1. def transpose\_matrix(in\_list):

output = []

for ele\_1 in range(len(in\_list[0])):

temp = []

for ele\_2 in in\_list:

temp.append(ele\_2[ele\_1])

output.append(temp)

print(f'transpose\_matrix({in\_list}) ➞ {output}')

transpose\_matrix([[1, 1, 1],[2, 2, 2],[3, 3, 3]])

transpose\_matrix([[5, 5],[6, 7],[9, 1]])

Output:

transpose\_matrix([[1, 1, 1], [2, 2, 2], [3, 3, 3]]) ➞ [[1, 2, 3], [1, 2, 3], [1, 2, 3]]

transpose\_matrix([[5, 5], [6, 7], [9, 1]]) ➞ [[5, 6, 9], [5, 7, 1]]

1. def is\_valid\_hex\_code(in\_string):

out\_string = True

for ele in in\_string:

if ele.lower() not in '#abcdef0123456789' or len(in\_string) != 7:

out\_string = False

print(f'is\_valid\_hex\_code({in\_string}) ➞ {out\_string}')

is\_valid\_hex\_code("#CD5C5C")

is\_valid\_hex\_code("#EAECEE")

is\_valid\_hex\_code("#eaecee")

is\_valid\_hex\_code("#CD5C58C")

is\_valid\_hex\_code("#CD5C5Z")

is\_valid\_hex\_code("#CD5C&C")

is\_valid\_hex\_code("CD5C5C")

Output:

is\_valid\_hex\_code(#CD5C5C) ➞ True

is\_valid\_hex\_code(#EAECEE) ➞ True

is\_valid\_hex\_code(#eaecee) ➞ True

is\_valid\_hex\_code(#CD5C58C) ➞ False

is\_valid\_hex\_code(#CD5C5Z) ➞ False

is\_valid\_hex\_code(#CD5C&C) ➞ False

is\_valid\_hex\_code(CD5C5C) ➞ False

1. import math

def mark\_maths(in\_list):

out\_list = []

for ele in in\_list:

ele = ele.split("=")

out\_list.append(eval(ele[0]) == int(ele[1]))

print(f'mark\_maths({in\_list}) ➞ {str(math.ceil((sum(out\_list)/len(out\_list))\*100))}%')

mark\_maths(["2+2=4", "3+2=5", "10-3=3", "5+5=10"])

mark\_maths(["1-2=-2"])

mark\_maths(["2+3=5", "4+4=9", "3-1=2"])

Output:

mark\_maths(['2+2=4', '3+2=5', '10-3=3', '5+5=10']) ➞ 75%

mark\_maths(['1-2=-2']) ➞ 0%

mark\_maths(['2+3=5', '4+4=9', '3-1=2']) ➞ 67%

1. def magic\_square\_game(in\_one,in\_two):

output = False

if in\_two[1][in\_one[0]-1] == in\_one[1][in\_two[0]-1]:

output = True

print(f'magic\_square\_game{in\_one,in\_two} ➞ {output}')

magic\_square\_game([2, "100"], [1, "101"])

magic\_square\_game([2, "001"], [1, "101"])

magic\_square\_game([3, "111"], [2, "011"])

magic\_square\_game([1, "010"], [3, "101"])

Output:

magic\_square\_game([2, '100'], [1, '101']) ➞ False

magic\_square\_game([2, '001'], [1, '101']) ➞ True

magic\_square\_game([3, '111'], [2, '011']) ➞ True

magic\_square\_game([1, '010'], [3, '101']) ➞ False

1. import math

def lets\_meet(in\_dist,in\_va,in\_vb):

total\_time = in\_dist/(in\_va+in\_vb)

Hours = math.floor(total\_time)

Minutes = math.floor((total\_time-Hours)\*60)

Seconds = math.floor(((((total\_time)-Hours)\*60)-Minutes)\*60)

print(f'lets\_meet{in\_dist,in\_va,in\_vb} ➞ {Hours}h {Minutes}min {Seconds}s')

lets\_meet(100, 10, 30)

lets\_meet(280, 70, 80)

lets\_meet(90, 75, 65)

Output:

lets\_meet(100, 10, 30) ➞ 2h 30min 0s

lets\_meet(280, 70, 80) ➞ 1h 52min 0s

lets\_meet(90, 75, 65) ➞ 0h 38min 34s