Relatório de Projeto - Psychomon

`https://github.com/RenatoGeh/Psychomon`

Guilherme Schützer - NUSP 8658544
Renato Geh - NUSP 8536030
Ricardo Lira - NUSP 8536131
Yan Couto - NUSP 8536023

Instruções

Para rodar o projeto basta usar

```
python3 Psychomon.py
```

O programa apresentará um menu com as opções, que são:

Read Pokémon
: Lê o Pokémon como está no enunciado

List Pokémon
: Lista o nome de todos os Pokémons já lidos

Battle!
: Começa uma batalha com Pokémons que já foram lidos (os nomes deles são usados como seus identificadores)
Não funciona caso nenhum Pokémon tenha sido inserido

Open Server
: Inicializa o Modo Servidor, é necessário utilizar os Pokémons já inseridos para lutar, como em **Battle!**

Open Client
: Inicializa o Modo Cliente, é necessário escolher um Pokémon que já foi lido, como em **Battle!**

Quit
: Termina o programa

Para não ter o trabalho de inserir os Pokémons novamente toda vez que quisessemos rodar o projeto, criamos um arquivo que já tem alguns Pokémons feitos, o `Pokefile`. Para usá-lo basta rodar

```
cat input/Pokefile - | python3 Psychomon.py
```

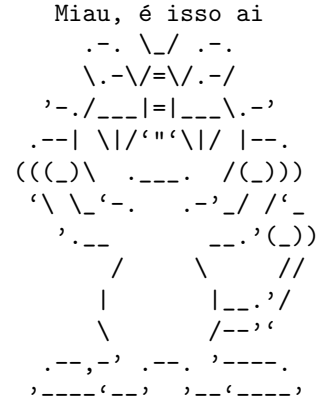
Note que algum texto "inutil" aparecerá no terminal pois o programa imprime a interação com o usuário mesmo quando está lendo a entrada do arquivo, mas depois disso já existirão 3 Pokémons lidos, basta usar **List Pokémon** para saber seus nomes e começar a usá-los em batalhas.

Para testar o programa, basta usar

```
python3 Poketest.py
```

O programa `Poketest` executa todos os testes que estão na pasta `tests`, que usam o pacote *unittest*.

1 Primeira Fase



Na primeira fase do projeto, primeiramente planejamos o que fariamos e começamos a primeira parte do diagrama de classes. Após debater e discutir como cada parte desa fase iria se interagir, e com o diagrama feito, ficou mais fácil a programação e implementação das ideias.

Inicialmente fizemos a base do programa, incluindo a definição de um Pokémon, seus atributos, seus ataques, o que é um ataque, e como estes seriam usados. Ainda sem implementar as fórmulas de combate ou outras funções mais específicas, criamos duas classes que auxiliariam em organizar as batalhas e dados para o programa. A classe "Pokedex" serve para "coletar" e armazenar os pokemons e seus respectivos atributos/ataques, enquanto a classe "Pokestadium" tem como objetivo ser um criador de batalhas, que organiza quais pokemons lutam, e inicializa o combate.

Com essa base já preparada, começamos a planejar as lutas ("pokebattles"), que já teriam uma interface simples implementada para o usuário, de forma que seja visível o combate entre dois pokemons. A classe "Pokebattle" então definiria quem começaria (o pokemon mais rapido), e organizaria os turnos de cada pokemon, mostrando os movimentos possíveis e imprimindo alguns de seus atributos e do seu oponente.

Implementar o dano foi um pouco trabalhoso, a fórmula não é tão simples quanto pensávamos, mas tirando escrever toda a tabela de type effectiveness o resto foi bem rápido. Como já tinhamos deixado preparado o cálculo do ataque antes, não foi necessário mudar nada fora da classe ataque, o que ajudou bastante.

Apesar de vários testes terem sido feitos durante todo esse processo, ao terminar o sistema de batalha mais simples, começamos a criar Pokémons com a finalidade de testar o combate. Essa criação inicial foi feita "criativamente", inventando nomes e atributos aleatórios, sem se basear em pokemons existentes.

Conforme fossemos arrumando alguns erros e bugs, começamos a pegar dados de sites como bulbapedia.com para nossos pokemons, adaptando as informações ao formato que nosso programa aceita como input.

2 Segunda Fase

A segunda fase do projeto foi bem diferente. Já tinhamos toda a base da primeira fase feita, o grande desafio foi aprender a usar novas bibliotecas para conseguir fazer tudo aquilo funcionar 'online'. Tinhamos pouca familiaridade com XML, então foi por ali que começamos, já que o servidor e o cliente precisavam daquilo para funcionar.

Depois de pesquisar algumas bibliotecas e tutoriais de como criar, ler e validar XML's em Python, decidimos usar a *lxml*, que também permite a validação dos XML's usando um xsd, que ao avançar no projeto, se mostrou muito útil

para encontrar erros, especialmente quando começamos a criar o servidor e o cliente.

Porém, tivemos uma grande dificuldade de usar essa biblioteca no Windows, tentando até mesmo compilar o seu código fonte, sem sucesso. Resolvemos portanto desenvolver o projeto no Linux, onde está biblioteca já vem instalada por padrão.

Tirando essa dificuldade inicial, o resto de XML's não foi muito trabalhoso, apenas esforço braçal.

Depois de pronta essa parte, já conseguimos um programa que cria XML's a partir de Pokemons e Pokemons a partir de XML's. Então começamos a criação da interface servidor/cliente. Para o servidor usamos a biblioteca *flask*, que tornou bem fácil a criação de um servidor funcional. O fluxo da batalha foi um pouco diferente do normal, por isso tivemos que adaptar parte do programa para isso. Para o cliente usamos *requests*, e o processo foi bem parecido com a batalha normal.

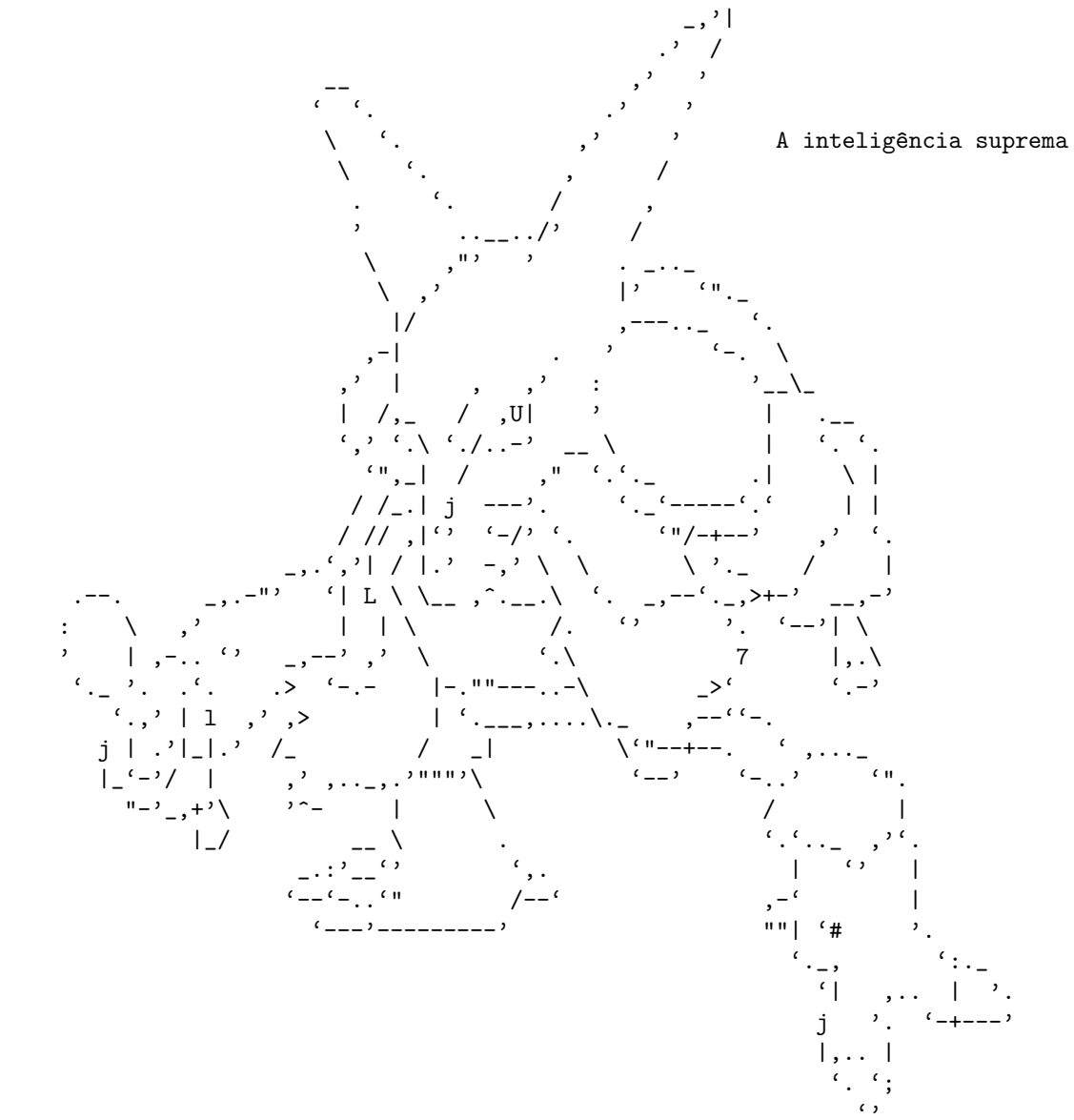
A interação entre os dois demorou um pouco para ficar boa, ocorreram algumas dificuldades para organizar o fluxo de dados, mas com um pouco de tempo conseguimos resolver. A primeira batalha usando dois terminais diferentes foi bem divertida.

Um 'problema' que encontramos é que fazendo as coisas exatamente como é dito no enunciado do EP, não é possível saber o HP inicial e PP's iniciais dos ataques da parte do servidor (a menos que *gambiarras* sejam utilizadas) e muito menos da parte do cliente. Além disso, o cliente fica sem nenhuma indicação de quais ataques foram usados pelo servidor e quanto dano os ataques causaram (além das informações adicionais).

Mudamos os testes para um diretório apropriado. Para isso tivemos de aprender a usar o sistema de módulos de python e como funciona para se dar "import" em subdiretórios.

Por último, também tivemos que arrumar o idioma do EP, já que em partes a comunicação com o usuário se dava em inglês e outras em português. Decidimos mudar, por fim, tudo para o inglês.

3 Terceira Fase



A última fase foi bem mais simples que as outras. Nesta, bastava a "Inteligência Artificial" de nosso programa escolher os ataques sozinhos quando usando o Modo Cliente ou Servidor.

O jeito que os ataques foram implementados foi uma tarefa bem simples, pois os ataques só dão dano direto e não causam efeitos secundários. Além disso, como não é possível mudar de pokemon durante a batalha, e são sempre batalhas individuais entre dois pokemons, não existe necessidade de *economizar* PPs para usar em batalhas futuras, logo basta apenas escolher o ataque disponível que tem maior "dano médio" até que a batalha acabe.

A fórmula para o "dano médio" que definimos foi:

$$\text{dano médio} = \text{chance acerto} \times (1 + \text{chance crítico}) \times \text{dano base}$$

Onde *dano base* é o dano do ataque, sem contar o crítico e o modificador aleatório.

Com isso feito, focamos em melhorar o programa em uma visão maior, aperfeiçoando um pouco os testes e o código onde achamos necessário. Novamente, utilizamos essa última parte do projeto para olhar nosso programa de forma mais abrangente, analisando melhor como cada parte interage entre sí, tentando aprimorar o que possível.

4 Comentários Finais

Como comentário do grupo em relação à essa última parte do projeto, achamos que testar o programa contra o de outros grupos não parece que vai dar muito certo, pois existe uma chance considerável de os programas não serem compatíveis um com o outro. Isso devido a algumas coisas não ficaram muito padronizadas no enunciado, como a ordem que os Pokémons tem que ficar no XML (nosso grupo assumiu que o Pokémon do Cliente fica sempre na primeira posição e o do Servidor na segunda). Além disso, como os ataques são um tanto simples, sem ter a parte interessante (e desafiante) de uma batalha Pokemon com vários modificadores e ataques variados, a vitória vai se inclinar fortemente nos fatores aleatórios (modificador aleatório e críticos) e nos Pokémons escolhidos (que podem ter valores arbitrarios de atributos se não forem balanceados para todos os grupos).

Apesar desse pormenor, o projeto teve seus estimulos, especialmente no começo, no qual tivemos que aprender a fazer nosso primeiro "grande" projeto em Python, utilizando dos conhecimentos de Patterns e outras ferramentas que aprendemos no curso, quanto na parte de aprender e trabalhar com bibliotecas pra transformar nosso programa em algo online.

Entendemos que o curso foi em boa parte experimental, e por isso teve seus contraventos, mas a ideia de fazer um Pokemon acabou sendo bem divertida, e tem potencial para, caso em futuros semestre repita tal projeto, de ser algo maior e mais "aventuroso" para os alunos.