**Solution:**

**Question 1:**

**Given a sentence, return the number of words that have the same first and last letter:**

def count\_same\_ends(sentence):

words = sentence.split()

count = 0

for word in words:

word = word.strip("!.,?")

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

count += 1

return count

# Example Usage:

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

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

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

**Question 2:**

**The Atbash cipher is an encryption method in which each letter is replaced by its mirror letter in the alphabet.**

def atbash(s):

result = []

for char in s:

if char.isalpha():

result.append(chr(219 - ord(char.lower())) if char.islower() else chr(155 - ord(char)))

else:

result.append(char)

return ''.join(result)

# Example Usage:

print(atbash("apple")) # ➞ "zkkov"

print(atbash("Hello world!")) # ➞ "Svool dliow!"

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

**Question 3:**

**Create a class Employee that takes a full name and optional keywords, and stores the attributes accordingly.**

class Employee:

def \_\_init\_\_(self, full\_name, \*\*kwargs):

self.name, self.lastname = full\_name.split() # Assuming 2-part name

for key, value in kwargs.items():

setattr(self, key, value)

# Example Usage:

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(john.name) # ➞ "John"

print(mary.lastname) # ➞ "Major"

print(richard.height) # ➞ 178

print(giancarlo.nationality) # ➞ "Italian"

**Question 4:**

**Determine if each seat can see the front stage in a grid of numbers.**

def can\_see\_stage(seats):

for row in seats:

if any(row[i] <= row[i-1] for i in range(1, len(row))):

return False

return True

# Example Usage:

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

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

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

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

**Question 5:**

**Create a Pizza class with ingredients, order number, and predefined pizza flavours.**

class Pizza:

order\_count = 0

flavours = {

"hawaiian": ["ham", "pineapple"],

"meat\_festival": ["beef", "meatball", "bacon"],

"garden\_feast": ["spinach", "olives", "mushroom"]

}

def \_\_init\_\_(self, ingredients):

Pizza.order\_count += 1

self.order\_number = Pizza.order\_count

self.ingredients = ingredients

@classmethod

def hawaiian(cls):

return cls(cls.flavours["hawaiian"])

@classmethod

def meat\_festival(cls):

return cls(cls.flavours["meat\_festival"])

@classmethod

def garden\_feast(cls):

return cls(cls.flavours["garden\_feast"])

# Example Usage:

p1 = Pizza(["bacon", "parmesan", "ham"]) # order 1

p2 = Pizza.garden\_feast() # order 2

print(p1.ingredients) # ➞ ["bacon", "parmesan", "ham"]

print(p2.ingredients) # ➞ ["spinach", "olives", "mushroom"]

print(p1.order\_number) # ➞ 1

print(p2.order\_number) # ➞ 2