Homework 2 (Intro to DS)

Colab Link:

Student Info: NCU MIS 109403019 鄒翔宇

Table of contents

Probability

- I have 2 kids. Given that one of them is a boy. What is the probability that both are boys? (10 points)

  Ans:

  $P(A)$ = one of two kids is a boy = $\frac{1}{2}$
  $P(B)$ = the other kid is a boy = $\frac{1}{2}$
  $P(A \cup B)$ = at least one kid is a boy = $\frac{3}{4}$
  $P(A \cap B)$ = two kids are boys = $P(A)*P(B) = \frac{1}{4}$

  $$\begin{aligned} P(A \cap B \mid A \cup B) &= P(A \cup B \mid A \cap B) * P(A \cap B) / P(A \cup B) \\ &= 1 * \tfrac{1}{4} / \tfrac{3}{4} \\ &= \tfrac{1}{3} \end{aligned}$$

  The probability is $\frac{1}{3}$.

- A fair six-sided die is rolled twice. What is the probability of getting 2 on the first roll and not getting 4 on the second roll? (10 points)

  Ans:

  $P(A)$ = not getting 4 on the second roll = $\frac{5}{6}$
  $P(B)$ = getting 2 on the first roll = $\frac{1}{6}$
  $P(A|B) = P(A \cap B) / P(B) = \frac{5}{6} * \frac{1}{6} / \frac{1}{6} = \frac{5}{6}$

  The probability is $\frac{5}{6}$.

Python programming

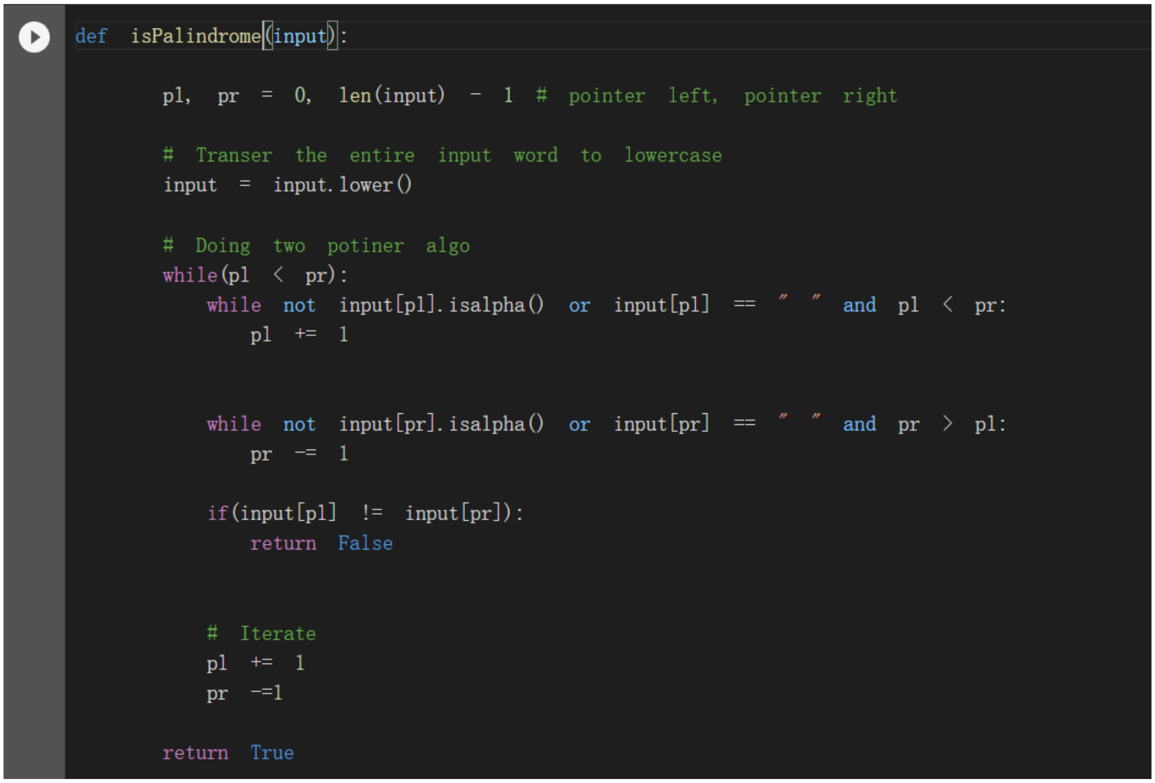
- Write a function that takes in a string and determines whether it is a palindrome. Ignore casing, punctuation, and spaces. (30 points)
    a. “palindrome” is a word, phrase, or sequence that reads the same backward as forward
    b. Ex: “aaabbaaa” - True
    c. Ex: “Mr. Owl ate my metal worm” - True
    d. Ex: “abcd” - False

Ans:

I want to use ***two pointer*** method to solve this problem.

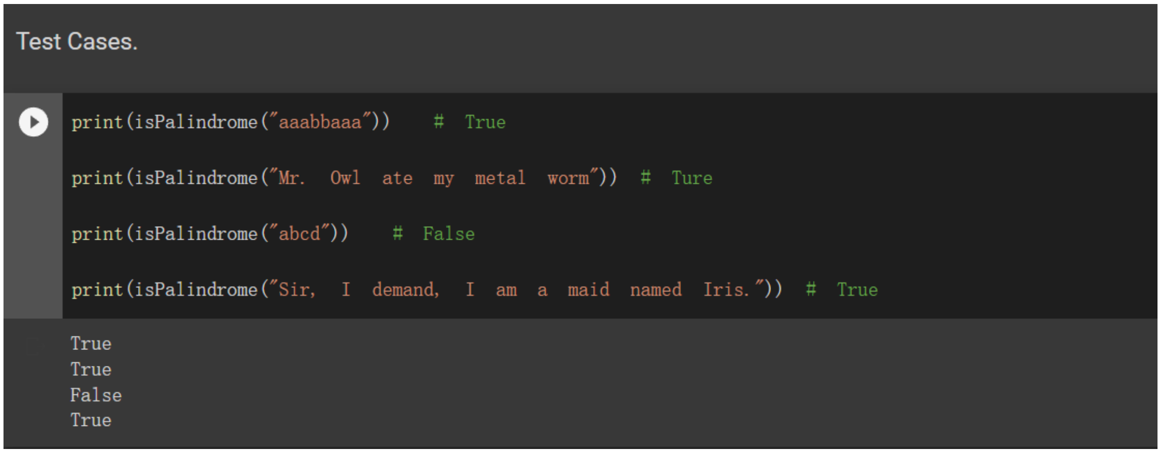
1. Because we need to ignore casing, I would set the entire input word to lowercase first.
2. Ignore punctuation and spaces: if one pointer to a string is not an alphabet or is a space, then set the pointer to the next position.

Writing down ***isPalindrome*** function



```
def isPalindrome(input):

    pl, pr = 0, len(input) - 1 # pointer left, pointer right

    # Transer the entire input word to lowercase
    input = input.lower()

    # Doing two potiner algo
    while(pl < pr):
        while not input[pl].isalpha() or input[pl] == " " and pl < pr:
            pl += 1


        while not input[pr].isalpha() or input[pr] == " " and pr > pl:
            pr -= 1

        if(input[pl] != input[pr]):
            return False


        # Iterate
        pl += 1
        pr -=1

    return True
```

Test case



Test Cases.

```
print(isPalindrome("aaabbaaa"))   # True
print(isPalindrome("Mr. Owl ate my metal worm")) # Ture
print(isPalindrome("abcd"))   # False
print(isPalindrome("Sir, I demand, I am a maid named Iris.")) # True
```

```
True
True
False
True
```

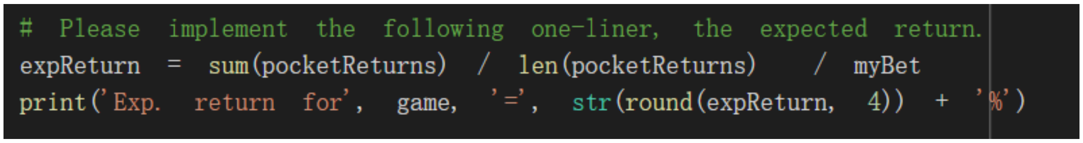
That seems the function works fine!

Monte Carlo Simulation

- Roulette is a casino game. For simplicity, we only allow betting on numbers. One number at a time. The following is such a Python implementation. Let’s bet $1 on pocket number 7.
    1. Please complete the expectedReturn (the statement located next to the last line). (20 points)

       Ans:

       Expected value should be the total of pocketReturns divide the length of pocketReturns, then divide *myBet* because we want to get the ***rate of return*** rather than total amount of return. Also, don’t forget to multiply *100* because we are going to print out the percentage of rate of return.

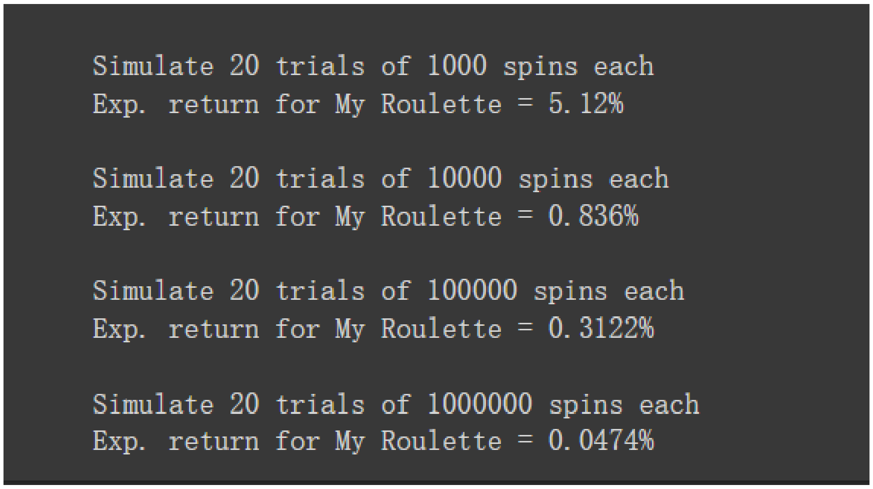


```
# Please implement the following one-liner, the expected return.
expReturn = sum(pocketReturns) / len(pocketReturns) / myBet
print('Exp. return for', game, '=', str(round(expReturn, 4)) + '%')
```

    2. Please report the simulation results after completing the expectedReturn. (15 points)

       Ans:

       The simulation result is **5.12%**, **0.836%**, **0.3122%**, and **0.0474%** respectively.



```
Simulate 20 trials of 1000 spins each
Exp. return for My Roulette = 5.12%

Simulate 20 trials of 10000 spins each
Exp. return for My Roulette = 0.836%

Simulate 20 trials of 100000 spins each
Exp. return for My Roulette = 0.3122%

Simulate 20 trials of 1000000 spins each
Exp. return for My Roulette = 0.0474%
```

3. Please calculate the analytical answer of the expected return. (15 points)

   Ans:

   Expected return rate = (35 * 1/36 - 1 * 35/36) / 1 = **0.0%**

   To sum up, except for those who are doing this game for fun, hosting this game or playing this game in this way is wasting time in the long run.

```
import random

class MyRoulette():
    def __init__(self):
        self.pockets = []
        for i in range(1, 37):
            self.pockets.append(i)
        self.ball = None
        self.pocketOdds = len(self.pockets) - 1
    def spin(self):
        self.ball = random.choice(self.pockets)
    def betPocket(self, pocket, amt):
        if str(pocket) == str(self.ball):
            return amt*self.pocketOdds
        else: return -amt
    def __str__(self):
        return 'My Roulette'

def playRoulette(game, numSpins, pocket, bet, toPrint):
    totPocket = 0
    for i in range(numSpins):
        game.spin()
        totPocket += game.betPocket(pocket, bet)
    if toPrint:
        print(numSpins, 'spins of', game)
        print('Expected return betting', pocket, '=',\
              str(100*totPocket/numSpins) + '%\n')
    return (totPocket/numSpins)

def findPocketReturn(game, numTrials, trialSize, toPrint, pocket, bet):
    pocketReturns = []
    for t in range(numTrials):
        trialVals = playRoulette(game, trialSize, pocket, bet, toPrint)
        pocketReturns.append(trialVals)
    return pocketReturns

# Monte Carlo simulation begins.
random.seed(0)
numTrials = 20
# Instantiate the Roulette game.
game = MyRoulette()
myPocket = 7
myBet = 1
for numSpins in (1000, 10000, 100000, 1000000):
    print('\nSimulate', numTrials, 'trials of', numSpins, 'spins each')
    # The list of my simulation results.
    pocketReturns = findPocketReturn(game, numTrials, numSpins, False, myPocket, myBet)
    # Please implement the following one-liner, the expected return.
    expReturn =
    print('Exp. return for', game, '=', str(round(expReturn, 4)) + '%')
```