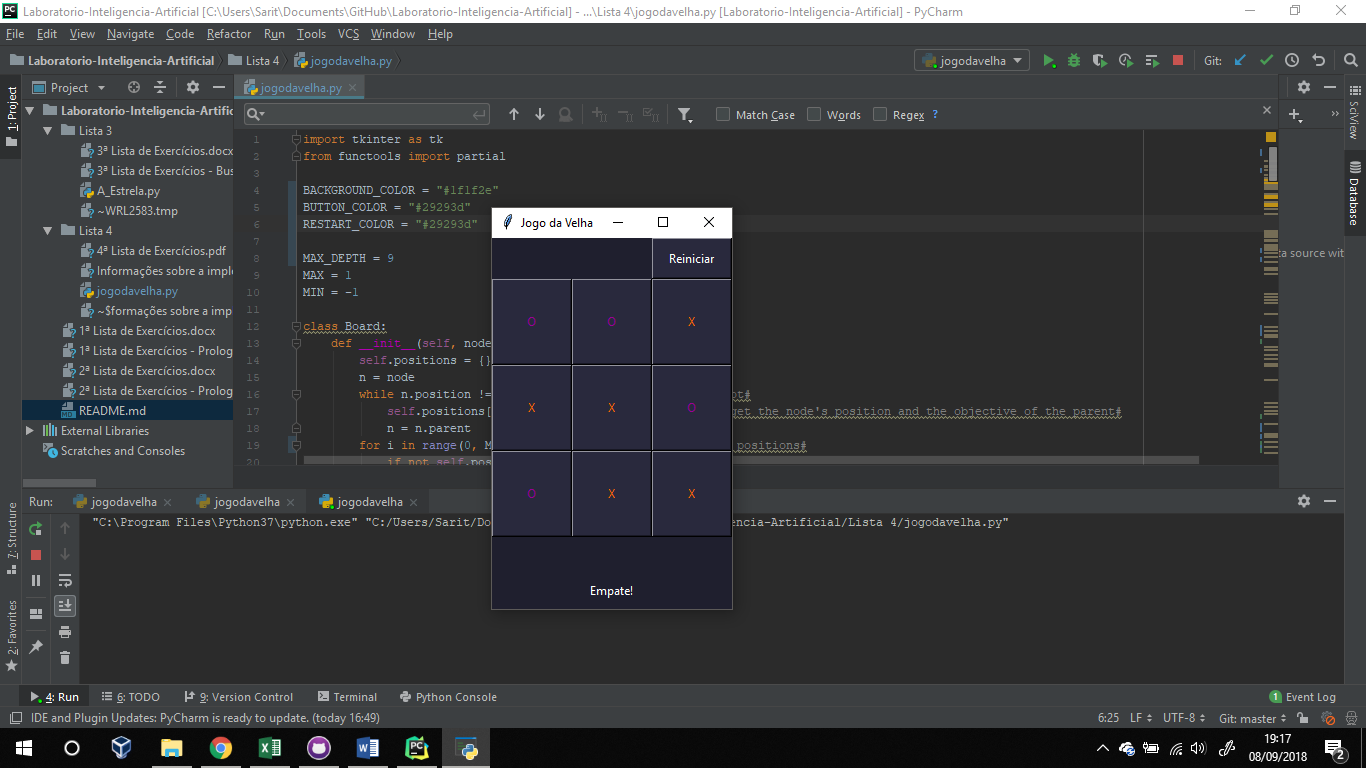
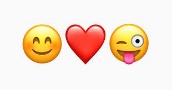
**Algumas informações sobre a implementação do algoritmo:**

* O código está em inglês, porque, como já expliquei anteriormente, só programo em Python em inglês (problemas com acentuação de comentários).

Link do repositório no GitHub [aqui](https://github.com/sarahrms/Laboratorio-Inteligencia-Artificial%20%20) (caso queira comprovar a autoria do código, será possível ver as alterações desde o início, bugs encontrados durante o processo e a adição de classes e funcionalidades aos poucos).

* Implementei uma interface gráfica para o jogo da velha, porque ficar digitando no teclado a posição no tabuleiro é ~~uma bosta~~ meio chato e a biblioteca utilizada (TKinter) é de muito fácil utilização.



Olha ti binitu!

* O programa gera a árvore com todas as possibilidades e a avalia primeiro, para depois criar a interface gráfica, ou seja, leva mais ou menos uns 15 segundos para ele começar. Depois ele percorre a árvore durante o jogo.
* Foram implementadas as seguintes classes:

**Tree:** A árvore de busca. Contém um nó raiz e os métodos **buildTree** e **minMaxSearch**.

**Node:** Nós da árvore. Contém o nó pai, um vetor de nós filhos, o nível, a posição (jogada), o objetivo (min ou max) e o valor avaliado (retornado na busca).

**BuildTree** chama **createChildren**, que cria os filhos recursivamente até chegar no nível 9. Por questões de otimização, ao chegar no nível 5, ela já começa a checar se há um vencedor e se houver ela não cria filhos para aquele nó (reduziu de 362880 nós folhas para 255168).

**MinMaxSearch** chama **evaluate,** que avalia os nós recursivamente. Quando encontra um nó folha, ele cria um **Board** com as posições e retorna o resultado do jogo. Depois passa o resultado para os pais, que pega o menor ou maior valor, dependendo do objetivo.

**Board:** Possui as posições no tabuleiro e o método checkWinner, que percorre o tabuleiro e verifica o vencedor (vitória Min, vitória Max, empate).

**TicTacToe:** Possui a interface gráfica e as funções de desenho dos botões, além de funções para detectar o vencedor e reiniciar o jogo. Recebe o nó raiz da árvore e cria um Board, depois cria a janela e os botões. **UpdateValue** trata a entrada do player e chama **chooseMove** e **makeMove** para decidir a jogada do pc. **ChooseMove** percorre a árvore e procura pela próxima jogada(o player é Max e o pc é Min), **makeMove** faz as alterações nas posições do board.