Teste de Avaliação 2
Curso: Fundamentos de Python
UFCD/Módulo/Temática: UFCD: 10793 - Fundamentos de Python
Ação: 10793_02/L
Formador/a: Rogério Jorge
Data-limite de entrega: 17 de julho
Cotação: 20 valores
Nome do Formando/a:
Classificação:

Instruções de submissão: submeta um único ficheiro em Word (*.doc ou .docx)* ou em PDF contendo o código, capturas de tela (*printscreens*) dos resultados do código, um pequeno relatório explicando a estrutura do seu código, as decisões de design tomadas e quaisquer instruções especiais para a execução do programa. Alternativamente, pode submeter o seu código para o GitHub, tendo pelo menos 1 *git commit* e 1 *git push* por cada elemento do grupo. Esta modalidade não necessita de relatório, sendo apenas necessário enviar o Word ou PDF especificando: o link dos repositórios GitHub e o número total de *commits*.

Objetivo: Criar um simulador de futebol

Neste projeto, vais desenvolver um Simulador de Gestão de Equipa de Futebol. Poderás criar uma aplicação web utilizando Flask **ou** uma aplicação de desktop com Tkinter. O objetivo é permitir que o utilizador crie uma equipa fictícia e simule jogos entre equipas. Cada grupo deve adicionar pelo menos dois elementos ao jogo para o tornar mais interessante, por exemplo: seleccionar a formação tática para cada jogo (4-4-2, 3-5-2), sistema de treinos semanal, estatísticas dos jogadores da equipa, compra/venda de jogadores, ligas e taças com jogos adicionais, finanças e orçamento, interação com a imprensa, suporte de adeptos, logotipo ou equipamentos próprios.

Requisitos:

1. Criação de Equipa: O utilizador poderá criar uma equipa com um nome à sua escolha.

2. Simulação de Jogos: O utilizador poderá simular um jogo entre a sua equipa e outra equipa fictícia. O resultado do jogo deve ser determinado com base em alguma lógica (ex: número aleatório, atributos dos jogadores).

3. 2 Elementos Adicionais: Torne o jogo mais envolvente e realista adicionando pelo menos dois elementos novos aos exemplos dados no final deste enunciado.

4. O código deverá ter pelo menos um teste unitário.

Exemplos de funções e ficheiros a utilizar

Flask - Aplicação Web

Estrutura de diretórios:

```
simulador_futebol_web/
|
├── templates/
|   └── index.html
|
└── app.py
```

Ficheiro `app.py`:

```
from flask import Flask, render_template
import random

app = Flask(__name__)

equipa = {"nome": "Minha Equipa", "jogadores": [], "saldo": 1000}

@app.route('/')
def index():
    return render_template('index.html', equipa=equipa)

@app.route('/simular_jogo')
def simular_jogo():
    return f"Resultado do jogo: {random.choice(['Vitória', 'Derrota', 'Empate'])}"

if __name__ == '__main__':
    app.run(debug=True)
```

Ficheiro `templates/index.html`:

```
<!DOCTYPE html>
<html lang="pt">
<head>
    <meta charset="UTF-8">
    <title>Simulador de Equipa</title>
```

```
</head>
<body>

<h1>{{ equipa.nome }}</h1>
<h2>Saldo: {{ equipa.saldo }}</h2>
<a href="/simular_jogo">Simular Jogo</a>

</body>
</html>
```

Tkinter - Aplicação de Desktop

Estrutura de diretórios:

```
simulador_futebol_desktop/
|
└── app.py
```

Ficheiro `app.py`:

```
import tkinter as tk
import random

class SimuladorFutebol(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("Simulador de Equipa")

        self.equipa = {"nome": "Minha Equipa", "jogadores": [], "saldo": 1000}

        tk.Label(self, text=self.equipa["nome"]).pack()
        tk.Label(self, text=f"Saldo: {self.equipa['saldo']}").pack()

        tk.Button(self, text="Simular Jogo", command=self.simular_jogo).pack()

    def simular_jogo(self):
        resultado = random.choice(['Vitória', 'Derrota', 'Empate'])
        tk.Label(self, text=f"Resultado do jogo: {resultado}").
```

```
pack()

if __name__ == '__main__':
    app = SimuladorFutebol()
    app.mainloop()
```

Boa Sorte,

O formador

Rogério Jorge