Projeto Final – Nave OnLine

Criar um jogo interativo em Python com interface gráfica (Tkinter), onde o jogador controla uma nave que deve capturar planetas.

O jogo deve evoluir automaticamente de fase, tornando-se mais desafiante e visualmente atrativo.

A cada novo nível, o nome de um planeta verdadeiro deve ser obtido através da API da **NASA (Exoplanet Archive)**.

Trabalho em Dupla ou Individual

- Pode ser realizado **em dupla** ou **individualmente**.
- Duplas devem entregar **um único ficheiro** e identificar claramente os dois nomes no cabeçalho do código.

Requisitos obrigatórios (mínimos):

O projeto **deve obrigatoriamente** incluir:

1. **Interface com Tkinter** (Canvas, eventos de teclado etc.)
2. **Movimento da nave** com as teclas ↑ **(Up)** e ↓ **(Down)**
3. **Planetas em movimento** da direita para a esquerda.
4. **Deteção de colisão** entre nave e planeta.
5. **Pontuação** que aumenta a cada planeta capturado.
6. **Mudança automática de nível a cada 10 pontos.**
7. **Consulta à API da NASA** para exibir o nome de um planeta real a cada novo nível.
8. A cada novo nível, devem mudar:
    - a **cor dos planetas**
    - o **emoji do planeta**
    - o **tamanho dos planetas**
    - o **fundo do jogo**
    - o **título do nível**, a incluir o **nome do planeta real**
9. Uso de **funções** para organizar e modularizar o código
10. **Comentários no código**, a explicar blocos principais.
11. Entrega em **.py**, com código **funcional e testado**.
12. Cabeçalho do código com:
    - Nome(s)
    - Título do projeto

Dicas:

➲ Usa o exemplo abaixo como base para desenvolver o teu projeto:

```
canvas.create_text(x, y, text="🪐", font=("Arial", tamanho), fill=cor)
```

➲ Usar listas para mudar a variável cor e emoji com base no nível para criar variedade visual. Usar listas como:

```
emojis = ["🪐", "🌕", "🌑", "🛸", "☄️", "🌟", "🌀", "🌍"]
emoji = emojis[nivel % len(emojis)]
```

➲ Para alterar a cor do fundo:

```
canvas.config(bg="darkblue")
```

Sugestão: criar uma função **`fundo_do_nivel(n)`** que retorna a cor de fundo para o nível n.

➲ Para alterar a cor do planeta:

```
cores = ["white", "yellow", "orange", "lightblue", "lightgreen", "violet"]
cor = cores[(nivel - 1) % len(cores)]
```

➲ Buscar planeta verdadeiro com **`requests`**:

```
url = "https://exoplanetarchive.ipac.caltech.edu/TAP/sync?..."
resposta = requests.get(url)
dados = resposta.json()
nome = dados[0]["pl_name"]
```

➲ Sugestão: para evitar travamentos na mudança de fase:

```
import threading
threading.Thread(target=tarefa, daemon=True).start()
```

➲ Outras sugestões:

- **`canvas.after(0, lambda: ...)`** para atualizar texto dentro de uma **`thread`**.
- **`janela.after(ms, função)`** para criar eventos no tempo, como movimento dos planetas.
- Evite repetir código: cria funções como **`criar_planeta()`** e **`mover_planetas()`** com **`after`**.

🗓️ Entrega e Apresentação – Obrigatória

- **Data-limite:** 06/06/2025 (sexta-feira) - **até às 10h00** (antes do início da aula)
- **Forma de entrega:** pasta compactada com todos os ficheiros, via classroom
- **Apresentação obrigatória:** durante a aula do dia **06/06/2025**

Regras importantes:

- O aluno OU dupla que **não apresentar** o projeto no **dia e horário marcados** terá o trabalho **desconsiderado** e nota zero
- A apresentação deve ser feita **por todos os participantes do projeto**, com todos os autores presentes.
- O projeto será executado **ao vivo na máquina do aluno ou partilhando o ecrã**.
- **Não serão aceitos vídeos gravados ou apresentações externas.**

O que deve constar na apresentação:

Durante a apresentação, o grupo deverá:

- ➲ Executar o jogo
- ➲ Jogar até pelo menos **nível 2**
- ➲ Mostrar claramente:
  - Pontuação a aumentar
  - Mudança de fundo, emoji, cor, tamanho dos planetas
  - Nome do planeta verdadeiro (ou *fallback*: uma mensagem alternativa, caso a API não responda) no cabeçalho
- ➲ Explicar brevemente a **organização do código**, com destaque para:
  - Uma função criada no projeto e sua utilidade
  - Como foi feita a integração com a API da NASA
    - O que mais poderia ser feito com a API: exibir distância do planeta, nome da estrela, temperatura estimada, comparar com a Terra, …

Critérios de Avaliação

Critério	Valores
Funcionamento básico do jogo: nave se move, planetas aparecem, colisão funciona, pontuação aumenta.	**5**
Consulta à API da NASA: exibe planeta real por nível, com fallback (mensagem alternativa) se falhar.	**2**
Personalização visual por nível: emoji, cor, tamanho dos planetas e fundo do jogo variam a cada fase.	**5**
Modularização e organização do código: uso de funções, sem código duplicado.	**2**
Apresentação no dia 06/06 até às 10h: demonstração funcional, explicação do código e da API usada.	**4**
Comentário e clareza do código: código comentado, fácil de entender, com cabeçalho de identificação.	**2**