Zofia Dusińska

Katarzyna Krawczyk

Sprawozdanie 2

Symulator gry w blackjacka

Opis problemu:

Głównym celem gry jest uzyskanie sumy punktów jak najbliższej bądź równej wartości 21. Ważne jest, aby liczba ta nie została przekroczona, gdyż powoduje to automatyczną przegraną danego gracza. Jeżeli podczas gry jeden z graczy zdobędzie 21 punktów, wówczas kompletuje on „blackjacka", co oznacza jego automatyczną wygraną (nawet na samym początku rozgrywki). Jednakże w przypadku uzyskania takiej samej punktacji (nawet 21 punktów przez obu), rozgrywka kończy się remisem.

Wartości kart:

- Walet/Jopek (J), Król (K) i Dama (Q) mają wartość równą 10 punktów;
- As może przyjmować dwie wartości punktowe 1 lub 11 punktów, zależnie od korzyści dla uczestnika;
- Karty od 2 do 10 mają wartość równą odpowiednio swojej numeracji;

Do rozpoczęcia gry potrzebne są dwie osoby: gracz oraz krupier, którzy na początku rozgrywki otrzymują po dwie karty (w przypadku krupiera – jedna jest zakryta). Pierwszeństwo ruchu ma gracz, który może poprosić o następną kartę lub zrezygnować z tej możliwości, jednocześnie przechodząc do podsumowania wyników. W przypadku podjęcia decyzji przez gracza o zakończeniu tury, krupier musi dobrać karty, jeśli suma jego punktów jest mniejsza niż 17. Kiedy wartość jego kart jest większa bądź równa 17 punktów, gra zostaje zakończona – sprawdzany jest wynik danej rozgrywki.

Opis ideowy opracowanego schematu rozwiązania:

Ideą naszego projektu jest stworzenie programu, który pozwoli użytkownikowi grać w grę Blackjack. Gracz zostanie zapytany czy chce rozpocząć grę. Po rozpoczęciu na ekranie zobaczy dwie swoje karty, które zostały mu przydzielone w losowy sposób przez program oraz dwie karty krupiera (jedna karta pozostaje zakryta). Od tego momentu gracz będzie mógł podjąć decyzję, czy chce dobrać kolejną kartę czy przejść do podsumowania punktów, zgodnie z poleceniami wyświetlanymi w konsoli. Kiedy gracz zdecyduje, że chce dobrać kartę, punkty zostaną automatycznie doliczone do jego puli. W przypadku obu graczy, jeżeli punktacja jednego z nich przekroczy próg 21 punktów, gra zostaje automatycznie zakończona z komunikatem, iż wygrywa

przeciwnik danego gracza. Kiedy jednak gracz nie będzie chciał dobrać więcej kart może zakończyć swoją turę, a tym samym karty zacznie dobierać krupier (jeśli suma jego punktów jest mniejsza niż 17). Tak samo jak u gracza są one losowo dobierane przez program, lecz od razu wszystkie stają się widoczne dla gracza. Program dobiera krupierowi karty aż do momentu, kiedy ich wartość będzie równa 17 przy tej wartości wypisuje on wynik werdyktu. Ponadto jedną z możliwości werdyktu powinien być remis, kiedy suma kart krupiera i gracza jest równa.

Demonstracje użycia opracowanego programu:

Demonstrację programu przedstawimy na kilku różnych przykładach bez podanych kolorów kart, gdyż nie wpływają one na wynik gry. Na wstępie program zadaje nam pytanie: „Czy chcesz zagrać w Blackjacka?", więc aby rozpocząć grę należy wprowadzić literę „y" do konsoli.Wielkość litery wprowadzonej w konsolę nie ma znaczenia, kod programu sam sobie ją zmniejszy, jeśli będzie taka potrzeba.

Po wprowadzeniu litery „y" program losuje graczowi i krupierowi po dwie karty, tak jak wcześniej wskazałyśmy w zasadach - jedna karta krupiera pozostaje zakryta. Pod wyświetlanymi kartami znajdują się dwa komunikaty z wartością naszych kart z pierwszego rozdania oraz zapytanie dotyczące dobrania kolejnej karty. W zależności od tego jakie karty wylosują uczestnicy gry, może ona mieć od tego momentu różne warianty:

a) Gracz w pierwszym rozdaniu może dostać Asa (11pkt.) i Króla (10pkt.), w wyniku czego wartość jego kart będzie równa 21, a tym samym kompletuje on „blackjacka" i automatycznie wygra grę;
b) Gracz może wylosować dwa Asy, przez co wartość jednego z nich będzie równa 11 a drugiego 1, aby nie dopuścić do przekroczenia sumy 21. Gracz mając w ręce karty o wartości 12, zostanie zapytany; ,,Czy chce dobrać kolejną kartę?''. Jeżeli odpowie twierdząco, program wylosuje dla niego kolejną kartę np. 2, więc jego aktualny wynik to 14. Gracz wie tylko, że wartość jednej karty krupiera jest równa 7, bo drugiej nie zna, gdyż jest zakryta. Jednakże jej wartość gracz pozna dopiero kiedy przestanie dobierać karty, lub suma jego punktów przekroczy 21. Jeśli gracz zdecyduje, że chce dobrać jeszcze jedną kartę, suma jego punktów wzrośnie, np. do 18 (bo dobrał kartę o wartości 4) i w tym momencie rezygnuje on z możliwości dobrania kolejnej z kart. W przypadku zakończenia tury przez gracza, karty krupiera zostają odkryte, a ze względu na wynik mniejszy niż 17 punktów (wylosował karty 7 i 8) musi dobrać kolejną. Wylosowaną kartą okazała się 10, co daje mu łącznie 25 punktów. Ze względu na przekroczenie progu 21 punktów, gra zostaje automatycznie zakończona, z komunikatem: „Wygrywa gracz".
c) Kolejną możliwością jest, kiedy gracz w pierwszym rozdaniu dostanie króla (10pkt.), oraz 8, a przy dobraniu kolejnej karty, np. 4 wartość jego kart przekroczy 21 punktów (suma = 22 pkt.). W takim przypadku gracz przegrywa, a na konsoli wyświetli się komunikat: ,,Wynik końcowy: Wygrywa krupier''.

d) Kiedy w dwóch pierwszych kartach gracza znajduje się Walet (10 pkt.) i Król (10 pkt.), suma jego punktów jest równa 20. Gracz podejmuje decyzję o zakończeniu swojej tury. Pierwszą kartą krupiera jest 4, a po odkryciu drugiej okazuje się, że jego suma jest mniejsza niż 17 (wylosował 6). Krupier musi dobrać kartę, którą okazuje się być 7. Jego łączna suma punktów to 17, jest to próg, od którego nie dobiera on kart. Jego suma kart wynosi 17 a suma gracza 20. Program informuje nas o wygranej gracza, bo jego wartość kart jest bliższa 21 niż wartość kart krupiera.
e) Między graczem a krupierem może dojść do remisu. Dzieje się tak kiedy np. dwoma pierwszymi kartami gracza jest dama (10pkt.) oraz 9, co daje sumę 19. Gracz nie dobierze kolejnej karty, a krupier miał widoczną 5 oraz po odkryciu drugiej karty waleta (10pkt.), a dobraną kartą jest 4 – jego suma także wynosi 19. W konsoli widnieje wynik końcowy: ,,Remis''.

Po każdorazowej grze w konsoli wyświetlany jest komunikat:,, Czy chcesz zagrać w Blackjack? (y/n)'', aby gracz nie musiał od nowa włączać programu.

Przepływ danych w programie w kolejności :

1. Na samym początku program importuje moduł random oraz funkcję pobierz_grafike z pliku blackjack_cards.py.

2. Program przechodzi do pętli while, w której znajduje się pytanie: "Czy chcesz zagrać w Blackjack? (y/n) ". Gracz z pozycji konsoli musi odpowiedzieć na dane pytanie. Jeśli wpisze „n" lub jakąkolwiek inną literę, warunek pętli nie zostanie spełniony, a tym samym program się zakończy, natomiast „y" spowoduje przejście do następnej części kodu, czyli funkcji **blackjack**.

3. W funkcji **blackjack** zostają utworzone zmienne: gracz i krupier jako puste listy oraz talia kart, która zawiera wartości utworzone przez funkcję **karty_do_blackjack**. Początkowo w danej funkcji zdefiniowane zostają kolory kart (pik, trefl, karo oraz kier) jako klucze słownika, a ich wartościami będą utworzone słowniki kart dla danego koloru. W każdym z nich zachodzi iteracja przez wartości 2 – 10, w wyniku czego powstaje słownik zawierający karty, w których kluczem jest nazwa danej karty w postaci ciągu znaków (np. „5"), a wartości danych kluczy odpowiadają numerowi danej karty (np. w przypadku klucza „5", jego wartość wyniesie 5 -> „5" : 5). Następnie dochodzi do połączenia utworzonego słownika ze słownikiem zawierającym figury (Jopek, Dama, Król oraz As) wraz z odpowiadającymi im wartościami (np. „K": 10), poprzez zastosowanie operatora union „|". Podsumowując, funkcja **karty_do_blackjack** zwraca słownik reprezentujący talię kart dla każdego z koloru, w którym są one kluczami, a ich wartościami są zagnieżdżone słowniki zawierające zestawy kart dla danego koloru.

4. W pętli for i in range, powtarzającej się dwa razy, do listy gracz i krupier dodawane są karty za pomocą funkcji **losowanie_kart**, odpowiadającej za wybranie z funkcji **karty_do_blackjack** losowego koloru karty (pik/trefl/karo/kier), a następnie losowej karty wraz z odpowiadającą jej

punktacją (karty 2-10 wartość im równa, A - wartość 11/1 , Q/ J/ K - wartość 10). Ponadto wylosowana karta zostaje usunięta ze słownika, co zapobiega wylosowaniu tej samej karty ponownie podczas jednej rozgrywki. Opisana funkcja zwraca karty w postaci krotek/tuples (kolor, nazwa karty, punkty), co będzie potrzebne w dalszej części kodu. Korzystając z pętli for oraz z tej funkcji, do listy gracz i krupier dodane zostają po dwie karty z talii.

5. Dane z pętli for trafiają do następnej pętli - while True, gdzie funkcja **drukowanie_kart** drukuje karty krupiera i gracza w konsoli według funkcji **grafika_kart** [przyjmuje parametry - widzialność kart (True/False) oraz karty w postaci listy (gracz/krupier)].

Na początku w tej funkcji (**grafika_kart**) utworzona zostaje pusta lista wszystkie_karty. Następnie pętla for z wykorzystaniem funkcji enumerate iteruje po elementach krotki/tuple (kolor, nazwa_karty, punkty), z jednoczesnym uzyskaniem indeksu dla tej krotki/tuple. Kolejno zostaje utworzony słownik znak_koloru, w którym sprawdza się kolor wylosowanej karty (klucz), a przypisaną wartością jest symbol danego koloru (pik": "♠" ,trefl": ♣" ,"karo": "♦", "kier": "♥). Następnie mamy warunek sprawdzający czy dana karta jest pierwsza lub czy jest odkryta. Jeśli warunek zostanie spełniony, to zmienna karta pobiera wartości z funkcji **pobierz_grafike**, przekazując jako parametry nazwa_karty, znak_koloru[True] oraz widzialnosc_kart = True, a w przeciwnym wypadku, zmieniona zostaje jedynie wartość parametru widzialnosc_kart na False. Zastosowanie znak_koloru[True] powoduje, że z danego słownika zostanie zwrócona wartość prawdziwa.

Przechodząc do funkcji **pobierz_grafike** (na samym początku kodu została zaimportowana), w zależności od tego czy karta ma być widzialna (odkryta) bądź nie (zakryta), wybierana jest karta. Jednakże w przypadku odkrytej, występuje zależność, czy nazwa_karty nie jest “10” (zastosowanie tej samej karty dla wszystkich, spowodowałoby zniekształcenie karty w konsoli). Funkcja ta zwraca kartę w postaci listy, a w przypadku odkrytej - z uzupełnioną nazwą karty oraz jej znakiem koloru.

Wracając do funkcji **grafika_kart**, dane przechodzą do pętli for i in range, tutaj pętla dla każdego i w zakresie od długości listy karty sprawdza czy długość listy wszystkie_karty jest mniejsza bądź równa i (długość tej listy jest równa 4, gdyż każda posiada 5 stringów). Jeżeli tak, to dodaje do listy wszystkie_karty elementy karty w zależności od i, a w przypadku kiedy ten warunek nie jest spełniony - najpierw dodany zostanie pusty string, a następnie elementy karty w zależności od i (dzięki temu możliwe jest ułożenie kart obok siebie, a nie jedna pod drugą). Na końcu funkcja **grafika_kart** zwraca listę wszystkie_karty.

Dwie listy wszystkie_karty (jedna od krupiera, druga od gracza) przechodzą do funkcji **drukowanie_kart** (przyjmuje parametry karty_krupiera i karty_gracza, czyli przekazane listy wszystkie_karty). W funkcji tej drukowane są najpierw karty krupiera z wykorzystaniem pętli for na parametrze karty_krupiera, a następnie w ten sam sposób karty gracza. Nad kartami drukowane są jeszcze zdania “Karty krupiera/gracza: “ odpowiednio nad danymi kartami.

Po przejściu danych przez powyższe funkcje otrzymujemy cztery karty wypisane w konsoli (tylko jedna karta krupiera jest odkryta, zgodnie ze zmienną karta_odkryta), które zostały wylosowane przez program wraz z wypisanym jedynie aktualnym wynikiem gracza , który został wyświetlony dzięki instrukcji print sumującej punkty listy gracza, poprzez zastosowanie funkcji **suma_punktow**, która przyjmuje parametr karty (w postaci krotki/tuple), czyli naszą listę gracz/krupier.

Funkcja ta sumuje punkty poprzez iterację elementu punkty w krotce/tuple karty z wykorzystaniem list comprehension, co zostaje przypisane do zmiennej suma. Następnie ponownie przy wykorzystaniu tej techniki zostaje utworzona nowa lista karta_as, która zawiera tylko elementy z listy karty, w którym występuje ciąg znaków ''A'', czyli karty As. Jeżeli zmienna suma jest większa od 21 i zmienna karta_as jest prawdziwa, to iterując po elementach krotki/tuple w karcie, w przypadku kiedy trafimy na kartę As, to od sumy kart odejmowane jest 10 punktów. Jednakże jeśli suma jest mniejsza lub równa 21, pętla jest przerywana. Funkcja **suma_punktow** zwraca zmienną suma.

6. Dane trafiają do warunku, który sprawdza: czy suma punktów krupiera lub gracza jest równa 21 lub czy suma punktów gracza jest większa od 21. Funkcja **suma_punktow** działa tak samo jak zostało to opisane w poprzednim punkcie (5.) Jeśli warunek jest spełniony to pętla while True zostaje przerwana i dane przechodzą do następnej pętli while. Jeśli jednak nie został on spełniony, to dane przechodzą do kolejnego warunku pytającego gracza, czy chce dobrać kolejną kartę. Jeśli odpowie twierdząco (wpisze w konsolę dowolnej wielkości “y”), zostanie mu dodana karta z wykorzystaniem funkcji **losowanie_kart** (przyjmująca parametr talia_kart), którą opisano w punkcie 4. W przypadku kiedy nie zdecyduje się dobrać karty, pętla zostanie przerwana.

7. Następnie dane przechodzą do pętli while z warunkiem, który sprawdza, czy suma punktów krupiera jest mniejsza od 17 oraz czy suma punktów gracza przekroczyła 21 punktów (funkcja **suma_punktow** została opisana w punkcie 5.). Do momentu, kiedy któryś z warunków nie będzie spełniony, to spowoduje dobranie kart przez krupiera z wykorzystaniem funkcji **losowanie_kart** (opisaną w punkcie 4.) z parametrem talia_kart.

8. Przedostatnim etapem w funkcji **blackjack** jest wydrukowanie kart za pomocą funkcji **drukowanie_kart** (opisana w punkcie 5.), która jako parametry przyjmuje “grafikę kart” z funkcji **grafika_kart** krupiera oraz gracza (dana funkcja została opisana w punkcie 5.). Jednakże w tym przypadku wszystkie karty (zarówno gracza, jak i krupiera) są odkryte.

9. Ostatnim etapem w głównej funkcji gry **blackjack** jest wydrukowanie porównania wyników gracza i krupiera za pomocą funkcji **porownanie_wynikow**, która jako parametry przyjmuje sumy punktów gracza i krupiera (f. **suma_punktow** opisana w pkt. 5.). Funkcja ta zwraca opisane za pomocą f-stringów punkty krupiera, gracza oraz pierwszą wartość logiczną (True) opisującą wynik porównań zdefiniowanych w słowniku.

Uwagi i komentarze:

Podczas tworzenia programu zauważyłyśmy, że w niektórych przypadkach możliwe jest zmniejszenie ilości wykorzystywania zmiennych lokalnych/ globalnych oraz budowania akcji przy użyciu następstw instrukcji, poprzez zastosowanie innych metod. Oto kilka z nich:

- W celu zmniejszenia ilości zmiennych stworzyłyśmy funkcję, która zwraca nam utworzoną talię kart w postaci zagnieżdżonych słowników w słowniku, zamiast zastosowania zmiennej talia_kart jako słownika już z wypisanymi elementami;
- W funkcji porownanie_wyników dla zminimalizowania ilości użytych instrukcji warunkowych, zamiast w warunkach przyrównywać liczbę punktów gracza i krupiera do danej liczby lub do siebie (np. if punkty_krupiera == punkty_gracza: zwróć 'remis'), to wykorzystałyśmy słownik (nieprzypisany do żadnej zmiennej), w którym porównywane są parametry funkcji porownanie_wynikow do liczb lub ze sobą. Ostatecznie z danego słownika program będzie szukał pierwszego z kluczy, który ma wartość "True" (dla danych parametrów) i zwróci jego wartość;
- Podobnie postąpiliśmy w początkowej części funkcji grafika_kart, używającej najpierw instrukcji warunkowych (if, elif), aby przypisać kolor karty (pik, trefl, karo, kier) do odpowiadającego mu symbolu (♠, ♣, ♦, ♥). Po zastanowieniu się nad możliwościami kodu stwierdziłyśmy, że używając jedynie pętli for oraz funkcji enumerate możemy iterować po elementach listy karty. Dzięki temu każdy element listy jest krotką/tuple (kolor, nazwa_karty, punkty), a enumerate dostarcza indeks dla każdego elementu (unikamy kilkukrotnego powtarzania warunku elif). Zostaje utworzony słownik znak_koloru, w którym sprawdza się kolor wylosowanej karty (klucz), a przypisaną wartością jest symbol danego koloru (pik": "♠" ,trefl": ♣" ,"karo": "♦", "kier": "♥). Wartość prawdziwą dla danego klucza otrzymuje się poprzez wykorzystanie znak_koloru[True] w dalszej części kodu.