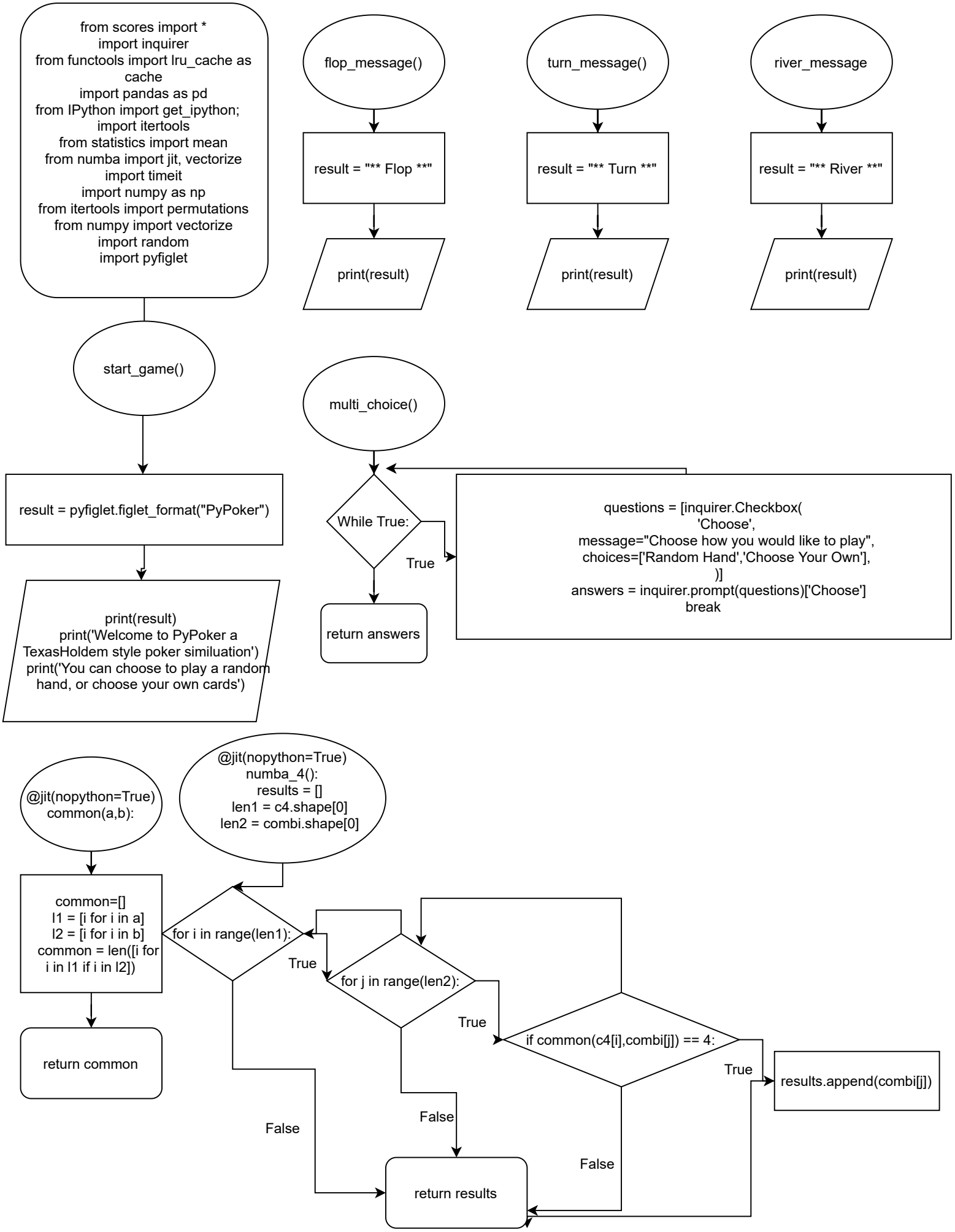
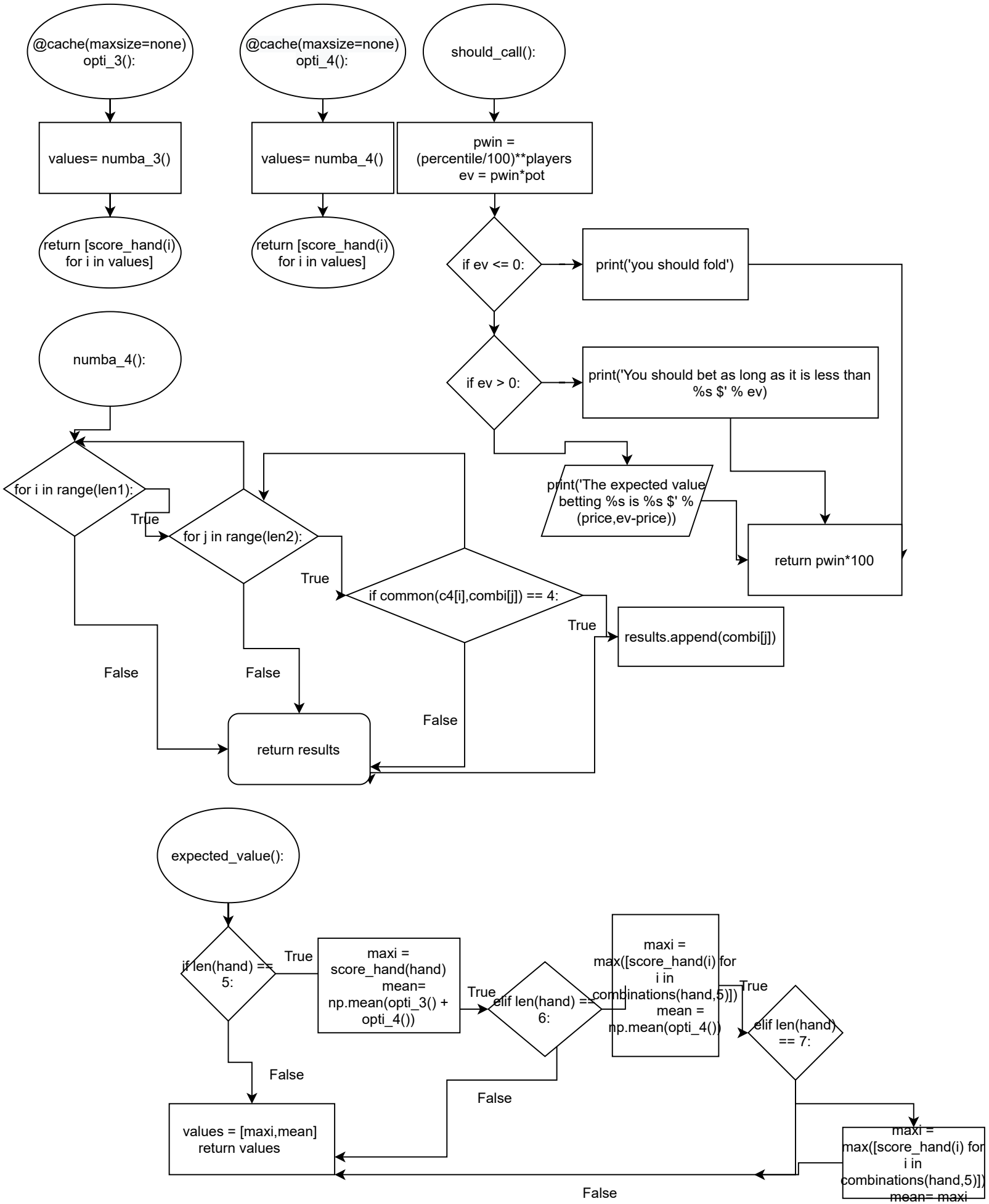
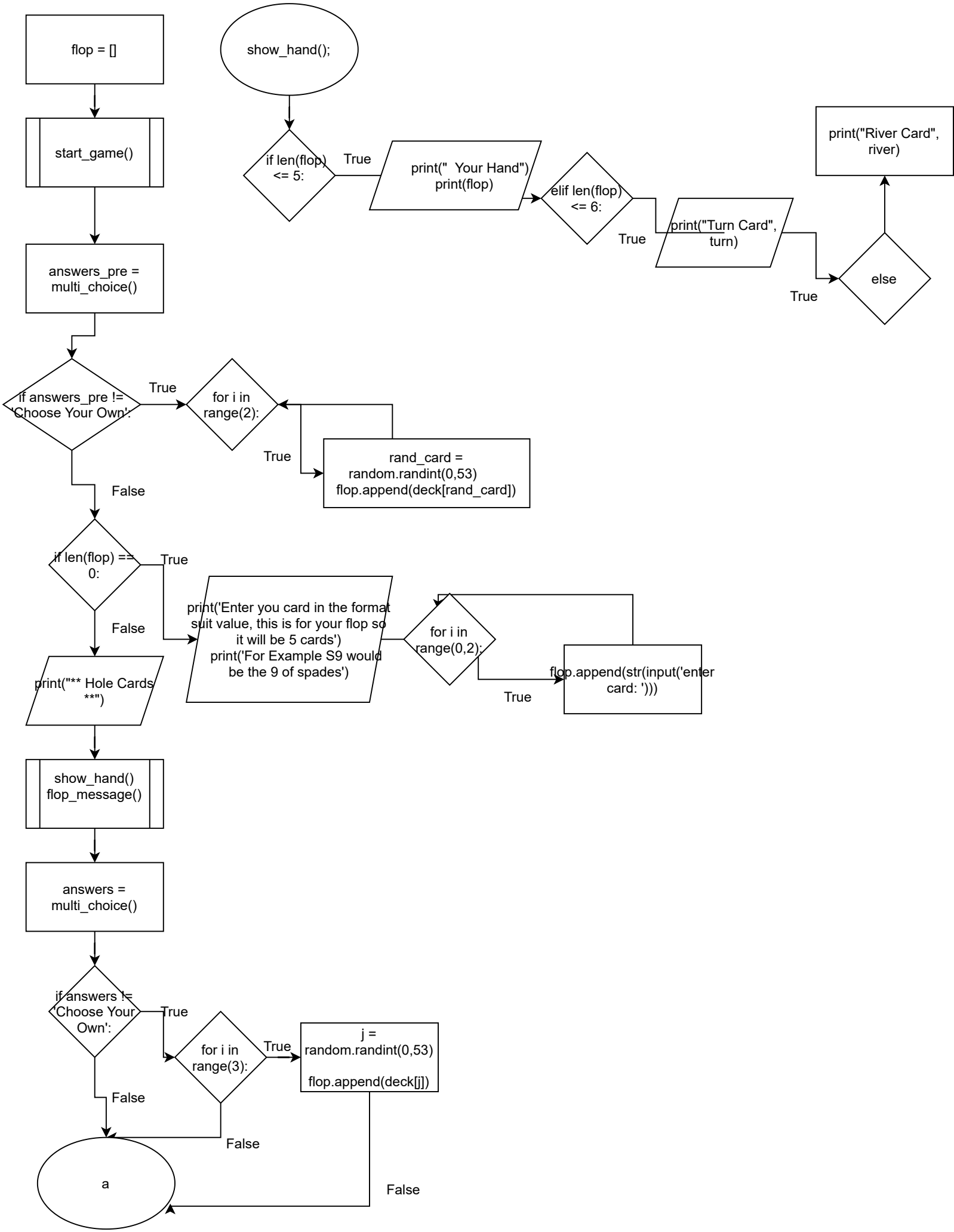
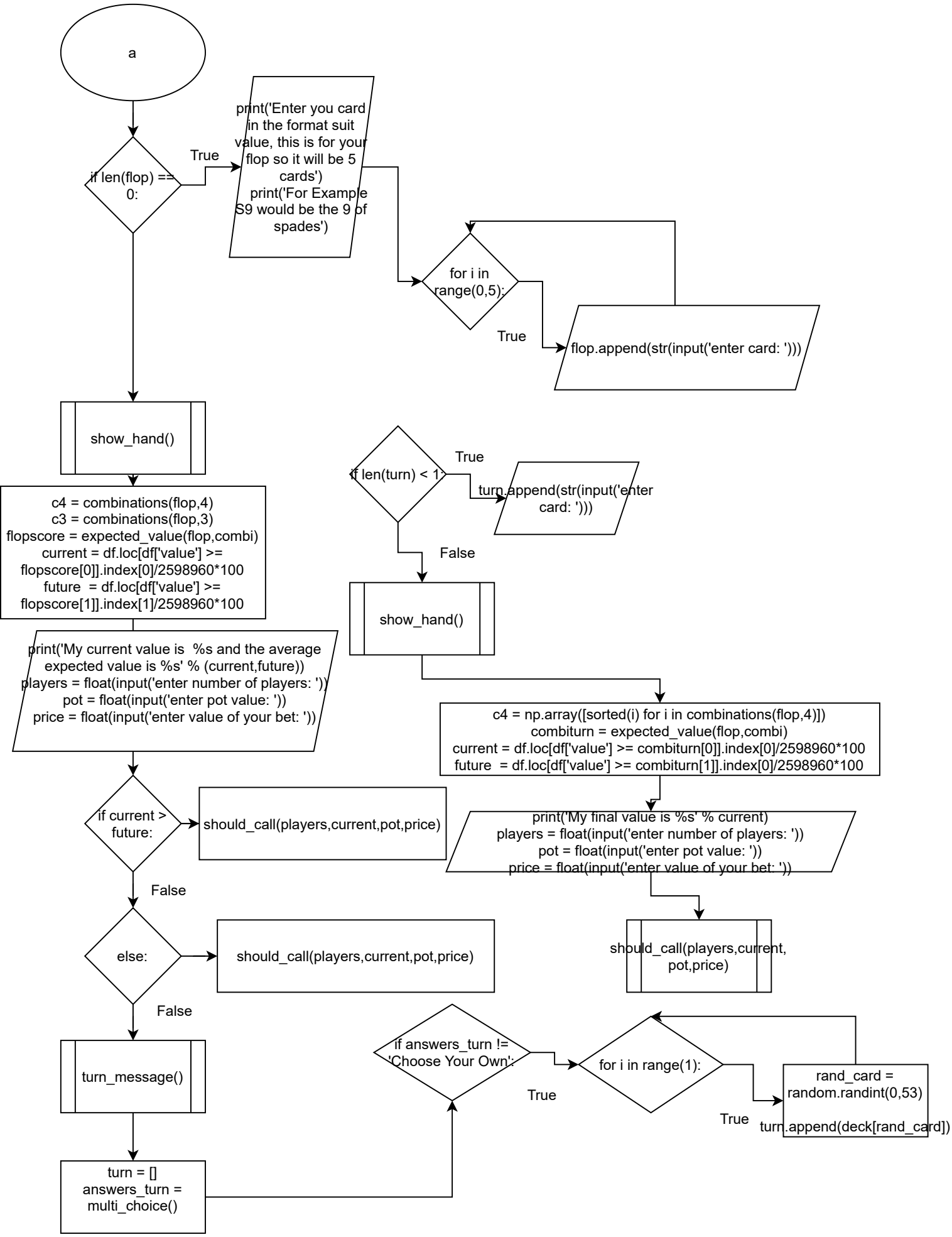

from scores import *
import inquirer
from functools import lru_cache as cache
import pandas as pd
from IPython import get_ipython;
import itertools
from statistics import mean
from numba import jit, vectorize
import timeit
import numpy as np
from itertools import permutations
from numpy import vectorize
import random
import pyfiglet
start_game()
result = pyfiglet.figlet_format("PyPoker")
print(result)
print('Welcome to PyPoker a TexasHoldem style poker similuation')
print('You can choose to play a random hand, or choose your own cards')
flop_message()
result = "** Flop **"
print(result)
turn_message()
result = "** Turn **"
print(result)
river_message
result = "** River **"
print(result)
multi_choice()
While True:
True
questions = [inquirer.Checkbox(
'Choose',
message="Choose how you would like to play",
choices=['Random Hand','Choose Your Own'],
)]
answers = inquirer.prompt(questions)['Choose']
break
return answers
@jit(nopython=True)
common(a,b):
common=[]
l1 = [i for i in a]
l2 = [i for i in b]
common = len([i for i in l1 if i in l2])
return common
@jit(nopython=True)
numba_4():
results = []
len1 = c4.shape[0]
len2 = combi.shape[0]
for i in range(len1):
True
for j in range(len2):
True
if common(c4[i],combi[j]) == 4:
True
results.append(combi[j])
False
False
False
return results


@cache(maxsize=none)
opti_3():
values= numba_3()
return [score_hand(i) for i in values]
@cache(maxsize=none)
opti_4():
values= numba_4()
return [score_hand(i) for i in values]
should_call():
pwin = (percentile/100)**players
ev = pwin*pot
if ev <= 0:
print('you should fold')
if ev > 0:
print('You should bet as long as it is less than %s $' % ev)
print('The expected value betting %s is %s $' % (price,ev-price))
return pwin*100
numba_4():
for i in range(len1):
True
for j in range(len2):
True
if common(c4[i],combi[j]) == 4:
True
results.append(combi[j])
False
False
False
return results
expected_value():
if len(hand) == 5:
True
maxi = score_hand(hand)
mean= np.mean(opti_3() + opti_4())
True
elif len(hand) == 6:
maxi = max([score_hand(i) for i in combinations(hand,5)])
mean = np.mean(opti_4())
True
elif len(hand) == 7:
False
False
values = [maxi,mean]
return values
maxi = max([score_hand(i) for i in combinations(hand,5)])
mean= maxi
False


flop = []
start_game()
answers_pre = multi_choice()
if answers_pre != 'Choose Your Own':
True
for i in range(2):
True
rand_card = random.randint(0,53)
flop.append(deck[rand_card])
False
if len(flop) == 0:
True
print('Enter you card in the format suit value, this is for your flop so it will be 5 cards')
print('For Example S9 would be the 9 of spades')
for i in range(0,2):
True
flop.append(str(input('enter card: ')))
False
print("** Hole Cards **")
show_hand()
flop_message()
answers = multi_choice()
if answers != 'Choose Your Own':
True
for i in range(3):
True
j = random.randint(0,53)
flop.append(deck[j])
False
False
False
a
show_hand();
if len(flop) <= 5:
True
print(" Your Hand")
print(flop)
elif len(flop) <= 6:
True
print("Turn Card", turn)
True
else
print("River Card", river)


a
if len(flop) == 0:
True
print('Enter you card in the format suit value, this is for your flop so it will be 5 cards')
print('For Example S9 would be the 9 of spades')
for i in range(0,5):
True
flop.append(str(input('enter card: ')))
show_hand()
c4 = combinations(flop,4)
c3 = combinations(flop,3)
flopscore = expected_value(flop,combi)
current = df.loc[df['value'] >= flopscore[0]].index[0]/2598960*100
future = df.loc[df['value'] >= flopscore[1]].index[1]/2598960*100
print('My current value is %s and the average expected value is %s' % (current,future))
players = float(input('enter number of players: '))
pot = float(input('enter pot value: '))
price = float(input('enter value of your bet: '))
if current > future:
should_call(players,current,pot,price)
False
else:
should_call(players,current,pot,price)
False
turn_message()
turn = []
answers_turn = multi_choice()
if answers_turn != 'Choose Your Own':
True
for i in range(1):
True
rand_card = random.randint(0,53)
turn.append(deck[rand_card])
if len(turn) < 1:
True
turn.append(str(input('enter card: ')))
False
show_hand()
c4 = np.array([sorted(i) for i in combinations(flop,4)])
combiturn = expected_value(flop,combi)
current = df.loc[df['value'] >= combiturn[0]].index[0]/2598960*100
future = df.loc[df['value'] >= combiturn[1]].index[0]/2598960*100
print('My final value is %s' % current)
players = float(input('enter number of players: '))
pot = float(input('enter pot value: '))
price = float(input('enter value of your bet: '))
should_call(players,current, pot,price)