Computing

Lesson 6: List Methods

Programming Part 5: Strings and Lists

Rebecca Franks

Create a deck of cards

Code Snippets

```python
words = ["python", "java", "C+", "ruby"]
for item in words:
    if item == "python":
        print(f"{item} is the best")
    else:
        print(f"{item} is a programming language")
```

```python
numbers = [4, 5, 3, 2, 9, 10]

location = numbers.index(9)

print(location)
```

Code Snippets

```python
numbers = [4, 5, 3, 2, 9, 10]

numbers.insert(2,10)

print(numbers)
```

```python
numbers = [4, 5, 3, 2, 9, 10]

numbers.pop(0)

print(numbers)
```

Code Snippets

```python
numbers = [4, 10, 5, 3, 2, 9, 10]

total = numbers.count(10)

print(total)
```

```python
numbers = [4, 10, 5, 3, 2, 9, 10]

numbers.reverse()

print(numbers)
```

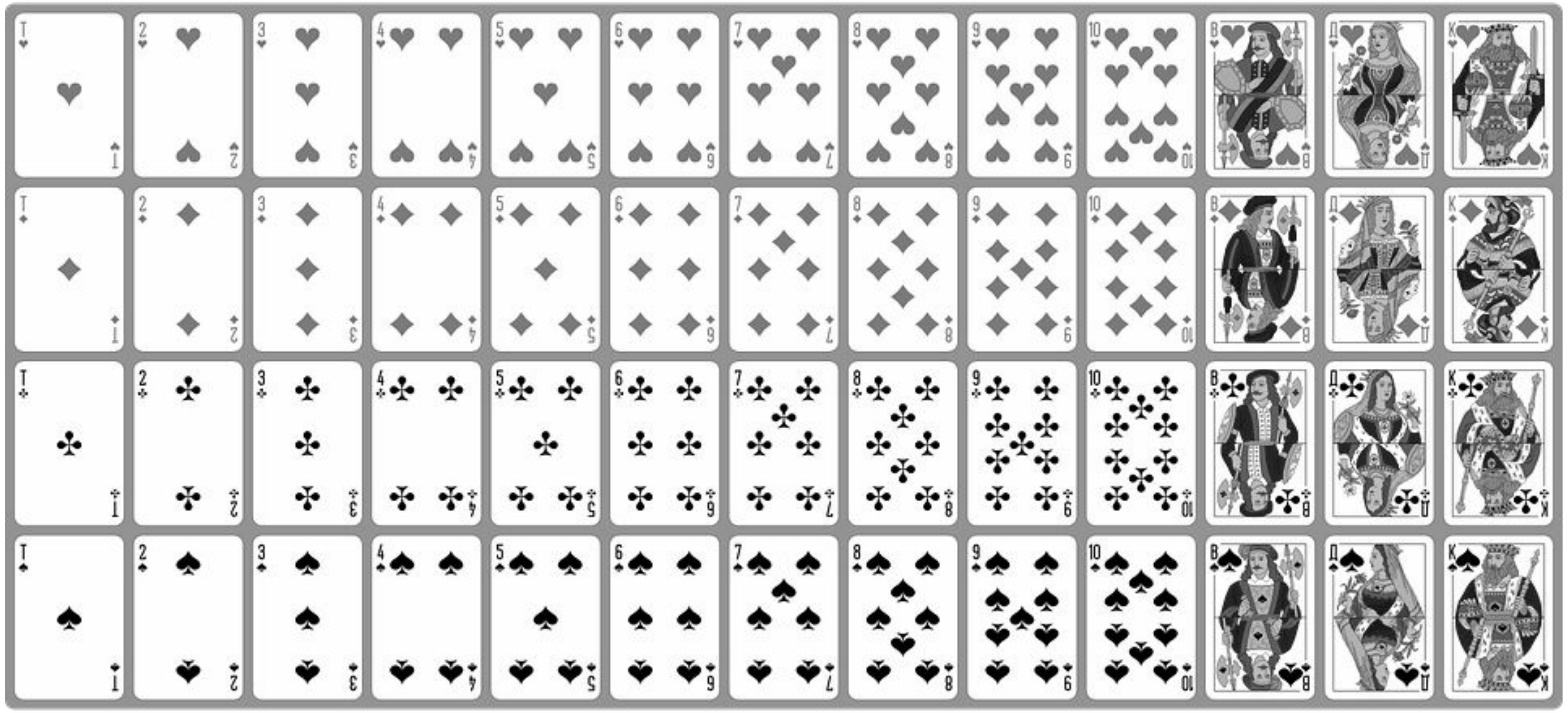
```python
numbers = [4, 10, 5, 3, 2, 9, 10]

numbers.sort()

print(numbers)
```

Scenario

For this activity you will be populating a list with a deck of cards. A deck of cards looks like the image below:



Source: Pixabay

Scenario

Each card in the pack is different but there is a clear pattern that can be replicated through an algorithm. A deck of cards contains 52 cards (not including the Joker cards).

The pack is divided into 4 suits. The suits are:

- Hearts
- Diamonds
- Clubs
- Spades

Scenario

Each suit contains 13 cards. The cards appear in this order:

- Ace (A)
- 2 to 10
- Jack (J)
- Queen (Q)
- King (K)

Your pack of cards should be a deck that includes 52 cards that are placed in the same order as the example image above.

Task 1

Use these lists as your starting code. It contains the suit symbols that will be required for the deck. It also creates a blank deck that will be populated during execution of the program.

```
suits = ["♥️", "♦️", "♣️",
"♠️"]
deck = []
```

You can find the starting code using this shortlink: **oaknat.uk/comp-oak-cards**

Task 2

For every `suit` in the deck you will need to create a card that includes the number or the letter required for that `suit`.

Start by creating a for loop that will execute for each `suit` in `suits`.

Test the loop by adding `print(suit)` inside the loop. The output should look like this:

```
♥️
♦️
♣️
♠️
>>>
```

Task 3

You now have a `for` loop that will execute for each suit in suits. Now you can start populating your deck list.

Instead of printing the suit, change the line of code so that it appends to the `deck` list with the `suit` instead. Complete the code snippet below and place it in your program.

```
deck.          (suit)
```

Task 4

Print the **deck** list at the end of your program to check that it has worked. The output should look like this:

```
['♥️', '♦️', '♣️', '♠️']
```

Task 5

Each suit has 13 cards. The cards start at 1 (Ace) and finish at 13 (King). For each `suit` in the pack you will need to create **13 cards** for the `deck`. In order to do this you will need to use a `for` loop inside a `for` loop (a nested loop).

The code below will get this started. Complete the code by filling in the blanks and make the changes in your program.

```
for suit in suits:
    for card in       (    ):
        deck.        (suit)
```

Task 6

The output from `print(deck)` should now look like this:

```
['♥', '♥', '♥', '♥', '♥', '♥', '♥', '♥', '♥', '♥', '♥', '♥',
'♥', '♦', '♦', '♦', '♦', '♦', '♦', '♦', '♦', '♦', '♦', '♦',
'♦', '♦', '♣', '♣', '♣', '♣', '♣', '♣', '♣', '♣', '♣', '♣',
'♣', '♣', '♣', '♠', '♠', '♠', '♠', '♠', '♠', '♠', '♠', '♠',
'♠', '♠', '♠', '♠']
```

Task 7

Check that you have 52 cards by printing the **length** of the cards at the end of your program. The output should look like this:

```
['❤️', '❤️', '❤️', '❤️', '❤️', '❤️', '❤️', '❤️', '❤️', '❤️', '❤️', '❤️',
'❤️', '♦️', '♦️', '♦️', '♦️', '♦️', '♦️', '♦️', '♦️', '♦️', '♦️', '♦️',
'♦️', '♦️', '♣️', '♣️', '♣️', '♣️', '♣️', '♣️', '♣️', '♣️', '♣️', '♣️',
'♣️', '♣️', '♣️', '♠️', '♠️', '♠️', '♠️', '♠️', '♠️', '♠️', '♠️', '♠️',
'♠️', '♠️', '♠️', '♠️']

52
```

Note: if it is displaying a different number then you probably have an error in the **range** that you selected in task 5.

Task 8

Now that your program is creating 52 cards, you need to add in the correct values before each suit. The variable card is holding a value from 1 to 13. You can use this in your program to add values to your suits.

Think carefully about what will need to be changed in the line of code below in order to complete this task. Don't worry about `A, J, Q, K` at the moment, this will be handled next.

Hint: concatenation will need to be used for this task to be successful. Look at old pieces of code where you have concatenated the string in order to help you with this.

```
        deck.append(        )
```

Task 9

Test your program by printing the deck again. It should output like the example below:

```
['1❤️', '2❤️', '3❤️', '4❤️', '5❤️', '6❤️', '7❤️', '8❤️', '9❤️', '10❤️',
'11❤️', '12❤️', '13❤️', '1♦️', '2♦️', '3♦️', '4♦️', '5♦️', '6♦️', '7♦️',
'8♦️', '9♦️', '10♦️', '11♦️', '12♦️', '13♦️', '1♣️', '2♣️', '3♣️', '4♣️',
'5♣️', '6♣️', '7♣️', '8♣️', '9♣️', '10♣️', '11♣️', '12♣️', '13♣️', '1♠️',
'2♠️', '3♠️', '4♠️', '5♠️', '6♠️', '7♠️', '8♠️', '9♠️', '10♠️', '11♠️',
'12♠️', '13♠️']
```

If your program has an error. Check that you have converted the value `card` to a string.

Task 10

Now is the time to add in `A, J, Q, K` in place of `1, 11, 12, 13`. Use `if-elif-else` to place the letters before the suit instead of the numbers.

Test your program by printing the deck again. It should output like the example below:

```
['A❤️', '2❤️', '3❤️', '4❤️', '5❤️', '6❤️', '7❤️', '8❤️', '9❤️', '10❤️',
'J❤️', 'Q❤️', 'K❤️', 'A♦️', '2♦️', '3♦️', '4♦️', '5♦️', '6♦️', '7♦️', '8♦️',
'9♦️', '10♦️', 'J♦️', 'Q♦️', 'K♦️', 'A♣️', '2♣️', '3♣️', '4♣️', '5♣️',
'6♣️', '7♣️', '8♣️', '9♣️', '10♣️', 'J♣️', 'Q♣️', 'K♣️', 'A♠️', '2♠️',
'3♠️', '4♠️', '5♠️', '6♠️', '7♠️', '8♠️', '9♠️', '10♠️', 'J♠️', 'Q♠️',
'K♠️']
```

Well done, you now have a deck of cards!

Task 11

Now that you have a deck of cards you can perform lots of operations that would work with a card game like Snap! for example.

Here is a code snippet that can be used to shuffle a list. Incorporate this into your program.

```
from random import shuffle
shuffle(list)
```

Print your list after it has been shuffled to check that it has worked.

Task 12

Step 1

You might have a two player game that requires each player to start with half of the shuffled deck each.

Create two new lists for `player1` and `player2`

Step 2

Populate each new list with half of the shuffled deck. Here is some sample code used to slice a list of months to support you with this:

```
summer = months[5:8]
```

Task 12

Step 3

Test your program by printing the player1 and player2 lists to see if the deck has been split.

Perform a further check by ensuring that each list contains 26 items.

The output should look like this (note that the items are randomised so will be different to your own output):

```
['10♦', '10♣', '2♥', 'K♥', '3♣', 'Q♣', '9♠', 'K♠', '4♥', '9♥', '10♥', '2♣',
'3♦', '9♣', '7♠', '6♦', '7♣', '9♦', 'J♣', '10♠', '7♦', '6♣', '4♦', '6♥',
'Q♥', '4♠']

26

['8♠', 'A♥', '4♣', 'K♣', '5♠', '8♥', 'K♦', 'A♣', '3♥', 'A♦', '8♦', '2♠',
'J♠', 'Q♠', '5♦', '7♥', 'J♦', 'Q♦', 'A♠', '8♣', '3♠', '5♣', '5♥', 'J♥',
'6♠', '2♦']
```

```
26
```