









```
[1] \ - [2] \ - [3] \ - [4] \ - [5] | / / - - -
```

Welcome to PyPoker a TexasHoldem style poker simulation

You can choose to play a random hand, or choose your own cards

[?] Choose how you would like to play:

- > o Random Hand
- o Choose Your Own

\*\* Hole Cards \*\*

Your Hand

['D10', 'H11']

\*\* Flop \*\*

[?] Choose how you would like to play:

- > o Random Hand
- o Choose Your Own

Your Hand

['D10', 'H11', 'D6', 'C14', 'H3']

My current value is 38.14756671899529 and the average expected value is 50.11777788038292

enter number of players: 2

enter pot value: 100

enter value of your bet: 10

You should bet as long as it is less than 25.117916596673993 \$

The expected value betting 10.0 is 15.117916596673993 \$

\*\* Turn \*\*

[?] Choose how you would like to play:

- > o Random Hand
- o Choose Your Own

Your Hand

Turn Card ['S7']

['D10', 'H11', 'D6', 'C14', 'H3', 'S7']

My current value is 38.42229199372056 and the average expected value is 50.11777788038292

enter number of players: 2

enter pot value: 100

enter value of your bet: 10

You should bet as long as it is less than 25.117916596673993 \$

The expected value betting 10.0 is 15.117916596673993 \$

\*\* River \*\*

[?] Choose how you would like to play:

- > o Random Hand
- o Choose Your Own

Your Hand

Turn Card ['S7']

['D10', 'H11', 'D6', 'C14', 'H3', 'S7']

My final value is 90.85280265952534

enter number of players: 2

enter pot value: 10

enter value of your bet: 100

You should bet as long as it is less than 8.254231751090654 \$

The expected value betting 100.0 is -91.74576824890934 \$

you win, you beat the bot

Your score: 90.85280265952534

Bot score: 46.664050235478804

```
1 from scores import *
2 import inquirer
3 from functools import lru_cache as cache
4 import pandas as pd
5 from IPython import get_ipython; get_ipython().run_line_magic('matplotlib', 'inline')
6 from statistics import mean
7 from numba import jit, vectorize
8 import timeit
9 import numpy as np
10 from itertools import permutations
11 from numpy import vectorize
12 import random
13 import pyfiglet
14
15
16 # TODO, add pre-flop 2 card hand, show stats for pre-flop
17 # TODO, make more game like
18 # TODO, declare winner
19
20 def start_game():
21     result = pyfiglet.figlet_format("PyPoker")
22     print(result)
23     print('Welcome to PyPoker a TexasHoldem style poker simulation')
24     print('You can choose to play a random hand, or choose your own cards')
25
26 def flop_message():
27     result = "** Flop **"
28     print(result)
29
30 def turn_message():
31     result = "** Turn **"
32     print(result)
33
34 def river_message():
35     result = "** River **"
36     print(result)
37
38 def multi_choice():
39     while True:
40         questions = [inquirer.Checkbox(
41             'Choose',
42             message="Choose how you would like to play",
43             choices=['Random Hand','Choose Your Own'],
44         )]
45         answers = inquirer.prompt(questions)['Choose']
46         break
47     return answers
48
49 # refactor, this is called multiple times
50 def calc_flop(c4, c3, flop):
51     flopscore = expected_value(flop,combi)
52     current = df.loc[df['value'] >= flopscore[0]].index[0]/2598960*100
53     future = df.loc[df['value'] >= flopscore[1]].index[1]/2598960*100
54     print('My current value is %s and the average expected value is %s' % (current,future))
55     players = float(input('enter number of players: '))
56     pot = float(input('enter pot value: '))
57     price = float(input('enter value of your bet: '))
58     if current > future:
59         should_call(players,current,pot,price)
60     else:
61         should_call(players,future,pot,price)
62
63 def calc_winner(current, pre_flop):
64     pre_flop = []
65     for i in range(7):
66         rand_card = random.randint(0,53)
67         pre_flop.append(deck[rand_card])
68
69     flopscore = expected_value(pre_flop,combi)
70     bot_score = df.loc[df['value'] >= flopscore[0]].index[0]/2598960*100
71
72     if current < bot_score:
73         print("you lose, bot had a better hand")
74     else:
75         print("you win, you beat the bot")
76
77     print("Your score:", current)
78     print("Bot score:", bot_score)
79
80 # Numba has two compilation modes: nopython and object
81 # Produces much faster code, to prevent Numba from falling back, and instead raise an error pass nopython=true
82 @jit(nopython=True)
83 def common(a,b):
84     common=[]
85     l1 = [i for i in a]
86     l2 = [i for i in b]
87     common = len([i for i in l1 if i in l2])
```

```

    return common
86
85 @jit(nopython=True)
84 def numba_4():
83     results = []
82     len1 = c4.shape[0]
81     len2 = combi.shape[0]
80     for i in range(len1):
79         for j in range(len2):
78             if common(c4[i],combi[j]) == 4:
77                 results.append(combi[j])
76     return results
75
74 @jit(nopython=True)
73 def numba_3():
72     results = []
71     len1 = c3.shape[0]
70     len2 = combi.shape[0]
69     for i in range(len1):
68         for j in range(len2):
67             if common(c3[i],combi[j]) == 4:
66                 results.append(combi[j])
65     return results
64
63 @cache(maxsize=None)
62 def opti_3():
61     values= numba_3()
60     return [score_hand(i) for i in values]
59
58 @cache(maxsize=None)
57 def opti_4():
56     values= numba_4()
55     return [score_hand(i) for i in values]
54
53 def expected_value(hand,combi):
52     if len(hand) == 5:
51         maxi = score_hand(hand)
50         mean= np.mean(opti_3() + opti_4())
49     elif len(hand) == 6:
48         maxi = max([score_hand(i) for i in combinations(hand,5)])
47         mean = np.mean(opti_4())
46     elif len(hand) == 7:
45         maxi = max([score_hand(i) for i in combinations(hand,5)])
44         mean= maxi
43     values = [maxi,mean]
42     return values
41
40 def should_call(players,percentile,pot,price):
39     pwin = (percentile/100)**players
38     ev = pwin*pot
37     if ev <= 0:
36         print('you should fold')
35     if ev > 0:
34         print('You should bet as long as it is less than %s $' % ev)
33     print('The expected value betting %s is %s $' % (price,ev-price))
32     return pwin*100
31
30 # this cannot be in a method or it breaks the whole thing due to scope
29 flop = []
28 pre_flop = []
27 start_game()
26
25 """
24 Pre-Flop
23 """
22
21 # Multiple choice prompt
20 answers_pre = multi_choice()
19 if answers_pre != 'Choose Your Own':
18     for i in range(2):
17         rand_card = random.randint(0,53)
16         flop.append(deck[rand_card])
15         pre_flop.append(deck[rand_card])
14
13 # if the user chooses to pick their own cards
12 if len(flop) == 0:
11     # Prompt the user what they're about to enter
10     print('Enter your card in the format suit value, this is for your flop so it will be 5 cards')
9     print('For Example S9 would be the 9 of spades')
8     for i in range(0,2):
7         flop.append(str(input('enter card: ')))
6
5 def show_hand():
4 # show the hand
3     if len(flop) <= 5:
2         print("____")
1         print("      Your Hand")
0         print(flop)
176

```

```

87     print("_____")
88     print("      Your Hand")
85     print("River Card", river)
84     print(flop)
83     print("_____")
82
81 print("** Hole Cards **")
80 show_hand()
79
78 """
77   Flop
76 """
75
74 flop_message()
73
72 # Multiple choice prompt
71 answers = multi_choice()
70 if answers != 'Choose Your Own':
69   for i in range(3):
68     card_rand = random.randint(0,53)
67   try:
66     flop.append(deck[card_rand])
65   except:
64     flop = []
63   flop.append(deck[card_rand])
62
61 # if the user chooses to pick their own cards
60 if len(flop) == 0:
59   # Prompt the user what they're about to enter
58   print('Enter you card in the format suit value, this is for your flop so it will be 5 cards')
57   print('For Example S9 would be the 9 of spades')
56   for i in range(0,5):
55     flop.append(str(input('enter card: ')))
54
53 show_hand()
52
51 c4 = combinations(flop,4)
50 c3 = combinations(flop,3)
49 calc_flop(c4, c3, flop)
48
47 """
46 Turn
45 """
44
43 turn_message()
42
41 turn = []
40 # Multiple choice prompt
39 answers_turn = multi_choice()
38 if answers_turn != 'Choose Your Own':
37   for i in range(1):
36     rand_card = random.randint(0,53)
35   turn.append(deck[rand_card])
34
33 if len(turn) < 1:
32   turn.append(str(input('enter card: ')))
31 flop.append(turn[0])
30
29 # show the hand
28 show_hand()
27 calc_flop(c4, c3, flop)
26 """
25 River
24 """
23
22 river_message()
21
20 river = []
19 answers_turn = multi_choice()
18 if answers_turn != 'Choose Your Own':
17   for i in range(1):
16     j = random.randint(0,53)
15   river.append(deck[j])
14
13 if len(turn) < 1:
12   river.append(str(input('enter card: ')))
11 # show the hand
10 show_hand()
9
8 flop.append(river[0])
7 combiriver = expected_value(flop,combi)
6 current = df.loc[df['value'] >= combiriver[0]].index[0]/2598960*100
5 print('My final value is %s' % current)
4 players = float(input('enter number of players: '))
3 pot = float(input('enter pot value: '))
2 price = float(input('enter value of your bet: '))
1 should_call(players,current,pot,price)
2 calc_winner(current, pre_flop)
272 calc_winner(current, pre_flop)

```

