1. 請寫一個程式，輸入一個字串，先判定字串是否只包含 0~9，請將此字串轉為十進制數字，將十進制數字輸出 2 進位，並且計算 2 進位數字裡有多少個”1”

```
import re

"""
Issue1: 將輸入
1. 判斷是否字串
2. 判斷是否可轉數字 (dec)
3. 將 dec 轉 bin
4. 判斷 bin 中有幾個 1
"""

MAX_INT = pow(2, 31) - 1
MIN_INT = - pow(2, 31)
MAX_LENGTH = len(str(MAX_INT)) + 1  # 設置最大輸入長度，10 (可輸入負為 11)

def limitInput(value) -> int:

    while True:
        inputStr = input(value)

        if len(inputStr) > MAX_LENGTH:
            print("輸入的字串長度不能超過", MAX_LENGTH, "個字符。請重新輸入。")
            continue

        # if not isString(inputStr):
        #     print("非字串")
        #     continue

        if not isNumericString(inputStr):
            print("輸入的字串包含非數字字符。請輸入只包含 0-9 正負整數字串。")
            continue
```

```
        inputValue = int(inputStr)
        if not (MIN_INT <= int(inputValue) <= MAX_INT):
            print("超過 INT 上限數:", MAX_INT)
            continue

        return inputValue

# def isString(inputStr:str) -> bool:
#     return isinstance(inputStr, str)

def isNumericString(inputStr:str) -> bool:
    return bool(re.match(r'^-?\d+$', inputStr))

def decimalToBinary(value:int) -> int:
    # print(bin(value))
    # 濾掉 0b
    return bin(value)[2:]

def oriDecimalToBinary(value:int) -> str:
    binStr = ""

    if value == 0:
        return '0'

    # 2’s Complement
    if value < 0:
        value = 2**32 + value
    else:
        value = value

    while value > 0:
        binStr = str(value % 2) + binStr
        value = value // 2

    return binStr


def oriBinaryCountOne(binStr:str) -> int:
    count = 0
```

```
    for char in binStr:
        if char == '1':
            count += 1

    return count

def binaryCountOne(binStr:str) -> int:
    return binStr.count('1')

def main():
    value = limitInput("輸入: ")

    # binValue = decimalToBinary(value)
    binValue = oriDecimalToBinary(value)
    print("轉為 bin:", binValue)

    # count = binaryCountOne(binValue)
    count = oriBinaryCountOne(binValue)
    print("總共有", count ,"個 1")


if __name__ == "__main__":
    main()
```

2. 請計算 2~100 之間的所有質數並輸出

```python
import math

"""
Issue2:
1. 判斷 1 數是否為質數
2. 範圍為 2-100
"""

def isPrime(n: int) -> bool:

    if n <= 1:
        return False

    if n == 2:
        return True

    if n % 2 == 0:
        return False

    #  16 = 4, 25 = 5, 36 = 6
    sqrtN = int(math.sqrt(n))

    # O(sqrt(n)/2)
    # 只檢索奇數: 3, 5, 7
    for i in range(3, sqrtN + 1, 2):
        if n % i == 0:
            return False
    return True

print("2 到 100 所有質數：")
for num in range(2, 101):
    if isPrime(num):
        print(num, end=" ")
```

3. 設計一程式，提示使用者輸入兩個圓形的中心點(x,y)和半徑。判斷第二個圓形是在第一個圓形的內部或是兩圓重疊或是兩圓無相交

```
import math

"""
Issue3:
1. 判斷 2 圓之間 「距離」
2. 根據 Reference 的公式，判斷 2 圓之間關係
"""

# Reference: https://www.liveism.com/live-concept.php?q=%E5%85%A9%E5%9C%93%E7%9A%84%E4%BD%8D%E7%BD%AE%E9%97%9C%E4%BF%82

def distance(x1, y1, x2, y2):
    return math.sqrt((x2 - x1)**2 + (y2 - y1)**2)

def circleRelation(x1, y1, r1, x2, y2, r2):
    dist = distance(x1, y1, x2, y2)
    print("2 圓心距離：", dist)
    if r1 == r2:
        if dist == 0:
            print("重合(完全重合)")
        elif dist < r1:
            print("內離(相交於 1 點)，圓內含於另 1 圓")
        elif dist == r1:
            print("內切(相交於 1 點)，圓內含於另 1 圓")
        else:
            print("內離(不相交)，圓外含於另一圓")
    elif dist < abs(r1 - r2):
        if r1 < r2:
            print("內離(不相交)，第 1 個圓在第 2 個圓內部，2 圓重疊")
        else:
            print("內離(不相交)，第 2 個圓在第 1 個圓內部，2 圓重疊")
    elif dist < (r1 + r2):
        print("相交(相交於 2 點)，2 圓重疊")
```

```
    elif dist == abs(r1 - r2):
        print("內切(相交於 1 點)")
    elif dist == (r1 + r2):
        print("外切(相交於 1 點)")
    else:
        # dist > (r1 + r2)
        print("外離(不相交)，2 圓無相交")

# 第 1 個圓
x1 = float(input("第 1 個圓中心 x 座標："))
y1 = float(input("第 1 個圓中心 y 座標："))
r1 = float(input("第 1 個圓半徑 r："))

# 第 2 個圓
x2 = float(input("第 2 個圓中心 x 座標："))
y2 = float(input("第 2 個圓中心 y 座標："))
r2 = float(input("第 2 個圓半徑 r："))

relation = circleRelation(x1, y1, r1, x2, y2, r2)
```

4. 寫個程式其輸入為 "Let's take LeetCode contest"，其輸出為"s'teL ekat edoCteeL tsetnoc"

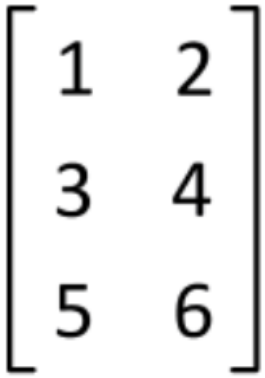
```
"""
Issue4:
1. 將字段反轉輸出
"""

def reverse_words(s: str) -> str:
    # 使用 s.split 以空白(space) 切出字段
    words = s.split()

    # 使用 list 反輸出，達到逆轉字段功用
    reversed_words = [word[::-1] for word in words]
    return ' '.join(reversed_words)

s = "Let's take LeetCode contest"
result = reverse_words(s)
print(result)
```

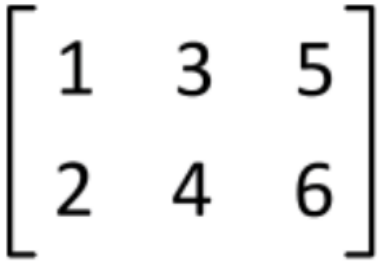
5. 有 N 個人圍成一圈並順序排號，從第一個人開始報數, 1, 2, 3 等順序，每報到 3 的人退出，最後留下的是編號幾號的人

```
"""
Issue5:
1. N 個人圍成一圈並順序排號
2. 從第 1 個人開始報數，1，2，3 等順序，每報到 3 的人退出
3. 最後留下編號的會是幾？
"""


# 約瑟夫環 (LinkList)

class Node:
    def __init__(self, value):
        self.value = value
        self.next = None

def josephus(n, step):
    head = Node(1)
    current = head
    for i in range(2, n + 1):
        current.next = Node(i)
        current = current.next

    # 閉合環
    current.next = head

    # 模擬：報數移除節點
    current = head
    while current.next != current:
        # 跳過當前節點 的 前 2 個節點
        for _ in range(step - 1):
            prev = current
            current = current.next
        # 移除當前節點
        prev.next = current.next
        current = prev.next
```

```
    return current.value

n = 10
step = 3
winner = josephus(n, step)
print("最後留下的編號是:", winner)
```

6. 寫一程式計算兩個矩陣的乘法，先輸入兩個二維矩陣 A 與 B，並逐步輸入矩陣中的所有數字，請輸出兩矩陣相乘後的結果(Ex: A 是 3x4 的矩陣, B 4x2 的矩陣, AxB = C 為 3x2 矩陣) ，再將 C 矩陣進行轉置並列印出結果 (C 矩陣為 3 x 2 矩陣，轉置後成為 2 x 3 矩陣)

(C 轉置前) 

(C 轉置後) 

```
"""
Issue6:
1. 輸入 2 個 2 維矩陣 A 與 B
2. 輸出 2 矩陣相乘後 結果(C)
3. 再將 C 矩陣 反轉 列印
"""

def matrix_multiplication(A, B):
    if len(A[0]) != len(B):
        print("無法矩陣相乘： A 矩阵的 col != B 矩阵的 row")
        return None

    result = [[0] * len(B[0]) for _ in range(len(A))]
```

```
    for i in range(len(A)):
        for j in range(len(B[0])):
            for k in range(len(B)):
                result[i][j] += A[i][k] * B[k][j]

    return result

def transpose_matrix(matrix):
    return [[matrix[j][i] for j in range(len(matrix))] for i in
range(len(matrix[0]))]

def input_matrix(rows, cols):
    matrix = []
    print(f"请輸入 {rows} x {cols} 矩阵的元素：")

    while True:
        for i in range(rows):
            row = input(f"請輸入第 {i+1} row 的元素（用空格分隔）：
").split()
            if len(row) != cols:
                print("row != cow，請重新輸入")
                matrix = []
                break
            matrix.append([int(num) for num in row])
        else:
            break
    return matrix

def print_matrix(matrix):
    for row in matrix:
        print(" ".join(map(str, row)))


# 統一使用原文，便於理解，因台灣為「直行橫列」，對岸則反過來
def main():
    rows_A = int(input("矩阵 A 的 row 數："))
    cols_A = int(input("矩阵 A 的 col 數："))
    print("矩阵 A：")
```

```
    A = input_matrix(rows_A, cols_A)

    rows_B = int(input("矩阵 B 的 row 數："))
    cols_B = int(input("矩阵 B 的 col 數："))
    print("矩阵 B：")
    B = input_matrix(rows_B, cols_B)

    if not A or not B:
        return

    print("矩阵 A：")
    print_matrix(A)
    print("矩阵 B：")
    print_matrix(B)

    result = matrix_multiplication(A, B)
    if result:
        print("AxB 矩阵结果：")
        print_matrix(result)

        transposed_result = transpose_matrix(result)
        print("轉置後：")
        print_matrix(transposed_result)

if __name__ == "__main__":
    main()
```