COMP20270
OOP in Python

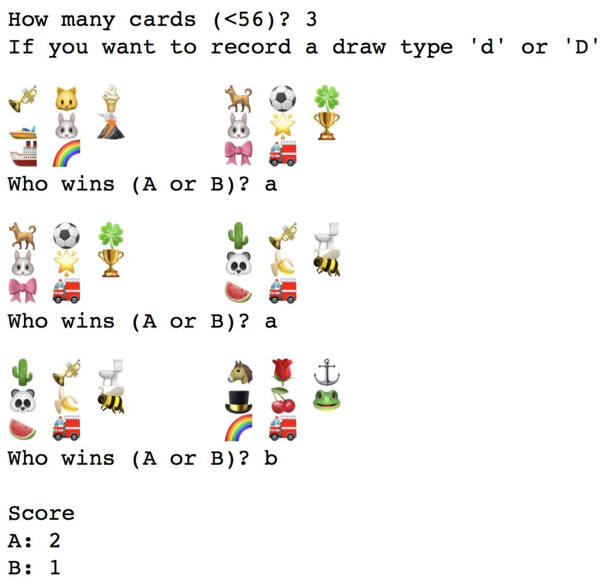
Assignment 2

Dobble

Due Thursday 28th November.

Objective

The objective of this exercise is to develop an implementation of the Dobble card game that will run in a Python Notebook. Four Dobble cards are shown in the image below. There are eight images on each card and each pair of cards will have exactly one image in common. The objective is to spot the matching image as pairs of cards are dealt.



```
How many cards (<56)? 3
If you want to record a draw type 'd' or 'D'

🎺 🐱 🍦        🐕 ⚽ 🍀
🚢 🐰 🌋        🐰 🌟 🏆
   🌈           🎀 🚒
Who wins (A or B)? a

🐕 ⚽ 🍀        🌵 🎺 🚽
🐰 🌟 🏆        🐼 🍌 🐝
🎀 🚒           🍉 🚒
Who wins (A or B)? a

🌵 🎺 🚽        🐴 🌹 ⚓
🐼 🍌 🐝        🎩 🍒 🐸
🍉 🚒           🌈 🚒
Who wins (A or B)? b

Score
A: 2
B: 1
```

Figure 1. (a) Real Dobble cards. (b) A sample game in a Python Notebook.

Stages

1. **Generate image IDs to produce valid cards.** The Notebook DobbleGenerator produces IDs for valid Dobble decks, e.g. a deck with 13 cards and 4 image-ids.

   ```
   Card  1: 1 2 3 4
   Card  2: 1 5 6 7
   Card  3: 1 8 9 10
   Card  4: 1 11 12 13
   Card  5: 2 5 8 11
   Card  6: 2 6 9 12
   Card  7: 2 7 10 13
   Card  8: 3 5 9 13
   Card  9: 3 6 10 11
   Card 10: 3 7 8 12
   Card 11: 4 5 10 12
   Card 12: 4 6 8 13
   Card 13: 4 7 9 11
   ```

   Dobble requires that each pair of cards match on exactly one image. This is not possible for all image counts. It is only possible when the number of images on the card is a prime number plus 1, e.g. 3,4,6,8.

   Modify the code from DobbleGenerator so that the deck is stored in a dictionary with the card No as key and the image ids (in a set) as the value.

   Write a check_validity function that will take a deck as argument and check for validity, i.e. check that each card matches on one and only one image. This function should be callable in two modes:

   `check_validity(deck, verbose = True)`

   `check_validity(deck)`

   The verbose mode should produce output as each pair is checked.

   Hint: If the image ids are stored in a set, the intersection method can be used for the check.

2. **DobbleDeck and DobbleCard classes:** Develop two classes DobbleCard and DobbleDeck where DobbleDeck holds a deck of cards stored as DobbleCard instances. These classes need only work for 8 image cards as shown in Figure 1 (b). Code for loading emoji images is also provided in the DobbleGenerator notebook.

   DobbleDeck should have `add_card`, `remove_card` and `play_card` methods.

3. **Dobble Game:** Write a function that will allow two users to play a Dobble game. The function will present 'cards' as lists of emojis. The users decide who wins a hand and the function records the win counts.

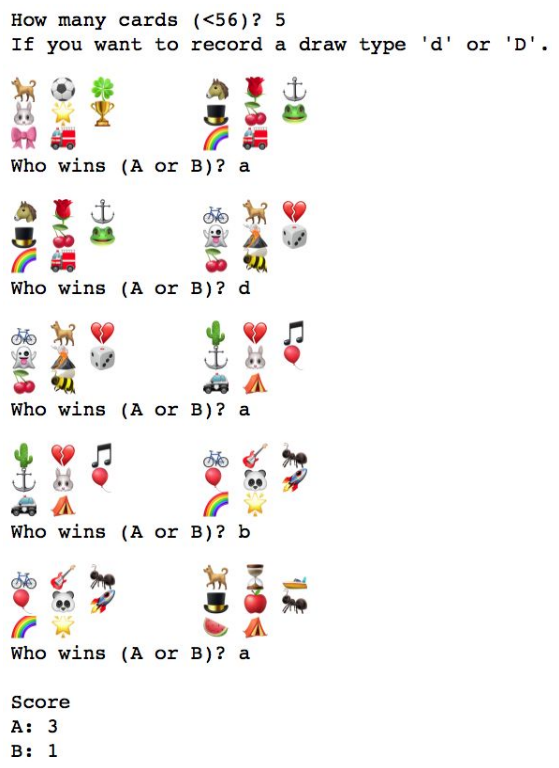
Each card in Dobble features eight different symbols. Any two cards have exactly one symbol in common. Altogether there should be 57 valid cards. For the basic game, reveal one card, then another. Whoever spots the symbol in common on both cards claims the first card, then another card is revealed for players to search, and so on. In this way, each card is involved in two tests. At the beginning of the game the players decide how many cards (tests) to play. In the sample games shown below, the players are A and B and a draw can be recorded as a 'd' or 'D.

Submission: This is an individual (not group) project. Submission is through the Moodle page. Your submission should comprise your notebook only. Clear all outputs in the notebook before saving for submission. You can use markdown cells in the notebook to explain any design decisions you have made.

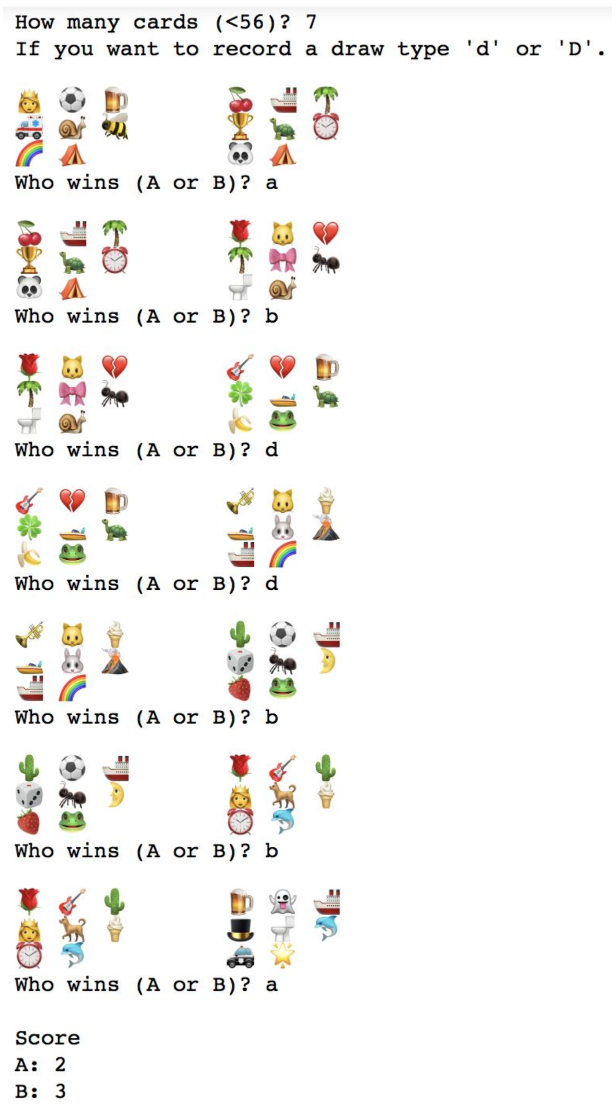
What is provided?

A Python Notebook (DobbleGenerator) is available on the Brightspace page that provides some useful code, code for generating valid image ids and code for loading emoji images.

Sample games



```
How many cards (<56)? 5
If you want to record a draw type 'd' or 'D'.

Who wins (A or B)? a

Who wins (A or B)? d

Who wins (A or B)? a

Who wins (A or B)? b

Who wins (A or B)? a

Score
A: 3
B: 1
```



```
How many cards (<56)? 7
If you want to record a draw type 'd' or 'D'.

Who wins (A or B)? a

Who wins (A or B)? b

Who wins (A or B)? d

Who wins (A or B)? d

Who wins (A or B)? b

Who wins (A or B)? b

Who wins (A or B)? a

Score
A: 2
B: 3
```