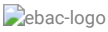


Library 01 | Python: Estudo & desenvolvimento de projetos

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

---

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)
```

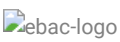
Com o retrono

```
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))
```

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)
```

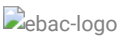
---

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 `1.5811388300841898`.

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**.

```
# 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)


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