

```
1 ##### 문제 1. 정수범위의 수 모두 더하기
2 def sum_range(start,stop,step):
3     if step > 0:
4         if start < stop:
5
6             return start + sum_range(start+step,stop,step)
7         else:
8             return 0
9     else:
10        return None
11
12 ### (1) 꼬리재귀 함수 [2점]
13 def sum_range(start,stop,step):
14     def loop(start,total):
15         if start < stop:
16             return loop(start+step,total+start)
17         else:
18             return total
19     if step > 0:
20         return loop(start,0)
21     else:
22         return None
23
24 ### (2) while 함수 [2점]
25 def sum_range(start,stop,step):
26     if step > 0:
27         total = 0
28         while start < stop:
29             total += start
30             start += step
31         return total
32     else:
33         return None
34
35 ### (3) for 함수 [2점]
36 def sum_range(start,stop,step):
37     if step > 0:
38         total = 0
39         for n in range(start,stop,step):
40             total += n
41         return total
42     else:
43         return None
44
45 ##print(sum_range(5,13,0)) # None
46 ##print(sum_range(13,5,-2)) # None
47 ##print(sum_range(13,5,2)) # 0
48 ##print(sum_range(5,13,1)) # 68
49 ##print(sum_range(5,13,2)) # 32
50
51
52 ##### 문제 2. 리스트에서 가장 작은 원소 찾기 (for 루프 활용) [4점]
53 def find_smallest(xs):
54     if xs != []:
55         smallest = xs[0]
56         for x in xs[1:]:
57             if x < smallest:
58                 smallest = x
59     return smallest
```

```

60     else:
61         return None
62
63 ##print(find_smallest([])) # None
64 ##print(find_smallest([3])) # 3
65 ##print(find_smallest([1,2,3,4,5])) # 1
66 ##print(find_smallest([5,4,3,2,1])) # 1
67 ##print(find_smallest([5,4,3,4,5])) # 3
68
69 ##### 문제 3. 리스트에서 지정한 원소 하나 제거 하기
70 def remove_one(xs,x):
71     if xs != []:
72         if x == xs[0]:
73             return xs[1:]
74         else:
75             return [xs[0]] + remove_one(xs[1:],x)
76     else:
77         return []
78
79 ### (1) 꼬리재귀 함수 [2점]
80 def remove_one(xs,x):
81     def loop(xs,zs):
82         if xs != []:
83             if x == xs[0]:
84                 return zs + xs[1:]
85             else:
86                 return loop(xs[1:],zs+[xs[0]])
87         else:
88             return zs
89     return loop(xs,[])
90
91 def remove_one(xs,x):
92     def loop(xs,zs):
93         if xs != []:
94             if x == xs[0]:
95                 return zs + xs[1:]
96             else:
97                 zs.append(xs[0])
98                 return loop(xs[1:],zs)
99         else:
100            return zs
101    return loop(xs,[])
102
103 ### (2) while 루프 함수 [2점]
104 def remove_one(xs,x):
105     zs = []
106     while xs != []:
107         if x == xs[0]:
108             return zs + xs[1:]
109         else:
110             zs.append(xs[0])
111             xs = xs[1:]
112     return zs
113
114 ##print(remove_one([],3)) # []
115 ##print(remove_one([4,2,3,4,1],4)) # [2,3,4,1]
116 ##print(remove_one([4,2,3,4,1],1)) # [4,2,3,4]
117 ##print(remove_one([4,2,3,4,1],5)) # [4,2,3,4,1]
118
119 ##### 문제 4. 리스트에서 지정한 원소 모두 제거 하기

```

```

120
121 ### (1) 재귀 함수 [2점]
122 def remove_all(xs,x):
123     if xs != []:
124         if x == xs[0]:
125             return remove_all(xs[1:],x)
126         else:
127             return [xs[0]] + remove_all(xs[1:],x)
128     else:
129         return []
130
131 ### (2) 꼬리재귀 함수 [2점]
132 def remove_all(xs,x):
133     def loop(xs,zs):
134         if xs != []:
135             if x == xs[0]:
136                 return loop(xs[1:],zs)
137             else:
138                 zs.append(xs[0])
139                 return loop(xs[1:],zs)
140         else:
141             return zs
142     return loop(xs,[])
143
144 ### (3) while 루프 함수 [2점]
145 def remove_all(xs,x):
146     zs = []
147     while xs != []:
148         if x != xs[0]:
149             zs.append(xs[0])
150         xs = xs[1:]
151     return zs
152
153
154 ##print(remove_all([],3)) # []
155 ##print(remove_all([4,2,3,4,1],4)) # [2,3,1]
156 ##print(remove_all([4,2,3,4,1],1)) # [4,2,3,4]
157 ##print(remove_all([4,2,3,4,1],5)) # [4,2,3,4,1]
158
159 ##### 문제 5. 자연수 문자열에서 쉼표(,)가 천 단위로 잘 삽입되어 있는지 확인
160 def check_number_with_comma(s):
161     def loop(s):
162         (front,comma,rest) = s.partition(",")
163         if len(front) == 3:
164             if comma == ",":
165                 return loop(rest)
166             else: # comma and rest must be ""
167                 return True
168         else:
169             return False
170     (front,comma,rest) = s.partition(",")
171     if len(front) == 3:
172         if int(front) < 100:
173             return False
174         else:
175             if comma == ",":
176                 return loop(rest)
177             else: # comma and rest must be ""
178                 return True
179     elif len(front) == 2:

```

```

180         if int(front) < 10:
181             return False
182         else:
183             if comma == ",":
184                 return loop(rest)
185             else: # comma and rest must be ""
186                 return True
187     elif len(front) == 1:
188         if int(front) == 0:
189             return False
190         else:
191             if comma == ",":
192                 return loop(rest)
193             else:
194                 return True
195     else:
196         return False
197
198 ### 논리식을 채워서 함수 완성 [6점]
199 def check_number_with_comma(s):
200     def loop(s):
201         (front,comma,rest) = s.partition(",")
202         return len(front) == 3 and (comma == "," and loop(rest) or \
203                                     comma != ",")
204         (front,comma,rest) = s.partition(",")
205         return len(front) == 3 and int(front) >= 100 and (comma == "," and
loop(rest) or comma != ",") or \
206         len(front) == 2 and int(front) >= 10 and (comma == "," and loop(rest)
or comma != ",") or \
207         len(front) == 1 and int(front) != 0 and (comma == "," and loop(rest)
or comma != ",")
208
209 ##print(check_number_with_comma("")) # False
210 ##print(check_number_with_comma("1")) # True
211 ##print(check_number_with_comma("0")) # False
212 ##print(check_number_with_comma("11")) # True
213 ##print(check_number_with_comma("01")) # False
214 ##print(check_number_with_comma("111")) # True
215 ##print(check_number_with_comma("011")) # False
216 ##print(check_number_with_comma("1111")) # False
217 ##print(check_number_with_comma("0111")) # False
218 ##print(check_number_with_comma("1,111")) # True
219 ##print(check_number_with_comma("1,000,011")) # True
220 ##print(check_number_with_comma("1,000,011,001")) # True
221 ##print(check_number_with_comma("01,000,011,001")) # False
222 ##print(check_number_with_comma("1,00,011,001")) # False
223 ##print(check_number_with_comma("1,000,11,001")) # False
224 ##print(check_number_with_comma("1,000,011,1")) # False
225
226 ##### 문제 6. 자연수에 천 단위로 쉼표(,)를 삽입하기
227
228 ### (1) 재귀 함수 [5점]
229 def to_number_with_comma(n):
230     if n >= 1000:
231         r = n % 1000
232         if r == 0:
233             digits = "000"
234         elif r < 10:
235             digits = "00" + str(r)
236         elif r < 100:

```

```
237         digits = "0" + str(r)
238     else:
239         digits = str(r)
240     return to_number_with_comma(n//1000) + "," + digits
241 else:
242     return str(n)
243
244 ### (2) 꼬리재귀 함수 [2점]
245 def to_number_with_comma(n):
246     def loop(n, acc):
247         if n >= 1000:
248             r = n % 1000
249             if r == 0:
250                 digits = "000"
251             elif r < 10:
252                 digits = "00" + str(r)
253             elif r < 100:
254                 digits = "0" + str(r)
255             else:
256                 digits = str(r)
257             return loop(n//1000, "," + digits + acc)
258         else:
259             return str(n) + acc
260     return loop(n, "")
261
262 ### (3) while 루프 함수 [2점]
263 def to_number_with_comma(n):
264     acc = ""
265     while n >= 1000:
266         r = n % 1000
267         if r == 0:
268             digits = "000"
269         elif r < 10:
270             digits = "00" + str(r)
271         elif r < 100:
272             digits = "0" + str(r)
273         else:
274             digits = str(r)
275         acc = "," + digits + acc
276         n //= 1000
277     return str(n) + acc
278
279 ##print(to_number_with_comma(0)) # "0"
280 ##print(to_number_with_comma(111)) # "111"
281 ##print(to_number_with_comma(1000)) # "1,000"
282 ##print(to_number_with_comma(1001)) # "1,001"
283 ##print(to_number_with_comma(1011)) # "1,011"
284 ##print(to_number_with_comma(1111)) # "1,111"
285 ##print(to_number_with_comma(11000)) # "11,000"
286 ##print(to_number_with_comma(11001)) # "11,001"
287 ##print(to_number_with_comma(11011)) # "11,011"
288 ##print(to_number_with_comma(11111)) # "11,111"
289 ##print(to_number_with_comma(1111111)) # "1,111,111"
290 ##print(to_number_with_comma(11111111)) # "11,111,111"
291 ##print(to_number_with_comma(111111111)) # "111,111,111"
292 ##print(to_number_with_comma(1111111111)) # "1,111,111,111"
293 ##print(to_number_with_comma(11111111111)) # "11,111,111,111"
294
295
```