p. 100 True/False questions:
2) False
   list.index(x)
   Return the index in the list of the first item whose value is x. It is an error if there is no such item.

3) False
   random order is done in shuffle

6) False
   Python's list is heterogeneous

p. 102 Short-Answer Questions:

1)
   a) d = Deck()  # create an object of type Deck
      print(d)       # display all the cards
      in case if method print is undefined, then we can do
      d = Deck()  # create an object of type Deck
      for card in d.cards:
         print(card)
      although this is not good – we are accessing to the “private” attribute of Deck object.

   b) d = Deck()  # create an object of type Deck
      d.shuffle()   # shuffle the cards in the deck
      for i in range(13):   # deal and display 13 cards
         print(d.deal())

   c) d = Deck()  # create an object of type Deck
      d.shuffle()   # shuffle the cards in the deck
      h = Hand(“North”)   # create an empty bridge hand, with label North
      for i in range(13):   # deal 13 cards to the hand
         card=d.deal()
         h.add(card)
      h.sort()   # sort them in bridge order
      h.dump()  # display all the cards in hand North

   d) d = Deck()  # create an object of type Deck
      d.shuffle()   # shuffle the cards in the deck
      hn = Hand(“North”)   # create an empty bridge hand, with label North
      hs = Hand(“South”)   # create an empty bridge hand, with label South
      he = Hand(“East”)   # create an empty bridge hand, with label East
      hw = Hand(“West”)   # create an empty bridge hand, with label West
      for i in range(13):   # deal 13 cards to each hand
         hn.add(d.deal()) # deal one card to bridge hand, with label North
         hs.add(d.deal()) # deal one card to bridge hand, with label South
he.add(d.deal()) # deal one card to bridge hand, with label East
hw.add(d.deal()) # deal one card to bridge hand, with label West

hn.sort()  # sort them in bridge order, although we were not asked for it
hs.sort()  # sort them in bridge order, although we were not asked for it
he.sort()  # sort them in bridge order, although we were not asked for it
hw.sort()  # sort them in bridge order, although we were not asked for it

hn.dump()  # display all the cards in hand North
hs.dump()  # display all the cards in hand South
he.dump()  # display all the cards in hand East
hw.dump()  # display all the cards in hand West

2) The algorithm using two lists has $\Theta(n^2)$ efficiency, and the algorithm doing shuffling in place is of $\Theta(n)$ efficiency.

Comments to the first algorithm: operation len() applied to a list with n elements has $\Theta(n)$ efficiency, and we have n iterations of the while loop.