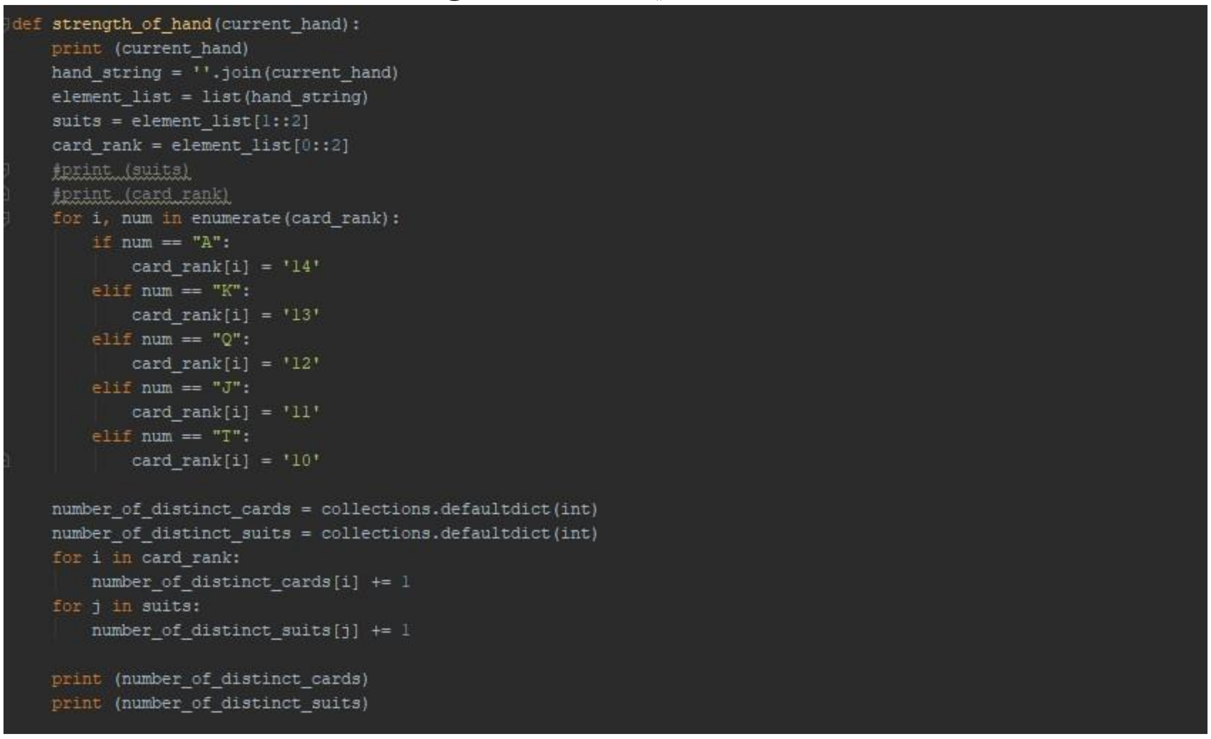
Poker Project Report

Name: Shajahan Abdul Rehman Basith **Name:** Habeebulla Shaik

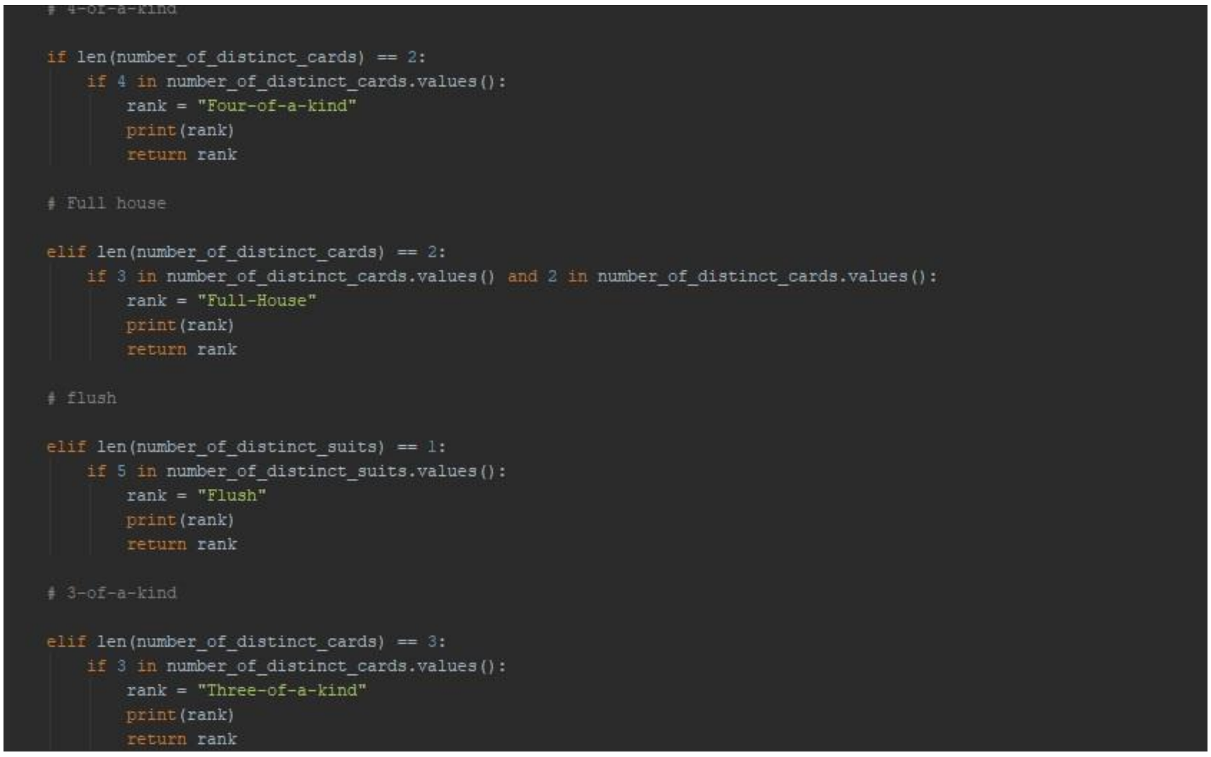
Description of the project:

- We have implemented a reflex agent where the agent calls an action based on the strength of the agent's hand.
- We added a function called strength_of_hand() which analyses the current hand of the player and returns the type of hand the player has.
- We also added another function called action which receives the return value from strength_of_hand() function and returns a number to the queryOpenAction() and queryCallRaiseAction() functions.
- Based on the return value from the action() function the agent makes decision to open a bet or call or raise the bet or to fold.
- We have made a change in the queryCardsToThrow() function such that, if the agent hand has a four-of-a-kind, full-house, flush or straight it will not throw any card from the hand, else, it we randomly select a card and throw.

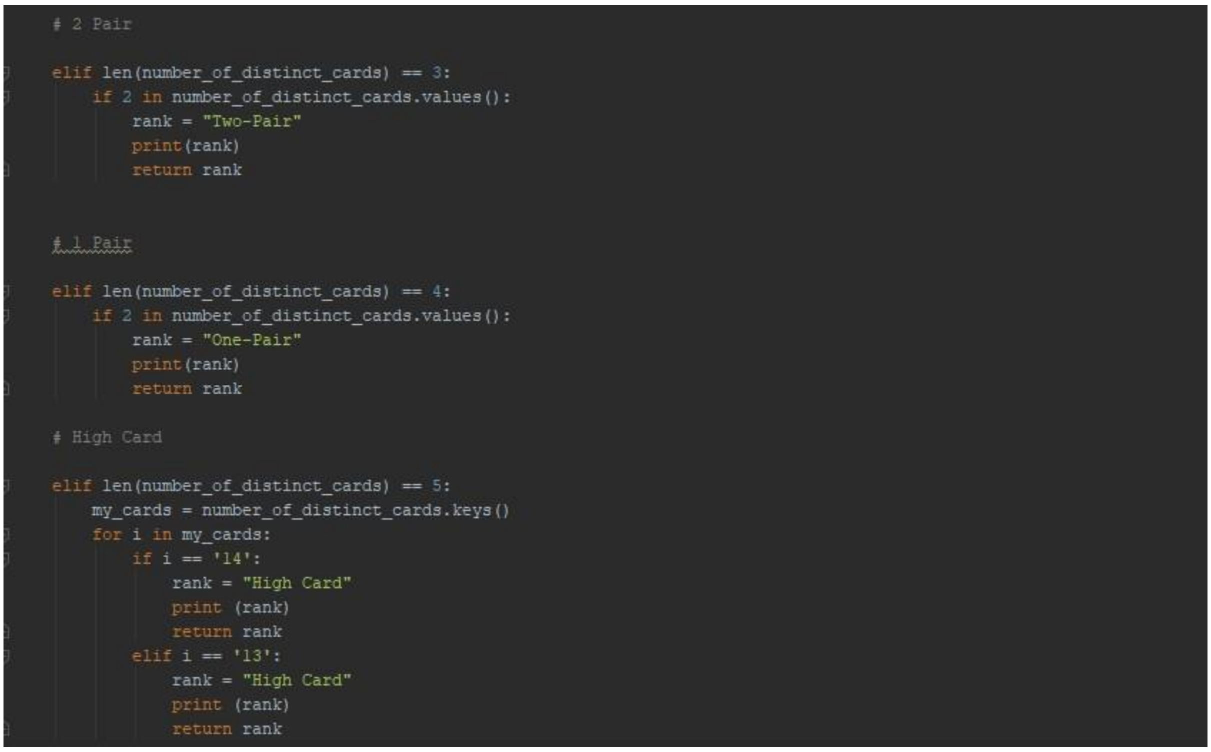
strength_of_hand() function:



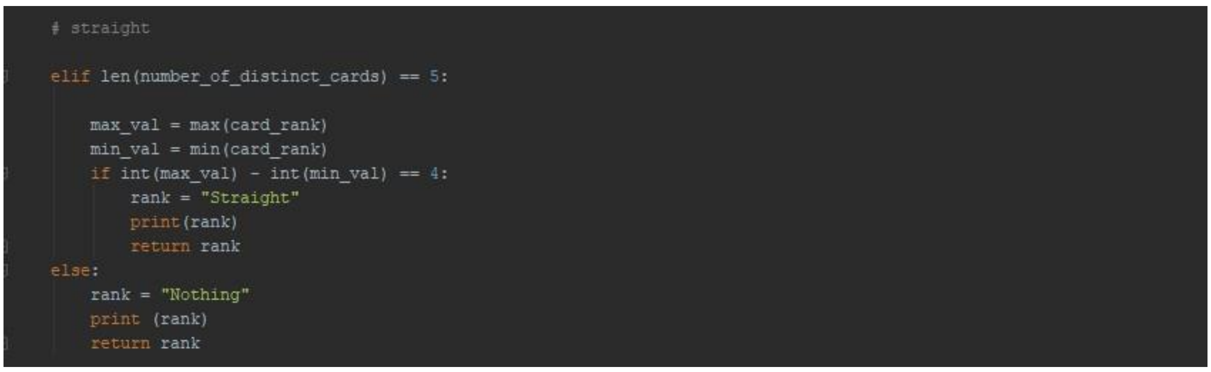
```python
def strength_of_hand(current_hand):
    print (current_hand)
    hand_string = ''.join(current_hand)
    element_list = list(hand_string)
    suits = element_list[1::2]
    card_rank = element_list[0::2]
    #print (suits)
    #print (card_rank)
    for i, num in enumerate(card_rank):
        if num == "A":
            card_rank[i] = '14'
        elif num == "K":
            card_rank[i] = '13'
        elif num == "Q":
            card_rank[i] = '12'
        elif num == "J":
            card_rank[i] = '11'
        elif num == "T":
            card_rank[i] = '10'

    number_of_distinct_cards = collections.defaultdict(int)
    number_of_distinct_suits = collections.defaultdict(int)
    for i in card_rank:
        number_of_distinct_cards[i] += 1
    for j in suits:
        number_of_distinct_suits[j] += 1

    print (number_of_distinct_cards)
    print (number_of_distinct_suits)
```



```python
# 4-of-a-kind

if len(number_of_distinct_cards) == 2:
    if 4 in number_of_distinct_cards.values():
        rank = "Four-of-a-kind"
        print(rank)
        return rank

# Full house

elif len(number_of_distinct_cards) == 2:
    if 3 in number_of_distinct_cards.values() and 2 in number_of_distinct_cards.values():
        rank = "Full-House"
        print(rank)
        return rank

# flush

elif len(number_of_distinct_suits) == 1:
    if 5 in number_of_distinct_suits.values():
        rank = "Flush"
        print(rank)
        return rank

# 3-of-a-kind

elif len(number_of_distinct_cards) == 3:
    if 3 in number_of_distinct_cards.values():
        rank = "Three-of-a-kind"
        print(rank)
        return rank
```

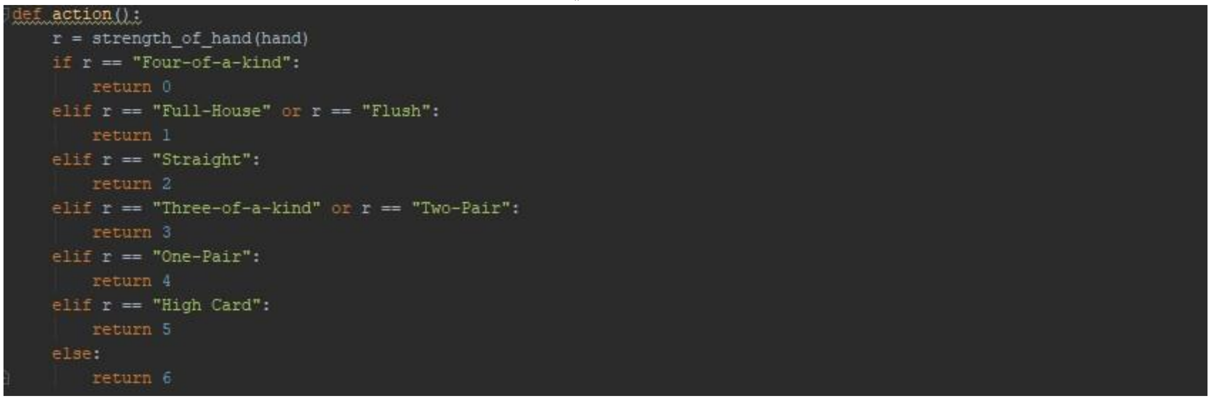


```python
# 2 Pair

elif len(number_of_distinct_cards) == 3:
    if 2 in number_of_distinct_cards.values():
        rank = "Two-Pair"
        print(rank)
        return rank


# 1 Pair

elif len(number_of_distinct_cards) == 4:
    if 2 in number_of_distinct_cards.values():
        rank = "One-Pair"
        print(rank)
        return rank

# High Card

elif len(number_of_distinct_cards) == 5:
    my_cards = number_of_distinct_cards.keys()
    for i in my_cards:
        if i == '14':
            rank = "High Card"
            print (rank)
            return rank
        elif i == '13':
            rank = "High Card"
            print (rank)
            return rank
```



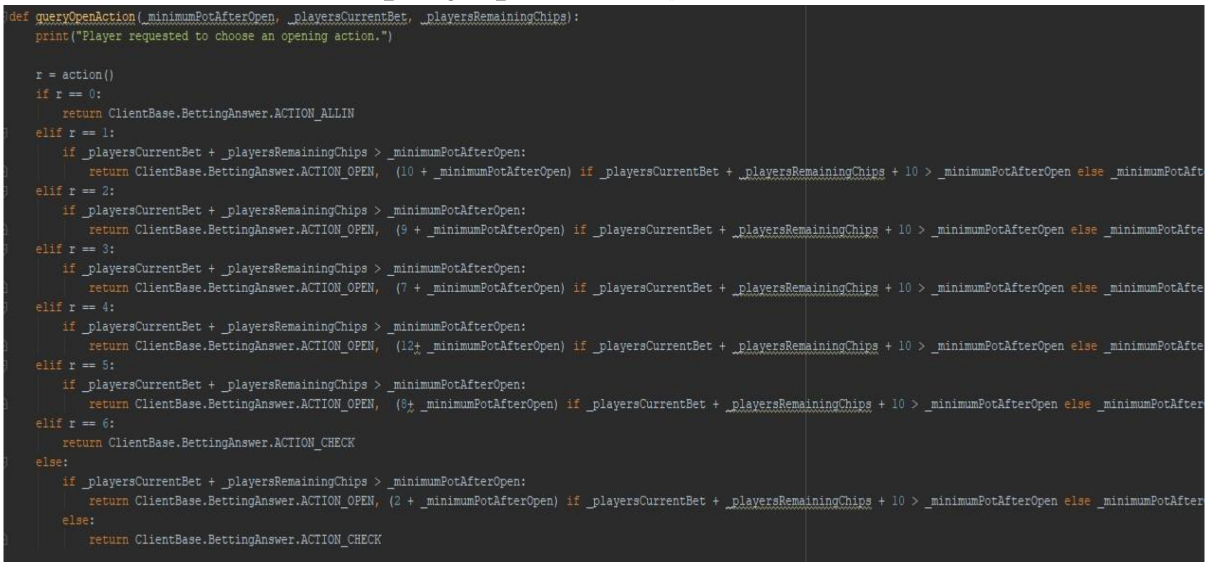
```python
# straight

elif len(number_of_distinct_cards) == 5:

    max_val = max(card_rank)
    min_val = min(card_rank)
    if int(max_val) - int(min_val) == 4:
        rank = "Straight"
        print(rank)
        return rank
else:
    rank = "Nothing"
    print (rank)
    return rank
```

action() function:



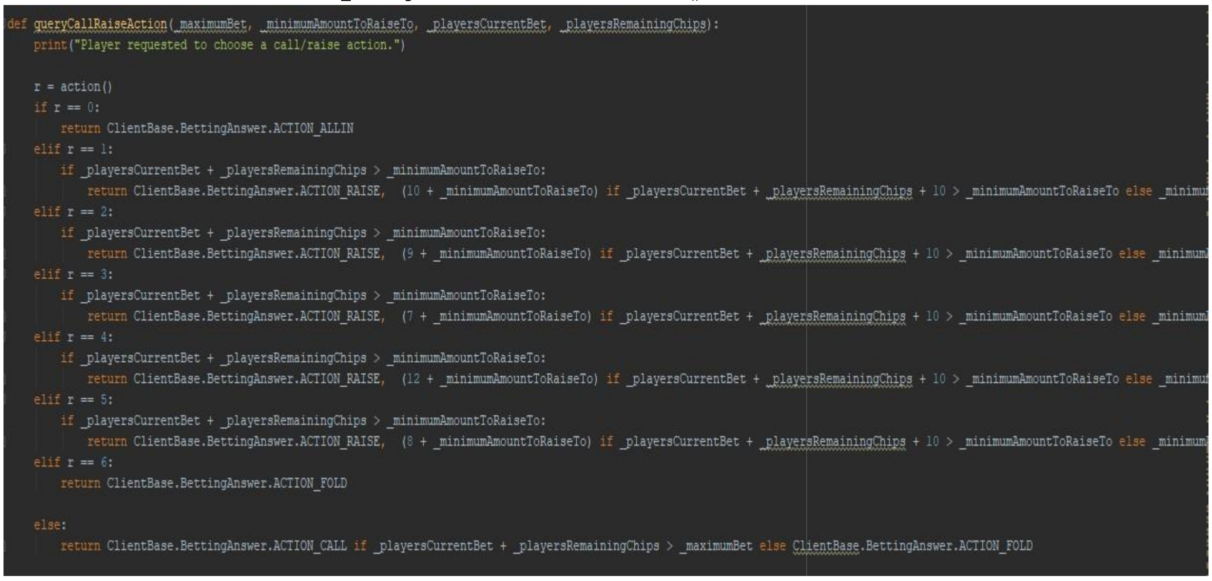
```python
def action():
    r = strength_of_hand(hand)
    if r == "Four-of-a-kind":
        return 0
    elif r == "Full-House" or r == "Flush":
        return 1
    elif r == "Straight":
        return 2
    elif r == "Three-of-a-kind" or r == "Two-Pair":
        return 3
    elif r == "One-Pair":
        return 4
    elif r == "High Card":
        return 5
    else:
        return 6
```

queryOpenAction() function:



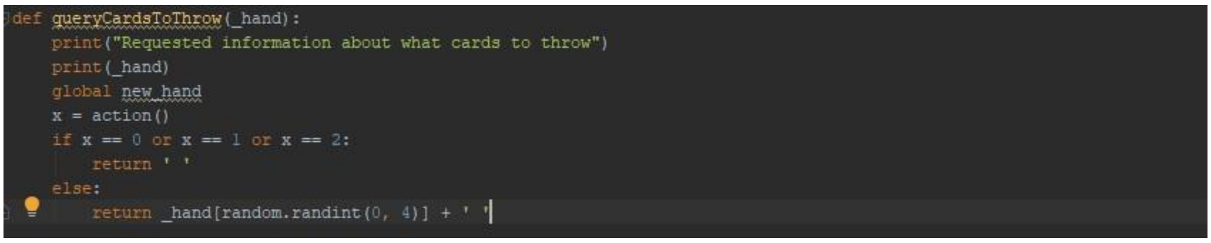
```python
def queryOpenAction(_minimumPotAfterOpen, _playersCurrentBet, _playersRemainingChips):
    print("Player requested to choose an opening action.")

    r = action()
    if r == 0:
        return ClientBase.BettingAnswer.ACTION_ALLIN
    elif r == 1:
        if _playersCurrentBet + _playersRemainingChips > _minimumPotAfterOpen:
            return ClientBase.BettingAnswer.ACTION_OPEN,  (10 + _minimumPotAfterOpen) if _playersCurrentBet + _playersRemainingChips + 10 > _minimumPotAfterOpen else _minimumPotAft
    elif r == 2:
        if _playersCurrentBet + _playersRemainingChips > _minimumPotAfterOpen:
            return ClientBase.BettingAnswer.ACTION_OPEN,  (9 + _minimumPotAfterOpen) if _playersCurrentBet + _playersRemainingChips + 10 > _minimumPotAfterOpen else _minimumPotAfte
    elif r == 3:
        if _playersCurrentBet + _playersRemainingChips > _minimumPotAfterOpen:
            return ClientBase.BettingAnswer.ACTION_OPEN,  (7 + _minimumPotAfterOpen) if _playersCurrentBet + _playersRemainingChips + 10 > _minimumPotAfterOpen else _minimumPotAfte
    elif r == 4:
        if _playersCurrentBet + _playersRemainingChips > _minimumPotAfterOpen:
            return ClientBase.BettingAnswer.ACTION_OPEN,  (12+ _minimumPotAfterOpen) if _playersCurrentBet + _playersRemainingChips + 10 > _minimumPotAfterOpen else _minimumPotAfte
    elif r == 5:
        if _playersCurrentBet + _playersRemainingChips > _minimumPotAfterOpen:
            return ClientBase.BettingAnswer.ACTION_OPEN,  (8+ _minimumPotAfterOpen) if _playersCurrentBet + _playersRemainingChips + 10 > _minimumPotAfterOpen else _minimumPotAfter
    elif r == 6:
        return ClientBase.BettingAnswer.ACTION_CHECK
    else:
        if _playersCurrentBet + _playersRemainingChips > _minimumPotAfterOpen:
            return ClientBase.BettingAnswer.ACTION_OPEN, (2 + _minimumPotAfterOpen) if _playersCurrentBet + _playersRemainingChips + 10 > _minimumPotAfterOpen else _minimumPotAfter
        else:
            return ClientBase.BettingAnswer.ACTION_CHECK
```

queryCallRaiseAction() function:



```python
def queryCallRaiseAction(_maximumBet, _minimumAmountToRaiseTo, _playersCurrentBet, _playersRemainingChips):
    print("Player requested to choose a call/raise action.")

    r = action()
    if r == 0:
        return ClientBase.BettingAnswer.ACTION_ALLIN
    elif r == 1:
        if _playersCurrentBet + _playersRemainingChips > _minimumAmountToRaiseTo:
            return ClientBase.BettingAnswer.ACTION_RAISE,  (10 + _minimumAmountToRaiseTo) if _playersCurrentBet + _playersRemainingChips + 10 > _minimumAmountToRaiseTo else _minimu
    elif r == 2:
        if _playersCurrentBet + _playersRemainingChips > _minimumAmountToRaiseTo:
            return ClientBase.BettingAnswer.ACTION_RAISE,  (9 + _minimumAmountToRaiseTo) if _playersCurrentBet + _playersRemainingChips + 10 > _minimumAmountToRaiseTo else _minimum
    elif r == 3:
        if _playersCurrentBet + _playersRemainingChips > _minimumAmountToRaiseTo:
            return ClientBase.BettingAnswer.ACTION_RAISE,  (7 + _minimumAmountToRaiseTo) if _playersCurrentBet + _playersRemainingChips + 10 > _minimumAmountToRaiseTo else _minimum
    elif r == 4:
        if _playersCurrentBet + _playersRemainingChips > _minimumAmountToRaiseTo:
            return ClientBase.BettingAnswer.ACTION_RAISE,  (12 + _minimumAmountToRaiseTo) if _playersCurrentBet + _playersRemainingChips + 10 > _minimumAmountToRaiseTo else _minimu
    elif r == 5:
        if _playersCurrentBet + _playersRemainingChips > _minimumAmountToRaiseTo:
            return ClientBase.BettingAnswer.ACTION_RAISE,  (8 + _minimumAmountToRaiseTo) if _playersCurrentBet + _playersRemainingChips + 10 > _minimumAmountToRaiseTo else _minimum
    elif r == 6:
        return ClientBase.BettingAnswer.ACTION_FOLD

    else:
        return ClientBase.BettingAnswer.ACTION_CALL if _playersCurrentBet + _playersRemainingChips > _maximumBet else ClientBase.BettingAnswer.ACTION_FOLD
```

queryCardsToThrow() function:



```python
def queryCardsToThrow(_hand):
    print("Requested information about what cards to throw")
    print(_hand)
    global new_hand
    x = action()
    if x == 0 or x == 1 or x == 2:
        return ' '
    else:
        return _hand[random.randint(0, 4)] + ' '
```