


Exercise 01: Barra de progresso personalizada

| | |
|---|-------------|
|  | Exercise 01 |
| Criar uma função <code>ft_progress(lst)</code> que exibe uma barra de progresso para um loop <code>for</code> | |
| Diretório de entrega: <code>ex01/</code> | |
| Arquivos a serem entregues: <code>loading.py</code> | |
| Funções ou bibliotecas permitidas: Nenhuma, especialmente <code>tqdm</code> ou qualquer biblioteca para barra de progresso automática | |

- Instruções:

1. Implemente a função `ft_progress(lst)` que aceita um iterável como argumento.
2. A função deve atuar como um gerador, usando o operador `'yield'`.
3. A barra de progresso deve exibir:
 - ETA (Tempo estimado para conclusão)
 - Porcentagem concluída
 - Barra visual do progresso
 - Número atual/total de iterações
 - Tempo decorrido

- Exemplo de uso:

```
a_list = range(1000)
ret = 0
for elem in ft_progress(a_list):
    ret += (elem + 3) % 5
    sleep(0.01)
print()
print(ret)
```

- Saída esperada:

```
ETA: 8.67s [ 23%][====> ] 233/1000 | elapsed time 2.33s
...
2000
```

- Dicas:

- O operador `'yield'` é usado para criar geradores. Ele "pausa" a função e retorna um valor, permitindo que a função seja retomada do mesmo ponto na próxima iteração.
- Para calcular o ETA, considere o tempo decorrido e a proporção de itens processados.
- Use `'\r'` para sobrescrever a linha atual no terminal, criando a ilusão de uma barra de progresso em movimento.

Exercise 01: Barra de progresso personalizada

- A função `time.time()` pode ser útil para medir o tempo decorrido.



Lembre-se de que a criação de uma barra de progresso personalizada requer atenção aos detalhes de formatação e cálculos de tempo precisos.