1. Create a function to perform basic arithmetic operations that includes addition, subtraction, multiplication and division on a string number (e.g. "12 + 24" or "23 - 21" or "12 // 12" or "12 \* 21").

Here, we have 1 followed by a space, operator followed by another space and 2. For the challenge, we are going to have only two numbers between 1 valid operator. The return value should be a number.

eval() is not allowed. In case of division, whenever the second number equals "0" return -1.

For example:

"15 // 0" ➞ -1

**Examples**

arithmetic\_operation("12 + 12") ➞ 24 // 12 + 12 = 24

arithmetic\_operation("12 - 12") ➞ 24 // 12 - 12 = 0

arithmetic\_operation("12 \* 12") ➞ 144 // 12 \* 12 = 144

arithmetic\_operation("12 // 0") ➞ -1 // 12 / 0 = -1

def arithmetic\_operation(string):

l=string.split()

operator,n1,n2=l[1],int(l[0]),int(l[2])

if operator=='+':

result=(n1+n2)

if operator=='-':

result=(n1-n2)

if operator=='\*':

result=(n1\*n2)

if operator=='//':

if n2==0:

result=-1

else:

result=(n1//n2)

return result

print(arithmetic\_operation('12 + 12'))

2. Write a function that takes the coordinates of three points in the form of a 2d array and returns the perimeter of the triangle. The given points are the vertices of a triangle on a two-dimensional plane.

**Examples**

perimeter( [ [15, 7], [5, 22], [11, 1] ] ) ➞ 47.08

perimeter( [ [0, 0], [0, 1], [1, 0] ] ) ➞ 3.42

perimeter( [ [-10, -10], [10, 10 ], [-10, 10] ] ) ➞ 68.28

import numpy as np

import math

def perimeter(coordinates):

p1=coordinates[0]

p2=coordinates[1]

p3=coordinates[2]

d1=round(math.sqrt((p2[0]-p1[0])\*\*2 + (p2[1]-p1[1])\*\*2),2)

d2=round(math.sqrt((p3[0]-p2[0])\*\*2 + (p3[1]-p2[1])\*\*2),2)

d3=round(math.sqrt((p1[0]-p3[0])\*\*2 + (p1[1]-p3[1])\*\*2),2)

return round(d1+d2+d3,2)

print(perimeter( [ [15, 7], [5, 22], [11, 1] ] ))

3. A city skyline can be represented as a 2-D list with 1s representing buildings. In the example below, the height of the tallest building is 4 (second-most right column).

[[0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 1, 0],

[0, 0, 1, 0, 1, 0],

[0, 1, 1, 1, 1, 0],

[1, 1, 1, 1, 1, 1]]

Create a function that takes a skyline (2-D list of 0's and 1's) and returns the height of the tallest skyscraper.

**Examples**

tallest\_skyscraper([

[0, 0, 0, 0],

[0, 1, 0, 0],

[0, 1, 1, 0],

[1, 1, 1, 1]

]) ➞ 3

tallest\_skyscraper([

[0, 1, 0, 0],

[0, 1, 0, 0],

[0, 1, 1, 0],

[1, 1, 1, 1]

]) ➞ 4

tallest\_skyscraper([

[0, 0, 0, 0],

[0, 0, 0, 0],

[1, 1, 1, 0],

[1, 1, 1, 1]

]) ➞ 2

def tallest\_skyscraper(tall):

return sum(1 for eachlist in tall if sum(eachlist)>0)

print(tallest\_skyscraper([[0, 0, 0, 0],[0, 1, 0, 0],[0, 1, 1, 0],[1, 1, 1, 1]]))

4. A financial institution provides professional services to banks and claims charges from the customers based on the number of man-days provided. Internally, it has set a scheme to motivate and reward staff to meet and exceed targeted billable utilization and revenues by paying a bonus for each day claimed from customers in excess of a threshold target.

This quarterly scheme is calculated with a threshold target of 32 days per quarter, and the incentive payment for each billable day in excess of such threshold target is shown as follows:

Days Bonus

0 to 32 days Zero

33 to 40 days SGD$325 per billable day

41 to 48 days SGD$550 per billable day

Greater than 48 days SGD$600 per billable day

Please note that incentive payment is calculated progressively. As an example, if an employee reached total billable days of 45 in a quarter, his/her incentive payment is computed as follows:

32\*0 + 8\*325 + 5\*550 = 5350

Write a function to read the billable days of an employee and return the bonus he/she has obtained in that quarter.

**Examples**

bonus(15) ➞ 0

bonus(37) ➞ 1625

bonus(50) ➞ 8200

def bonus(int\_num):

if int\_num > 48:

output = 0+(8\*325)+(8\*550)+((int\_num-48)\*600)

elif int\_num < 48 and int\_num >= 41:

output = 0+(8\*325)+((int\_num-41+1)\*550)

elif int\_num >33 and int\_num <= 40:

output = 0+((int\_num-33+1)\*325)

else:

output = 0

return output

print(bonus(44))

5. A number is said to be Disarium if the sum of its digits raised to their respective positions is the number itself.

Create a function that determines whether a number is a Disarium or not.

**Examples**

is\_disarium(75) ➞ False

# 7^1 + 5^2 = 7 + 25 = 32

is\_disarium(135) ➞ True

# 1^1 + 3^2 + 5^3 = 1 + 9 + 125 = 135

is\_disarium(544) ➞ False

is\_disarium(518) ➞ True

is\_disarium(466) ➞ False

is\_disarium(8) ➞ True

def is\_disarium(n) :

sum=0

count = len(str(n))

x = n

while (x!=0):

r = x%10

sum = sum + (r\*\*count)

count = count-1

x = x//10

if (sum == n) :

return True

else :

return False

print(is\_disarium(135))