파이썬프로그래밍 2분반 ## <7주차 과제> ---

정보보안공학과 ### 202121556 ### 곽지현

2023-04-18

In [59]:
```
# 3. 리스트 연산자의 이해를 위해 다음 코드를 실행하여 결과를 확인해보세요.

list3 = [] # 리스트 선언

print(list3 + [1, 2, 3], "길이:", len(list3 + [1, 2, 3]))
# list3에 [1, 2, 3]을 연결하고 그 길이를 출력
print(list3 + [1, 2] * 2, "길이:", len(list3 + [1, 2] * 2))
# list3에 [1, 2]를 연결하고 곱하기 2를 한뒤, 총 길이를 출력
print(list3, "길이:", len(list3))
# list3와 그 길이를 출력
```

```
[1, 2, 3] 길이: 3
[1, 2, 1, 2] 길이: 4
[] 길이: 0
```

In [57]:
```
# 4. 리스트의 append, insert, extend 함수를 통한 요소추가의 이해를 위해 다음 코드를
# 실행하여 결과를 확인해보세요.

list4 = [] # 리스트 선언
list4.append(1) # list4에 1을 추가
list4.append("abc") # list4에 "abc"를 추가
print(list4) # list4를 출력

list4.insert(0, "New") # list4의 0번째 인덱스에 "New"를 추가
print(list4) # list4를 출력

list4.extend([2, 3, 4]) # list4 뒤에 2, 3, 4를 추가
print(list4) # list4를 출력

list4.extend("defg") # list4 뒤에 "defg"를 추가
print(list4) # list4를 출력
```

```
[1, 'abc']
['New', 1, 'abc']
['New', 1, 'abc', 2, 3, 4]
['New', 1, 'abc', 2, 3, 4, 'd', 'e', 'f', 'g']
```

In [56]:
```
# 5. 리스트의 pop, remove, clear, del 함수를 통한 요소제거의 이해를 위해 다음 코드를
# 실행하여 결과를 확인해보세요.

list5 = [1, 2, 3, 4, 5] # 리스트 선언

pop_data = list5.pop() # pop_data에 list5의 5를 제거해 대입
print(pop_data, list5) # pop_data와 list5를 출력
```

```
pop_data2 = list5.pop(0) # pop_data2에 list5의 0번째 인덱스인 1을 제거해 대입
print(pop_data2, list5) # pop_data2와 list5를 출력

list5.remove(3) # list5의 3을 제거
print(list5) # list5를 출력

del list5[0] # list5의 0번째 인덱스인 2를 제거
print(list5) # list5를 출력

list5.clear() # list5 모두 제거
print(list5) # list5를 출력
```

```
5 [1, 2, 3, 4]
1 [2, 3, 4]
[2, 4]
[4]
[]
```

In [14]:
```
# 6. 리스트 슬라이싱의 이해를 위해 다음 코드를 실행하여 결과를 확인해보세요.

list6 = [[1, 2], [3, 4], [5, 6], [7, 8]] # 리스트안에 리스트 선언

print(list6[0:2]) # list6의 0번, 1번 인덱스 출력
print(list6[0:2][1]) # list6의 0번과 1번 인덱스 중 1번 인덱스 출력
print(list6[0:2][1:]) # list6의 0번과 1번 인덱스 중 1번부터 인덱스 출력
```

```
[[1, 2], [3, 4]]
[3, 4]
[[3, 4]]
```

In [9]:
```
# 7. 리스트 연산 함수에 대한 이해를 위해 다음 코드를 실행하여 결과를 확인해보세요.

list7 = [5, 2, 4, 10, 5, 7, 8, 5] # 리스트 선언

list7.sort() # list7을 오름차순으로 정렬
print(list7) # list7을 출력

list7.reverse() # list7을 내림차순으로 정렬
print(list7) # list7을 출력

print(list7, "7위치 :", list7.index(7)) # list7에서 7번의 위치 출력
print(list7, "5의 개수 :", list7.count(5)) # list7에서 5의 개수를 출력
```

```
[2, 4, 5, 5, 5, 7, 8, 10]
[10, 8, 7, 5, 5, 5, 4, 2]
[10, 8, 7, 5, 5, 5, 4, 2] 7위치 : 2
[10, 8, 7, 5, 5, 5, 4, 2] 5의 개수 : 3
```

In [3]:
```
# 8. 리스트와 in 연산자의 이해를 위해 다음 코드를 실행하여 결과를 확인해보세요.

list8 = [1, 3, 5, 7, 9] # 리스트 선언

print(8 in list8)
# list8안에 8이 없기 때문에 False를 출력
print(1 in list8)
# list8안에 1이 있기 때문에 True를 출력
print([1, 3] in list8)
# list8안에 [1, 3]리스트가 없기 때문에 False를 출력
print([1, 3, 5, 7, 9] in list8)
# list8안에 [1, 3, 5, 7, 9]리스트가 있기 때문에 True를 출력
print(list8[0] in list8)
```

```
# list8안에 list8의 0번 인덱스인 10이 있기 때문에 True를 출력
print(list8[:2] in list8)
# list8안에 list8의 0번, 1번 인덱스인 [1, 3]리스트가 없기 때문에 False를 출력
```

```
False
True
False
False
True
False
```

In [1]:
```
# 9. for문의 이해를 위해 다음 코드를 실행하여 결과를 확인해보세요.

for x in range(10): # 10번 반복
    print(x+1, "회차 반복") # 출력
```

```
1 회차 반복
2 회차 반복
3 회차 반복
4 회차 반복
5 회차 반복
6 회차 반복
7 회차 반복
8 회차 반복
9 회차 반복
10 회차 반복
```

In [3]:
```
# 10. 리스트를 활용한 for문의 이해를 위해 다음 코드를 실행하여 결과를 확인해보세요.

for item in [1, 2, 3]: # [1, 2, 3]리스트의 요소가 하나씩 item안에 들어감
    print(item) # item을 출력

print() # 공백 출력
for item1, item2 in [[1,2], [3, 4]]: # [1, 2]리스트의 요소는 item1에 [3, 4]리스트의 요
    print("[1, 2], [3, 4]가 자동으로 언패킹 됨")
    print(item1, item2) # item1, item2를 출력

print() # 공백 출력
for x in [[1,2], [3,4]]: # [1. 2], [3, 4]리스트의 요소가 하나씩 x안에 들어감
    print(x[0], x[1]) # x의 0번, 1번 인덱스 출력
```

```
1
2
3

[1, 2], [3, 4]가 자동으로 언패킹 됨
1 2
[1, 2], [3, 4]가 자동으로 언패킹 됨
3 4

1 2
3 4
```

In [36]:
```
# 11. 반복하여 사용자 입력을 리스트에 넣어주는 프로그램입니다.
# 다음과 같은 결과가 나오도록 빈칸을 채워 실행하세요.

result = [] # 리스트 선언
user = input("몇 개 넣을건가요 ? > ") # input함수 선언
for num in range(int(user)): # user에 입력된 숫자만큼 반복
    user2 = input("item input > ") # input함수 선언
```

```
    result.append(user2) # result에 user2를 추가
print(result) # result 출력
```

```
몇 개 넣을건가요 ? > 3
item input > 1
item input > abc
item input > 10
['1', 'abc', '10']
```