Create a deck of cards

# Support .

**Python cheat sheets**

Use the ‘Python cheat sheets’ resource to support you with the activities on this sheet.

# Scenario .

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

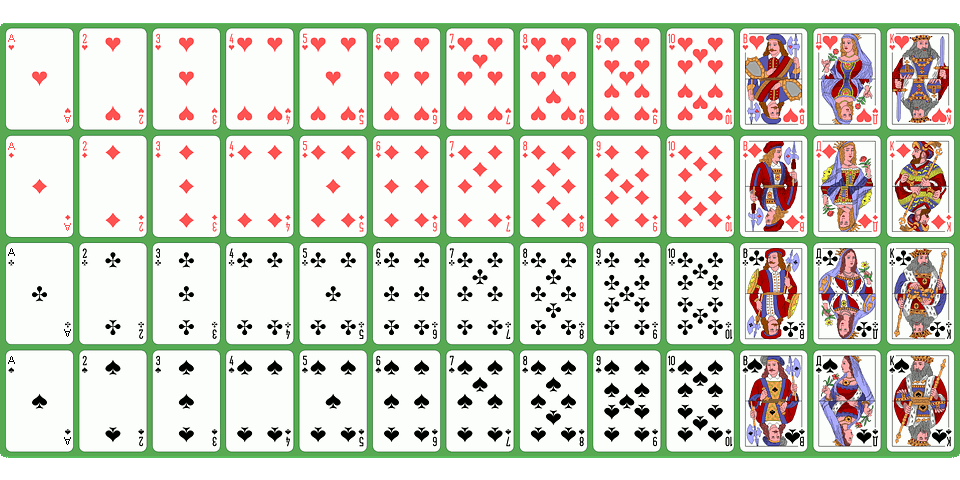


Image source: <https://pixabay.com/vectors/atlasnye-deck-playing-cards-game-884206/> (image modified to include A instead of T)

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

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 .

**Step 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 = [] |
| --- | --- |

**Step 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:

|  | ♥  ♦  ♣  ♠  >>> |
| --- | --- |

**Step 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 adds to the deck list with the suit instead. Complete the code snippet below and place it in your program.

|  | deck. (suit) |
| --- | --- |

**Step 4**

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

['♥', '♦', '♣', '♠']

**Step 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) |
| --- | --- |

**Step 6**

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

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

**Step 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 step 5.

**Step 8**

Now that your program is creating 52 cards, you need to add in the correct values before each suit. The variable x 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.

|  | deck.append( ) |
| --- | --- |

**Hint:** Concatenation will need to be used for this task to be successful. Look at the ‘Python cheat sheets’ resource to support you with this.

**Step 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 x to a string.

**Step 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.

**Step 11**

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!**

# Shuffle the deck! .

**Step 1**

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) |
| --- | --- |

**Step 2**

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

# Split the deck! .

**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. Use the cheat sheets to find out how to take a slice from the list and incorporate this into your program.

**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

# Explorer task .

**Task 1**

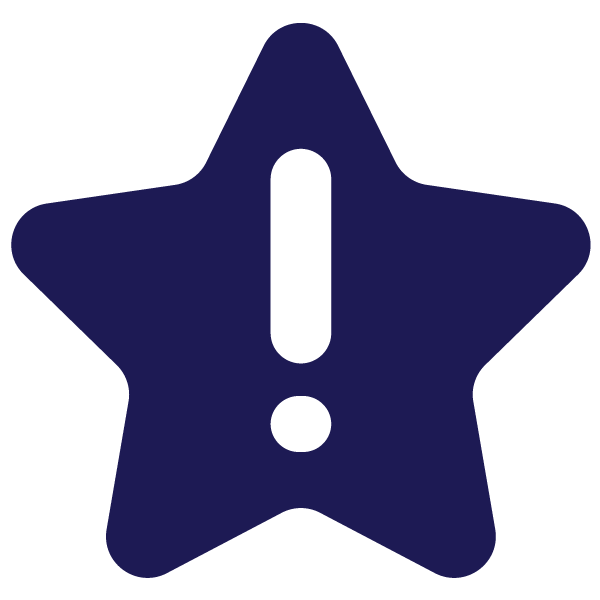
The suit symbols were created using the unicode codes. You can find more about them on the [Playing cards in Unicode Wikipedia page](https://en.wikipedia.org/wiki/Playing_cards_in_Unicode).

In order to create a unicode symbol, you need this code snippet:

|  | heart = "\u2665" |
| --- | --- |

Adapt your program so that it displays white hearts instead of black hearts. Use the Unicode Wikipedia page to help with this.

# Further explorer task .

**Note:** This will not work in Mu, but it does work in repl.it.

Make the hearts and diamonds appear in red text. A code snippet has been provided below to help you with this.

|  | def red(text):  return "\033[1;31m" + text + "\033[0m"    print(red("Hello")) |
| --- | --- |

**Your Final Code:**

|  |
| --- |