1. def count\_same\_ends(in\_string):

special\_chars = '!@#$%^&\*.'

cleaned\_string = ''

out\_num = 0

for ele in in\_string:

if ele not in special\_chars:

cleaned\_string += ele

for ele in cleaned\_string.split(" "):

if ele[0].lower() == ele[-1].lower():

if len(ele) != 1:

out\_num +=1

print(f'count\_same\_ends({in\_string}) ➞ {out\_num}')

count\_same\_ends("Pop! goes the balloon")

count\_same\_ends("And the crowd goes wild!")

count\_same\_ends("No I am not in a gang.")

Output:

count\_same\_ends(Pop! goes the balloon) ➞ 1

count\_same\_ends(And the crowd goes wild!) ➞ 0

count\_same\_ends(No I am not in a gang.) ➞ 1

1. def atbash(in\_string):

alpha = 'abcdefghijklmnopqrstuvwxyz'

r\_alpha = 'zyxwvutsrqponmlkjihgfedcba'

out\_string = ''

for ele in in\_string:

if ele not in " !1234567890":

out\_string += r\_alpha[alpha.index(ele.lower())].upper() if ele.isupper() else r\_alpha[alpha.index(ele.lower())]

else:

out\_string += ele

print(f'atbash({in\_string}) ➞ {out\_string}')

atbash("apple")

atbash("Hello world!")

atbash("Christmas is the 25th of December")

Output:

atbash(apple) ➞ zkkov

atbash(Hello world!) ➞ Svool dliow!

atbash(Christmas is the 25th of December) ➞ Xsirhgnzh rh gsv 25gs lu Wvxvnyvi

1. class Employee:

def \_\_init\_\_(self,name=None,salary=None,height=None,nationality=None):

self.name = name

self.firstname = name.split(" ")[0]

self.lastname = name.split(" ")[1]

self.salary = salary

self.height = height

self.nationality = nationality

john = Employee("John Doe")

mary = Employee("Mary Major",salary=120000)

richard = Employee("Richard Roe", salary=110000, height=178)

giancarlo = Employee("Giancarlo Rossi", salary=115000, height=182, nationality="Italian")

print(f'john.name ➞ "{john.name}"')

print(f'mary.lastname ➞ "{mary.lastname}"')

print(f'richard.height ➞ {richard.height}')

print(f'giancarlo.nationality ➞ "{giancarlo.nationality}"')

Output:

john.name ➞ "John Doe"

mary.lastname ➞ "Major"

richard.height ➞ 178

giancarlo.nationality ➞ "Italian"

1. def can\_see\_stage(in\_list):

transposed\_list = []

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

temp\_list = []

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

temp\_list.append(in\_list[item][ele])

transposed\_list.append(temp\_list)

output = True

for ele in transposed\_list:

if ele != sorted(ele) or len(ele) != len(set(ele)):

output = False

break

print(f'can\_see\_stage({in\_list}) ➞ {output}')

can\_see\_stage([[1, 2, 3],[4, 5, 6],[7, 8, 9]])

can\_see\_stage([[0, 0, 0],[1, 1, 1],[2, 2, 2]])

can\_see\_stage([[2, 0, 0],[1, 1, 1],[2, 2, 2]])

can\_see\_stage([[1, 0, 0],[1, 1, 1],[2, 2, 2]])

Output:

can\_see\_stage([[1, 2, 3], [4, 5, 6], [7, 8, 9]]) ➞ True

can\_see\_stage([[0, 0, 0], [1, 1, 1], [2, 2, 2]]) ➞ True

can\_see\_stage([[2, 0, 0], [1, 1, 1], [2, 2, 2]]) ➞ False

can\_see\_stage([[1, 0, 0], [1, 1, 1], [2, 2, 2]]) ➞ False

1. class Pizza:

order\_count = 0

def \_\_init\_\_(self,ingredients=None):

self.ingredients = ingredients

self.order\_number = Pizza.order\_count+1

Pizza.order\_count = self.order\_number

def hawaiian(self):

self.ingredients = ['ham', 'pineapple']

def meat\_festival(self):

self.ingredients = ['beef', 'meatball', 'bacon']

def garden\_feast(self):

self.ingredients = ['spinach', 'olives', 'mushroom']

p1 = Pizza(["bacon", "parmesan", "ham"])

p2 = Pizza()

p2.garden\_feast()

print(f'p1.ingredients ➞ {p1.ingredients}')

print(f'p2.ingredients ➞ {p2.ingredients}')

print(f'p1.order\_number ➞ {p1.order\_number}')

print(f'p2.order\_number ➞ {p2.order\_number}')

Output:

p1.ingredients ➞ ['bacon', 'parmesan', 'ham']

p2.ingredients ➞ ['spinach', 'olives', 'mushroom']

p1.order\_number ➞ 1

p2.order\_number ➞ 2