1. **Real Card Games**

|  |
| --- |
| **10.1 Aim**  To real Card Games Drawing Cards from a Deck, make deck and draw, draw a hand of cards, and deal. perform the program in Jupyter Notebook. |

**10.2 Program and Input Data**

Write Python Code & also Few input data. Include all data tables and/or observation.

|  |
| --- |
| **Program 1:**   1. Drawing cards from a deck; make deck and draw  Drawing cards from a deck; draw a hand of cardsProgram 2:a) Drawing cards from a deck; deal i. Drawing cards from a deck; analyse results (1)  Analyse the no of pairs or n-of-a-kind in a hand  ii. Analyse the no of combinations of the same suit  **10.3 Programs and output**  **Program 1:** Drawing Cards from a Deck #Make a deck of cards:  # A: ace, J: jack, Q: queen, K: king  # C: clubs, D: diamonds, H: hearts, S: spades  def make\_deck ():  ranks = ['A', '2', '3', '4', '5', '6', '7', '8', '9', '10', 'J', 'Q', 'K']  suits = ['C', 'D', 'H', 'S']  deck = []  for s in suits: # s0, s1, s2, s3 - each have 13 cards - shapes are clubs, spade, diamond, heart  for r in ranks:  deck.append(s + r)  random\_number.shuffle(deck)  return deck  print(make\_deck())  print(len(make\_deck())) #count the cards  **Output:**  ['CA', 'HQ', 'D8', 'HK', 'C2', 'H10', 'DA', 'D5', 'CQ', 'SK', 'CK', 'S10', 'H3', 'H4', 'DJ', 'HA', 'D3', 'H5', 'S2', 'C4', 'C9', 'DQ', 'HJ', 'H2', 'SQ', 'S7', 'SJ', 'C8', 'D2', 'S5', 'H7', 'D7', 'C6', 'H6', 'CJ', 'D4', 'C7', 'S6', 'C5', 'H9', 'S8', 'H8', 'C3', 'DK', 'S3', 'S4', 'S9', 'D9', 'D6', 'C10', 'D10', 'SA']  52 Drawing cards from a deck; draw a hand of cards Note: deck is returned since the function changes the list deck is changed in-place so the change aects the deck object in the calling code anyway, but returning changed arguments is a Python convention and good habit.  def deal\_hand (n, deck):  hand = [deck[i] for i in range(n)]  del deck[:n]  return hand, deck # DELETE ALL THE 52 CARDS IF USE THIS COMMANDS  def deal\_hand(n, deck):  hand = [deck[i] for i in range(n)]  del deck[:n]  return hand, deck  deal\_hand(10,deck)  **Output:**  (['DK', 'SK', 'HK', 'D4', 'H3', 'C2', 'C4', 'D8', 'C7', 'H4'],  ['S9',  'DQ',  'S6',  'DA',  'H5',  'H6',  'D7',  'S4',  'D2',  'H9',  'H10',  'D9',  'CA',  'CK',  'HQ',  'CQ',  'H2',  'HA',  'D5',  'S2',  'CJ',  'H7',  'C9',  'HJ',  'C5',  'C10',  'SA',  'S8',  'C8',  'SQ',  'D10',  'S7',  'DJ',  'S3',  'D6',  'C3',  'S10',  'S5',  'SJ',  'C6',  'H8']) Program 2:Drawing cards from a deck; deal #Deal hands for a set of players:  def deal(cards\_per\_hand, no\_of\_players):  deck = make\_deck() #52 cards in deck  print(deck)  print(len(deck))  hands = [] #empty array--> count=0; consists of nothing  for i in range(no\_of\_players): # 4players(i=0,1,2,3 ---> 4times  hand, deck = deal\_hand(cards\_per\_hand, deck) #5,52  print(hand)  print(deck)  hands.append(hand) #adding to the empty array-- store hand values to hands array  return hands  players = deal(5, 4) #hand contains 5 cards, 4 players  import pprint; pprint.pprint(players)  **OUTPUT:**  ['H7', 'C9', 'C10', 'H4', 'H5', 'D2', 'S10', 'D4', 'CQ', 'C2', 'SJ', 'D6', 'C8', 'DJ', 'SK', 'HA', 'S7', 'D8', 'D9', 'DA', 'C3', 'SQ', 'S4', 'D10', 'HQ', 'D5', 'CK', 'H8', 'S2', 'C7', 'H2', 'CA', 'C4', 'D3', 'HJ', 'H10', 'D7', 'SA', 'S8', 'H3', 'HK', 'CJ', 'C6', 'C5', 'S9', 'DQ', 'S3', 'S6', 'DK', 'H9', 'S5', 'H6']  52  ['H7', 'C9', 'C10', 'H4', 'H5']  ['D2', 'S10', 'D4', 'CQ', 'C2', 'SJ', 'D6', 'C8', 'DJ', 'SK', 'HA', 'S7', 'D8', 'D9', 'DA', 'C3', 'SQ', 'S4', 'D10', 'HQ', 'D5', 'CK', 'H8', 'S2', 'C7', 'H2', 'CA', 'C4', 'D3', 'HJ', 'H10', 'D7', 'SA', 'S8', 'H3', 'HK', 'CJ', 'C6', 'C5', 'S9', 'DQ', 'S3', 'S6', 'DK', 'H9', 'S5', 'H6']  ['D2', 'S10', 'D4', 'CQ', 'C2']  ['SJ', 'D6', 'C8', 'DJ', 'SK', 'HA', 'S7', 'D8', 'D9', 'DA', 'C3', 'SQ', 'S4', 'D10', 'HQ', 'D5', 'CK', 'H8', 'S2', 'C7', 'H2', 'CA', 'C4', 'D3', 'HJ', 'H10', 'D7', 'SA', 'S8', 'H3', 'HK', 'CJ', 'C6', 'C5', 'S9', 'DQ', 'S3', 'S6', 'DK', 'H9', 'S5', 'H6']  ['SJ', 'D6', 'C8', 'DJ', 'SK']  ['HA', 'S7', 'D8', 'D9', 'DA', 'C3', 'SQ', 'S4', 'D10', 'HQ', 'D5', 'CK', 'H8', 'S2', 'C7', 'H2', 'CA', 'C4', 'D3', 'HJ', 'H10', 'D7', 'SA', 'S8', 'H3', 'HK', 'CJ', 'C6', 'C5', 'S9', 'DQ', 'S3', 'S6', 'DK', 'H9', 'S5', 'H6']  ['HA', 'S7', 'D8', 'D9', 'DA']  ['C3', 'SQ', 'S4', 'D10', 'HQ', 'D5', 'CK', 'H8', 'S2', 'C7', 'H2', 'CA', 'C4', 'D3', 'HJ', 'H10', 'D7', 'SA', 'S8', 'H3', 'HK', 'CJ', 'C6', 'C5', 'S9', 'DQ', 'S3', 'S6', 'DK', 'H9', 'S5', 'H6']  [['H7', 'C9', 'C10', 'H4', 'H5'],  ['D2', 'S10', 'D4', 'CQ', 'C2'],  ['SJ', 'D6', 'C8', 'DJ', 'SK'],  ['HA', 'S7', 'D8', 'D9', 'DA']]  **a) Drawing cards from a deck; analyze results (1)**  **Analyze the no of pairs or n-of-a-kind in a hand**:  def same\_rank(hand, n\_of\_a\_kind):  ranks = [card[1:] for card in hand]  counter = 0  already\_counted = []  for rank in ranks:  if rank not in already\_counted and \  ranks.count(rank) == n\_of\_a\_kind:  counter += 1  already\_counted.append(rank)  return counter  #same\_rank('hand,C9')  **Drawing cards from a deck; analyze results**  **Analyze the no of combinations of the same suit:**  def same\_suit(hand):  suits = [card[0] for card in hand]  counter = {} # counter[suit] = how many cards of suit  for suit in suits:  count = suits.count(suit)  if count > 1:  counter[suit] = count  return counter  for hand in players:  print ( """\The hand %s has %d pairs, %s 3-of-a-kind and  %s cards of the same suit.""" % \  (', '.join(hand), same\_rank(hand, 2),  same\_rank(hand, 3),  '+'.join([str(s) for s in same\_suit(hand).values()])))  **OUTPUTS**  \The hand DQ, H7, SJ, HA, SK has 0 pairs, 0 3-of-a-kind and  2+2 cards of the same suit.  \The hand H9, S7, D8, S5, S2 has 0 pairs, 0 3-of-a-kind and  3 cards of the same suit.  \The hand S6, S10, DA, CJ, H4 has 0 pairs, 0 3-of-a-kind and  2 cards of the same suit.  \The hand H6, C7, H8, H5, D4 has 0 pairs, 0 3-of-a-kind and  3 cards of the same suit. |

**10.4 PRE-LAB**

|  |
| --- |
| 1. Write the correct function from the following list to get the random integer between 99 to 200, which is divisible by 3 ?  **10.5 POST LAB**  1. To generate a random float number between 20.5 to 50.5, which function of a random module I need to use ?  2. Summarize your results in the introductory sentence. Relate your results to your objective.  **10.6 Result** |