1. **(Kamil) Inicjalizacja pustej tablicy (*funkcja init\_board()*)**

* plansza 3 x 3
* utworzona za pomocą listy list
* każda komórka wypełniona “.”
* Umieszczenie tu funkcji print\_board(), żeby wyświetliło pustą tablicę
* Zwrócenie tablicy
* Poniżej dałem możliwy sposób zapisu, dzięki któremu wydaje mi się łatwo można by później modyfikować zawartość poszczególnych komórek. Do rozważenia.  
  Obraz zawierający tekst

  Opis wygenerowany automatycznie

Też myślałam by zrobić to w ten sposób 😊

1. **(Ewelina) Wczytywanie współrzędnych od użytkowników - rundy (*funkcja get\_move()*)**

* Jeśli użytkownik poda współrzędne spoza zakresu, pytaj dalej
* Jeśli użytkownik poda współrzędne zajętego już miejsca - pytaj dalej
* Jeśli użytkownik poda cokolwiek innego niż współrzędne, pytaj dalej
* Jeśli użytkownik wpisze exit lub quit to wyjdzie z programu (opcjonalne)
* Użytkownik ma wpisywać koordynaty np. A2, C1 itd
* Funkcja ma zwrócić tuplę w której będą przechowywane wartości współrzędnych z tablicy (np A1 to (0,0) - myślałam by wykorzystać też do tego celu słownik - np {“A1” : (0, 0), “A2” : (0, 1) itd}
* Pomyślałam że można stworzyć listę dostępnych pól, lista ta byłaby modyfikowana po każdym wykonanym ruchu użytkownika to mogłoby rozwiązać 3 pierwsze punkty ( jeżeli input jest w tej liście to …, a jeśli nie to pytaj dalej bez rozróżniania na to czemu jest błędny input - ale nie wiem czy to nie błędne podejście), można by też przed ruchem użytkownika wyświetlić listę dostępnych pól?
* W tej funkcji musiałaby być kolejność graczy, też mogłoby to być zwrócone z funkcji

1. (Ewelina) Po każdej rundzie wstawianie w odpowiednie miejsce X lub O (funkcja mark())

* Argumenty to który gracz (od tego zależy czy wstawiane będzie X czy O) i tupla która reprezentuje współrzędne na tablicy
* Zwracana jest plansza, która następnie jest wyświetlana
* Zastanawiam się jak widzisz samo wstawianie znaków X/O w miejscu kropek? Poprzez funkcę .replace() i podanie odpowiedniego indeksu? Choć w sumie całoby się to chyba zrobić po prostu przez przypisanie nowej wartości. Np. linia\_A[2] = „O”

Myślę że oba sposoby są dobre

1. (Kamil) Wyświetlanie stanu tablicy po każdym ruchu (print\_board())

* Korzystanie z gotowego szablonu, może da się za pomocą f string dodać w odpowiednie miejsce odpowiednie komórki z listy ?? Myślę, że jest to bardzo dobry i łatwy do zrealizowania pomysł!
* Funkcja musiałaby przyjmować jako argument planszę A gdyby to było na zasadzie print, tak jak mówiliśmy wcześniej. Miałoby zdefiniowaną planszę w f”…” a w nawiasach {} wstawialibyśmy elementy listy o odpowiednich indeksach. Każdy input gracza/komputera modyfikowałby konkretne elementy listy i zgodnie z tym potem by drukowało aktualny stan planszy.

Żle się wyraziłam, chodziło mi że argument to byłaby ta lista list już po wykonaniu zmian, bo ta funkcja print mogłaby być wykonywana w funkcji mark()

Chodzi mi konkretnie o coś takiego: Obraz zawierający tekst

Opis wygenerowany automatycznie

Obraz zawierający tekst

Opis wygenerowany automatycznie

1. Sprawdzanie czy nie ma wygranej - myślałam by uzależnić to od ilości wykonanych ruchów, czyli jak player 1 zrobił 3 ruchy to od tego momentu sprawdzać czy wygrał Tak, czyli sprawdzenia dopiero od 5 ruchu

* 8 kombinacji na wygranie – umieszczenie ich w strukturze danych ?? Lista? Tutaj masz na myśli porównywanie ze sobą list? Czy jak to widzisz?

Właśnie sama nie wiem jak się za to zabrać, pierwszy pomysł to było właśnie stworzenie listy np. [“A1B1C1”, itd.] i trzeba by łączyć w stringa to co wpisał użytkownik i sprawdzać czy jest to elementem tablicy, ale znowu musielibyśmy robić różne kombinacje z tego co wpisał użytkownik . Cały czas myślę nad tym jak to rozwiązać, póki co chyba najtrudniejsza część do napisania

* Mi po głowie krążą tutaj znowu if, konkretnie coś typu: if „.” not in linie[a] and „O” not in linie[a]: print(„X wygrał”). Ale obawiam się, że było by to baaaardzo dużo ręcznej roboty….

1. Sprawdzenie czy tablica jest pełna, też można uzależnić od ilości ruchów bo w sumie może być ich tylko 9, trochę bez sensu ta funkcja w wymaganiach bo wystarczy właśnie kontrolować chyba ilość ruchów Tak, myślę, że to jest dobry tok rozumowania. Ewentualnie można to zrobić przez if „.” not in linie, gdybyśmy chcieli zrobić coś bardziej uniwersalnego, do rozbudowy w przyszłości na większą ilość pól.
2. Wyświetlenie na koniec gry komunikatu o wygranej / przegranej / remisie
3. Funkcja która zarządzałaby rozgrywką

* Najpierw inicjalizacja planszy
* Get\_move()
* Print\_board()
* Mogłoby to być w while True, albo while uzależniony od ilości ruchów