Teste de Python: Questões Teóricas:

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

Programação síncrona o código é executado passo a passo, uma operação por vez. Se uma operação demora por algum motivo (como uma requisição de rede ou leitura de arquivo), o programa fica bloqueado até que termine.

Programação assíncrona permite que o programa execute várias tarefas ao mesmo tempo, sem bloquear a execução. Usa async e await para gerenciar operações que podem demorar.

O uso pode ser feito em operações de I/O como requisições de rede, leitura de arquivos ou chamadas a APIs

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

Metaclasses são classes geradas dinamicamente pela função type() passando três parâmetros.

São uteis em frameworks ou bibliotecas que precisam de controle avançado sobre a criação de classes.

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

Garbage collector e o gerenciador automático de memória usado para remover objetos que não estão mais sendo usados.

Usando a biblioteca GC, podemos usar para desativá-lo (gc.disable()), forçar uma coleta (gc.collect()) ou manipular referências

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

A diferencia que o copy cria uma cópia superficial de um objeto. Objetos internos (como listas dentro de listas) são referenciados, não copiados, e o deepcopy cria uma cópia profunda de um objeto, copiando recursivamente todos os objetos internos.

5) O que são decorators e como eles funcionam?

Decorators são funções que modificam o comportamento de outras funções ou métodos

Exemplo: @staticmethod ou @property são decorators embutidos. Você também pode criados

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

GIL: É um mecanismo que permite apenas uma thread executar código Python por vez, mesmo em CPUs multi-core.

Impacto no multi-threading: O GIL pode limitar o desempenho em tarefas que usam muitas threads para processamento intensivo de CPU. No entanto, para tarefas de I/O (como leitura de arquivos ou requisições de rede), o multi-threading ainda é útil.

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

Tipagem dinâmica e um tipo de uma variável é inferido em tempo de execução, não precisa ser declarado explicitamente.

Exemplo: x = 1(x é um inteiro), depois x = "texto" (x agora é uma string).

Tipagem forte no python não faz conversões implícitas entre tipos incompatíveis.

Exemplo: "10" + 1 resulta em um erro, pois não converte automaticamente a string para inteiro ou vice-versa.