Practice Problems in Python

BSDS, 1st Year 2024–2025

Date: 13 August, 2024

PRACTICING INSTRUCTIONS

1. Write down the best possible implementation of the problems listed below.
2. Take care of the error cases too.
3. After you finish the coding, look at the model answers at the end and understand.

NOTE: The programs are to be written in Python and should be well commented. All programs should take the required inputs from standard input buffer and print the desired outputs to the standard output buffer, until otherwise stated.

Q1. (EASY) Given an string, write a program to verify whether the string has equal number of vowels and consonants or not.

Q2. (EASY) An n-digit number is `SPECIAL` if the addition of its sum of the digits and the product of its digits equals to the original number. E.g., 19 is a `SPECIAL` 2-digit number. Write a program to verify whether a given number is `SPECIAL` or not. Extend this program to verify whether there exists any `SPECIAL` number for a given value of number of digits n.

Q3. (EASY) The `abundancy` of a natural number n is defined as the rational number $\frac{\sigma(n)}{n}$, the ratio between the sum of divisors of the number and the number itself. A number n is defined as `friendly` if it shares `abundancy` with one or more other numbers. This means there might exist a pair of numbers i and j such that $i \neq j$ but $\frac{\sigma(i)}{i} = \frac{\sigma(j)}{j}$. For example, 6 and 28 are `friendly` with each other because $\frac{\sigma(6)}{6} = \frac{\sigma(28)}{28} = 2$. Write a program to verify whether a pair of integers given as user input are `friendly` or not.

Q4. (EASY) Write a program to crop out the words that are palindromes from a sentence given as user input.

Q5. (EASY) Write a program to check whether a number given as user input is a power of 32 or not. Accordingly, print POWER OF 32 or NOT POWER OF 32.

Q6. (MEDIUM) Two elements $A[i]$ and $A[j]$ of a list A are said to form an *inversion pair* if $A[i] > A[j]$ but $i < j$. Write a program to count the number of inversion pairs in a list A containing distinct integers.
Note that, for the array $A = \{8, 4, 2, 1\}$, the *inversion pairs* are (8, 4), (4, 2), (8, 2), (8, 1), (4, 1) and (2, 1).

Q7. (MEDIUM) Suppose you are playing a game in turn with the computer. Total n number of sticks are to be picked up in this game. Whoever picks the last one loses the game. Neither

the computer nor you can pick up more than 3 sticks at a time. Nobody can skip a turn, i.e. at least one stick is to be picked up in a turn. Write a program to ensure that the computer wins optimally (whenever there is a chance) irrespective of the turn.

Q8. (MEDIUM) The following pseudocode gives a simple way to calculate a transformation of a string into another string that can be easily compressed. It assumes that the input string s contains special characters '^' and '$' (not to be taken as inputs), which are the first and last characters, respectively, and occurs nowhere else in the text. E.g., '^CAGAGA$' gets converted to 'A$GGC^AA' through this transformation.

```
function TRANSFORM(string s){
    R <- All possible rotations of s as elements // Right circular shift
    SortedR <- The elements of R in lexicographic order
    return(Last character of each element in SortedR in the order they appear)
}
```

Note that the alphabets get a higher priority than the character '^', and it gets a higher priority than '$' in lexicographic ordering.

Q9. (MEDIUM) Write a program to print the following pattern given the line number as user input.

```
        *
      * * *
    *   *   *
   *    *    *
  *     *     *
* * * * * * * * *
  *     *     *
   *    *    *
    *   *   *
      * * *
        *
```

Q10. (MEDIUM) Let us define the `value` of a string as the sum of ASCII values of its characters. For example, `value` of the string "In2Comp" is $632 = (73 + 110 + 50 + 67 + 111 + 109 + 112)$. Write a program that will take a set of strings as inputs and show them in the ascending order of `value` as the output.

Q11. (HARD) Write a program to print the following pattern given the line number as user input. Note that the numbers starting from top left corner are in the ascending order following the path toward right, then down, left, top, then right again, and so on.

```
 1  2  3  4  5
16 17 18 19  6
```

```
15 24 25 20  7
14 23 22 21  8
13 12 11 10  9
```

Q12. (HARD) A group of n friends $F_1, F_2, \ldots, F_n$ decide to try their luck at a lottery. Each person F_i buys a number X_i of lottery tickets. Each lottery ticket has a digit (0-9) printed on it. The rule for winning the jackpot is as follows. Each person is asked to announce the largest multiple of 3 that can be formed by selecting and arranging the digits on his lottery tickets. The person who has the highest such multiple wins the jackpot. Note that a person's tickets may not have distinct digits on them.

Q13. (HARD) There are N petrol pumps $P_1, P_2, \ldots, P_N$ arranged in a clockwise direction along a circular road. Consider a truck with a fuel tank that is initially empty, but which has infinite capacity. The truck will initially fuel up at some P_i and move in a clockwise direction along the circular road. Each time it reaches a petrol pump, it will take up all the petrol available at that pump. Write a program to determine the first P_i such that if the truck starts from P_i, it will be able to completely traverse the circle and return to P_i. The amount of petrol that every petrol pump has (in litres), and the distance from that petrol pump to the next petrol pump (in km) will be given to you as inputs. Assume that the truck needs 1 litre of fuel to travel 1 km.

Q14. (HARD) Let us define a sequence $a_0, a_1, a_2, \ldots, a_{n-1}$ as Λ-*bitonic* if there exists a j, $0 \leq j < n$, such that $a_0 < a_1 < \cdots < a_j > a_{j+1} > \cdots > a_{n-1}$. Consider an $m \times n$ matrix A consisting of integer entries, such that each row and each column of the matrix forms a Λ-bitonic sequence. Write a program to efficiently find the largest element of the matrix.

Q15. (HARD) The power of any arbitrary square matrix A, denoted as A^n, for a nonnegative integer n, is defined as the matrix product of n copies of A. This means

$$A^n = \underbrace{A \cdots A}_{n \text{ times}}.$$

Note that, a matrix to the zeroth power is defined to be the identity matrix of the same dimensions, i.e. $A^0 = I$.

Given the square matrix A and the nonnegative integer n, write a program that can recursively compute A^n involving minimal multiplications.

Q16. (HARD) Let us define the `SQUEEZE` operation on a number that decreases the number of occurrences of all of its digits same number of times retaining their ordering in the original number. E.g., if a number '110011' is SQUEEZED by a factor of 2 then it will turn into 101. Write a program that will take a number and show the maximum possible SQUEEZED number as the output. Note that the `SQUEEZE` operation is valid if and only if it is applicable to all the segments (series of 0s or 1s) of the number.

Q17. (OPEN) Dyadic rationals are fractions where the denominator is a power of two, expressed as $\frac{k}{2^n}$. As computers operate in base-2 (binary), dyadic rationals align perfectly with the binary system. They can be represented exactly in binary floating-point formats. For example: 1/2 = 0.5 = 0.1 in base 2; 3/8 = 0.375 = 0.011 in base 2. While dyadic rationals do not cover all real numbers, they are crucial in digital computations for representing fractions exactly where possible, reducing rounding errors in certain calculations. By using dyadic rationals, one can perform arithmetic operations without introducing rounding errors that typically occur with non-dyadic fractions. Write a program to make the calculations of non-dyadic fractions more accurate by means of performing calculations on dyadic fractions to the extent possible. You are welcome to choose special cases of input patterns to attempt.

Model Answers:

Q1. GENERAL APPROACH

```
str = input('Enter a string: ')
vowels = 'aeiouAEIOU'
countV, countC = 0, 0
for c in str:
    if c in vowels:
        countV = countV + 1
    else:
        countC = countC + 1
if countV == countC:
    print('EQUAL')
else:
    print('NOT EQUAL')
```

BETTER APPROACH

```
str = input('Enter a string: ')
vowels = "aeiouAEIOU"
countV = sum(str.count(c) for c in vowels)
print('NOT EQUAL' if len(str)-(countV<<1) else 'EQUAL')
```

Q2. GENERAL APPROACH

```
def checkSPECIAL(num):
    sum, prod = 0, 1
    for d in num:
        sum += int(d)
        prod *= int(d)
    if sum + prod == int(num):
        return('SPECIAL')
    else:
        return('NOT SPECIAL')
num = input('Enter a number: ')
print(checkSPECIAL(num))
n = input('Enter the number of digits: ')
print('The '+n+'-digit SPECIAL numbers are:')
for i in range(10**(int(n)-1), 10**int(n)):
    if checkSPECIAL(str(i)) == "SPECIAL":
        print(i)
```

BETTER APPROACH

```
Try mathematically!!!
```

Q3. GENERAL APPROACH

```
def abundancy(num):
    sum = 0
    for i in range(1, num+1):
        if num%i == 0:
            sum += i
    return sum
num1, num2 = input('Enter a pair of numbers: ').split()
if abundancy(int(num1))/int(num1) == abundancy(int(num2))/int(num2):
    print('friendly')
else:
    print('not friendly')
```

BETTER APPROACH

```
num1, num2 = input('Enter a pair of numbers: ').split()
divisors1 = [i for i in range(1, int(num1)+1) if int(num1)%i == 0]
divisors2 = [i for i in range(1, int(num2)+1) if int(num2)%i == 0]
if sum(divisors1)*int(num2) == sum(divisors2)*int(num1):
    print('friendly')
else:
    print('not friendly')
```