TDP002 DAT1 2024-08-27 08--13

I denna tenta förekommer flera specialsymboler i en given fil och i uppgifterna. De särskilda symbolerna är ♠, ♥, ♦, ♣. Dessa är desutom färgsatta i de givna kodfilerna. OM du stöter på problem med att jämföra eller visa dessa symboler så är det helt ok att inte använda dem – Du hittar en variant av den givna filen ”deck.py” där dessa symboler ersatts med en bokstav ”deck_no_symbols.py” som du kan använda istället.

Uppgift 1 - Blackjack

Blackjack är ett kortspel mellan en (eller flera) spelare och en dealer, och för denna uppgift ska du skapa en förenklad version av spelet som körs i terminalen. Spelet går ut på att få en hand med kort som har ett totalt värde så nära 21 som möjligt, utan att överskrida 21. Varje kort har ett värde som motsvarar kortets valör, förutom knekt (J), dam (Q) och kung (K) som alla har värdet 10, och ess (A) som kan ha värdet 1 eller 11 beroende på vad som är mest fördelaktigt för spelaren / dealern. (ex. A + 10 = 21, A + 10 + 10 = 21, A + 9 = 20, osv.)

Spelet är i dealerns favör. Om spelaren och dealern har samma poäng vinner dealern. Om båda spelaren och dealern är över 21 poäng vinner dealern. Om både spelaren och dealern har under 21 poäng vinner den som har högst poäng.

Spelaren och dealern ska turas om att dra ett kort vardera i denna förenklade variant av Blackjack, och spelaren får sin tur först. Spelaren kan välja att dra ett kort eller att stanna, och när man stannar drar man inga fler kort. Dealern ska dra kort tills dess att deras hand är ett värde som är lika med eller större 17.

Om dealern eller spelaren stannar ska den andra parten fortsätta att dra kort tills de stannar eller är över 21.

Körexempel 1 (användarinmatning i fetstil):

```
Dina kort är: [], total valör: 0
Dealerns kort är: [], total valör: 0
Vill du ta ett kort? (j/n) j
Du drog: ("A", "♥")
Dealern drog: ("K", "♦")
Dina kort är: [("A", "♥")], total valör: 11
Dealerns kort är: [("K", "♦")], total valör: 10
Vill du ta ett kort? (j/n) j
Du drog: ("J", "♣")
Dealern drog: ("2", "♠")
Dina kort är: [("A", "♥"), ("J", "♣")], total valör: 21
Dealerns kort är: [("K", "♦"), ("2", "♠")], total valör: 12
Vill du ta ett kort? (j/n) n
Du stannar.
Dealern drog: ("8", "♣")
Dealerns kort är: [("K", "♦"), ("2", "♠"), ("8", "♣")], total valör: 20
Dealern stannar.
Du vinner! Din hand är bättre än dealerns.
```

Uppgift 2 – Kortstatistik

En av de ovanligaste händerna att dra i poker är en färgstege. Om man drar fem kort från en kortlek är sannolikheten att dra en färgstege 0.00139%. En färgstege (straight flush) är fem kort i samma färg och i följd. (ess kan vara både högst och lägst i en stege)

ex: [(A, ♥), (2, ♥), (3, ♥), (4, ♥), (5, ♥)] # Ess är lågt (1) i denna stege

ex: [(10, ♦), (J, ♦), (Q, ♦), (K, ♦), (A, ♦)] # Ess är högt (14) i denna stege

En färg (flush) är en hand där alla fem kort har samma färg, utan någon särskild valör. Detta är en hand som är markant mycket enklare att dra än en färgstege, med en sannolikhet på 0.1965%.

ex: [(A, ♥), (5, ♥), (9, ♥), (Q, ♥), (K, ♥)]

En vanlig stege (straight) är fem kort där valörerna är i följd, men färgen spelar ingen roll. (ess kan även här vara både högst och lägst) Sannolikheten att dra en vanlig stege på fem kort är 0.3925%, alltså ungefär dubbelt så stor som att dra en färg.

ex: [(A, ♥), (2, ♥), (3, ♦), (4, ♣), (5, ♠)] # Som straight flush, fast utan hänsyn till färg

ex: [(10, ♦), (J, ♥), (Q, ♦), (K, ♦), (A, ♦)] # Även här kan ess vara både högst och lägst

Din uppgift är att skriva en funktion `count_hands` som ska simulera att vi drar fem kort tills vi får en färgstege. Denna funktion ska köra en oändlig loop där vi varje varv i loopen skapar en ny kortlek med new_deck(), blandar den, och drar fem kort. Funktionen ska sedan kontrollera om handen innehåller en färgstege, stege eller färg, och hålla koll på hur många gånger vardera hand dras. Så fort vi drar vår första färgstege ska loopen brytas (men färgstegen ska så klart räknas!). Drar vi varken en färgstege, stege eller färg ska dessa händer inte räknas

När loopen brutits ska funktionen skriva ut en tabell som visar hur många gånger vardera hand drogs. Av alla händer vi har räknat ska vi även skriva ut hur stor procentuell andel av händerna som var en färgstege, stege eller färg, avrundat till två decimaler. Tabellen ska även innehålla en stapelgraf som visar procentandelen av varje hand, där varje stapel (#) representerar 2 procentenheter (avrundat uppåt). Se körexemplet nedan.

Körexempel (användarinmatning i fetstil):

```
Handtyp            | Antal |  Procent | Staplar
-------------------|-------|----------|----------------------------------------
Straight Flush     |     1 |    2.04% | ##
Straight           |    32 |   65.31% | #################################
Flush              |    16 |   32.65% | ################
-------------------|-------|----------|----------------------------------------
Totalt antal matchande händer: 49
```

TIPS: För att förenkla testningen kan du säga till random-modulen att använda en fast seed, så att du alltid får samma ordning när du blandat korten. random.seed(51) har (under våra tester) generellt sett gett en färgstege inom 10 000 iterationer. Du kan avkommentera raden där detta står för att få ett deterministiskt beteende. Din lösning ska dock fungera för alla fall.

Uppgift 3 – Selection sort

Din uppgift är att implementera en sorteringsalgoritm som kallas för "selection sort" för att sortera en kortlek. Selection sort fungerar genom att man itererar över en lista, och för varje iteration så hittar man det minsta elementet (eller kortet, i detta fall) i listan och byter plats på det med det första elementet i listan. Därefter så "låser" man det första elementet, och tittar på nästa, letar efter det (kvarvarande) minsta elementet, och byter plats på det med det andra elementet i listan, och så vidare.

Uppgiften är att skapa en ny kortlek, blanda denna och sedan sortera den med din implementation av selection sort.

Först ska hjärter hamna i ordningen A -> K, sedan klöver i ordningen A -> K. Därefter ska ruter läggas in i ordningen K -> A, så att två kungar möts i mitten av kortleken (detta kallas ofta för "the kissing kings"). Sist men inte minst ska spader läggas in i ordningen K -> A. Vi kommer alltså att ha ess både på toppen och botten av kortleken. Titta på ordningen vi får från new_deck() för att se hur det ska se ut när det är sorterat!

TIPS: Det går bra att dela upp kortleken efter färg innan du sorterar dem, och sedan sätta ihop dem igen till en sorterad kortlek!

Exempel på selection sort:

Om vi har en lista med elementen [5, 3, 8, 1, 4], så kommer algoritmen att börja med att fastställa det första elementet i listan genom att iterera igenom hela listan, hitta det minsta elementet och byta plats på det minsta elementet och första elementet. I exemplet innebär det vi behöver undersöka 5 element i listan och att 5 byter plats med 1.

När första elementet är fastställt så fastställs det andra elementet genom att iterera igenom listan med start från det andra elementet, hitta det minsta elementet och byta plats på det och det andra elementet. I exemplet innebär det att vi behöver undersöka 4 element (eftersom första elementet redan är fastställt) och att 3 byter plats med 3 (ingenting händer). Detta upprepas tills alla element är fastställda.

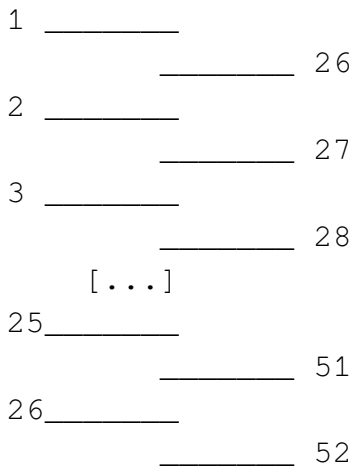
Körexempel:

```
Deck before sort: [('4', '♠'), ('8', '♥'), ('5', '♦'), ...
Deck after sort: [('A', '♥'), ('2', '♥'), ..., ('A', '♠')]
```

Uppgift 4 -- Faro shuffle (rekursion)

Ett knep som kortmagiker använder sig av för att ge intrycket av att de blandar en kortlek kallas för "Faro shuffle" (även kallad "Weave shuffle" eller "Dovetail shuffle"). Detta är ett sätt att genomföra en kontrollerad blandning av en kortlek som ger ett deterministiskt resultat. Med detta kan en skicklig magiker få det att se ut som att en kortlek blandas väldigt noga, när magikern i själva verket har full kontroll på vilken ordning korten hamnar i.

För att genomföra en Faro shuffle så delas en kortlek på mitten i två exakt lika stora delar, och därefter så "vävs" dessa halvor ihop genom att ta vartannat kort från varje halva när man "blandar" kortleken. För att illustrera detta med ett exempel kan en sådan blandning se ut såhär:



Din uppgift är att skriva en funktion som simulerar Faro shuffle tillräckligt många gånger för att kortleken ska hamna tillbaka i sitt ursprungliga skick -- alltså att alla kort ligger i sin ursprungliga ordning igen. Funktionen ska returnera antalet Faro shuffles som behövdes för att kortleken ska hamna i sitt ursprungliga skick igen. Ett krav för denna uppgift är att din funktion ska köras rekursivt, där varje anrop genomför en enskild Faro shuffle.

Du kan använda funktionen `new_deck()` för att skapa en ny kortlek. Du kan testa om kortleken är i sin ursprungliga ordning genom att jämföra om `deck == new_deck()`.

Körexempel:

```
Faro shuffles needed: 8
```

Uppgift 5 - Bästa möjliga hand

Att behärska konsten att snabbt identifiera den starkaste handen är en värdefull färdighet i många kortspel. I denna uppgift ska du utveckla ett program som kan analysera en hand med fem kort och bestämma den bästa möjliga kombinationen av kort som finns i den dragna handen.

Programmet ska dra och skriva ut fem kort från en kortlek, och sedan skriva ut den bästa kombinationen av kort som finns på handen. De enda kombinationerna som ska kontrolleras är följande, där kombinationerna längst up är bäst. Om det finns två kombinationer som överlappar ska endast den bästa kombinationen skrivas ut -- t.ex. så ska varken triss eller par skrivas ut om det finns en kåk på handen, då en kåk är mer värdefull än de andra händerna.

1. Fyrtal: Fyra kort med samma valör. Ex: A♠, A♥, A♦, A♣
   - Skriver ut: "Fyrtal, fyra ess"
2. Kåk: Tre kort med samma valör och två kort med samma valör. Ex: 3♠, 3♥, 3♦, J♠, J♥
   - Skriver ut: "Kåk, tre treor och två knektar"
3. Triss: Tre kort med samma valör. Ex: 2♠, 2♥, 2♦
   - Skriver ut: "Triss, tre tvåor"
4. Två par: Två kort med samma valör och två kort med samma valör. Ex: 5♠, 5♥, 7♦, 7♣
   - Skriver ut: "Två par, två femmor och två sjuor"
5. Par: Två kort med samma valör. Ex: Q♠, Q♥
   - Skriver ut: "Par, två damer"
6. Högt kort: Ett kort med högst valör (ess har högst valör). Ex: K♠
   - Skriver ut: "Högt kort, kung"

Körexempel:

```
$ python3 upg5.py
Din hand: [(3, ♠), (3, ♥), (3, ♦), (J, ♠), (J, ♥)]
Bästa möjliga hand: Kåk, tre treor och två knektar

$ python3 upg5.py
Din hand: [(4, ♠), (4, ♥), (4, ♦), (4, ♣), (3, ♠)]
Bästa möjliga hand: Fyrtal, fyra fyror

$ python3 upg5.py
Din hand: [(A, ♠), (Q, ♥), (K, ♦), (2, ♣), (3, ♠)]
Bästa möjliga hand: Högt kort, ess
```