```
"""
For this project, I first thought of using list of strings as card representation, because I thought that it would save me more time to write but then as i work more into this project
I realized that string isn't really the best choice. I thought of using a dictionary, but that would've been a hassle for me, so the best option I had was to go with tuples.
And I now believe tuples is a better option to go with for this project, it could clearly separate the rank and suits, it is also easy for me access the rank and suits at any time.

One part of the code that required significant refactoring was in hands.py. At first, I tried to write just four functions to handle Flush, Two Pair, One Pair, and High Card.
But this approach quickly became complicated, especially since Flush includes multiple variations like regular Flush, Straight Flush, and Royal Flush. So instead of forcing
those four functions, I wrote a bunch of different functions to identify the hands and evaluate them into these 4 groups.
"""

import random

ranks = ["2","3","4","5","6","7","8","9","10","J","Q","K","A"]
suits = ["Spades","Clubs","Hearts","Diamonds"]

def create():
    """
    Create a list of cards in tuple form and returns them.
    :return: a list of tuples that represent cards
```

```
20      """
21      deck = []
22      for rank in ranks:
23          for suit in suits:
24              deck.append((rank, suit))
25      return deck
26
27
28  def shuffle(deck):
29      """"
30      Shuffles the deck.
31      :param deck: a list of tuples that represent cards
32      """
33      random.shuffle(deck)
34
35  def deal(deck, n):
36      """
37      Deals n cards from the deck.
38      :param deck: a list of tuples that represent cards
39      :param n: the number of cards to deal
40      :return: the dealed cards
41      """
42      hands = []
43      for _ in range(n):
44          hands.append(deck.pop(0))
45      return hands
46
```

```python
#  Poker Hands
# • Flush (includes normal, royal, and straight flushes)
# • Two pair (includes two pair, four-of-a-kind, and full house)
# • Pair (includes pair and three-of-a-kind)
# • High card (includes high card and straight). Ace has the highest rank and Two has the lowest.

def isFlush(hands):
    """
    Check if a hand is a flush, all cards same suit.
    :param hands: list of 5 cards as tuples (rank, suit)
    :return: a boolean value
    """
    suits = []
    for card in hands:
        suits.append(card[1])
    removed = set(suits)
    return len(removed) == 1


def isStraight(hands):
    """
    Check if a hand is a straight, consecutive cards.
    :param hands: list of 5 cards as tuples (rank, suit)
    :return: a boolean value
    """
    rankOrder = {"2": 2, "3": 3, "4": 4, "5": 5, "6": 6, "7": 7, "8": 8, "9": 9, "10": 10, "J": 11, "Q": 12, "K": 13, "A": 14}
    ranks = []
    for card in hands:
        rank = card[0]
        ranks.append(rankOrder[rank])

```

```
32      ranks.sort()
33      for i in range(4):
34          if ranks[i] + 1 != ranks[i + 1]:
35              return False
36      return True
37
38
39  def rankCounts(hands):
40      """
41      Count how many cards of each rank are in the hand.
42      :param hands: list of 5 cards as tuples (rank, suit
    )
43      :return: dictionary with rank counts
44      {rank: count}
45      {key: value}
46      """
47      counts = {}
48      for card in hands:
49          rank = card[0]
50          if rank in counts:
51              counts[rank] += 1
52          else:
53              counts[rank] = 1
54      return counts
55
56
57  def isFourOfAKind(hands):
58      """
59      Check if a hand has four cards of the same rank.
60      :param hands: list of 5 cards as tuples (rank, suit
    )
61      :return: a boolean value
62      """
63      counts = rankCounts(hands)
64      if 4 in counts.values():
65          return True
66      return False
67
```

```
68
69 def isFullHouse(hands):
70     """
71     Check if a hand has three of one rank and two of
   another.
72     :param hands: list of 5 cards as tuples (rank,
   suit)
73     :return: a boolean value
74     """
75     counts = rankCounts(hands).values()
76     three = False
77     two = False
78
79     for count in counts:
80         if count == 3:
81             three = True
82         elif count == 2:
83             two = True
84
85     return three and two
86
87
88 def isThreeOfAKind(hands):
89     """
90     Check if a hand has exactly three cards of the
   same rank.
91     :param hands: list of 5 cards as tuples (rank,
   suit)
92     :return: a boolean value
93     """
94     counts = rankCounts(hands).values()
95     if 3 in counts and not isFullHouse(hands):
96         return True
97     return False
98
99
100 def hasPairs(hands, n):
101     """
```

```
102     Check if a hand has exactly n pairs.
103
104     precondition: n is a positive integer and n <= 2
105     :param hands: list of 5 cards as tuples (rank,
    suit)
106     :param n: number of pairs to check
107     :return: a boolean value
108     """
109     counts = rankCounts(hands)
110     pairCount = 0
111     for value in counts.values():
112         if value == 2:
113             pairCount += 1
114     return pairCount == n
115
116
117 def evaluate(hands):
118     """
119     Evaluate a hand and return its category.
120     Grouped into: flush, two pair, pair, or high card.
121     :param hands: list of 5 cards as tuples (rank,
    suit)
122     :return: string representing the hand category
123     """
124     flush = isFlush(hands)
125     straight = isStraight(hands)
126
127     if flush and straight:
128         return "flush"
129     elif flush:
130         return "flush"
131     elif isFourOfAKind(hands) or isFullHouse(hands) or
     hasPairs(hands, 2):
132         return "two pair"
133     elif isThreeOfAKind(hands) or hasPairs(hands, 1):
134         return "pair"
135     else:
136         return "high card"
```

137
138
139
140
141

```
"""
Poker Game
:author: James Lin
:note: I affirm that I have carried out the attached
academic endeavors with full academic honesty, in
accordance with the Union College Honor Code and the
course syllabus.
"""


import cards
import hands

def play_rounds():
    """
    Play poker simulation rounds and print out results
in a table.
    """
    print("# of hands  pairs  %    2 pairs  %
flushes  %   high card  %")
    for i in range(10000, 100001, 10000):
        pair = 0
        twoPair = 0
        flush = 0
        highCard = 0
        handsDealt = 0

        while handsDealt < i:
            deck = cards.create()
            cards.shuffle(deck)

            while handsDealt < i and len(deck) >= 5:
                hand = cards.deal(deck, 5)
                category = hands.evaluate(hand)
                if category == "pair":
                    pair += 1
                elif category == "two pair":
                    twoPair += 1
```

```
34                  elif category == "flush":
35                      flush += 1
36                  elif category == "high card":
37                      highCard += 1
38
39                  handsDealt += 1
40
41          total = handsDealt
42          pairPercent = (pair / total) * 100
43          twoPairPercent = (twoPair / total) * 100
44          flushPercent = (flush / total) * 100
45          highCardPercent = (highCard / total) * 100
46
47          print(f"{total:>7,}" f"  {pair:>5}  {
    pairPercent:05.2f}"f"  {twoPair:>5}  {twoPairPercent:05
    .2f}"f"  {flush:>5}  {flushPercent:05.2f}"f"  {highCard
    :>5}  {highCardPercent:05.2f}")
48
49  if __name__ == "__main__":
50      play_rounds()
51
```