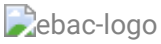


# **Library 01** | Python: Estudo & desenvolvimento de projetos

Caderno de **Conteúdos**

Desenvolvedor Enzo Schitini

## Descrição

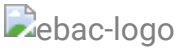
**Sobre:** `---`

## Tópicos

1. Tratamento de erros
2. Programação funcional
3. Consumindo dados de uma tabela
4. Nomes da tabela
5. Desvio padrão amostral
6. Random
7. Objetos e classes - Exemplos
8. Graficos
9. Converter imagens e Criar PDF
10. GitHub

---

# 1. `Tratamento de erros`



## 1.1. (**Desafio**) `Try e Except` para extrair o **tipo de erro**

## Encontrando e filtrando

```
%%writefile erros.csv
nome, classe
```

```
# Filtrando nome do erro                        | Enzo Schitini

anos = [2019, 2020, 2021]

try:
  ano_atual = anos[2]
  print(ano_atual)
except Exception as exc:
  # Encontrando a posição
  posicao_risco = str(type(exc)).find("'") + 1
  posicao_fim = str(type(exc)).find(">") - 1
  resposta = str(type(exc))[posicao_risco:posicao_fim]
  # Imprimindo resposta
  print("O programa apresenta falhas de sistema!")
  print("Erro:", resposta, "--> descrição:", str(exc))
  # Salvando erros
  with open(file='erros.csv', mode='a', encoding='utf8') as fp:
    line = str(resposta) + ',' + ' --> ' + str(exc) + '\n'
    fp.write(line)


# Mostrando a lista

conteudo = None

with open(file='erros.csv', mode='r', encoding='utf8') as fp:
    conteudo = fp.read()

print(conteudo)
```

## 2. Programação funcional


### Criando e chamando

Sem o retrono

```
def soma(v1: int):
    soma_resposta = v1 + 5
    print(soma_resposta)

d1 = 5
soma(v1=d1)
```

Com o retrono

```
def soma(v1: int, v2: int) -> int:
    soma_resposta = v1 + v2
    return soma_resposta

d1 = 5
d2 = 5
resposta = soma(v1=d1, v2=d2)
print(resposta)
```

## Função Lambida

```
ex_email = lambda email: email.split(sep='@')[1]


em = 'enzo@gmail.com'
vl = ex_email(em)


print(vl)


som = lambda v1, v2: True if (v1 + v2) / 2 == 16 else False


v = 10
vv = 22

valor = som(v,vv)
print(valor)


emails = ['enzo@gmail.com', 'gew@gamail.com']

provedores = map(lambda email: email.split(sep='@')[1], emails)
print(provedores)
print(list(provedores))


provedores = filter(lambda email: '@gmail.com' in email, emails)
print(list(provedores))


qtds = [34, 44, 68]
lista = []

for qtd in qtds:
    v = qtd + 10
    lista.append(v)

print("Lista com o FOR:", lista)

metade = map(lambda qtd: qtd + 10, qtds)
print("Lista com o Lambda:", list(metade))
```

## Com dicionários

```
emprestimos = [
    {'id_vendedor': '104271', 'valor_emprestimos': '448.0', 'quantidade_emprestimos': '1',
    {'id_vendedor': '21476', 'valor_emprestimos': '826.7', 'quantidade_emprestimos': '3',
    {'id_vendedor': '87440', 'valor_emprestimos': '313.6', 'quantidade_emprestimos': '3',
    {'id_vendedor': '15980', 'valor_emprestimos': '-8008.0', 'quantidade_emprestimos': '6',
    {'id_vendedor': '215906', 'valor_emprestimos': '2212.0', 'quantidade_emprestimos': '5',
    {'id_vendedor': '33696', 'valor_emprestimos': '2771.3', 'quantidade_emprestimos': '2',
    {'id_vendedor': '33893', 'valor_emprestimos': '2240.0', 'quantidade_emprestimos': '3',
    {'id_vendedor': '214946', 'valor_emprestimos': '-4151.0', 'quantidade_emprestimos': '18
    {'id_vendedor': '123974', 'valor_emprestimos': '2021.95', 'quantidade_emprestimos': '2'
'20161208'},
    {'id_vendedor':
'225870',
'valor_emprestimos':
'4039.0',
'quantidade_emprestimos':
'2',
'data':
'20161208'}
]

valor_emprestimos_lista = list(map(lambda x: float(x['valor_emprestimos']), emprestimos))

print(valor_emprestimos_lista)
```

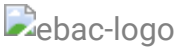
## 3. Consumindo dados de uma tabela


```
lista = []
ordem = 0

with open(file='./banco.csv', mode='r', encoding='utf8') as arq:
    linha = arq.readline()
    linha = arq.readline()
    while linha:
        list(linha)
        linha = linha.split(',')[5]
        lista.append(linha)
        if linha == '747':
            print("OK")
        linha = arq.readline()

for ele in lista:
    ordem = ordem + 1
    print("-----------------")
    if ele == '0':
```

```
        print(ordem, '->', ele, "##### NULO")
    else:
        print(ordem, '->', ele)

print("-----------------")


lista = []

with open(file='./reviews_data.csv', mode='r', encoding='utf8') as arq:
    linha = arq.readline()
    linha = arq.readline()
    while linha:
        list(linha)
        linha = linha.split(',')[0]
        lista.append(linha)
        linha = arq.readline()

print(lista)
```

---

## 4. Nomes da tabela

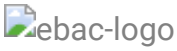



```
nomes = lista

# Se você quiser saber quais nomes se repetem na lista, você pode usar a função
# set() para criar um conjunto com os nomes únicos e depois iterar sobre esse
# conjunto para verificar quantas vezes cada nome aparece na lista:

nomes = lista
nomes_unicos = set(nomes)
qtd_nomes = []

for nome in nomes_unicos:
    if nomes.count(nome) > 1:
        #print(f"O nome {nome} aparece {nomes.count(nome)} vezes na lista.")
        qtd_nomes.append(nomes.count(nome))

maior_numero = max(qtd_nomes)
#print(f"O maior número na lista é {maior_numero}.")

for nome in nomes_unicos:
    if nomes.count(nome) == maior_numero:
        print(f"O nome {nome} aparece {nomes.count(nome)} vezes na lista.")
        qtd_nomes.append(nomes.count(nome))
```

```
# Criando tabela com os nomes repetidos

nomes = lista
nomes_unicos = set(nomes)
qtd_nomes = []
qtd_nomes_ordem = []

# Nome e repetições
for nome in nomes_unicos:
    if nomes.count(nome) > 1:
        rp = f"O nome {nome}, Aparece {nomes.count(nome)} vezes na lista."
        qtd_nomes.append(rp)

# Ordenando --------------
print(qtd_nomes)
r = qtd_nomes[1]
str(r)
va = r[r.find(':') + 2]
#print(va)
# ------------------------


ord = 0
with open(file='nomes.csv', mode='a', encoding='utf8') as fp:
  for scr in qtd_nomes:
    ord = ord + 1
    str(scr)
    linha = str(ord) + ',' + scr + '\n'
    fp.write(linha)


import os
import sys

def restart_program():
    python = sys.executable
    os.execl(python, python, *sys.argv)

while True:
    try:
        # Código aqui:
        break  # Se não houver erros, saia do loop
    except Exception as e:
        print(f"Ocorreu um erro: {e}")
        restart_program()  # Reinicie o programa em caso de erro



while True:
    try:
        # Seu código aqui
        print('OK')
        lista = [3, 9, 2]
        lista = lista[3]
        print(lista)
        break # Se não houver erros, saia do loop
```

```
    except Exception as e:
        print(f"Ocorreu um erro: {e}")
        continue  # Se ocorrer um erro, continue para a próxima iteração do loop
```

## 5. Desvio padrão amostral




### 1. Definição

O **desvio padrão amostral é uma medida estatística** que indica o quanto os valores de um conjunto de dados tendem a se afastar da média. Em Python, você pode calcular o desvio padrão amostral usando a `biblioteca statistics`. A função `stdev()` dessa biblioteca retorna o desvio padrão amostral.

```
#Aqui está um exemplo de como calcular o desvio padrão amostral em Python:

import statistics
from functools import reduce

data = [1, 2, 3, 4, 5]
sample_stdev = statistics.stdev(data)
soma = reduce(lambda x, y: x + y, data)
media = round(soma / 5)
print(f"Desvio padrão amostral: {sample_stdev} e media é: {media}")
```

Esse código calcula o desvio padrão amostral para um conjunto de dados representado pela lista data. No exemplo acima, o resultado será `1.58113883008418981`.

Lembre-se de que o desvio padrão amostral **é uma medida útil para entender a dispersão dos dados em torno da média**. Ele é frequentemente **usado em `análises estatísticas` e `científicas` para avaliar a variabilidade dos dados**.

## 6. Random


### 1. Definição

A função `random` é uma biblioteca padrão do Python que permite gerar números pseudoaleatórios. Ela pode ser usada para gerar números aleatórios inteiros ou em ponto flutuante, bem como para selecionar elementos aleatórios de uma sequência. A documentação oficial do Python 3.11.5 contém informações detalhadas sobre a biblioteca `random`.

A função `random` é usada para gerar números pseudoaleatórios uniformemente distribuídos em um intervalo semiaberto de 0,0 a 1,0. A função `randint(a, b)` pode ser usada para gerar um número inteiro aleatório entre `a` e `b`, inclusive.

```
from random import random

print(random())


# RANDOM

from random import random

numeros = [round(100 * random()) for _ in range(0, 100)]
print(numeros)

import random

numero_aleatorio = random.randint(0, 100)
#print(numero_aleatorio)

r = numeros[numero_aleatorio]
print("O valor na posição:", numero_aleatorio, "é", r)


# Números pares

pares = []
impares = []

for numero in numeros:
    num = numero % 2
    if num == 0:
        pares.append(numero)
    else:
        impares.append(numero)

print("Pares:", pares)
print("Impares:", impares)
```

### 2. Permutação aleatória

A função `random` também pode ser usada para gerar uma permutação aleatória de uma lista internamente e para amostragem aleatória sem substituição.

A biblioteca NumPy também fornece a função `numpy.random.permutation()` que é usada principalmente para obter uma cópia permutada aleatoriamente de uma sequência e obter um intervalo permutado aleatoriamente em Python.

**3. Arredondar um números**

Por fim, a função `round()` é usada para arredondar um número com precisão especificada.

## 7. Objetos e classes - Exemplos

```
class Pessoa(object):

  def __init__(self, nome: str, idade: int, documento: str):
    self.nome = nome
    self.idade = idade
    self.documento = documento
```

## 8. Gráficos

### # Aula 01 - Introdução

```
def graficos():

  import matplotlib.pyplot as plt
  x = [1, 2, 3, 4, 5, 6]
  y = [1, 4, 9, 16, 25, 36]

  plt.plot(x,y,color='blue')
  plt.scatter(x,y,color='black')
  plt.title('Climate Change')
  plt.xlabel('x')
  plt.ylabel('y')
  plt.grid()
  plt.legend(['Estimativa', 'Dado'])
  plt.show()

graficos()
```

### # Aula 02 - Gráfico matplotlib: Alterando formatação das linhas

```
def graficos():

  import matplotlib.pyplot as plt
  x = [1, 2, 3, 4, 5, 6]
  y = [1, 4, 9, 16, 25, 36]

  plt.plot(x,y,'b--') # b = blue
  #plt.plot(x,y,'b-.')
  plt.scatter(x,y,marker='v',color='red')
  plt.show()

graficos()
```

# Aula 03 - Uso de label em legend para o comando plot

```
def graficos():

  import matplotlib.pyplot as plt
  x = [1, 2, 3, 4, 5]
  y1 = [2, 4, 7, 5, 8]
  y2 = [3, 6, 9, 4, 5]

  plt.plot(x,y1,color='blue',label='Produção')
  plt.plot(x,y2,color='red',label='Demanda')
  plt.legend()
  plt.show()

graficos()
```

# Aula 04 - Uso conjunto de plot e bar

```
def graficos():

  import matplotlib.pyplot as plt
  x  = [1, 2, 3, 4, 5, 6, 7, 8, 9]
  y1 = [1, 3, 5, 3, 1, 3, 5, 3, 1]
  y2 = [2, 4, 6, 4, 2, 4, 6, 4, 2]

  plt.plot(x,y1,'r-.')
  plt.bar(x,y2,color='gray')
  plt.show()

graficos()
```

# Aula 05 - Dois gráficos de barras no mesmo gráfico

```
import matplotlib.pyplot as plt

# Look at index 4 and 6, which demonstrate overlapping cases.
x1 = [1, 3, 4, 5, 6, 7, 9] # Pareamento entre índice de satisfação (x) e número de clientes
y1 = [4, 7, 2, 4, 7, 8, 3]

x2 = [2, 4, 6, 8, 10] # Pareamento entre índice de satisfação (x) e número de clientes (y)
y2 = [5, 6, 2, 6, 2]

# Mais cores: https://matplotlib.org/api/colors_api.html

plt.bar(x1, y1, label="Empresa A", color='b')
plt.bar(x2, y2, label="Empresa B", color='g')
plt.plot()

plt.xlabel("Índice de Satisfação")
plt.ylabel("Número de clientes")
plt.title("Comparativo de desempenho entre empresas")
plt.legend()
plt.show()
```

## # Conclisão - Salvar gráfico matplotlib como imagem

```
def graficos():

  import matplotlib.pyplot as plt
  x = [1, 2, 3, 4, 5, 6]
  y = [1, 4, 9, 16, 25, 36]

  plt.plot(x,y,color='blue')
  plt.scatter(x,y,color='black')
  plt.title('Climate Change')
  plt.xlabel('x')
  plt.ylabel('y')
  plt.grid()
  plt.legend(['Estimativa', 'Dado'])
  plt.savefig('plt.jpg')
  plt.show()

graficos()
```

## # Exemplos - Gráficos práticos

```
# Importando a biblioteca para fazer gráficos.
import matplotlib.pyplot as plt

x = [2, 3, 4, 5, 6, 7, 8, 9]
y = [9, 2, 8, 3, 7, 4, 6, 5]

# Criando gráficos a partir dos valores contidos em x e y.
plt.plot(x, y, color='green')  # Desenhando linhas
```

```
plt.scatter(x, y, color='red') # Desenhando apenas os pontos
plt.title('Evolução das vendas') # Título do gráfico
plt.xlabel('Tempo')   # Definindo nome do eixo X
plt.ylabel('Vendas')   # Definindo nome do eixo Y
plt.legend(['Previsão','Verificado']) # Definindo legenda
plt.grid()                            # Criando uma grade

plt.show()
```

```
plt.plot(x, y, 'b--')
plt.scatter(x, y, marker="*", color='red')
plt.show()
```

```
import matplotlib.pyplot as plt

x  = [1, 2, 3, 4, 5, 6, 7, 8, 9]
y1 = [1, 3, 5, 3, 1, 3, 5, 3, 1]
y2 = [2, 4, 6, 4, 2, 4, 6, 4, 2]
plt.plot(x, y2, color = "blue", label="Produção")
plt.plot(x, y1, color = "orange", label="Demanda")

plt.xlabel("Eixo x")
plt.ylabel("Eixo y")
plt.title("Produção e demanda ao longo do tempo")
plt.legend()
plt.show()
```

```
plt.plot(x, y, 'b--')
plt.bar(x, y, color='gray')
plt.show()
```

## 9. Converter imagens e Criar PDF


`pip install/uninstall reportlab` - Recursos para a construção de PDF

```
from reportlab.pdfgen import canvas
from reportlab.lib.pagesizes import A4


def mm2p(milimetros):
    return milimetros / 0.352777


cnv = canvas.Canvas("Meu_pdf.pdf", pagesize=A4)
cnv.drawString(300, 150, "Ok")
cnv.circle(100, 200, 50)
cnv.line(100, 200, 22, 22)
```

```
cnv.rect(200, 250, 300, 350)
# Imagens
cnv.drawImage("Group2.png", 44, 650, width= 508, height= 174)
cnv.save()
```

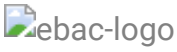
## Converter imagens

`pip install/uninstall Pillow` - Recursos para a conversão das imagens

```
# Converção simples

from PIL import Image

img = Image.open("Images/Group 11105.png").convert("RGB")
img.save("Group 11105.jpg")
```

```
from PIL import Image
import os

lista_arquivos = os.listdir("Images")
print(lista_arquivos)

for arquivo in lista_arquivos:
    img = Image.open(f"images/{arquivo}").convert("RGB")
    img.save(f"images/{arquivo.replace('png', 'jpg')}")
```

# 10. Programação orientada a objeto




## Classes

```
class ControleRemoto():
    pass
```

```
class ControleRemoto():
    def __init__(self, valor_da_cor, altura, profundidade, largura):
        self.cor = valor_da_cor
        self.altura = altura
        self.profundidade = profundidade
        self.largura = largura

    def passar_canal(self, botao):
        if botao == "+":
```

```
            print("Aumentar")
        elif botao == "-":
            print("Diminuir")
print("-------------------------")
controle_remoto = ControleRemoto('cinza', '10', '2', '2')
print(controle_remoto.cor)

controle_remoto.passar_canal("+")
print("-------------------------")
controle_remoto2 = ControleRemoto('Azul', '10', '2', '2')
print(controle_remoto2.cor)

controle_remoto2.passar_canal("-")
print("-------------------------")
```

## Projeto Netflix

```
# Programa Netflix
class Cliente():
    def __init__(self, nome, email, plano, salario) -> None:
        self.nome = nome
        self.email = email
        self.salario = salario
        self.lista_planos = ["basic", "premium"]
        if plano in self.lista_planos:
            self.plano = plano
        else:
            raise Exception("Plano erro!")

    def mudar_plano(self, novo_plano):
        if novo_plano in self.lista_planos:
            self.plano = novo_plano
        else:
            print("plano invalido!")

    def ver_filme(self, filme, plano_filme):
        if self.plano == plano_filme:
            print(f"Ver filme {filme}")
        elif self.plano == "premium":
            print(f"Ver filme {filme}")
        elif self.plano == "basic" and plano_filme == "premium":
            print("Mude seu plano")
        else:
            print("Plano invalido")

    def compras(self, preco):
        if self.salario >= preco:
            self.salario = self.salario - preco
            print("Comprado!")
        elif self.salario < preco:
            print("Nao")
        elif self.salario == None:
            print("Nao encontramos seu salario")
```

```
        else:
            print("Erro")

    def __str__(self) -> str:
        return f'Nome do usuário {self.nome}, plano {self.plano}, salário {self.salario}'

# Ações do usuário --------------
print("----------------------------")
# Print informações do usuário
cliente_use = Cliente("Enzo", "schitini@gmail.com", "basic", 100)
print(cliente_use.nome)
print(cliente_use.plano)
print(cliente_use.salario)
print("----------------------------")
# Tenta ver filme com plano basic mas não pode
cliente_use.ver_filme("Laura", "premium")
print("----------------------------")
# Muda o plano para premium
cliente_use.mudar_plano("premium")
print(cliente_use.plano)
# Tenta ver o filme com o novo plano
cliente_use.ver_filme("Laura", "premium")
print("----------------------------")
# Botão comprar filme
cliente_use.compras(60)
print(cliente_use.salario)
print("----------------------------")
# Tudo sobre o usuário
print(cliente_use)
print("----------------------------")
```

## Objetos

## Instâncias

## Herança

```
class Gerente(Cliente):

    def __init__(self, nome, email, plano, salario, pontos) -> None:
        super().__init__(nome, email, plano, salario)
        self.pontos = pontos
```

```
gerente_use = Gerente("Enzo", "schitini@gmail.com", "basic", 100, 10)
print(gerente_use.pontos)
```

## Projetos From Import

```
import senha
lista = [5, 6, 6, 8, 9, 8]
print(senha.minha_senha)
```

```
14
```

```
import senha
senha.verificar_senha('552')
```

```
No
```

```
catg = ['G', 'A']
td = ['G - 5', 'A - 8', 'A - 9', 'G - 7']
valores = []
# G = 12 A = 17

for x in catg:
 soma = sum(int(i.split(' - ')[1]) for i in td if i.split(' - ')[0] == x)
 valores.append(soma)

print(valores)
```

```
[12, 17]
```

```
def categoria(lista : list, catg : list) -> list:
   total = []
   for x in catg:
      soma = sum(int(item.split(', ')[1]) for item in lista if item.split(', ')[0] == x)
      total.append(soma)
   return total

c = ['Music, 50', 'Sport, 20', 'Music, 23', 'Sport, 70']
cc = ['Music', 'Sport']

i = categoria(c, cc)
print(i)
```

```
[73, 90]
```

## 11. GitHub

Para publicar

cd desktop mkdir enzo cd enzo echo "# Enzo" >> README.md git init git add README.md git add . git commit -m "first commit" git branch -M main git remote add origin https://github.com/enzoschitini/Enzo.git git push -u origin main

## Para baixar

git pull