1. def checker\_board(grid\_size,in\_one,in\_two):

if in\_one != in\_two:

input = [in\_one,in\_two]

output = []

for ele\_1 in range(grid\_size):

output.append([])

for ele\_2 in range(grid\_size):

output[ele\_1].append(input[(ele\_1+ele\_2)%2])

else:

output = 'Invalid'

print(f'checker\_board{grid\_size,in\_one,in\_two} ➞ {output}')

checker\_board(2, 7, 6)

checker\_board(3, "A", "B")

checker\_board(4, "c", "d")

checker\_board(4, "c", "c")

Output:

checker\_board(2, 7, 6) ➞ [[7, 6], [6, 7]]

checker\_board(3, 'A', 'B') ➞ [['A', 'B', 'A'], ['B', 'A', 'B'], ['A', 'B', 'A']]

checker\_board(4, 'c', 'd') ➞ [['c', 'd', 'c', 'd'], ['d', 'c', 'd', 'c'], ['c', 'd', 'c', 'd'], ['d', 'c', 'd', 'c']]

checker\_board(4, 'c', 'c') ➞ Invalid

1. def almost\_palindrome(in\_string):

in\_string\_rev = in\_string[::-1]

count = 0

for ele in range(len(in\_string)):

if in\_string[ele] != in\_string\_rev[ele]:

count +=1

output = True if count == 2 else False

print(f'almost\_palindrome({in\_string}) ➞ {output}')

almost\_palindrome("abcdcbg")

almost\_palindrome("abccia")

almost\_palindrome("abcdaaa")

almost\_palindrome("1234312")

Output:

almost\_palindrome(abcdcbg) ➞ True

almost\_palindrome(abccia) ➞ True

almost\_palindrome(abcdaaa) ➞ False

almost\_palindrome(1234312) ➞ False

1. def prime\_numbers(in\_num):

out\_num = 0

out\_list = [2,3]

for ele in range(1,in\_num+1):

if ele <= 3 and ele > 0:

out\_num = 2 if ele==3 else 1 if ele ==2 else 0

elif ele > 3 and (((ele-1)%6 == 0) or ((ele+1)%6 == 0)):

out\_num +=1

out\_list.append(ele)

for top in out\_list:

for bottom in out\_list:

if top != bottom:

if top%bottom == 0:

out\_num -= 1

print(f'prime\_numbers({in\_num}) ➞ {out\_num}')

prime\_numbers(10)

prime\_numbers(20)

prime\_numbers(30)

Output:

prime\_numbers(10) ➞ 4

prime\_numbers(20) ➞ 8

prime\_numbers(30) ➞ 10

1. def after\_n\_days(in\_list,in\_num):

week\_dict = {0:'Sunday',1:'Monday',2:'Tuesday',3:'Wednesday',4:'Thursday',5:'Friday',6:'Saturday'}

week\_days\_no = list(week\_dict.keys())

week\_days\_name = list(week\_dict.values())

output = []

for ele in in\_list:

output.append(week\_dict[(week\_days\_name.index(ele)+in\_num)%7])

print(f'after\_n\_days{in\_list,in\_num} ➞ {output}')

after\_n\_days(["Thursday", "Monday"], 4)

after\_n\_days(["Sunday", "Sunday", "Sunday"], 1)

after\_n\_days(["Monday", "Tuesday", "Friday"], 1)

Output:

after\_n\_days(['Thursday', 'Monday'], 4) ➞ ['Monday', 'Friday']

after\_n\_days(['Sunday', 'Sunday', 'Sunday'], 1) ➞ ['Monday', 'Monday', 'Monday']

after\_n\_days(['Monday', 'Tuesday', 'Friday'], 1) ➞ ['Tuesday', 'Wednesday', 'Saturday']

1. def is\_correct\_aliases(in\_list\_one, in\_list\_two):

output = False

if len(in\_list\_one) == len(in\_list\_two):

for ele in range(len(in\_list\_one)):

if in\_list\_one[ele].split(" ")[0][0] == in\_list\_two[ele].split(" ")[0][0] == in\_list\_two[ele].split(" ")[1][0]:

output = True

else:

output = False

break

print(f'is\_correct\_aliases{in\_list\_one,in\_list\_two}➞{output}')

is\_correct\_aliases(["Beth T."],["Brandishing Mimosa"])

is\_correct\_aliases(["Adrian M.","Harriet S.","Mandy T."], ["Amazing Artichoke", "Hopeful Hedgehog", "Marvelous Mouse"])

is\_correct\_aliases(["Rachel F.","Pam G.","Fred Z.","Nancy K."], ["Reassuring Rat", "Peaceful Panda", "Fantastic Frog", "Notable Nickel"])

Output:

is\_correct\_aliases(['Beth T.'], ['Brandishing Mimosa'])➞False

is\_correct\_aliases(['Adrian M.', 'Harriet S.', 'Mandy T.'], ['Amazing Artichoke', 'Hopeful Hedgehog', 'Marvelous Mouse'])➞True

is\_correct\_aliases(['Rachel F.', 'Pam G.', 'Fred Z.', 'Nancy K.'], ['Reassuring Rat', 'Peaceful Panda', 'Fantastic Frog', 'Notable Nickel'])➞True