parâmetro (keyword). Tornam o código mais legível quando há muitos parâmetros.

```
import numpy as np

x = np.array([1, 2, 3, 4])
print(np.mean(x))     # posicional
print(np.mean(a=x))    # nomeado (keyword)
```

O retorno pode ser usado diretamente em expressões ou armazenado em uma variável, tornando o código mais flexível.

Ecossistema de Pacotes em Python

Python tem uma biblioteca padrão robusta, mas ciência de dados depende muito de pacotes externos. O principal repositório é o **PyPI**, acessado via pip.

PyPI

Repositório oficial com milhares de pacotes prontos para instalação via pip.

GitHub

Versões de desenvolvimento podem ser instaladas diretamente de repositórios.

```
# PyPI (padrão)
pip install numpy pandas matplotlib

# GitHub (quando necessário)
# pip install git+https://github.com/usuario/projeto.git
```

CONFIGURAÇÃO

Instalação e Uso de Pacotes (passo a passo)

01

Crie/ative um ambiente virtual

Use venv para isolar dependências do projeto e evitar conflitos entre bibliotecas.

02

Instale com pip

Normalmente você faz isso uma vez por projeto. Mantenha comandos de instalação fora do script principal.

03

Importe no código

Sempre que abrir uma nova sessão ou executar o script, importe os pacotes necessários.

```
# criar venv
python -m venv .venv

# ativar (Windows PowerShell)
# .\.venv\Scripts\Activate.ps1

# ativar (Linux/macOS)
# source .venv/bin/activate

pip install numpy pandas matplotlib
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

PRÁTICA

Exemplo prático: gráfico rápido (scatter) com matplotlib

Vamos criar dois vetores numéricos e visualizar a relação entre eles usando uma função cúbica: $y = x^3$.

Em Python, o caminho direto para um gráfico rápido é matplotlib. A ideia é visualizar a forma da função a partir dos pontos gerados.

```
import numpy as np
import matplotlib.pyplot as plt

x = np.array([-1, -0.8, -0.6, -0.4, -0.2, 0,
 0.2, 0.4, 0.6, 0.8, 1.0])
y = x**3

plt.scatter(x, y)
plt.xlabel("x")
plt.ylabel("y")
plt.show()
```

Prototipando uma função no console

1

Teste passo a passo

Execute linha a linha para validar o comportamento antes de encapsular.

2

Valide a lógica

Verifique se os resultados estão corretos e fazem sentido.

3

Encapsule em função

Transforme em código reutilizável após validação completa.

O exemplo abaixo simula lançar dois dados e somar os valores resultantes:

```
import numpy as np

dado = np.arange(1, 7)
dados = np.random.choice(dado, size=2, replace=True)
print(dados, dados.sum())
```

Criando a função jogada()

Encapsulando a lógica

Agora que a lógica foi validada, encapsule em uma função. A função não recebe parâmetros e retorna a soma de dois lançamentos.

O return define o valor retornado para quem chamou a função. Manter variáveis locais dentro da função evita poluir o escopo global.

```
import numpy as np

def jogada():
 dado = np.arange(1, 7)
 dados = np.random.choice(dado, size=2, replace=True)
 return int(dados.sum())

print(jogada())
```

SIMULAÇÃO BÁSICA

Simulação mínima: 30 amostras + histograma

Vamos chamar `jogada()` várias vezes para gerar uma amostra pequena. Com 30 valores já dá para calcular uma média razoável, mas a distribuição ainda pode ficar irregular para visualização.

No histograma, usamos largura de classe 1 (passos inteiros) para representar cada soma possível dos dois dados.

```
import numpy as np
import matplotlib.pyplot as plt

amostras = np.array([jogada() for _ in range(30)])
bins = np.arange(amostras.min() - 0.5, amostras.max() + 1.5, 1)

plt.hist(amostras, bins=bins)
plt.xlabel("Soma dos dois dados")
plt.ylabel("Frequência")
plt.show()
```

Análise teórica: espaço de possibilidades (36 combinações)

Ao lançar dois dados, existem **36 pares possíveis** (D1, D2). Algumas somas aparecem mais vezes: 7 tem 6 combinações, enquanto 2 e 12 têm apenas 1.

Isso explica por que a distribuição das somas tem formato triangular. Podemos gerar essa tabela programaticamente e organizar em um DataFrame.

```
import pandas as pd

comb = []
for d1 in range(1, 7):
 for d2 in range(1, 7):
 comb.append((d1 + d2, (d1, d2)))

df = pd.DataFrame(comb, columns=["soma", "par"])
tabela = df.groupby("soma")["par"].apply(list).reset_index()
print(tabela)
```

SIMULAÇÃO COMPLETA

Simulação completa: 10.000 amostras

Precisão estatística

Com muitos lançamentos, a distribuição empírica se aproxima da teórica com alta fidelidade.

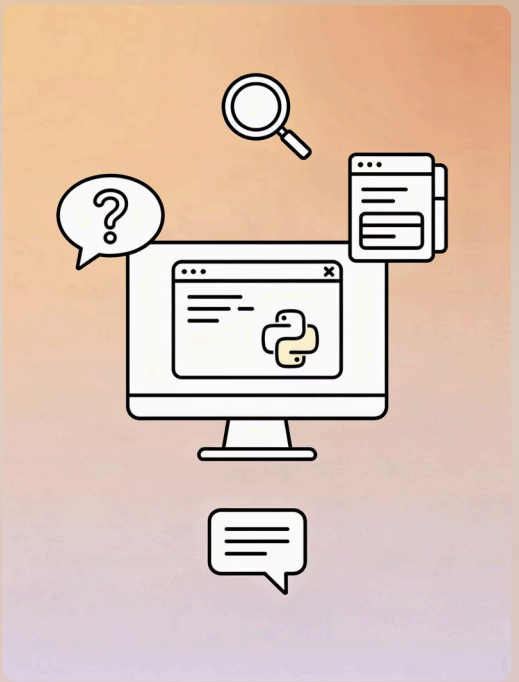
Valor mais frequente

O valor 7 deve aparecer como o mais frequente, com formato triangular claro.

```
import numpy as np
import matplotlib.pyplot as plt

amostras = np.array([jogada() for _ in range(10000)])
bins = np.arange(1.5, 12.6, 1)

plt.hist(amostras, bins=bins)
plt.xlabel("Soma dos dois dados")
plt.ylabel("Frequência")
plt.show()
```



Sistema de ajuda no Python

Você pode calcular estatísticas e consultar a documentação integrada diretamente no Python. A documentação mostra descrição, parâmetros, retorno e exemplos práticos.

help()

Abre a documentação completa de qualquer função ou classe no console.

? (IPython)

Atalho rápido que funciona em IPython e consoles interativos modernos.

```
import numpy as np


print(np.mean(amostras))
help(np.sqrt)
```

Executando exemplos da documentação

A documentação do Python e das bibliotecas traz exemplos executáveis que podemos reproduzir diretamente no código.

Aqui vamos plotar a raiz quadrada do valor absoluto, com suavização através de linha contínua. Isso ajuda a aprender novas funções de forma prática e visual.

```
import numpy as np
import matplotlib.pyplot as plt

xx = np.arange(-9, 10)
yy = np.sqrt(np.abs(xx))

plt.plot(xx, yy, 'o', label="pontos")
plt.plot(xx, yy, '-', label="linha")
plt.legend()
plt.show()
```

AVANÇADO

Amostragem com distribuição de probabilidades

Agora vamos gerar valores já ponderados pelas probabilidades teóricas. Cada soma recebe um peso proporcional ao número de combinações possíveis.

1 Elimina variação aleatória

Ao usar probabilidades corretas, reduzimos a variação devido ao acaso da amostragem.

2 Convergência mais rápida

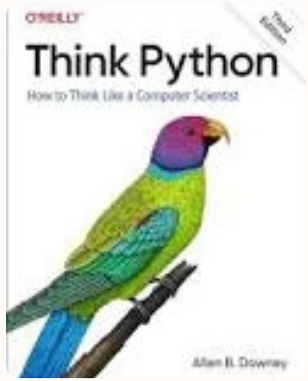
O histograma converge mais rapidamente para a distribuição teórica correta.

```
import numpy as np
import matplotlib.pyplot as plt

valores = np.arange(2, 13)
probs = np.array([1,2,3,4,5,6,5,4,3,2,1]) / 36
amostras = np.random.choice(valores, size=10000, p=probs)

bins = np.arange(1.5, 12.6, 1)
plt.hist(amostras, bins=bins)
plt.xlabel("Soma")
plt.ylabel("Frequência")
plt.show()
```

Referências



Think Python

Downey, A. *Think Python: How to Think Like a Computer Scientist.* O'Reilly Media.

An essential introduction to programming fundamentals and computational thinking using Python.

Python Data Science Handbook

VanderPlas, J. *Python Data Science Handbook.* O'Reilly Media.

A comprehensive guide to essential tools for working with data in Python, including NumPy, Pandas, and visualization libraries.

Data Science from Scratch

Grus, J. *Data Science from Scratch.* O'Reilly Media.

Learn data science fundamentals by building algorithms and tools from the ground up using Python.