```
"""
Deck of 52 cards created for shuffling and dealing.
"""

import random

import card_deck
import card_format as cf


def build_deck():
    """
    Create a deck of cards in 4 suits and 13 ranks.
    :return: Generated ranks and suits added to the
     empty deck list in the format from card_format
   file
    """

    ranks = ['Ace', '2', '3', '4', '5', '6', '7', '
   8', '9', '10', 'Jack', 'Queen', 'King']
    suits = ['clubs', 'diamonds', 'hearts', 'spades
   ']
    deck = []
    for rank in ranks:
        for suit in suits:
            match = cf.create(rank, suit)
            deck.append(match)

    return deck


def shuffle_deck(my_decks):
    """
    Shuffling deck of 52 cards copying the deck
   cards
    :param my_decks: To be shuffled deck
    :return A shuffled deck from a randomly
   generated pile of cards
    """
    return card_deck.shuffle_deck()

```

```
36
37 def deal_cards(shuffled_card_deck):
38     """
39     Dealt deck of cards after deck is shuffled.
40     :return: Dealed deck of cards
41     """
42
43     deck_shuffling = shuffle_deck()
44     dealed_cards = []
45     for deal in range(shuffled_card_deck):
46         taken_card = random.choice(shuffled_deck)
47         dealed_cards.append(taken_card)
48         shuffled_deck.remove(taken_card)
49     return dealed_cards
50
51
52 if __name__ == "__main__":
53     the_cards = deal_cards(2)
54     the_cards
55     shuffled_deck = shuffle_deck()
56
```

```
"""
Output table of percent of hand rankings with
pairs, 2 pairs, flushes, and high card generated
from
range of 10,000 - 100,000 hands
"""

import card_deck as cd


def deal(deck, param):
    pass


def hand_rank(hand):
    pass


def play_rounds(n=100000, hands=None, rank_names=
None):
    """
    Generate the number of hands from 10k to 100k
with percentage of hands in each incremental 10k
generated hands.
    :return: Percentage of type generated in total
number of hands and specific number, compared hands
.
    """
    rounds = {i: 0 for i in range(9)}
    for i in range(n):
        deck = cd.shuffled_deck()
        deal(deck, 5)
        rounds[hand_rank(hands)] += 1

    for j in range(9):
        print(f'{rank_names[j]}; {rounds[j] ({100
 * rounds[j] / n}%)')


table = {'# of hands': 10,000:
        'pairs': xx,
```

```
35          '%': xx.xx}
36 for number, hands in table.items():
37 print(f'{number:10} ==> {hands:10d}')
```

```
 1 """
 2 Formatting deck of cards.
 3
 4 I affirm that I have carried out the attached
   academic endeavors with full academic honesty, in
 5 accordance with the Union College Honor Code and
   the course syllabus.
 6
 7 Citing help from Chris. Collaborated with Shane
   Ford. Referenced class coding samples; python3.com
   resources, and
 8 resources inside the Project File.
 9 """
10
11
12 def add_string(cards):
13     """
14     Format the deck of cards in rank of suit format
    in a tuple.
15     :param cards: deck of cards
16     :return: format of cards when it's returned
17     """
18     return "[%r of %s]" % (get_r(cards), get_s(
   cards))
19
20
21 def create(r, s):
22     """
23     Formatting ranks and suits as a tuple.
24     :param r: Rank
25     :param s: Suit
26     :return: value of rank and suits
27     """
28     return r, s
29
30
31 def get_r(cards):
32     """
33     The value for ranks in deck of cards.
34     :param cards: Deck of cards.
35     :return: The left-hand side value of cards as a
```

```
35  tuple returning suits.
36      """
37      return cards[0]
38
39
40  def get_s(cards):
41      """
42      The value for suits in deck of cards.
43      :param cards: Deck of cards.
44      :return: The right-hand side value of cards as
    a tuple returning suits.
45      """
46      return cards[1]
47
```

```
"""
Generate poker hand rankings pairs, 2 pairs, flushes, and high card.
:param What is composed of in a hand dealing, the required to generate each of those in a hand dealing.
:return Flushes, two pair, pair, and high card.
"""

from random import randint
import card_deck as deck

ranks = ['Ace', '2', '3', '4', '5', '6', '7', '8', '9', '10', 'Jack', 'Queen', 'King']
suits = ['clubs', 'diamonds', 'hearts', 'spades']


def hand_dealings():
    """
    Create 5 hands from dealed deck of cards in the card_deck file.
    :return: The possible hands out of card dealings only with flush, two pair, pair, high card.
    """
    hands = {'Flush': 0,
             'Two pair': 0,
             'Pair': 0,
             'High card': 0}
    hand_dealings.shuffle_deck(hands)
    return hands


def is_flush(my_hands):
    """
    Defining hand dealings that are flush.
    :param my_hands: Representing my_hands as the possible hand dealings that would be a flush card.
    :return:
    """
    return all([suits(my_hands) == suits(my_hands[0
```

```
33 ]) for card in my_hands[1:]])
34
35
36 def is_two_pair(my_hands):
37     """
38     Composition of a two pair card.
39     :return: Two pair card generated from function.
40     """
41     suits2 = [h[1] for h in my_hands]
42     rank_deck = ['Ace', '2', '3', '4', '5', '6', '7
   ', '8', '9', '10', 'Jack', 'Queen', 'King']
43
44     for i in rank_deck:
45         my_hands[i] += 1
46     if sorted(my_hands.values()) == [1, 2, 2]:
47         return True
48     else:
49         return False
50
51
52 def is_pair(my_hands):
53     """
54     Composition of a pair card.
55     :return: Pair card generated from function.
56     """
57     suits2 = [h[1] for h in my_hands]
58     rank_deck = ['Ace', '2', '3', '4', '5', '6', '7
   ', '8', '9', '10', 'Jack', 'Queen', 'King']
59
60     for i in rank_deck:
61         my_hands[i] += 1
62         for j in rank_deck:
63             my_hands[j] += 1
64     if sorted(my_hands.values()) == [1, 2, 2, 2, 3
   , 3]:
65         return True
66     else:
67         return False
68
69
70 def high_card(my_hands):
```

```
    """
    Composition of a high card.
    :return: High card generated from function.
    """
    suits2 = [h[1] for h in my_hands]
# I have no clue.

if __name__ == "__main__":
    [is_flush(hand) for hand in my_hands]

```