def parse\_encoded\_string(encoded):

parts = list(filter(None, encoded.split('0')))

print(parts)

return {

"first\_name": parts[0],

"last\_name": parts[1],

"id": parts[2]

}

result = parse\_encoded\_string("Robert000Smith000123")

print(result)

#Output

['Robert', 'Smith', '123']

{'first\_name': 'Robert', 'last\_name': 'Smith', 'id': '123'}

from collections import Counter

def find\_extra\_char(str1, str2):

count1 = Counter(str1)

count2 = Counter(str2)

print(count1)

print(count2)

for char in count2:

if count2[char] != count1.get(char, 0):

return char

print(find\_extra\_char("eueiieo", "iieoedue"))

#Output

Counter({'e': 3, 'i': 2, 'u': 1, 'o': 1})

Counter({'e': 3, 'i': 2, 'o': 1, 'd': 1, 'u': 1})

d

def is\_shadow\_sentence(sentence1, sentence2):

words1 = sentence1.split()

words2 = sentence2.split()

if len(words1) != len(words2):

return False

for w1, w2 in zip(words1, words2):

if len(w1) != len(w2):

return False

if set(w1) & set(w2):

return False

return True

print(is\_shadow\_sentence("they are round", "fold two times"))

print(is\_shadow\_sentence("his friends", "our company"))

#Output

True

False

def has\_duplicate\_letters(sentence):

words = sentence.split()

for word in words:

letters = [char.lower() for char in word if char.isalpha()]

if len(letters) != len(set(letters)):

return True

return False

print(has\_duplicate\_letters("Each word here is fine"))

print(has\_duplicate\_letters("This sentence has a letter"))

#Output

True

True

def ascii\_to\_hex(ascii\_str):

return ' '.join(format(ord(char), '02x') for char in ascii\_str)

print(ascii\_to\_hex("Hello!"))

def find\_blocking\_spot(pos1, pos2):

winning\_lines = [

[0, 1, 2],

[3, 4, 5],

[6, 7, 8],

[0, 3, 6],

[1, 4, 7],

[2, 5, 8],

[0, 4, 8],

[2, 4, 6]

]

for line in winning\_lines:

if pos1 in line and pos2 in line:

for spot in line:

if spot != pos1 and spot != pos2:

return spot

return None

print(find\_blocking\_spot(0, 1))

print(find\_blocking\_spot(2, 5))

print(find\_blocking\_spot(0, 8))

#Output

48 65 6c 6c 6f 21

2

8

4

def to\_morse\_code(text):

morse\_dict = {

'A': '.-', 'B': '-...', 'C': '-.-.', 'D': '-..',

'E': '.', 'F': '..-.', 'G': '--.', 'H': '....',

'I': '..', 'J': '.---', 'K': '-.-', 'L': '.-..',

'M': '--', 'N': '-.', 'O': '---', 'P': '.--.',

'Q': '--.-', 'R': '.-.', 'S': '...', 'T': '-',

'U': '..-', 'V': '...-', 'W': '.--', 'X': '-..-',

'Y': '-.--', 'Z': '--..',

'0': '-----', '1': '.----', '2': '..---', '3': '...--',

'4': '....-', '5': '.....', '6': '-....', '7': '--...',

'8': '---..', '9': '----.',

'.': '.-.-.-', ',': '--..--', '?': '..--..', "'": '.----.',

'!': '-.-.--', ':': '---...',

' ': '/'

}

return ' '.join(morse\_dict[char.upper()] for char in text if char.upper() in morse\_dict)

print(to\_morse\_code("Hello, World!"))

# Output

.... . .-.. .-.. --- --..-- / .-- --- .-. .-.. -.. -.-.--

import datetime

def has\_friday\_13(month, year):

try:

date = datetime.date(year, month, 13)

return date.weekday() == 4 # 4 means Friday

except ValueError:

return False

print(has\_friday\_13(9, 2025))

print(has\_friday\_13(6, 2025))

#Output

False

True

def check\_input(value):

if not isinstance(value, int):

raise TypeError("Input must be an integer")

if value < 0:

raise ValueError("Number must not be negative")

if value > 100:

raise Exception("Number is too large")

print("Valid input:", value)

test\_values = ["abc", -10, 150, 50]

for val in test\_values:

try:

check\_input(val)

except TypeError as te:

print("Type Error:", te)

except ValueError as ve:

print("Value Error:", ve)

except Exception as e:

print("Other Error:", e)

#Output

Type Error: Input must be an integer

Value Error: Number must not be negative

Other Error: Number is too large

Valid input: 50

class LowBalanceError(Exception):

pass

def withdraw(amount):

balance = 500

if amount > balance:

raise LowBalanceError("Withdrawal amount exceeds account balance.")

else:

print("Withdrawal successful.")

try:

withdraw(600)

except LowBalanceError as e:

print("Caught an exception:", e)

#Output

Caught an exception: Withdrawal amount exceeds account balance.'''

s="hello"

print(s[2])

print(s[-2])

s="python"

print(s[-7])

from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("RepartitionExample").getOrCreate()

data = [("Tom", 25), ("Jerry", 22), ("Mike", 30),("Jack",34),("Rose",32)]

df = spark.createDataFrame(data, ["name", "age"])

print("Original partitions:", df.rdd.getNumPartitions())

print("Original Data:")

df.show()

df\_repart = df.repartition(3)

print("After repartition:", df\_repart.rdd.getNumPartitions())

print("Repartitioned Data:")

df\_repart.show()

#Output

Original partitions: 2

Original Data:

+-----+---+

| name|age|

+-----+---+

| Tom| 25|

|Jerry| 22|

| Mike| 30|

| Jack| 34|

| Rose| 32|

+-----+---+

After repartition: 3

Repartitioned Data:

+-----+---+

| name|age|

+-----+---+

|Jerry| 22|

| Mike| 30|

| Tom| 25|

| Rose| 32|

| Jack| 34|

+-----+---+

'''SELECT \* FROM products

SELECT product\_name, unit\_price

FROM products

WHERE unit\_price > (

SELECT AVG(unit\_price)

FROM products

);

SELECT customer\_id, company\_name

FROM customers

WHERE customer\_id IN (

SELECT customer\_id

FROM orders

GROUP BY customer\_id

HAVING COUNT(order\_id) > 5

);

SELECT first\_name, last\_name

FROM employees

WHERE employee\_id IN (

SELECT employee\_id

FROM orders

WHERE ship\_country = 'Germany'

);

SELECT product\_name FROM products WHERE supplier\_id IN (

SELECT supplier\_id

FROM suppliers

WHERE country = 'USA'

);

SELECT order\_id FROM order\_details GROUP BY order\_id HAVING SUM(quantity) > 50;'''

def wrap\_string(S, W):

for i in range(0, len(S), W):

print(S[i:i+W])

wrap\_string("ABCDEFGHIJKLIMNOQRSTUVWXYZ", 4)

#Output

ABCD

EFGH

IJKL

IMNO

QRST

UVWX

YZ

a="abcdefghijklmnopqrstuvwxyz"

def print\_rangoli(size):

lines = []

for row in range(size):

print\_ = "-".join(a[row:size])

lines.append(print\_[::-1] + print\_[1:])

width = len(lines[0])

for row in range(size-1, 0, -1):

print(lines[row].center(width, '-'))

for row in range(size):

print(lines[row].center(width, '-'))

if \_\_name\_\_ == '\_\_main\_\_':

n = int(input())

print\_rangoli(n)

#Output

5

--------e--------

------e-d-e------

----e-d-c-d-e----

--e-d-c-b-c-d-e--

e-d-c-b-a-b-c-d-e

--e-d-c-b-c-d-e--

----e-d-c-d-e----

------e-d-e------

--------e--------

import calendar

month, day, year = map(int, input().split())

weekday\_num = calendar.weekday(year, month, day)

weekday\_name = calendar.day\_name[weekday\_num].upper()

print(weekday\_name)

#Output

06 19 2025

THURSDAY

M = int(input())

set\_M = set(map(int, input().split()))

N = int(input())

set\_N = set(map(int, input().split()))

result = sorted(set\_M ^ set\_N)

for num in result:

print(num)

#Output

3

1 2 3

3

3 4 5

1

2

4

5

text = input()

reversed\_text = text[::-1]

print(reversed\_text)

#Output

rise to vote sir

ris etov ot esir

text = input()

result = text[::2]

print(result)

#Output

H1e2l3l4o5w6o7r8l9d

Helloworld

h = 34

l = 94

# 2c + 4r = l

# c + r = h

for i in range(h):

t\_l = 2 \* (h - i) + 4 \* i

if t\_l == l:

print(f"Chicken : {i}, Rabbit : {h-i}")

break

#Output

Chicken: 23

Rabbit: 12

list1 = [1, 3, 6, 78, 35, 55]

list2 = [12, 24, 35, 24, 88, 120, 155]

set1 = set(list1)

set2 = set(list2)

res=set(set1 & set2)

print(f"Intersection : {list(res)}")

#Output

Intersection : [35]

class Person:

def getGender(self):

return "Unknown"

class Male(Person):

def getGender(self):

return "Male"

class Female(Person):

def getGender(self):

return "Female"

m=Male()

f=Female()

print(m.getGender())

print(f.getGender())

#Output

Male

Female