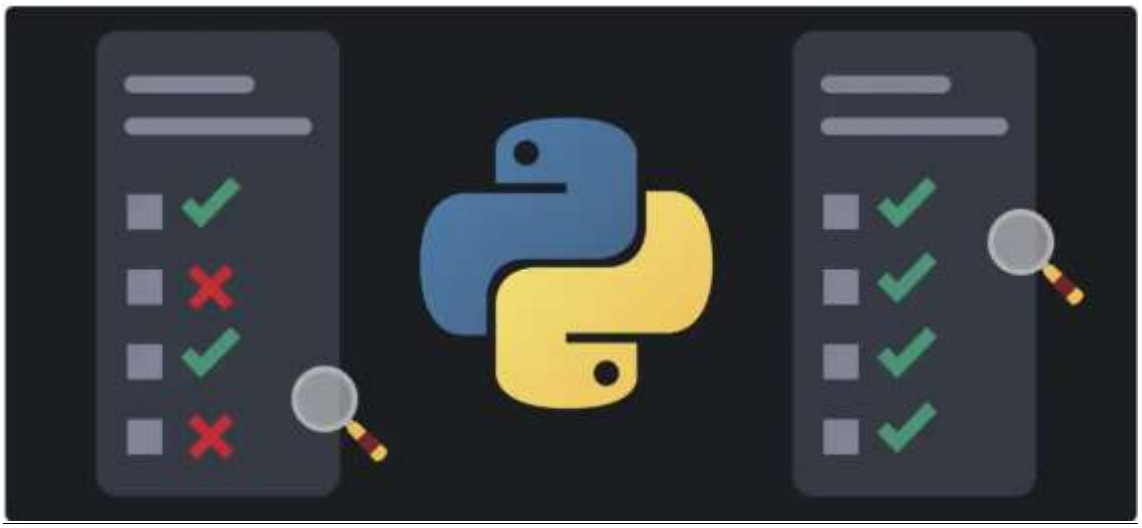
Teste de Python Salux



Teste de Python: Questões Teóricas:

1)Explique a diferença entre programação síncrona e assíncrona em Python. Quando você usaria cada uma?

2)O que são metaclasses em Python e como elas podem ser úteis?

3)Como funciona o garbage collector do Python e como podemos gerenciar manualmente a memória?

4)Qual a diferença entre deepcopy e copy em Python?

5)O que são decorators e como eles funcionam?

6)Explique o conceito de GIL (Global Interpreter Lock) e como ele afeta o multi-threading em Python?

7)Como funciona a tipagem dinâmica e forte do Python? Dê um exemplo prático.

Questões Práticas:

Implemente uma estrutura de dados LRUCache utilizando OrderedDict ou collections.deque. Sua implementação deve incluir os métodos:

```
class LRUCache:
    def __init__(self, capacity: int):
        pass

    def get(self, key: int) -> int:
        pass

    def put(self, key: int, value: int) -> None:
        pass
```

Dado o seguinte DataFrame:
Escreva uma função para calcular a média salarial dos funcionários com idade superior a 30 anos.

```
import pandas as pd

data = {
    'id': [1, 2, 3, 4, 5],
    'nome': ['Alice', 'Bob', 'Carlos', 'Daniel', 'Eva'],
    'idade': [25, 30, 35, 40, 45],
    'salario': [5000, 7000, 8000, 10000, 12000]
}

df = pd.DataFrame(data)
```

Implemente uma função assíncrona que faça requisições HTTP para três URLs diferentes em paralelo e retorne os conteúdos dessas páginas.

```
import asyncio
import aiohttp

async def fetch_urls(urls: list):
    pass
```

Caso Prático:

Você foi contratado para otimizar um sistema de processamento de logs que está enfrentando problemas de performance. O sistema recebe logs
em tempo real e precisa armazená-los em um banco de dados PostgreSQL. Atualmente, o código processa cada linha de log e insere os dados individualmente,
causando gargalos de performance.
Reescreva a função abaixo utilizando boas práticas para otimizar o desempenho:

```
---------------------------------------------------------------------------------
def process_log(logs):
    import psycopg2

    conn = psycopg2.connect("dbname=test user=postgres password=secret")
    cursor = conn.cursor()

    for log in logs:
        cursor.execute("INSERT INTO logs (timestamp, level, message) VALUES (%s, %s, %s)",
(log['timestamp'], log['level'], log['message']))

    conn.commit()
    cursor.close()
    conn.close()
---------------------------------------------------------------------------------
```

Dicas:

Considere a utilização de executemany ou COPY para inserção em lote.
Evite reconectar ao banco para cada execução.
Utilize context managers para garantir o fechamento correto da conexão.