

```
1 # Console 執行之程式碼
2 # =====
3 # 列表 (List) 練習
4 # =====
5 print("\n=== 列表 (List) 練習 ===")
6
7 # 建立列表
8 print("\n1. 建立和操作列表:")
9 list1 = []
10 list2 = [1, 2, 3, 4, 5]
11 list3 = [1, 2, 3, 3.5, 4.4, 5.1]
12 list4 = ["Mary", "Tim", "Nancy"]
13 list5 = ["Blue", "Red"]
14
15 print(f"空列表: {list1}")
16 print(f"整數列表: {list2}")
17 print(f"混合數字列表: {list3}")
18 print(f"字串列表: {list4}")
19 print(f"顏色列表: {list5}")
20
21 # 列表索引和切片
22 print("\n2. 列表索引和切片:")
23 print(f"list2[1]: {list2[1]}")
24 print(f"list2[1:3]: {list2[1:3]}")
25 print(f"list2[-1]: {list2[-1]}")
26 print(f"list2[:3]: {list2[:3]}")
27
```

```
27
28 print("\n3. 列表方法:")
29 list6 = [1, 2, 3, 4, 5]
30 print(f"原始列表: {list6}")
31
32 list6.append(6)
33 print(f"append(6)後: {list6}")
34
35 list6.insert(__index: 0, __object: 0)
36 print(f"insert(0, 0)後: {list6}")
37
38 list6.remove(3)
39 print(f"remove(3)後: {list6}")
40
41 popped = list6.pop()
42 print(f"pop()彈出的元素: {popped}")
43 print(f"pop()後的列表: {list6}")
44
45 list7 = [1, 2, 3, 4, 5]
46 print(f"\nlist7: {list7}")
47 print(f"count(3): {list7.count(3)}")
48
49 print(f"index(4): {list7.index(4)}")
50
51 list8 = [5, 2, 8, 1, 9]
52 print(f"\n排序前: {list8}")
53 list8.sort()
```

```
54 print(f"sort()後: {list8}")
55
56 list8.reverse()
57 print(f"reverse()後: {list8}")
58
59 print("\n4. 列表推導式:")
60 squares = [i ** 2 for i in range(5)]
61 print(f"平方數列表: {squares}")
62
63 even_numbers = [i for i in range(10) if i % 2 == 0]
64 print(f"偶數列表: {even_numbers}")
65
66 # =====
67 # 元組 (Tuple) 練習
68 # =====
69 print("\n=== 元組 (Tuple) 練習 ===")
70
71 print("\n1. 建立和操作元組:")
72 tuple1 = ()
73 tuple2 = (1, 2, 3, 4, 5)
74 tuple3 = ("go", "me", "we")
75 tuple4 = (2,)
76
77 print(f"空元組: {tuple1}")
78 print(f"數字元組: {tuple2}")
79 print(f"字串元組: {tuple3}")
80 print(f"單元素元組: {tuple4}")
```

```
74 tuple3 = ("go", "me", "we")
75 tuple4 = (2,)
76
77 print(f"空元組: {tuple1}")
78 print(f"數字元組: {tuple2}")
79 print(f"字串元組: {tuple3}")
80 print(f"單元素元組: {tuple4}")
81
82 print(f"\ntuple2[0]: {tuple2[0]}")
83 print(f"tuple2[1:3]: {tuple2[1:3]}")
84
85 print("\n2. 元組方法:")
86 print(f"tuple2.count(3): {tuple2.count(3)}")
87 print(f"tuple2.index(4): {tuple2.index(4)}")
88
89 print("\n3. 元組解包:")
90 point = (3, 4)
91 x, y = point
92 print(f"座標點: {point}")
93 print(f"x = {x}, y = {y}")
94
95 print("\n4. 元組與列表轉換:")
96 list_from_tuple = list(tuple2)
97 tuple_from_list = tuple(list2)
98 print(f"元組轉列表: {list_from_tuple}")
99 print(f"列表轉元組: {tuple_from_list}")
100
```

```
100
101 # =====
102 # 集合 (Set) 練習
103 # =====
104 print("\n=== 集合 (Set) 練習 ===")
105
106 print("\n1. 建立和操作集合:")
107 set1 = set()
108 set2 = {1, 2, 3, 4, 5}
109 set3 = {"go", "me", "we"}
110
111 print(f"空集合: {set1}")
112 print(f"數字集合: {set2}")
113 print(f"字串集合: {set3}")
114
115 print("\n2. 集合操作:")
116 set_a = {1, 2, 3, 4}
117 set_b = {3, 4, 5, 6}
118
119 print(f"集合A: {set_a}")
120 print(f"集合B: {set_b}")
121
122 # 聯集 (Union)
123 union_set = set_a | set_b # 或 set_a.union(set_b)
124 print(f"聯集 A|B: {union_set}")
125
126 # 交集 (Intersection)
```

```

124 print(f"交集 A&B: {intersection_set}")
125
126 # 交集 (Intersection)
127 intersection_set = set_a & set_b # 或 set_a.intersection(set_b)
128 print(f"交集 A&B: {intersection_set}")
129
130 # 差集 (Difference)
131 difference_set = set_a - set_b # 或 set_a.difference(set_b)
132 print(f"差集 A-B: {difference_set}")
133
134 # 對稱差集 (Symmetric Difference)
135 sym_diff_set = set_a ^ set_b # 或 set_a.symmetric_difference(set_b)
136 print(f"對稱差集 A^B: {sym_diff_set}")
137
138 print("\n3. 集合方法:")
139 test_set = {1, 2, 3}
140 print(f"原始集合: {test_set}")
141
142 test_set.add(4)
143 print(f"add(4)後: {test_set}")
144
145 test_set.remove(2)
146 print(f"remove(2)後: {test_set}")
147
148 test_set.discard(5)
149 print(f"discard(5)後: {test_set}")
150
151 print("\n4. 集合推導式:")

```

```
150
151 print("\n4. 集合推導式:")
152 squares_set = {i ** 2 for i in range(5)}
153 print(f"平方數集合: {squares_set}")
154
155 # =====
156 # 字典 (Dictionary) 練習
157 # =====
158 print("\n=== 字典 (Dictionary) 練習 ===")
159
160 print("\n1. 建立和操作字典:")
161 dict1 = {}
162 dict2 = {"name": "John", "age": 30, "city": "New"}
163 dict3 = {1: "one", 2: "two", 3: "three"}
164
165 print(f"空字典: {dict1}")
166 print(f"資訊字典: {dict2}")
167 print(f"數字字典: {dict3}")
168
169 print(f"\n取得值 dict2['name']: {dict2['name']}")
170 print(f"取得值 dict2.get('age'): {dict2.get('age')}")
171 print(f"取得值 dict2.get('country', 'Unknown'): {dict2.get('country', 'Unknown')}")
172
173 print(f"\n字典鍵: {dict2.keys()}")
174 print(f"字典值: {dict2.values()}")
175 print(f"字典項目: {dict2.items()}")
176 # 字典推導式
```

```
175 print(f"字典項目: {dict2.items()}")
176 # 字典推導式
177 print("\n2. 字典推導式:")
178 squares_dict = {i: i ** 2 for i in range(5)}
179 print(f"平方數字典: {squares_dict}")
180
181 # 12-1
182 print("stack")
183 n = 0
184 stack = []
185 while True:
186     print("-----")
187     print("stack operation")
188     print("option 1: Push")
189     print("option 2: Pop")
190     print("option 3: Display")
191     print("option 4: Quit")
192     print("-----")
193
194     choice = eval(input("Input your choice: "))
195     if choice == 1:
196         key = eval(input("Input your key: "))
197         stack.append(key)
198         n += 1
199     elif choice == 2:
200         if n > 0:
201             key = stack.pop()
```



```
1  # 12-1
2  print("stack")
3  n = 0
4  stack = []
5  while True:
6      print("-----")
7      print("stack operation")
8      print("option 1: Push")
9      print("option 2: Pop")
10     print("option 3: Display")
11     print("option 4: Quit")
12     print("-----")
13
14     choice = eval(input("Input your choice: "))
15     if choice == 1:
16         key = eval(input("Input your key: "))
17         stack.append(key)
18         n += 1
19     elif choice == 2:
20         if n > 0:
21             key = stack.pop()
22             print("Pop key: ", key)
23             n -= 1
24         else:
25             print("Stack is empty")
26     elif choice == 3:
27         print("Stack: ", stack)
28     else:
29         break
```

```
31 # 12-2
32 print("Queue")
33 n = 0
34 queue = []
35 while True:
36     print("-----")
37     print("Queue operation")
38     print("option 1: Enqueue")
39     print("option 2: dequeue")
40     print("option 3: Display")
41     print("option 4: Quit")
42     print("-----")
43
44     choice = eval(input("Input your choice: "))
45     if choice == 1:
46         key = eval(input("Input your key: "))
47         queue.append(key)
48         n += 1
49     elif choice == 2:
50         if n > 0:
51             key = queue.pop(0)
52             print("Pop key: ", key)
53             n -= 1
54         else:
55             print("Queue is empty")
56     elif choice == 3:
57         print("Queue: ", queue)
58     else:
59         break
```

```

61 # 12-3
62 a = [1, 2, 3, 4]
63 b = [2, 4, 3, 1]
64
65 c = [0 for i in range(4)]
66 for i in range(4):
67     c[i] = a[i] + b[i]
68 print("向量加法")
69 print(c)
70
71 scalar = 3
72 for i in range(4):
73     c[i] = scalar * a[i]
74 print("向量乘法")
75 print(c)
76
77 # 12-4
78 def Matrix_Add(A, B): 1 usage new *
79     n = len(A)
80     c = [[0 for j in range(n)] for i in range(n)]
81     for i in range(n):
82         for j in range(n):
83             c[i][j] = A[i][j] + B[i][j]
84     return c
85 def Matrix_Multiply(A, B): 1 usage new *
86     n = len(A)
87     c = [[0 for j in range(n)] for i in range(n)]
88     for i in range(n):
89         for j in range(n):

```

```

78 def Matrix_Add(A, B): 1 usage new *
79     c = [[0 for j in range(n)] for i in range(n)]
80
81     for i in range(n):
82         for j in range(n):
83             c[i][j] = A[i][j] + B[i][j]
84     return c
85 def Matrix_Multiply(A, B): 1 usage new *
86     n = len(A)
87     c = [[0 for j in range(n)] for i in range(n)]
88     for i in range(n):
89         for j in range(n):
90             for k in range(n):
91                 c[i][j] = c[i][j] + A[i][k] * B[k][j]
92
93     return c
94
95 A = [[1, 2], [3, 4]]
96 B = [[2, 4], [3, 1]]
97
98 c = Matrix_Add(A, B)
99 print("矩陣乘法")
100 print(c)
101
102 c = Matrix_Multiply(A, B)
103 print("矩陣乘法")
104 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 $A \cup B$ ： `{1, 2, 3, 4, 5, 6}`

交集 $A \cap B$ ： `{3, 4}`

差集 $A - B$ ： `{1, 2}`

對稱差集 $A \oplus B$ ： `{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(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'}

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
```