Python3 Backend - Exercícios

Descrição

Abaixo segue uma lista de exercícios utilizados para avaliar o conhecimento do candidato. Alguns tópicos abordarão a base da linguagem python bem como lógica de programação e estrutura de dados.
Caso não consiga terminar um exercício específico, a tentativa será avaliada.

Regras

1. Criar código de exemplo para cada exercício que exige código separadamente;
2. Publicá-los em um repositório público e compartilhá-lo para ser avaliado.
    a. Prover readme de como executar os códigos compartilhados
3. As questões que correspondem a respostas dissertativas podem ser enviadas por e-mail junto com o link do repositório.

Estrutura de Dados

1. Implemente um algoritmo de vetores não ordenados com capacidade limitada utilizando a linguagem **python3**.

2. Explique com suas palavras os conceitos **FIFO** (*First-In First-Out)* e **LIFO** (*Last-In First-Out).*

3. Crie um script python que mostre um array com o nome do país e sua capital. Crie métodos que ordenem a lista pelo nome do país e pelo nome da capital. Adicione pelo menos 8 entradas no array.

Lógica de Programação

1. Um atleta que mantém meticulosos registros de suas caminhadas. Durante sua última caminhada de exatamente **P** passos, para cada passo foi anotado se ele era subida **S** ou descida **D.** As caminhadas sempre começam e terminam no nível do mar, e cada passo para cima ou para baixo representa **1** unidade na alteração da altitude. Considere os seguintes termos:
    a. Uma montanha é uma sequência de passos consecutivos acima do nível do mar, começando com um passo acima do nível do mar e terminando com um passo até o nível do mar.
    b. Um vale é uma sequência de passos consecutivos abaixo do nível do mar, começando com um passo abaixo do nível do mar e terminando com um passo até o nível do mar.

Dada uma sequência de passos de subida ou descida durante uma caminhada, imprima o número de vales percorridos durante a caminhada.
Exemplo:

P = 8 passos = [**DDSSSSDD**]

O atleta primeiro entrou em um vale **2 unidades abaixo**, então ele subiu uma montanha **2 unidades acima**, chegando no nível do mar no final da caminhada.

Descrição da Função:
Complete a função **contandoVales** com os seguintes parâmetros:

- **int** *passos*: o número de passos na caminhada (passos=**9**)
- **string** caminho: a descrição da caminhada (caminho="**SSDDDSSSS**")

Constraints:

- 2 <= **passos** <= 10
- caminho[i] ∈ {**SD**}

Exemplo de Entrada:
passos=8
caminho=SDDDSDSS

Exemplo de Saída:
1

Explicação:
Se representarmos _ como nível do mar, um passo acima como /, um passo abaixo como \, a caminhada do exemplo de entrada acima poderia ser desenhada da seguinte forma:

```
_/\      _
   \    /
    \/\/
```

Exercício:
Crie uma função que receba como parâmetro os **passos** e o **caminho** e retorne a quantidade de vales percorridos.

Casos de Teste:
passos**=12**
caminho**=DDUUDDUDUUUD**
retorno**=2**

passos**=10**
caminho**=DUDDDUUDUU**
retorno**=2**

passos**=10**
caminho**=UDDDUDUDUU**
retorno**=1**

passos=**100**
caminho**=DUDUUUUUUUUUDUDDUUDUUDDDUUDDDDDUUDUUUUDDDUUUUUUUD**
DUDUDUUUDDDDUUDDDUDDDDUUDDUDDUUUDUUUDUUDUDUDDDDDDDDD
retorno**=2**

Django / MQTT / Docker / PostgreSQL

1. Explique com suas palavras os conceitos de Publisher, Exchange, Topic, Queue e consumer no MQTT.

2. Crie um docker-compose com o banco de dados PostgreSQL e um broker MQTT.

3. Crie uma API utilizando o Django Rest Framework com a seguinte estrutura:
    a. Usar Padrão REST API;
    b. Integrar com o banco de dados PostgreSQL docker.
    c. Criar Endpoint para cadastrar/listar/editar/deletar um *dispositivo* que possua pelo menos os atributos **nome** e **mac_adress**;
    d. Criar Endpoint para cadastrar/listar/editar/deletar um *dado_entrada* que possua pelo menos os atributos **valor**, **data_hora** e relacionamento com o **dispositivo**;
    e. Habilitar Django Admin;
    f. Prover documentação mínima para usar a API.

4. Crie um publisher mqtt que envie 100 dados de entrada aleatórios para um tópico no MQTT Broker no docker. Criar um consumer mqtt para receber essas 100 entradas e enviá-las para API.