Zadania

Twoim zadaniem będzie przygotowanie gry “Kółko i krzyżyk”. Poniższe polecenia pomogą Ci stopniowo przygotowywać pełen projekt.

1. Utwórz nowy plik “kolko_i_krzyzyk.py”
2. Zastanów się (i napisz w komentarzu) jakie czynności powinny zostać wykonane podczas gry. Po zakończeniu porównasz swoje założenia z tymi zaprezentowanymi w dalszej części instrukcji.

- Czynności wykonywane tylko raz, na początku gry,
- Czynności wykonywane co rundę, powtarzające się,
- Czynności wykonywane tylko raz, na koniec gry.

   Jakie zmienne będą potrzebne do nadzorowania gry?

3. Przygotuj zmienne dla symboli graczy oraz samą planszę.

Zmienne *gracz_x*, *gracz_o* powinny przechowywać pojedynczy znak **X** lub **O** jako napis. Zmienna *plansza* powinna być listą, która w środku zawiera trzy kolejne listy - odpowiadające wierszom na planszy. W każdej z komórek listy będzie litera alfabetu odpowiadająca miejscowi na planszy:

a	b	c
d	e	f
g	h	i

To właśnie za pomocą tych nazw będziemy wskazywać miejsce, do którego należy wpisać kółko lub krzyżyk.

4. Przygotuj funkcję o nazwie *game*, która nie przyjmuje żadnych argumentów oraz nie zwraca żadnych wartości. Zastosuj type hinting dla zwracanej wartości*. Dodaj komentarz opisujący tę funkcję jako “Główna funkcja gry”. Zastosuj słowo kluczowe pass.

Ta funkcja będzie zawierać pełen opis gry oraz wywołania do funkcji pomocniczych.

5. Wewnątrz funkcji *game* dodaj powitanie, które wyświetli się w konsoli.

   Witaj w grze 'Kółko i krzyżyk'!

6. Dodaj zmienną *aktualny_gracz*, której wartość początkowa powinna przyjąć znak **X**’a.

Czy powinieneś na sztywno wpisać tam napis **X**?

Odp.: [redacted]

7. Dodaj zmienną, która będzie liczyć wykonane ruchy (rundy) *licznik_rund*.

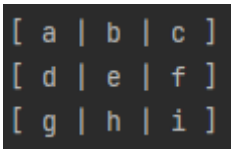
Jaką wartość początkową ustawisz tej zmiennej? Odp. [redacted]

8. Trzy poprzednie punkty opisywały kod wykonywany jednorazowo na początku gry. Wykorzystaj pętlę, aby przygotować miejsce dla kodu powtarzanego co rundę.

Jaką pętlę wykorzystasz? Z jakim warunkiem lub elementami?

Odp.: [redacted]

9. Pierwszym krokiem powtarzanym co rundę będzie rysowanie planszy w konsoli. Przygotuj funkcję *rysuj*, która wypisze zawartość zmiennej *plansza_gry* w konsoli w formacie:



Następnie należy wywołać funkcję *rysuj* wewnątrz pętli.

10. Kolejnym krokiem jest odczytanie ruchu gracza.

Wyświetlany komunikat powinien przyjąć format:

Gracz 'X': Podaj nazwę pola, które chcesz zaznaczyć:

Gracz 'O': Podaj nazwę pola, które chcesz zaznaczyć:

w zależności od tego kogo obecnie jest ruch. Odczytana nazwa pola powinna zostać zapisana w nowej zmiennej o nazwie *ruch*.

11. Następnie należy wyczyścić ekran.

Do tego celu wykorzystamy nową funkcję *wyczysc()*, której zadaniem będzie wyczyszczenie konsoli: pozorne lub prawdziwe. Pozorne czyszczenie -> wypisać kilka znaków nowej linii. Prawdziwe czyszczenie -> wyszukać w internecie jak w pythonie można czyścić konsolę - część dla chętnych.

12. Po wyczyszczeniu ekranu należy wykonać zadany ruch.

Należy jednak pamiętać o tym aby sprawdzić, czy wprowadzona nazwa pola jest poprawna oraz czy podane miejsce jest wolne (czy wcześniej nie wpisano tam znaku gracza). Do tego celu przygotuj funkcję, która będzie przyjmować dwa argumenty:

- wprowadzony przez użytkownika tekst (zmienna *ruch* z punktu 10) - oczekiwany typ argumentu to **str**
- znak X lub O w zależności jaki gracz obecnie wykonuje ruch - oczekiwany typ to **str**

Funkcja będzie zwracać wartość **bool**. True, jeśli udało się poprawnie wstawić znak w pole, False jeśli coś zawiodło.

Co należy sprawdzić wewnątrz funkcji? Kolejność wykonywanych działań:

- Czy wprowadzono tylko jeden znak? Jeśli jest inaczej - False
- Czy wprowadzono poprawna litere pola? Jeśli inaczej - False
- Wprowadzono poprawne pole, jakie to indeksy w liście? **Tutaj należy wywołać funkcję *dekoduj*, która opisana jest w kolejnym punkcie.**
- Odczytanie znaku znajdujący się na planszy - odczytanie elementu z listy *plansza* do zmiennej. Musimy zapisać sobie jaki znak znajduje się w wybranym polu. Jakie są opcje rezultatu?

Odp.: [redacted]

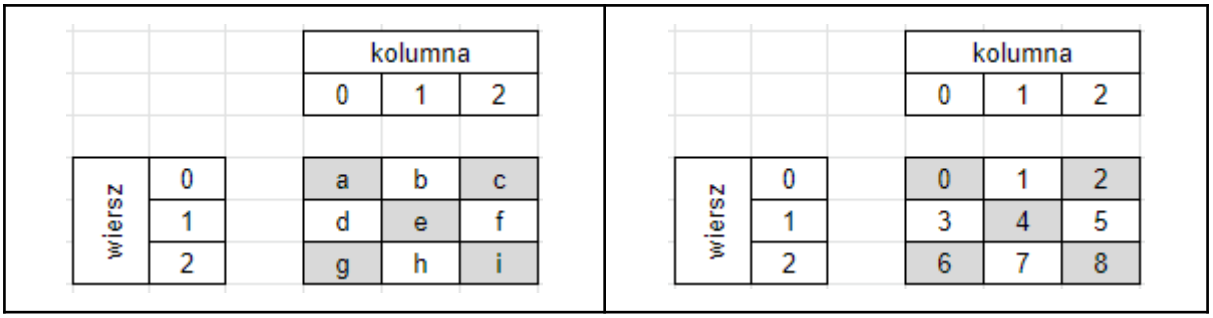
- Czy pole jest zajęte? Czy znak to X lub O? Jeśli tak jest należy zwrócić False
- Pole nie jest zajęte, można je zaznaczyć i zwrócić True, ponieważ ruch jest w pełni poprawny

13. Opis działania funkcji *dekoduj*. Zadaniem funkcji jest dekodowanie wartości tekstowej od a do i (pola planszy w kółko i krzyżyk) na wartości indeksów dla wierszy **r** oraz kolumn **c**. Funkcja przyjmuje pole wskazane przez użytkownika (typ **str** - od znaku *a* do znaku *i*), a zwraca parę indeksów, które odwołują się do tego miejsca w tablicach zmiennej *plansza*.

Zastanów się jak to rozwiązać, a następnie porównaj swoje pomysły z poniższymi rozwiązaniami.

Istnieje wiele sposobów, aby spełnić podane założenia:

- najprostsze z nich to przygotowanie dziewięciu if-ów, w których dla każdego pola zwracamy odpowiednią krotkę (Tuple) z indeksami,
- trudniejszym sposobem, ale zajmującym mniej miejsca w kodzie jest wyliczenie obu indeksów na podstawie numeru pola. Podpowiedź poniżej:



Dla każdej litery oczekiwany rezultat to wartość *(wiersz, kolumna)* odczytana z tabeli.

14. Rezultat działania funkcji *wykonaj* należy zapisać do zmiennej *czy_wykonano*. (W pętli wewnątrz funkcji *game*). Jeśli funkcja zwróciła wartość False należy wyświetlić komunikat:

Podano niepoprawną nazwę pola: zz

gdzie w miejsce *zz* wyświetli się wprowadzony przez użytkownika napis.
Następnie należy przejść do kolejnej iteracji pętli pomijając dalszą część kodu zawartego w ciele pętli. Jak to zrobić? Odp.: [redacted]

15. Po powyższej instrukcji warunkowej mamy już pewność, że wykonano ruch, który był poprawny. Należy więc sprawdzić, czy obecna sytuacja na planszy pozwala wygrać grę.

Tworzymy funkcję *wygrany*, która zwraca typ **str**:

- znak wygranego gracza X lub O
- pusty napis, jeśli nikt nie wygrał

Opis funkcji *wygrany*. Należy sprawdzić kombinacje pól, które mogą dać zwycięstwo:

- 3 poziome
- 3 pionowe
- 2 na skos

Jeśli, w którejś z tych kombinacji wszystkie 3 pola mają taki sam znak to oznacza, że mamy zwycięzcę i należy zwrócić znak znajdujący się w takim polu. Na trzech polach jest to samo, więc zwracamy znak z dowolnego pola (z tych trzech).
Po sprawdzeniu wszystkich kombinacji należy zwrócić pusty napis, ponieważ brak wygranej oznacza, że zwycięzcy brak i należy kontynuować grę.

16. Rezultat funkcji *wygrany* należy zapisać do zmiennej *wygrany_gracz*. Jeśli zawartość zmiennej różni się od pustego napisu należy:

- wywołać funkcję rysuj, aby zaprezentować finalną wersję planszy
- ogłosić zwycięzce za pomocą komunikatu o postaci:

Zwycięża gracz 'X'

Zwycięża gracz 'O'

- przerwać wykonywanie pętli (ponieważ mamy zwycięzcę).

Jak to zrobić? Odp. [redacted]

17. Po sprawdzeniu powyższego warunku (dalej wewnątrz pętli) zwiększamy licznik rund i sprawdzamy czy osiągnęliśmy maksymalną liczbę rund.

Jeśli osiągnięto maksimum należy:

- wywołać funkcję rysuj, aby zaprezentować finalną wersję planszy
- ogłosić remis za pomocą komunikatu
- przerwać wykonywanie pętli

Ile rund to maksimum? Odp. [redacted]

18. Ostatnim krokiem jest dodanie zmiany aktualnego gracza. Podpowiedzi:

[redacted]

19. Wywołaj gotową funkcję *game*.