Cycle 8 - Lab Exercises

Sorting with a data in a custom structure

This week we'll be working on a modified version of an exam question from last year.

The game of poker is played with a standard 52 card deck. This is a collection of 52 playing cards which has:

- four suits: hearts ♥, diamonds ♦, clubs ♣ and spades ♠
- thirteen face values: 2,3,4,5,6,7,8,9,10,J (Jack), Q (Queen), K (King), (A) Ace
  Every combination of suit and face value is present as a distinct card. For example, 2 of Spades (2♠) is valid card, as is Jack of Hearts (J♥) or Ace of Spades (A♠).
  A hand of cards is an ordered collection of cards that is held by one player; for example the five cards 3♠ J♥ J♠ A♦ 7♦ (read Three of Spades, Jack of Hearts, Jack of Spades, Ace of Diamonds, Seven of Diamonds).

Task 1:

Describe in words a data structure to represent a single card, and then using this structure, describe a data structure to represent a hand of cards. Write down a concrete example of the data structure in Python syntax for the three card hand: 2♥ 2♠ K♣

Task 2:

Write a function `greaterThan(card1, card2)` that takes two cards as input, and returns True if card1 is greater than card 2 by the following rules, False otherwise:

The rules for the ranking of an individual card are as follows:

- Cards are ranked first by face value:
- 2 < 3 < 4 < 5 <6 < 7 < 8 < 9 < 10 < Jack < Queen < King < Ace
- Cards which are tied for the same face value are ranked according to suits:
- ♦ Diamonds < ♣ Clubs < ♥ Hearts < ♠ Spades For example: 4♦ < 4♥ < 8♥ < A♥

Task 3:

Using your `greaterThan` function, implement a sorting algorithm of your choice that can take a list of cards and sort it in ranked order.

For example, given the input representing the hand 2♥ Q♠ 6♣ 4♦ Q♣ the sequence should be returned in the order 2♥ 4♦ 6♣ Q♣ Q♠.

Task 4:

In poker, when scoring the value of a hand, multiple cards must be taken into account. For example, a pair of cards with the same face value is worth more than a pair of unmatched cards with differing face values. To start implementing such rules, three functions need to be implemented:

- `is_consecutive(a, b)` must return True if a and b have consecutive face values regardless of face value and False otherwise (e.g. 2♦,3♦ is consecutive, as is J♣,Q♦ but not A♦,2♦ or 7♦,9♣ or 3♣,2♣)

- `is_matching_face(a,b)` must return True if a and b have the same face values but differing suits and False otherwise
- `is_matching_suit(a,b)` must return True if a and b have the same suits and False otherwise

Define these three functions in Python.

Task 5 - Stretch Task:

Write a function `input_hand(n)` that will take a number n, and request that number of cards from the user as text input. It should return the corresponding hand using the datastructure you described in (a). The user input will be of the form of one line per card: facevalue suit Your function must work where facevalue can be a word like five or King, as well as accepting a single a number/letter like A or 8. Your function must accept a suit that is a full word like hearts as well as a single letter like h or H. Each word will be separated with a space. The input must be case insensitive. For example input_hand(3) should read the following input correctly: five diamonds ACE S 5 CLuBs and return the representation of the hand 5♦ A♠ 5♣

You should deal with malformed input using an exception. You may use the following dictionary: `number_words = {"one":1, "two":2, "three":3, "four":4, "five":5, "six":6, "seven":7, "eight":8, "nine":9, "ten":10}`