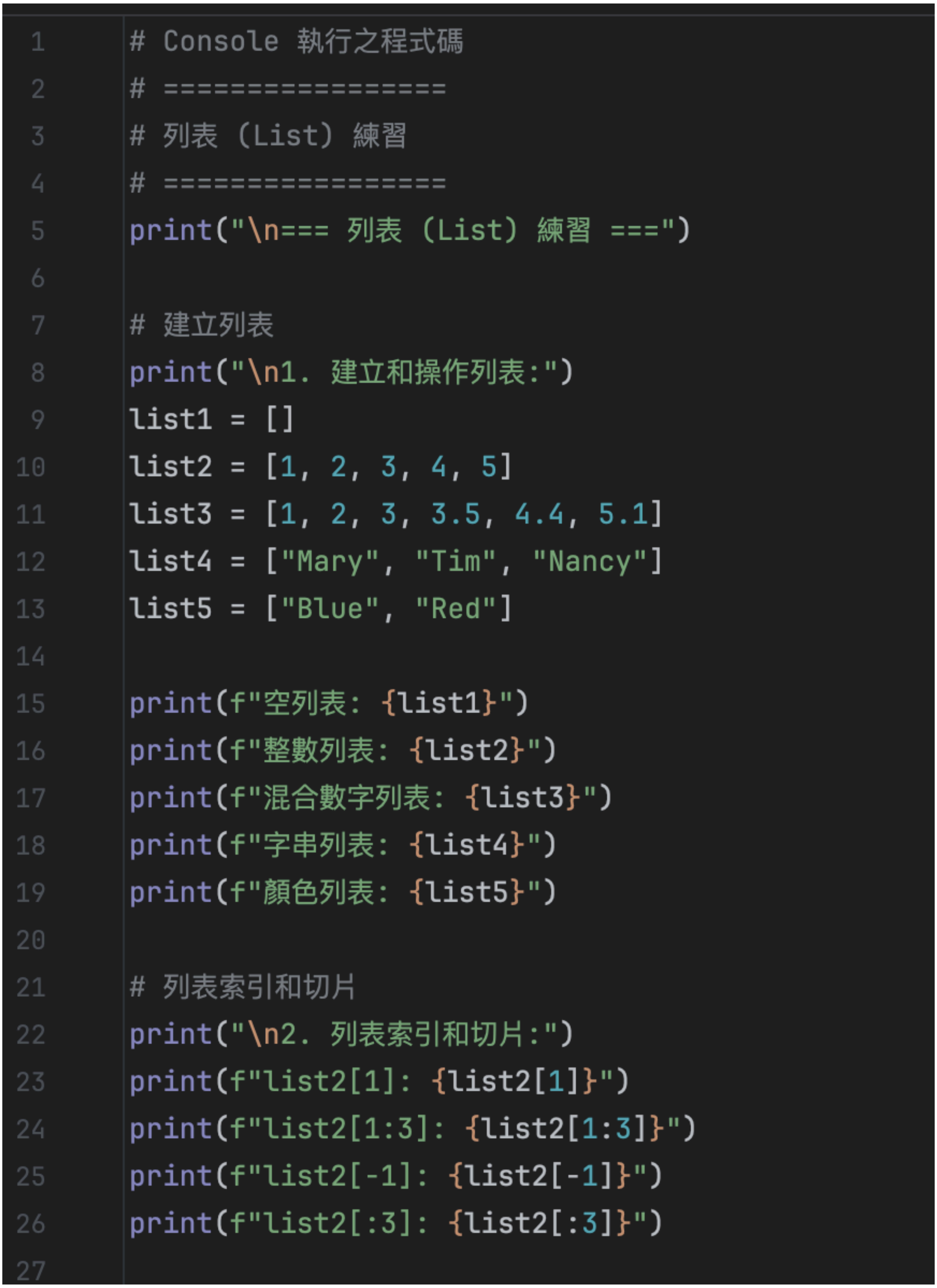
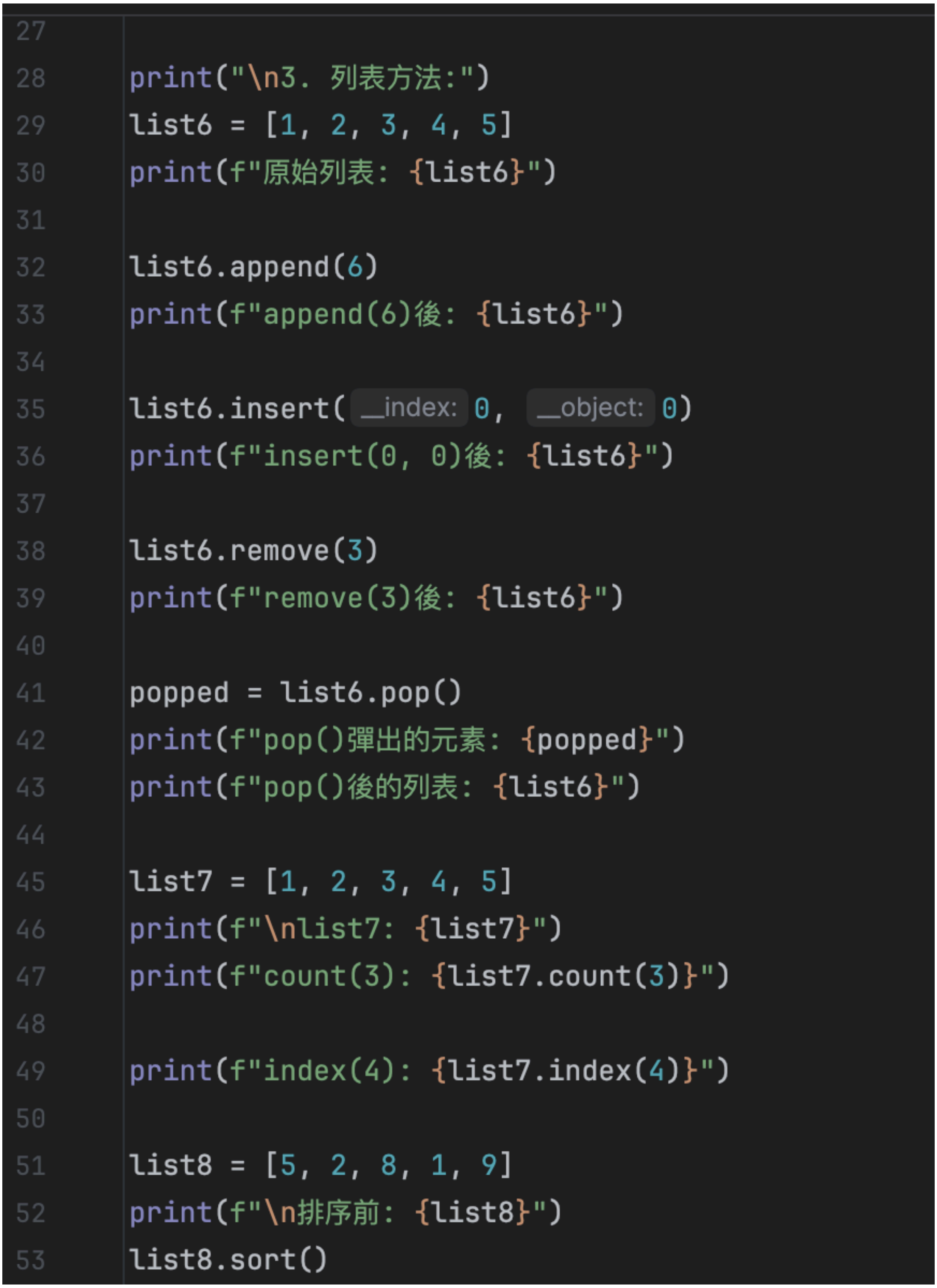
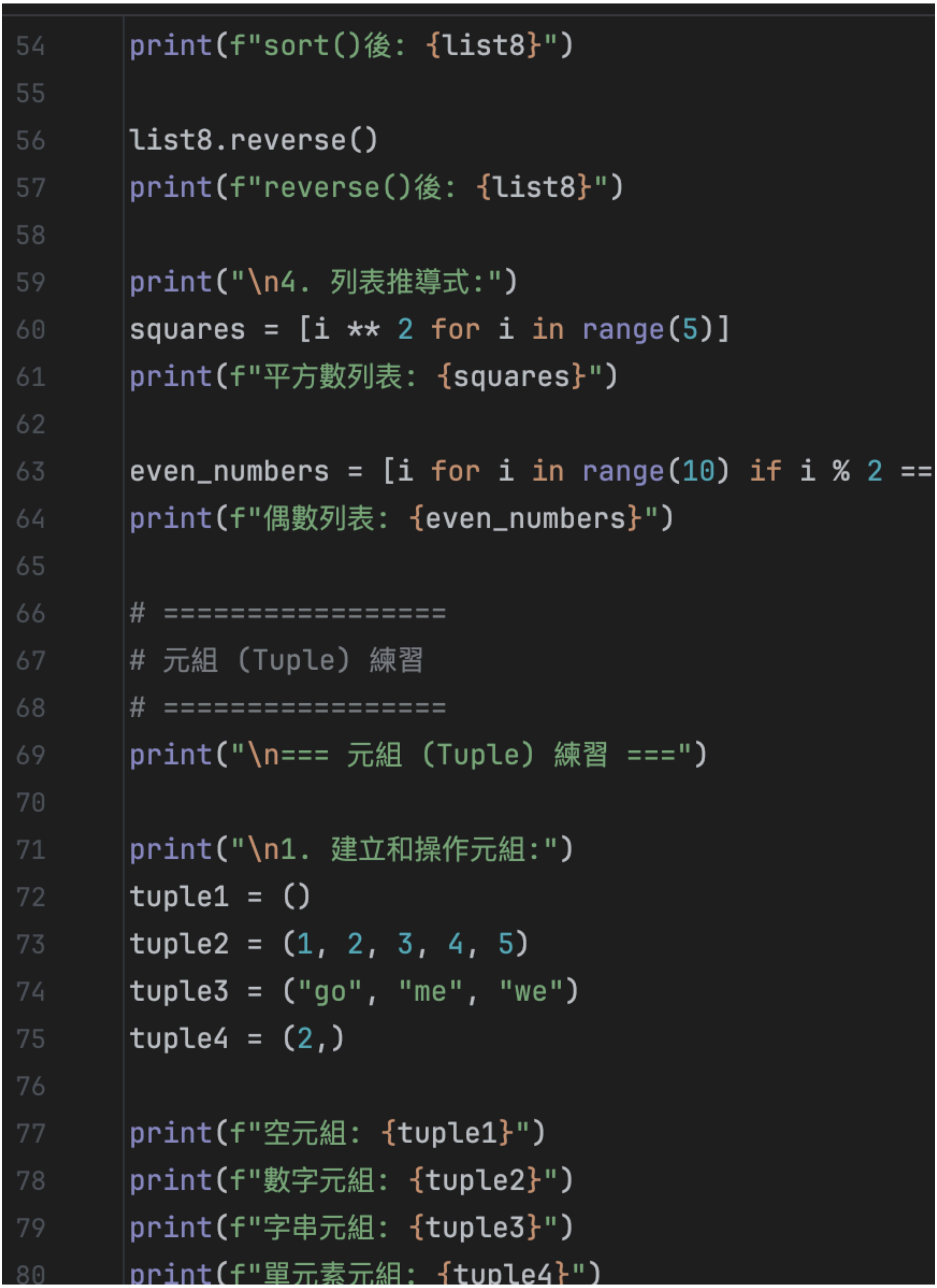


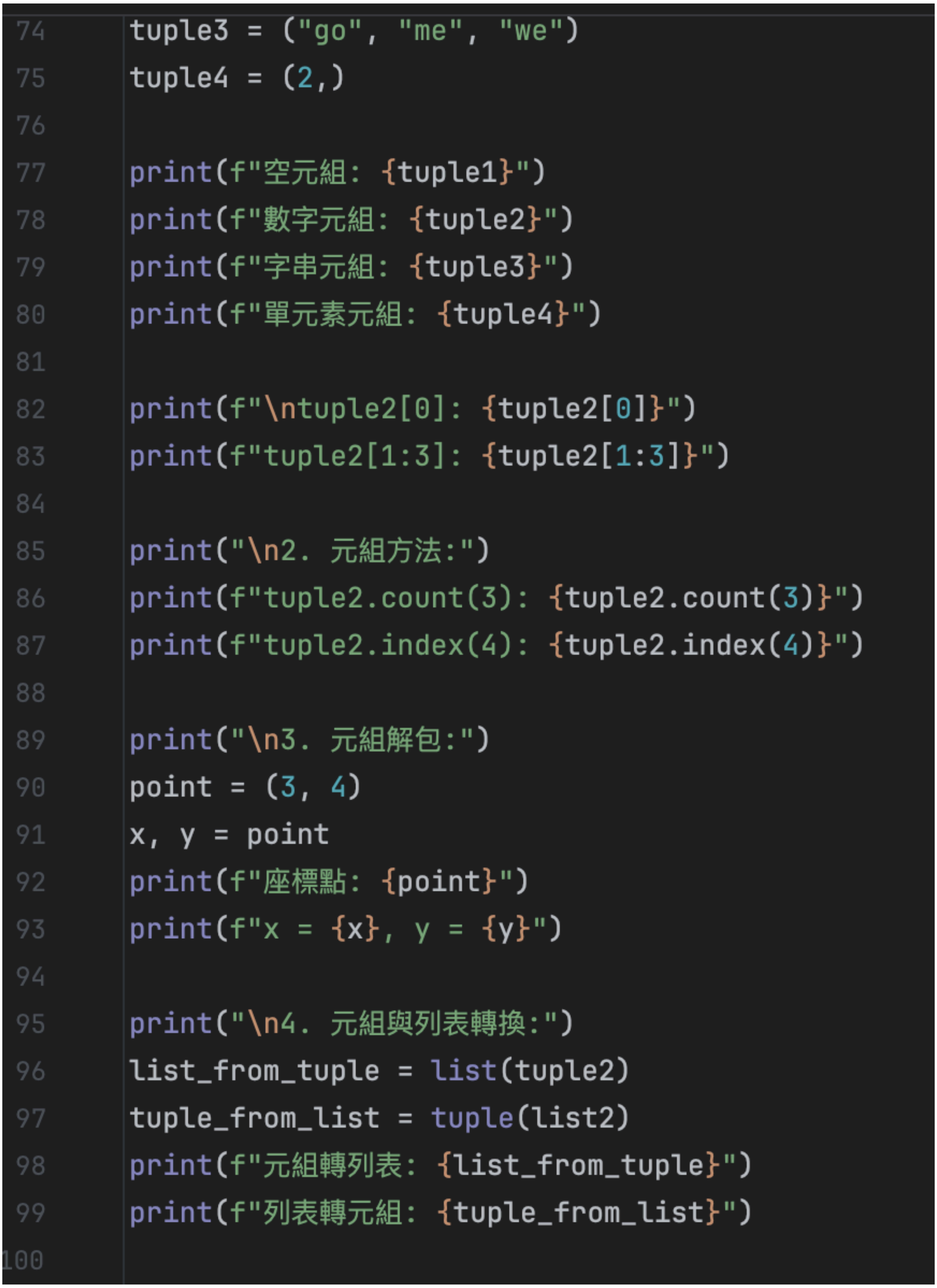
```python
# Console 執行之程式碼
# =================
# 列表 （List） 練習
# =================
print("\n=== 列表 （List） 練習 ===")

# 建立列表
print("\n1. 建立和操作列表:")
list1 = []
list2 = [1, 2, 3, 4, 5]
list3 = [1, 2, 3, 3.5, 4.4, 5.1]
list4 = ["Mary", "Tim", "Nancy"]
list5 = ["Blue", "Red"]

print(f"空列表: {list1}")
print(f"整數列表: {list2}")
print(f"混合數字列表: {list3}")
print(f"字串列表: {list4}")
print(f"顏色列表: {list5}")

# 列表索引和切片
print("\n2. 列表索引和切片:")
print(f"list2[1]: {list2[1]}")
print(f"list2[1:3]: {list2[1:3]}")
print(f"list2[-1]: {list2[-1]}")
print(f"list2[:3]: {list2[:3]}")

```



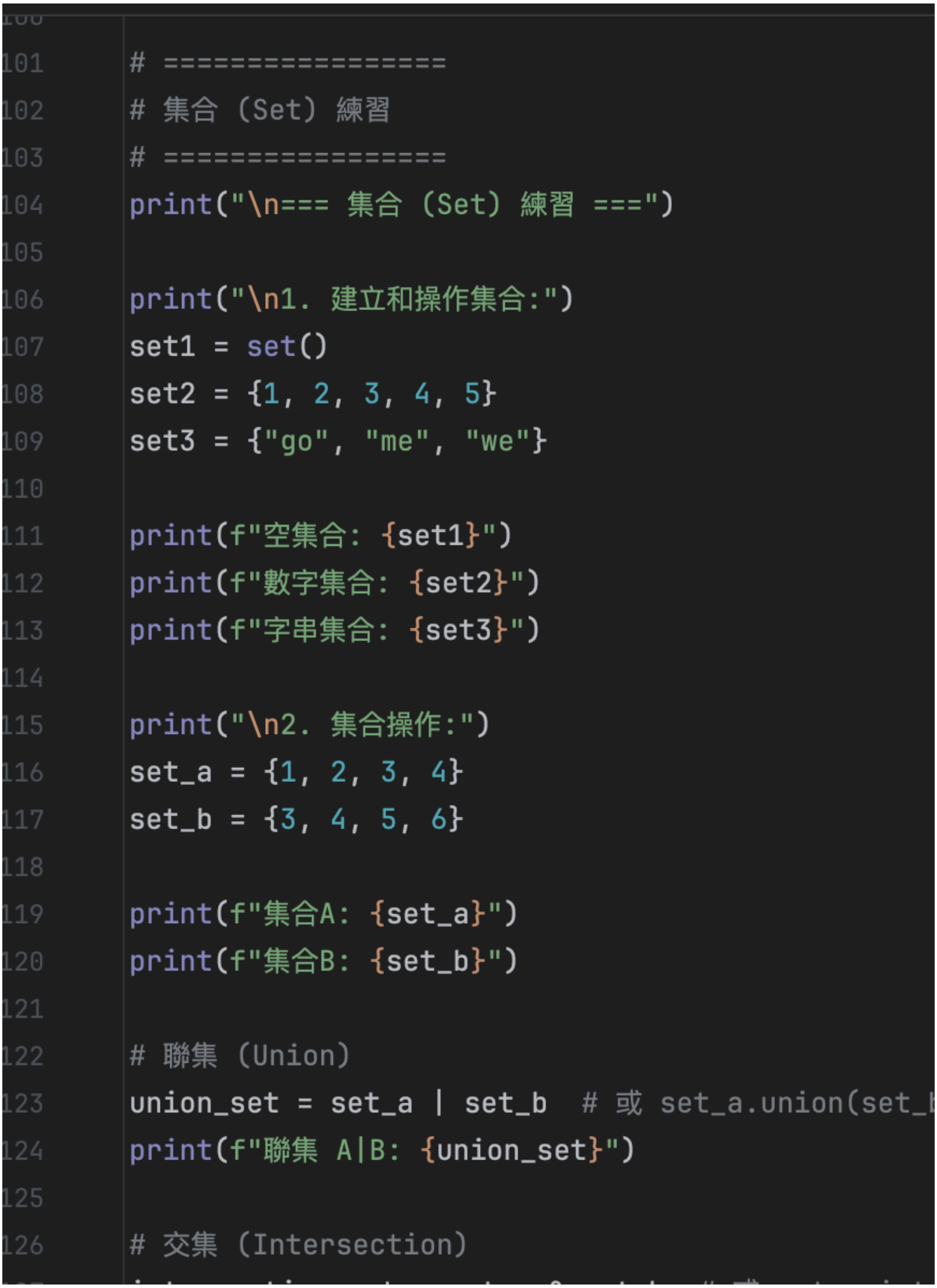
```python

print("\n3. 列表方法:")
list6 = [1, 2, 3, 4, 5]
print(f"原始列表: {list6}")

list6.append(6)
print(f"append(6)後: {list6}")

list6.insert( __index: 0,  __object: 0)
print(f"insert(0, 0)後: {list6}")

list6.remove(3)
print(f"remove(3)後: {list6}")

popped = list6.pop()
print(f"pop()彈出的元素: {popped}")
print(f"pop()後的列表: {list6}")

list7 = [1, 2, 3, 4, 5]
print(f"\nlist7: {list7}")
print(f"count(3): {list7.count(3)}")

print(f"index(4): {list7.index(4)}")

list8 = [5, 2, 8, 1, 9]
print(f"\n排序前: {list8}")
list8.sort()
```



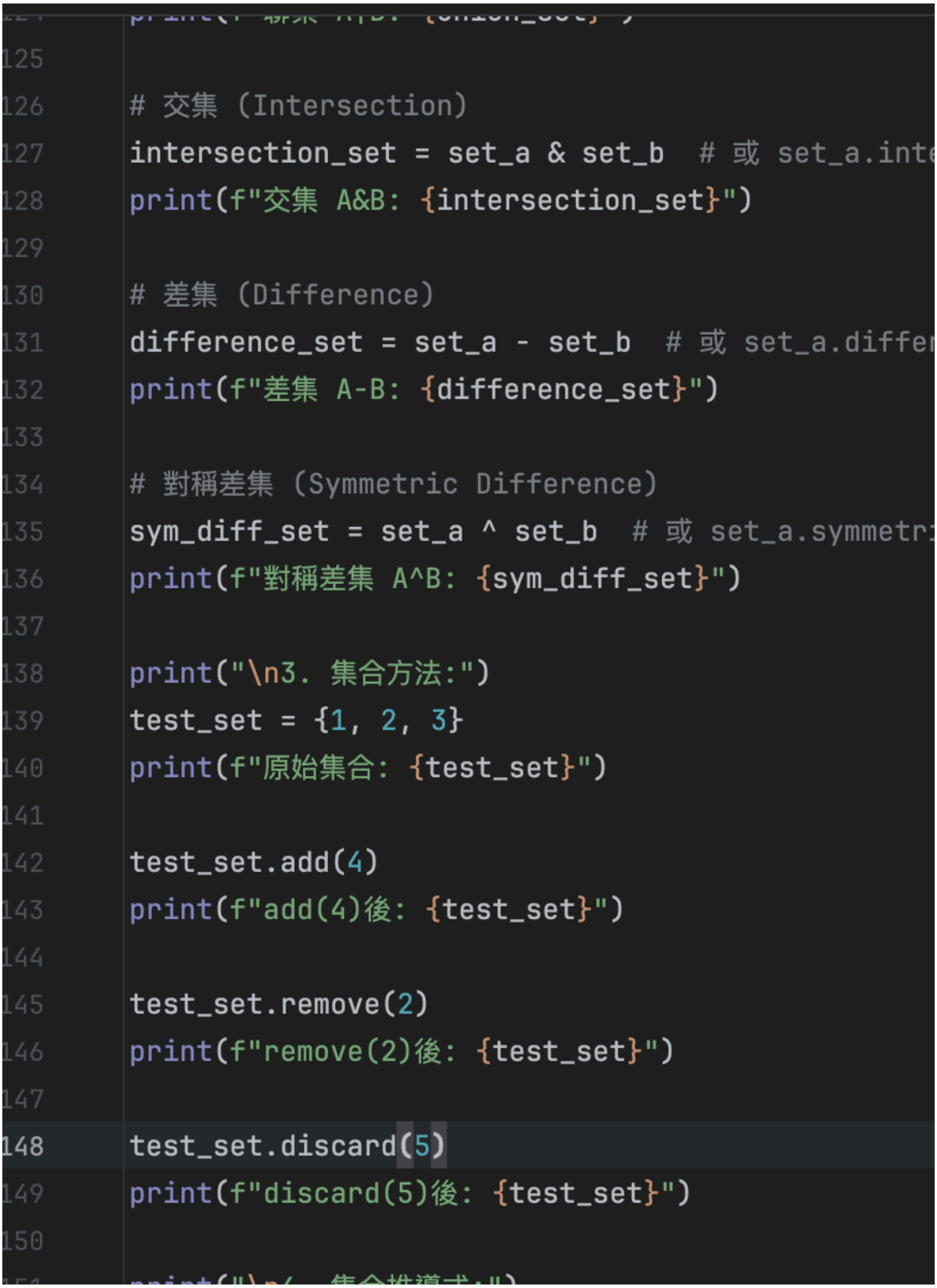
```python
print(f"sort()後: {list8}")

list8.reverse()
print(f"reverse()後: {list8}")

print("\n4. 列表推導式:")
squares = [i ** 2 for i in range(5)]
print(f"平方數列表: {squares}")

even_numbers = [i for i in range(10) if i % 2 ==
print(f"偶數列表: {even_numbers}")

# ================
# 元組 (Tuple) 練習
# ================
print("\n=== 元組 (Tuple) 練習 ===")

print("\n1. 建立和操作元組:")
tuple1 = ()
tuple2 = (1, 2, 3, 4, 5)
tuple3 = ("go", "me", "we")
tuple4 = (2,)

print(f"空元組: {tuple1}")
print(f"數字元組: {tuple2}")
print(f"字串元組: {tuple3}")
print(f"單元素元組: {tuple4}")
```



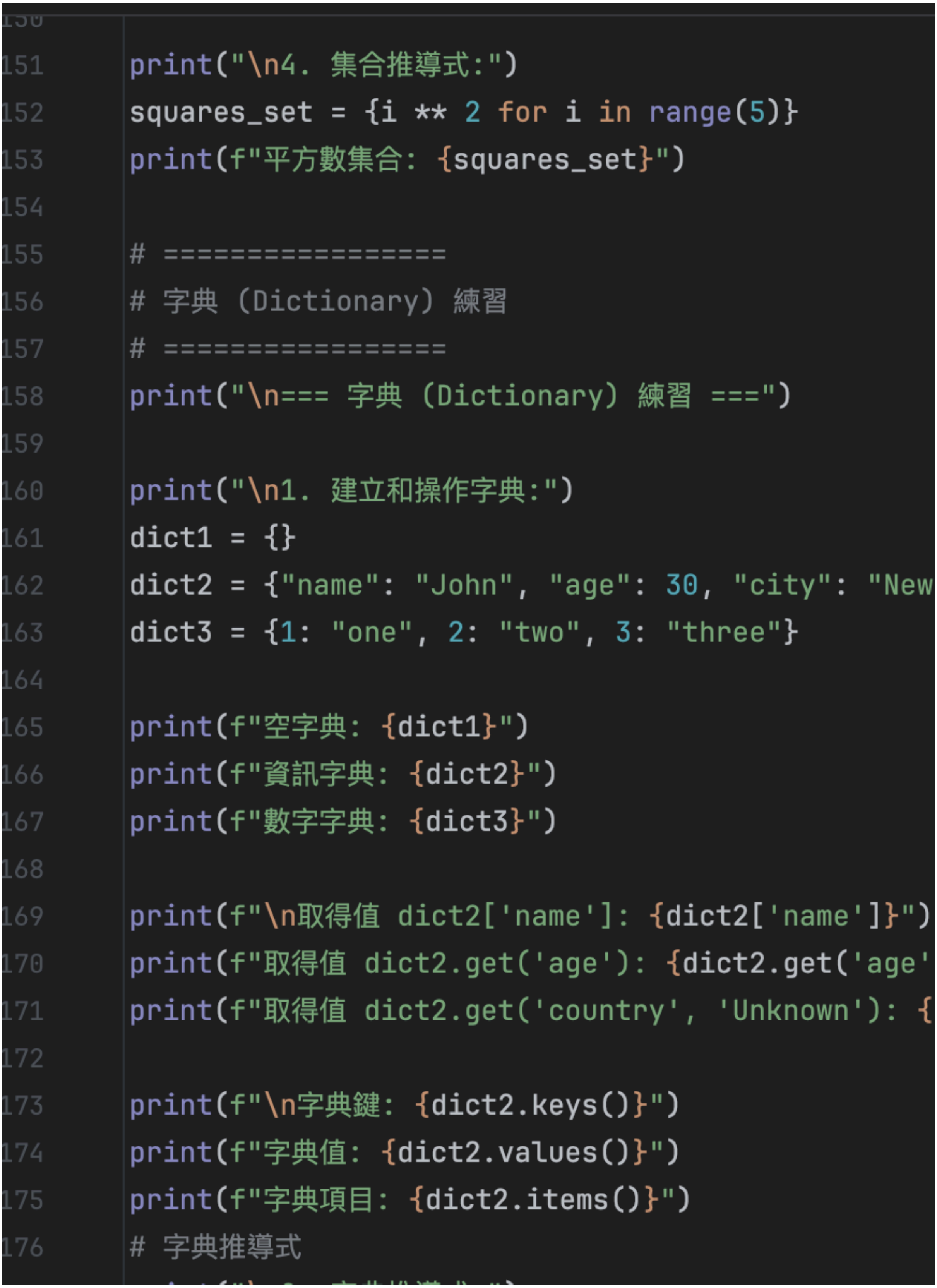
```python
tuple3 = ("go", "me", "we")
tuple4 = (2,)

print(f"空元組: {tuple1}")
print(f"數字元組: {tuple2}")
print(f"字串元組: {tuple3}")
print(f"單元素元組: {tuple4}")

print(f"\ntuple2[0]: {tuple2[0]}")
print(f"tuple2[1:3]: {tuple2[1:3]}")

print("\n2. 元組方法:")
print(f"tuple2.count(3): {tuple2.count(3)}")
print(f"tuple2.index(4): {tuple2.index(4)}")

print("\n3. 元組解包:")
point = (3, 4)
x, y = point
print(f"座標點: {point}")
print(f"x = {x}, y = {y}")

print("\n4. 元組與列表轉換:")
list_from_tuple = list(tuple2)
tuple_from_list = tuple(list2)
print(f"元組轉列表: {list_from_tuple}")
print(f"列表轉元組: {tuple_from_list}")

```



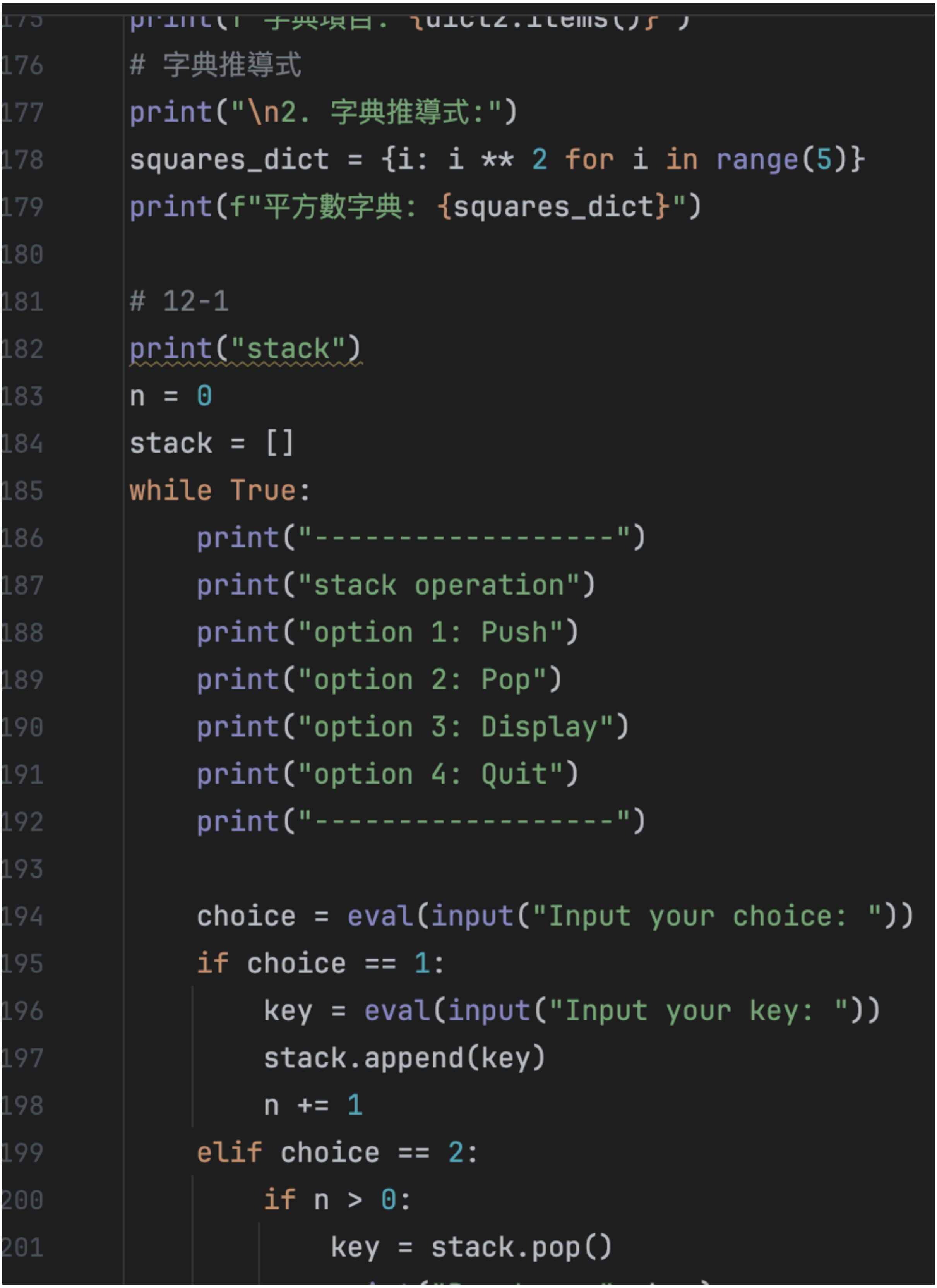
```python
# =================
# 集合 (Set) 練習
# =================
print("\n=== 集合 (Set) 練習 ===")

print("\n1. 建立和操作集合:")
set1 = set()
set2 = {1, 2, 3, 4, 5}
set3 = {"go", "me", "we"}

print(f"空集合: {set1}")
print(f"數字集合: {set2}")
print(f"字串集合: {set3}")

print("\n2. 集合操作:")
set_a = {1, 2, 3, 4}
set_b = {3, 4, 5, 6}

print(f"集合A: {set_a}")
print(f"集合B: {set_b}")

# 聯集 (Union)
union_set = set_a | set_b  # 或 set_a.union(set_
print(f"聯集 A|B: {union_set}")

# 交集 (Intersection)
```



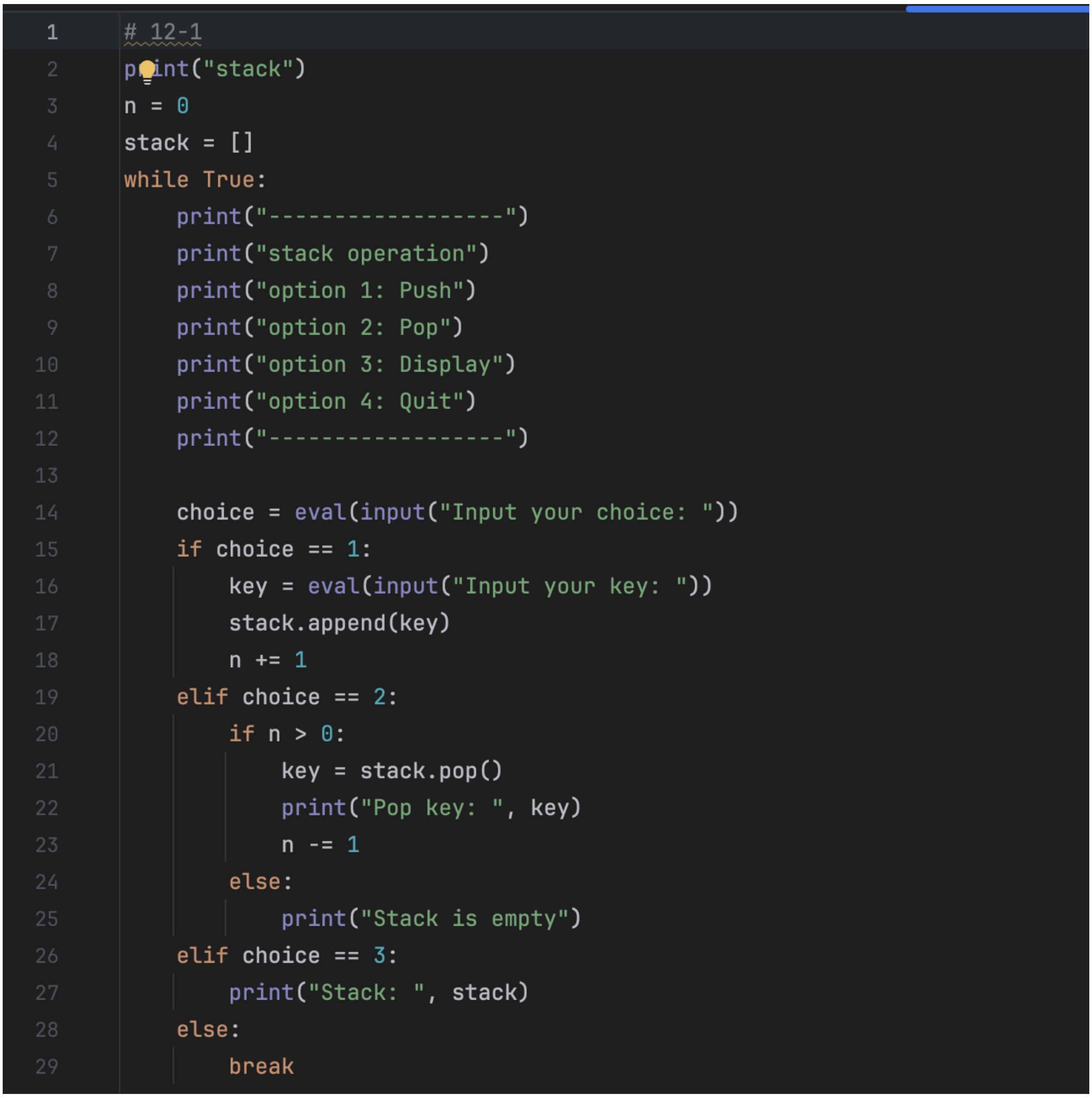
```python
print(f"聯集 A|B: {union_set}")

# 交集 (Intersection)
intersection_set = set_a & set_b  # 或 set_a.inte
print(f"交集 A&B: {intersection_set}")

# 差集 (Difference)
difference_set = set_a - set_b  # 或 set_a.differ
print(f"差集 A-B: {difference_set}")

# 對稱差集 (Symmetric Difference)
sym_diff_set = set_a ^ set_b  # 或 set_a.symmetr
print(f"對稱差集 A^B: {sym_diff_set}")

print("\n3. 集合方法:")
test_set = {1, 2, 3}
print(f"原始集合: {test_set}")

test_set.add(4)
print(f"add(4)後: {test_set}")

test_set.remove(2)
print(f"remove(2)後: {test_set}")

test_set.discard(5)
print(f"discard(5)後: {test_set}")

print("\n4. 集合推導式:")
```



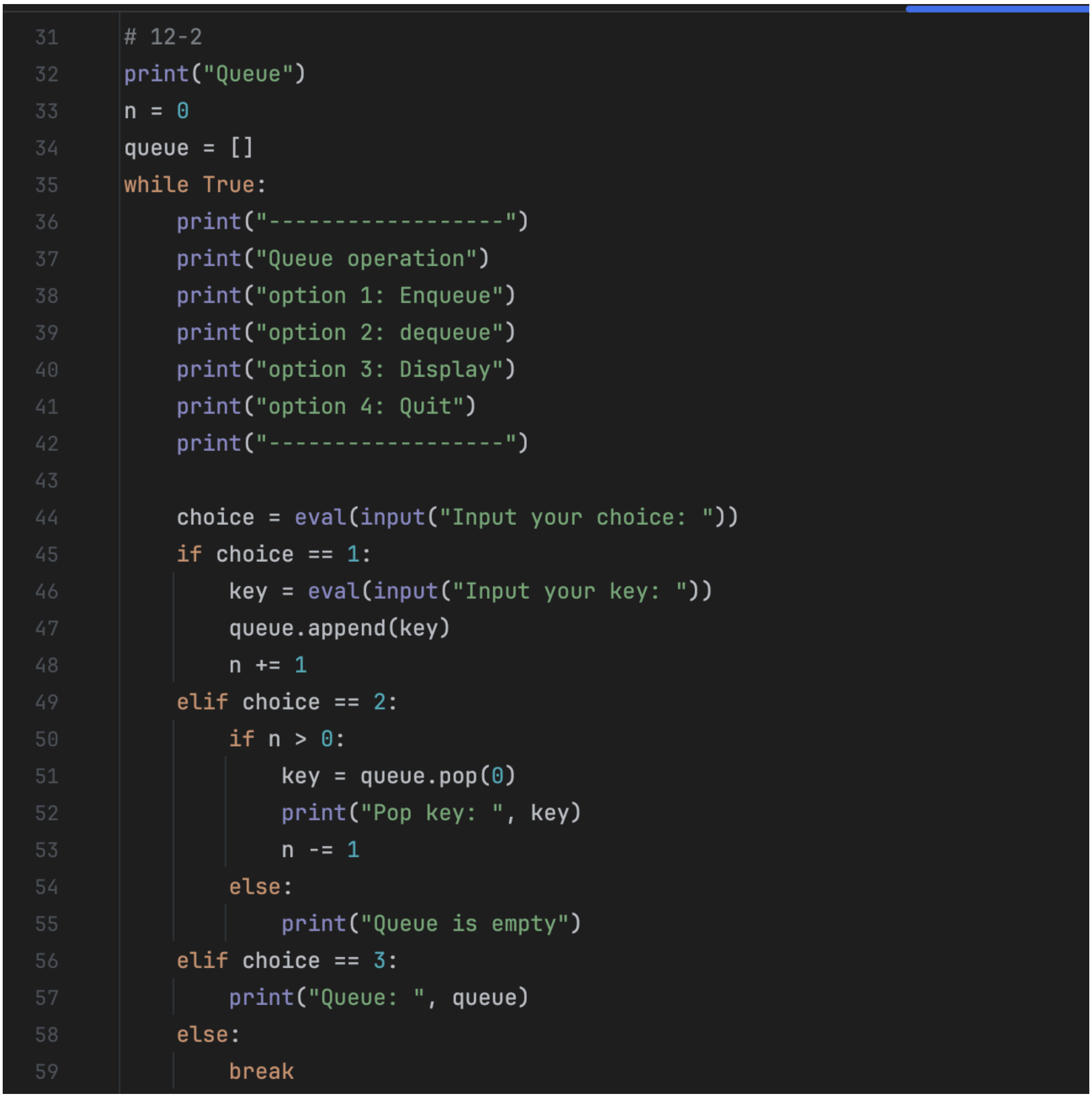
```
print("\n4. 集合推導式:")
squares_set = {i ** 2 for i in range(5)}
print(f"平方數集合: {squares_set}")

# =================
# 字典 (Dictionary) 練習
# =================
print("\n=== 字典 (Dictionary) 練習 ===")

print("\n1. 建立和操作字典:")
dict1 = {}
dict2 = {"name": "John", "age": 30, "city": "New
dict3 = {1: "one", 2: "two", 3: "three"}

print(f"空字典: {dict1}")
print(f"資訊字典: {dict2}")
print(f"數字字典: {dict3}")

print(f"\n取得值 dict2['name']: {dict2['name']}")
print(f"取得值 dict2.get('age'): {dict2.get('age'
print(f"取得值 dict2.get('country', 'Unknown'): {

print(f"\n字典鍵: {dict2.keys()}")
print(f"字典值: {dict2.values()}")
print(f"字典項目: {dict2.items()}")
# 字典推導式
```



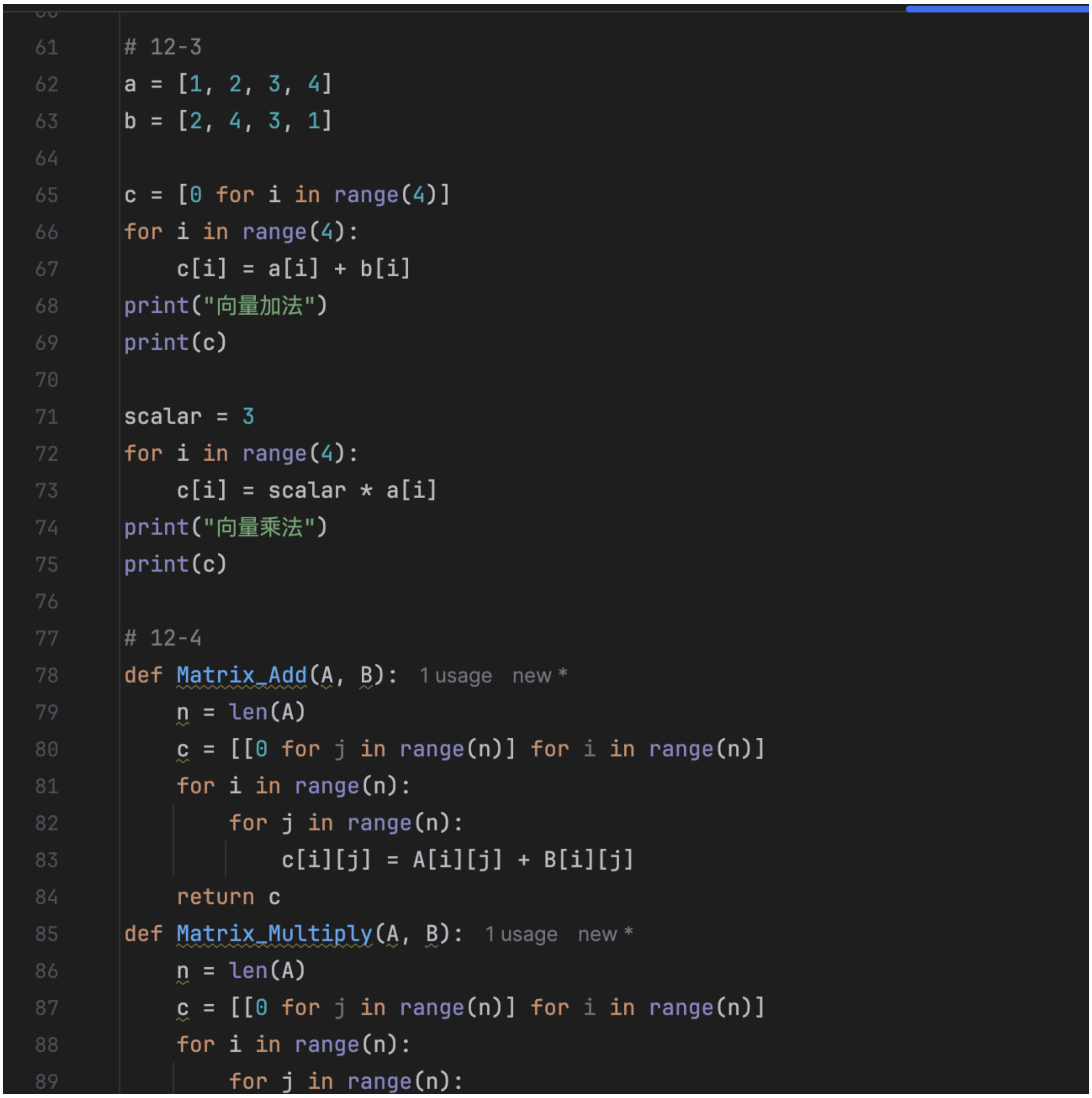
```python
print(f"字典項目: {dict2.items()}")
# 字典推導式
print("\n2. 字典推導式:")
squares_dict = {i: i ** 2 for i in range(5)}
print(f"平方數字典: {squares_dict}")

# 12-1
print("stack")
n = 0
stack = []
while True:
    print("------------------")
    print("stack operation")
    print("option 1: Push")
    print("option 2: Pop")
    print("option 3: Display")
    print("option 4: Quit")
    print("------------------")

    choice = eval(input("Input your choice: "))
    if choice == 1:
        key = eval(input("Input your key: "))
        stack.append(key)
        n += 1
    elif choice == 2:
        if n > 0:
            key = stack.pop()
```



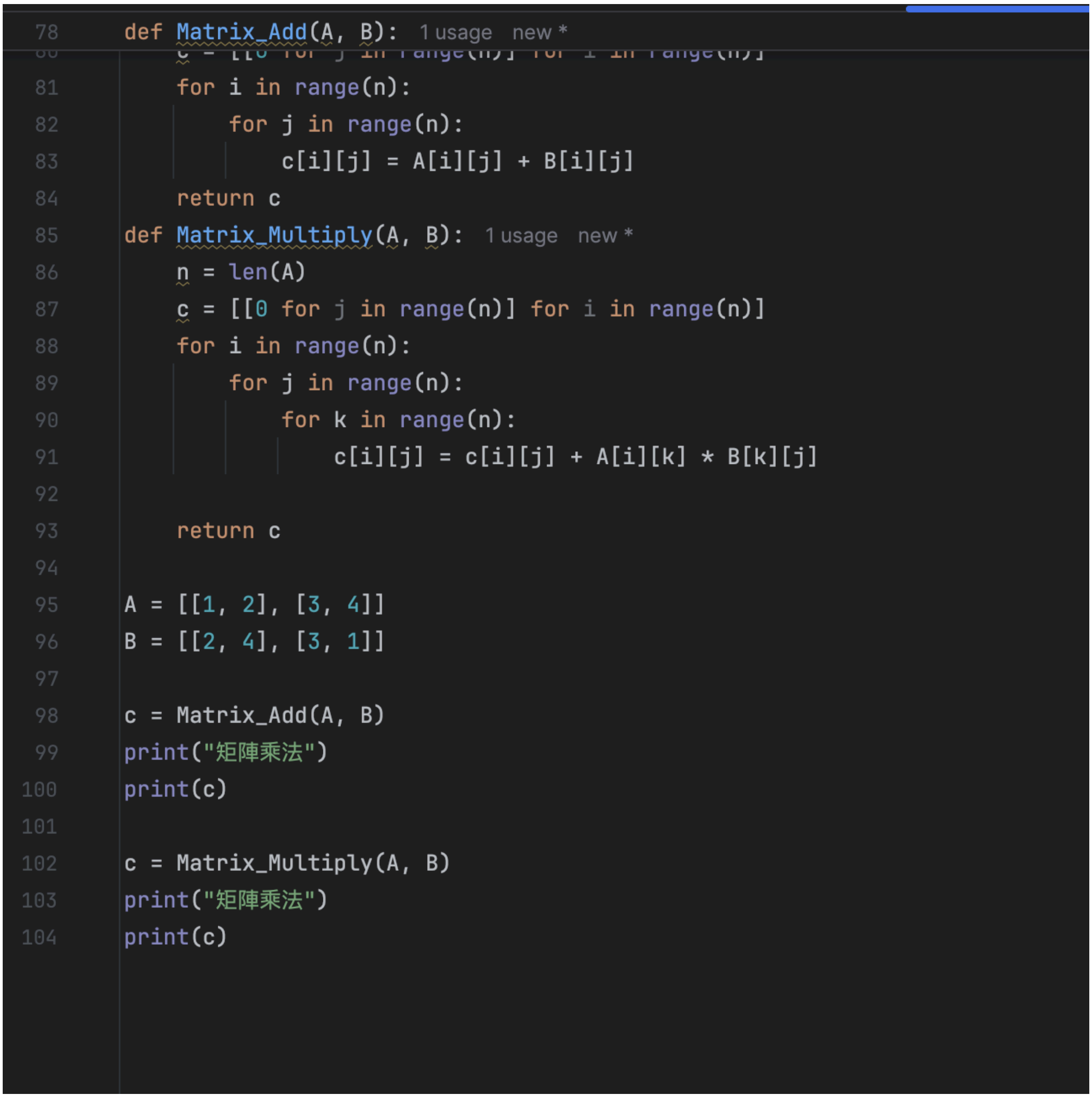
```python
# 12-1
print("stack")
n = 0
stack = []
while True:
    print("-----------------")
    print("stack operation")
    print("option 1: Push")
    print("option 2: Pop")
    print("option 3: Display")
    print("option 4: Quit")
    print("-----------------")

    choice = eval(input("Input your choice: "))
    if choice == 1:
        key = eval(input("Input your key: "))
        stack.append(key)
        n += 1
    elif choice == 2:
        if n > 0:
            key = stack.pop()
            print("Pop key: ", key)
            n -= 1
        else:
            print("Stack is empty")
    elif choice == 3:
        print("Stack: ", stack)
    else:
        break
```



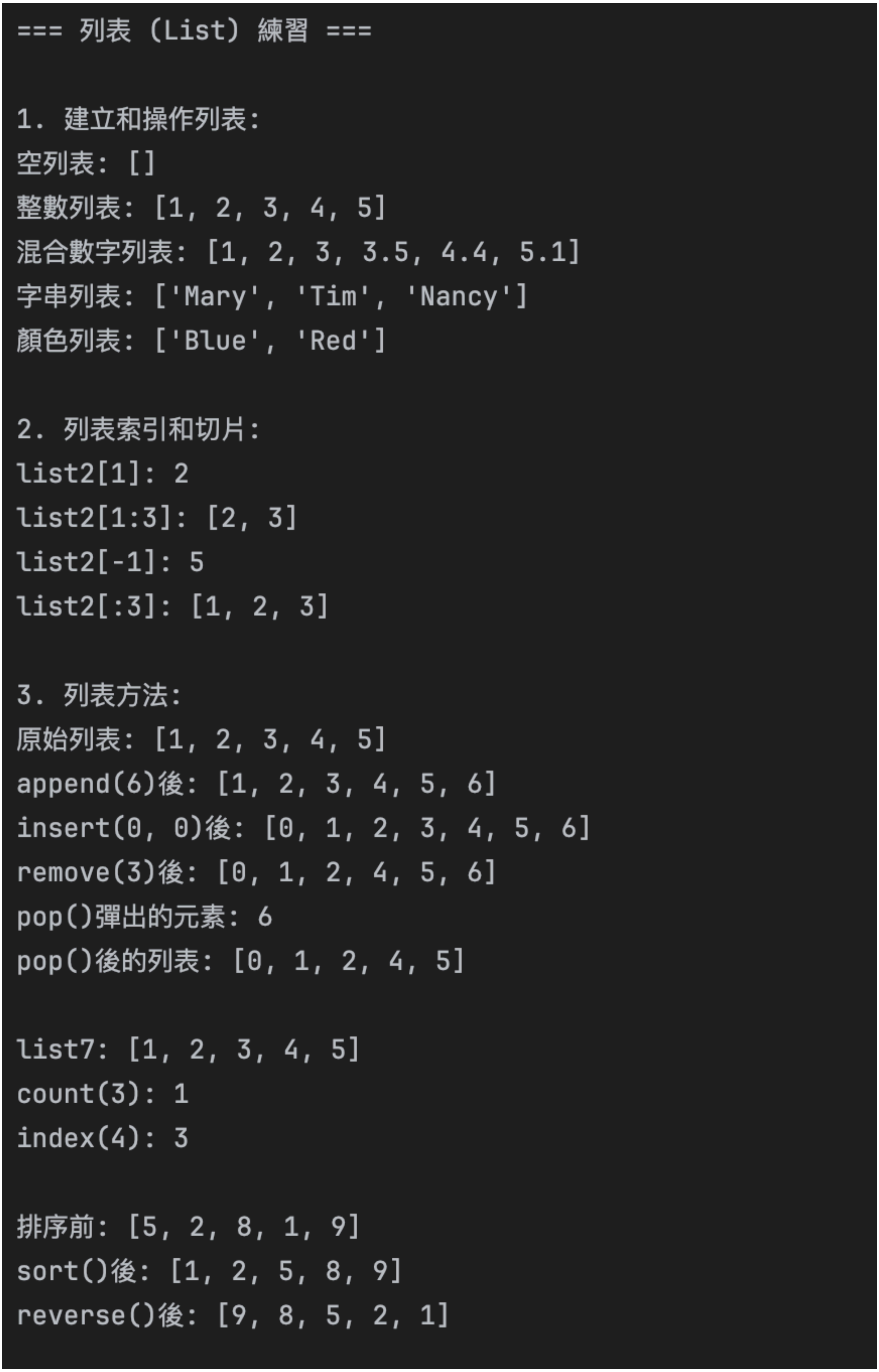
```python
# 12-2
print("Queue")
n = 0
queue = []
while True:
    print("------------------")
    print("Queue operation")
    print("option 1: Enqueue")
    print("option 2: dequeue")
    print("option 3: Display")
    print("option 4: Quit")
    print("------------------")

    choice = eval(input("Input your choice: "))
    if choice == 1:
        key = eval(input("Input your key: "))
        queue.append(key)
        n += 1
    elif choice == 2:
        if n > 0:
            key = queue.pop(0)
            print("Pop key: ", key)
            n -= 1
        else:
            print("Queue is empty")
    elif choice == 3:
        print("Queue: ", queue)
    else:
        break
```



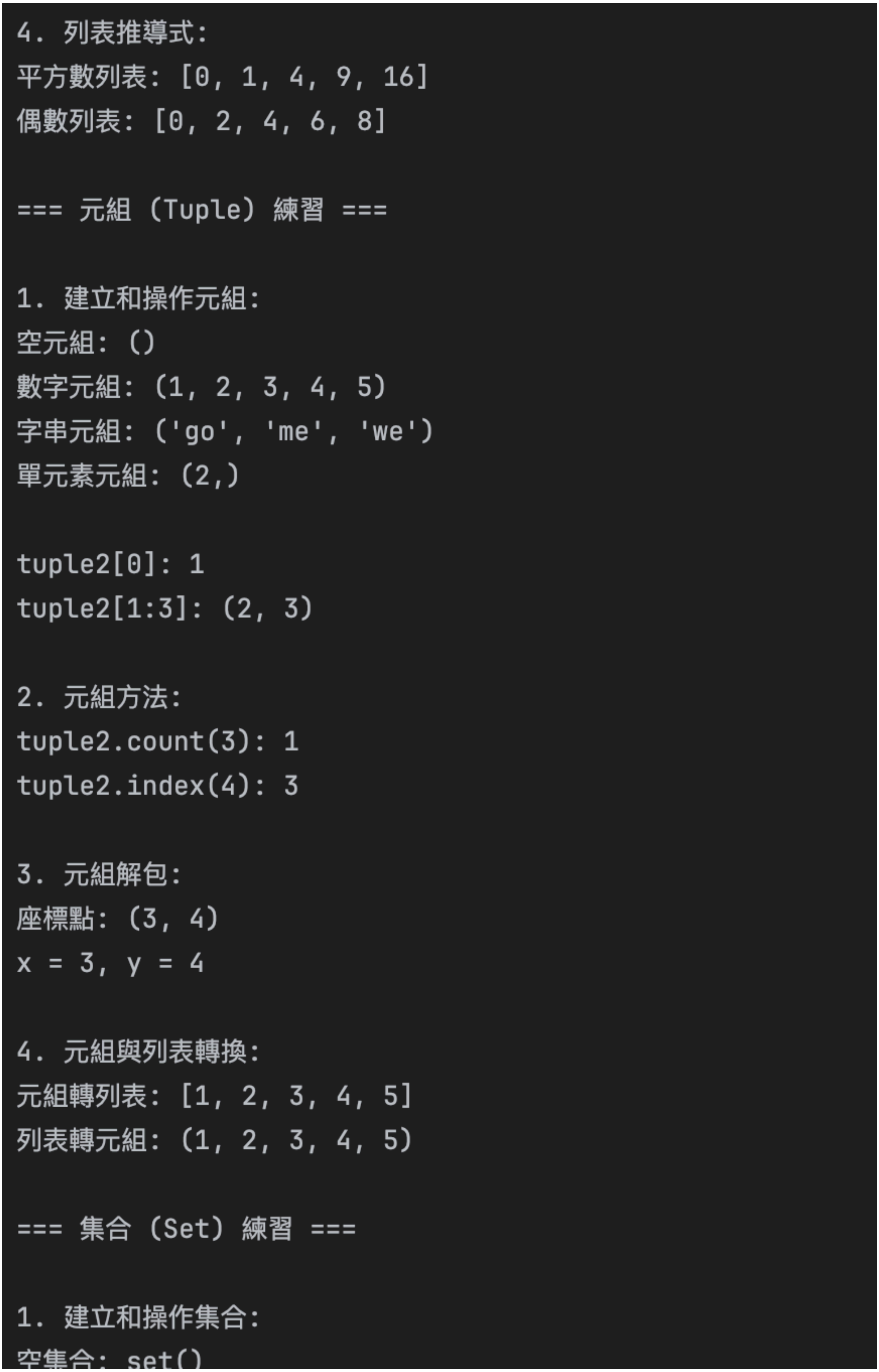
```python
# 12-3
a = [1, 2, 3, 4]
b = [2, 4, 3, 1]

c = [0 for i in range(4)]
for i in range(4):
    c[i] = a[i] + b[i]
print("向量加法")
print(c)

scalar = 3
for i in range(4):
    c[i] = scalar * a[i]
print("向量乘法")
print(c)

# 12-4
def Matrix_Add(A, B):
    n = len(A)
    c = [[0 for j in range(n)] for i in range(n)]
    for i in range(n):
        for j in range(n):
            c[i][j] = A[i][j] + B[i][j]
    return c
def Matrix_Multiply(A, B):
    n = len(A)
    c = [[0 for j in range(n)] for i in range(n)]
    for i in range(n):
        for j in range(n):
```



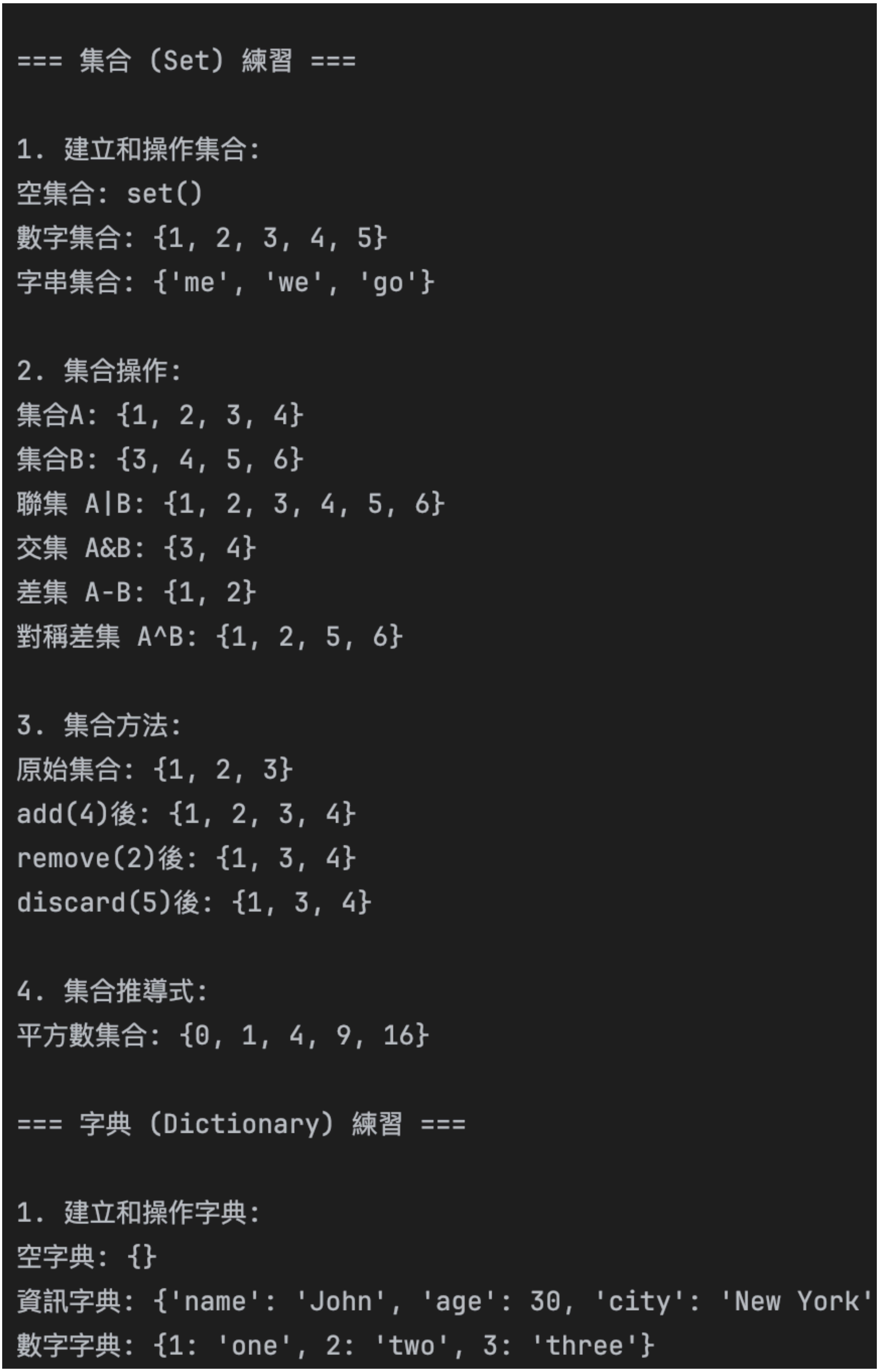
```python
def Matrix_Add(A, B):
    c = [[0 for j in range(n)] for i in range(n)]
    for i in range(n):
        for j in range(n):
            c[i][j] = A[i][j] + B[i][j]
    return c
def Matrix_Multiply(A, B):
    n = len(A)
    c = [[0 for j in range(n)] for i in range(n)]
    for i in range(n):
        for j in range(n):
            for k in range(n):
                c[i][j] = c[i][j] + A[i][k] * B[k][j]

    return c

A = [[1, 2], [3, 4]]
B = [[2, 4], [3, 1]]

c = Matrix_Add(A, B)
print("矩陣乘法")
print(c)

c = Matrix_Multiply(A, B)
print("矩陣乘法")
print(c)
```



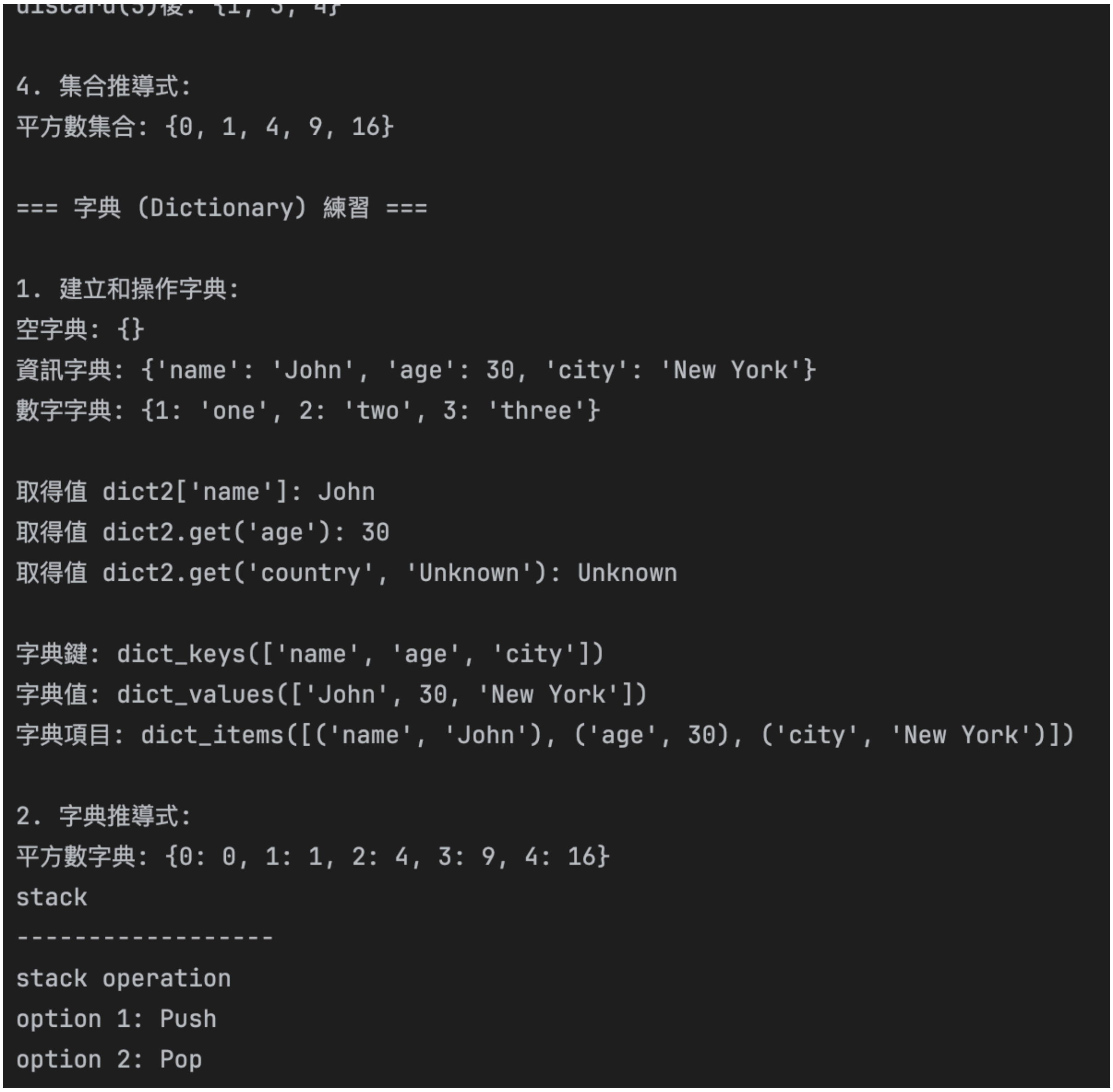
```
=== 列表 (List) 練習 ===

1. 建立和操作列表:
空列表: []
整數列表: [1, 2, 3, 4, 5]
混合數字列表: [1, 2, 3, 3.5, 4.4, 5.1]
字串列表: ['Mary', 'Tim', 'Nancy']
顏色列表: ['Blue', 'Red']

2. 列表索引和切片:
list2[1]: 2
list2[1:3]: [2, 3]
list2[-1]: 5
list2[:3]: [1, 2, 3]

3. 列表方法:
原始列表: [1, 2, 3, 4, 5]
append(6)後: [1, 2, 3, 4, 5, 6]
insert(0, 0)後: [0, 1, 2, 3, 4, 5, 6]
remove(3)後: [0, 1, 2, 4, 5, 6]
pop()彈出的元素: 6
pop()後的列表: [0, 1, 2, 4, 5]

list7: [1, 2, 3, 4, 5]
count(3): 1
index(4): 3

排序前: [5, 2, 8, 1, 9]
sort()後: [1, 2, 5, 8, 9]
reverse()後: [9, 8, 5, 2, 1]
```



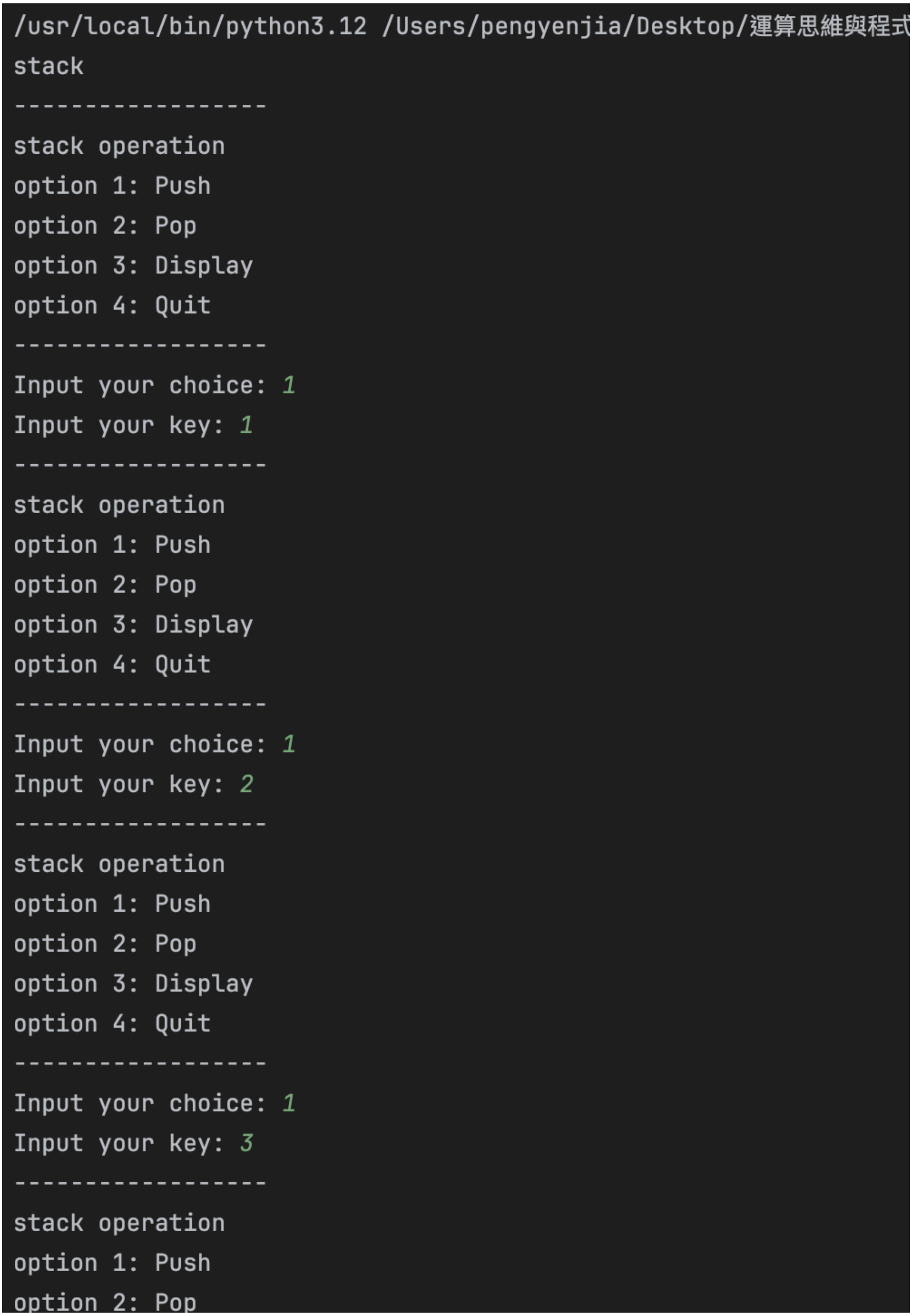
```
4. 列表推導式:
平方數列表: [0, 1, 4, 9, 16]
偶數列表: [0, 2, 4, 6, 8]

=== 元組 (Tuple) 練習 ===

1. 建立和操作元組:
空元組: ()
數字元組: (1, 2, 3, 4, 5)
字串元組: ('go', 'me', 'we')
單元素元組: (2,)

tuple2[0]: 1
tuple2[1:3]: (2, 3)

2. 元組方法:
tuple2.count(3): 1
tuple2.index(4): 3

3. 元組解包:
座標點: (3, 4)
x = 3, y = 4

4. 元組與列表轉換:
元組轉列表: [1, 2, 3, 4, 5]
列表轉元組: (1, 2, 3, 4, 5)

=== 集合 (Set) 練習 ===

1. 建立和操作集合:
空集合: set()
```



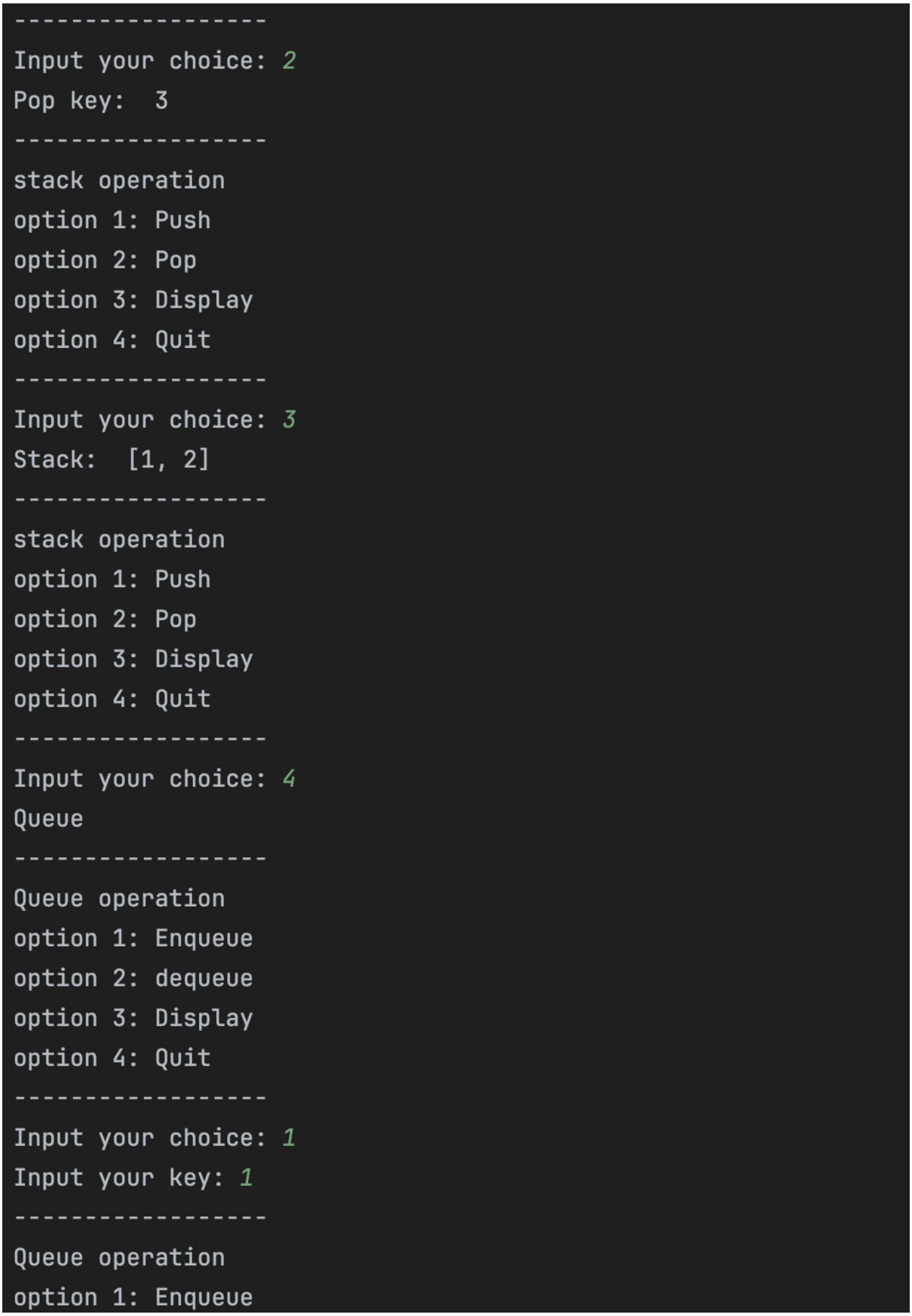
```
=== 集合 (Set) 練習 ===

1. 建立和操作集合:
空集合: set()
數字集合: {1, 2, 3, 4, 5}
字串集合: {'me', 'we', 'go'}

2. 集合操作:
集合A: {1, 2, 3, 4}
集合B: {3, 4, 5, 6}
聯集 A|B: {1, 2, 3, 4, 5, 6}
交集 A&B: {3, 4}
差集 A-B: {1, 2}
對稱差集 A^B: {1, 2, 5, 6}

3. 集合方法:
原始集合: {1, 2, 3}
add(4)後: {1, 2, 3, 4}
remove(2)後: {1, 3, 4}
discard(5)後: {1, 3, 4}

4. 集合推導式:
平方數集合: {0, 1, 4, 9, 16}

=== 字典 (Dictionary) 練習 ===

1. 建立和操作字典:
空字典: {}
資訊字典: {'name': 'John', 'age': 30, 'city': 'New York'
數字字典: {1: 'one', 2: 'two', 3: 'three'}
```



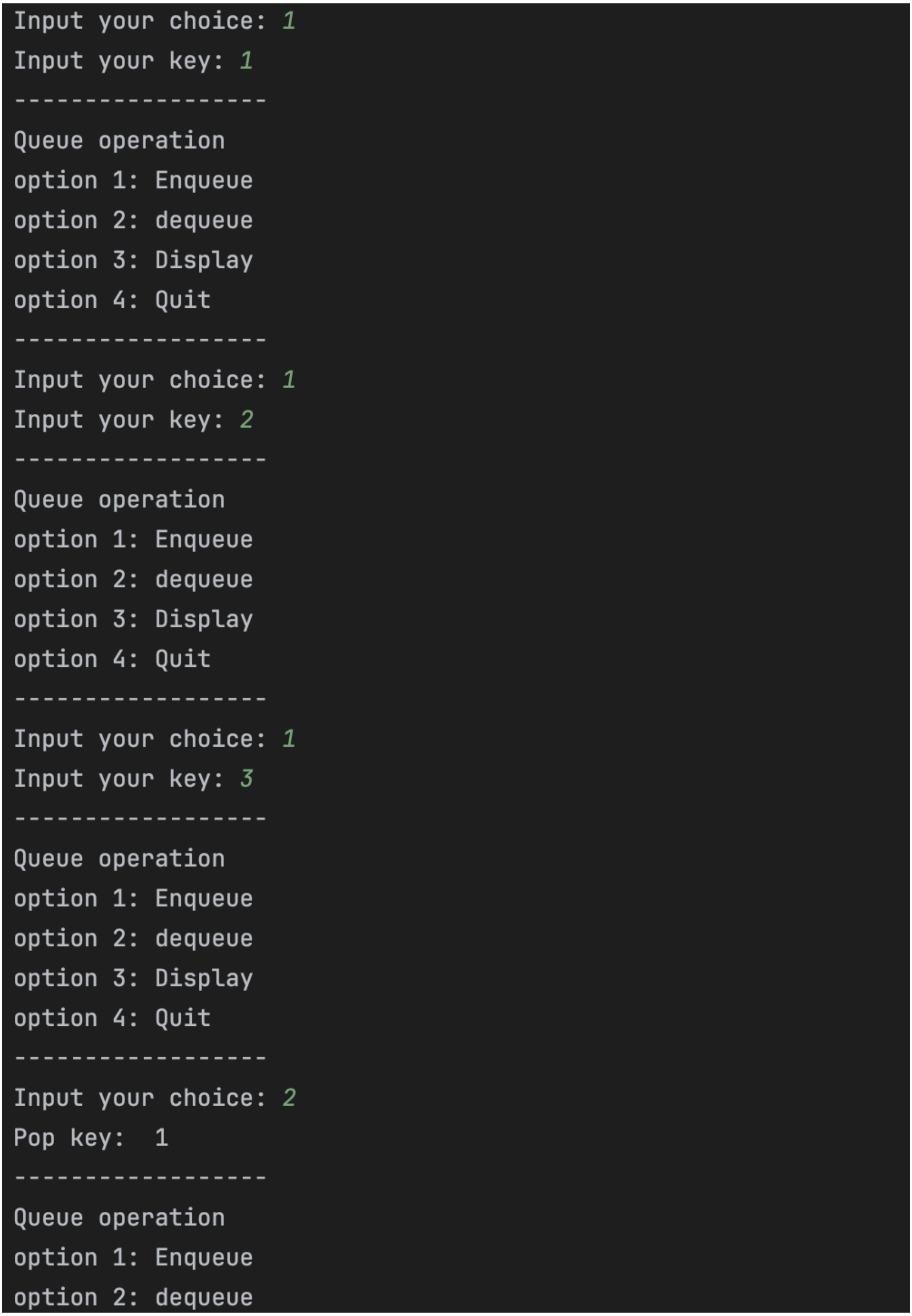
```
discard(3)後: {1, 3, 4}

4. 集合推導式:
平方數集合: {0, 1, 4, 9, 16}

=== 字典 (Dictionary) 練習 ===

1. 建立和操作字典:
空字典: {}
資訊字典: {'name': 'John', 'age': 30, 'city': 'New York'}
數字字典: {1: 'one', 2: 'two', 3: 'three'}

取得值 dict2['name']: John
取得值 dict2.get('age'): 30
取得值 dict2.get('country', 'Unknown'): Unknown

字典鍵: dict_keys(['name', 'age', 'city'])
字典值: dict_values(['John', 30, 'New York'])
字典項目: dict_items([('name', 'John'), ('age', 30), ('city', 'New York')])

2. 字典推導式:
平方數字典: {0: 0, 1: 1, 2: 4, 3: 9, 4: 16}
stack
------------------
stack operation
option 1: Push
option 2: Pop
```



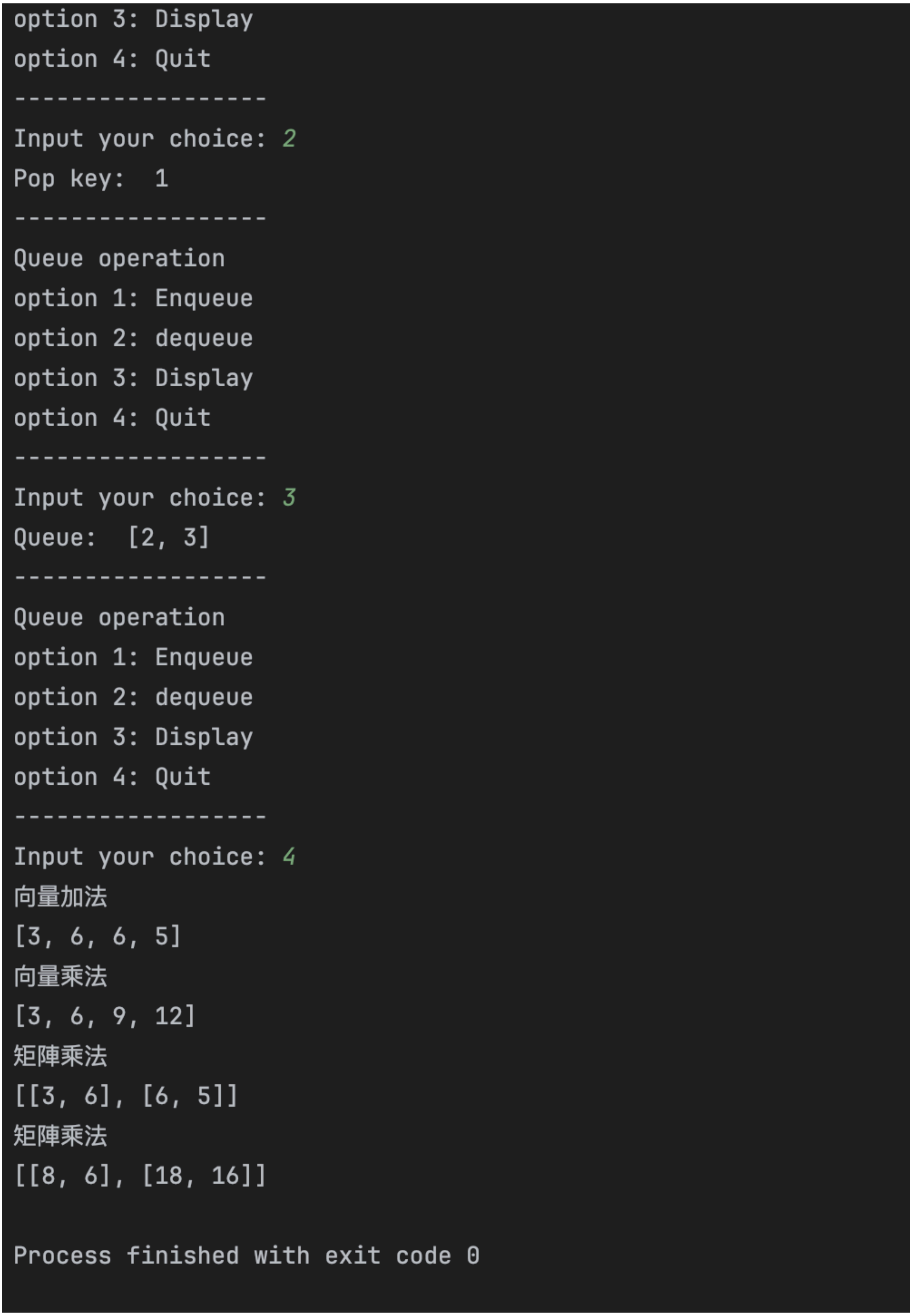
```
/usr/local/bin/python3.12 /Users/pengyenjia/Desktop/運算思維與程式
stack
------------------
stack operation
option 1: Push
option 2: Pop
option 3: Display
option 4: Quit
------------------
Input your choice: 1
Input your key: 1
------------------
stack operation
option 1: Push
option 2: Pop
option 3: Display
option 4: Quit
------------------
Input your choice: 1
Input your key: 2
------------------
stack operation
option 1: Push
option 2: Pop
option 3: Display
option 4: Quit
------------------
Input your choice: 1
Input your key: 3
------------------
stack operation
option 1: Push
option 2: Pop
```



```
------------------
Input your choice: 2
Pop key:  3
------------------
stack operation
option 1: Push
option 2: Pop
option 3: Display
option 4: Quit
------------------
Input your choice: 3
Stack:  [1, 2]
------------------
stack operation
option 1: Push
option 2: Pop
option 3: Display
option 4: Quit
------------------
Input your choice: 4
Queue
------------------
Queue operation
option 1: Enqueue
option 2: dequeue
option 3: Display
option 4: Quit
------------------
Input your choice: 1
Input your key: 1
------------------
Queue operation
option 1: Enqueue
```



```
Input your choice: 1
Input your key: 1
------------------
Queue operation
option 1: Enqueue
option 2: dequeue
option 3: Display
option 4: Quit
------------------
Input your choice: 1
Input your key: 2
------------------
Queue operation
option 1: Enqueue
option 2: dequeue
option 3: Display
option 4: Quit
------------------
Input your choice: 1
Input your key: 3
------------------
Queue operation
option 1: Enqueue
option 2: dequeue
option 3: Display
option 4: Quit
------------------
Input your choice: 2
Pop key:  1
------------------
Queue operation
option 1: Enqueue
option 2: dequeue
```



```
option 3: Display
option 4: Quit
------------------
Input your choice: 2
Pop key:  1
------------------
Queue operation
option 1: Enqueue
option 2: dequeue
option 3: Display
option 4: Quit
------------------
Input your choice: 3
Queue:  [2, 3]
------------------
Queue operation
option 1: Enqueue
option 2: dequeue
option 3: Display
option 4: Quit
------------------
Input your choice: 4
向量加法
[3, 6, 6, 5]
向量乘法
[3, 6, 9, 12]
矩陣乘法
[[3, 6], [6, 5]]
矩陣乘法
[[8, 6], [18, 16]]

Process finished with exit code 0
```