파이썬프로그래밍 2분반 ## <9주차 과제> ---

정보보안공학과 ### 202121556 ### 곽지현

2023-05-05

In [1]:
```
# 1. 딕셔너리 선언의 이해를 위해 다음 코드를 실행하여 결과를 확인해보세요.

dict1 = {
    "key1" : 1, "key2" : "2", # key1에 1을 저장, key2에 "2"를 저장
    "key3" : 3, "key4" : 4,   # key3에 3을 저장, key4에 4를 저장
    "key5" : [5, 6, 7],       # key5에 리스트[5, 6, 7]를 저장
    "key6" : {"key6-1" : [True, False]} # key6에 "key6-1"과 리스트[True, False]를 저장
}
print(dict1) # dict1 출력
dict2 = dict(key1=1, key2="2", key3=3, key4=4, key5=[5, 6, 7])
# dict2에 dict(key1=1, key2="2", key3=3, key4=4, key5=[5, 6, 7])를 저장
print(dict2) # dict2 출력
```

```
{'key1': 1, 'key2': '2', 'key3': 3, 'key4': 4, 'key5': [5, 6, 7], 'key6': {'key6-1': [T
rue, False]}}
{'key1': 1, 'key2': '2', 'key3': 3, 'key4': 4, 'key5': [5, 6, 7]}
```

In [2]:
```
# 2. 딕셔너리 요소 접근의 이해를 위해 다음 코드를 실행하여 결과를 확인해보세요.

dict1 = {
    "key1" : 1, "key2" : "2", # key1에 1을 저장, key2에 "2"를 저장
    "key3" : 3, "key4" : 4,   # key3에 3을 저장, key4에 4를 저장
    "key5" : [5, 6, 7],       # key5에 리스트[5, 6, 7]를 저장
    "key6" : {"key6-1" : [True, False]} # key6에 "key6-1"과 리스트[True, False]를 저장
}
print("dict1 키 'key1'의 값:", dict1["key1"]) # key1 값 출력
print("dict1 키 'key5'의 값:", dict1["key5"]) # key5 값 출력
print("dict1 키 'key5'의 값 [5, 6, 7]의 인덱스 2의 값:", dict1["key5"][2])
# key5 리스트에서 인덱스 2의 값 출력
print() # 공백 출력

if "key6" in dict1: # dict1 안에 key6의 값이 있다면
    print("dict1에 키 'key6'의 값:", dict1.get("key6"))
    # key6의 값을 dict1에서 추출하여 출력

if ("key7" in dict1) == False:
    # dict1 안"없는 키!에 key7이 없어서 False이다.
    # 그래서 False와 False가 같기 때문에 if문 실행
    print("dict1에 키 'key7'가 없음:", dict1.get("key7", !"))
    # dict1에 key7의 값이 없어서 "없는 키!!"로 사용할 값을 지정
    print("get 함수는 키 값이 없을 때 사용할 값을 지정해줄수 있음")
    # "get 함수는 키 값이 없을 때 사용할 값을 지정해줄수 있음"을 출력
```

```
dict1 키 'key1'의 값: 1
dict1 키 'key5'의 값: [5, 6, 7]
dict1 키 'key5'의 값 [5, 6, 7]의 인덱스 2의 값: 7

dict1에 키 'key6'의 값: {'key6-1': [True, False]}
dict1에 키 'key7'가 없음: 없는 키!!
get 함수는 키 값이 없을 때 사용할 값을 지정해줄수 있음
```

In [3]:
```
# 3. 딕셔너리 값 추가 및 제거의 이해를 위해 다음 코드를 실행하여 결과를 확인해보세요.

dict3 = {
    "key1" : "value1" # key1에 "value1"를 저장
}
print("dict3 키 'newKey' 생성 및 값을 추가")
dict3["newKey"] = "newValue" # dict3에 newKey : newValue 요소를 추가
print(dict3) # dict3의 값을 출력

print("dict3 키 'key1'의 값을 수정") # "dict3 키 'key1'의 값을 수정"을 출력
dict3["key1"] = "editValue" # dict3에 key1의 값을 editValue로 수정
print(dict3) # dict3의 값을 출력

print("dict3 키 'key1'와 그 값을 삭제") # "dict3 키 'key1'와 그 값을 삭제"를 출력
del dict3["key1"] # dict3에 key1을 제거
print(dict3) # dict3의 값을 출력
```

```
dict3 키 'newKey' 생성 및 값을 추가
{'key1': 'value1', 'newKey': 'newValue'}
dict3 키 'key1'의 값을 수정
{'key1': 'editValue', 'newKey': 'newValue'}
dict3 키 'key1'와 그 값을 삭제
{'newKey': 'newValue'}
```

In [4]:
```
# 4. 딕셔너리를 사용한 for문의 이해를 위해 다음 코드를 실행하여 결과를 확인해보세요.

dict4 = {
    "key1" : 1, "key2" : "2", # key1에 1을 저장, key2에 "2"를 저장
    "key3" : 3, "key4" : 4,   # key3에 3을 저장, key4에 4를 저장
    "key5" : [5, 6, 7],       # key5에 리스트[5, 6, 7]를 저장
    "key6" : {"key6-1" : [True, False]}
    # key6에 "key6-1"과 리스트[True, False]를 저장
}

for key in dict4: # dict4 안에 있는 키들이 key 변수에 들어간다.
    print(key, ":", dict4[key]) # key 출력
```

```
key1 : 1
key2 : 2
key3 : 3
key4 : 4
key5 : [5, 6, 7]
key6 : {'key6-1': [True, False]}
```

In [5]:
```
# 5. range 함수의 이해를 위해 다음 코드를 실행하여 결과를 확인해보세요.

print(list(range(10))) # 0 ~ 9 까지 리스트로 출력
print(list(range(3, 9))) # 3 ~ 8 까지 리스트로 출력
print(list(range(1, 10, 2))) # 1 ~ 9 까지 2씩 증가하면서 리스트로 출력
print(list(range(10, 0, -1))) # 10 ~ 1 까지 1씩 감소하면서 리스트로 출력
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[3, 4, 5, 6, 7, 8]
[1, 3, 5, 7, 9]
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

In [6]:
```python
# 6. range 함수를 사용한 for문의 이해를 위해 다음 코드를 실행하여 결과를 확인해보세요.

cnt = 0 # cnt에 0을 대입
for x in range(10): # 0 ~ 9 까지 10개를 x 변수에 대입
    print("x:", x) # x를 출력
    cnt += 1 # cnt에 1을 증가
print(cnt, "번 반복₩n") # cnt와 "번 반복₩n"을출력

cnt = 0 # cnt에 0을 대입
for x in range(3, 9): # 3 ~ 8 까지 6개를 x 변수에 대입
    print("x:", x) # x를 출력
    cnt += 1 # cnt에 1을 증가
print(cnt, "번 반복₩n") # cnt와 "번 반복₩n"을출력

cnt = 0 # cnt에 0을 대입
for x in range(1, 10, 2): # 1 ~ 9 까지 2씩 증가하여 x 변수에 대입
    print("x:", x) # x를 출력
    cnt += 1 # cnt에 1을 증가
print(cnt, "번 반복₩n") # cnt와 "번 반복₩n"을출력

cnt = 0 # cnt에 0을 대입
for x in range(10, 0, -1): # 10 ~ 1 까지 1씩 감소하여 x 변수에 대입
    print("x:", x) # x를 출력
    cnt += 1 # cnt에 1을 증가
print(cnt, "번 반복₩n") # cnt와 "번 반복₩n"을출력
```

```
x: 0
x: 1
x: 2
x: 3
x: 4
x: 5
x: 6
x: 7
x: 8
x: 9
10 번 반복

x: 3
x: 4
x: 5
x: 6
x: 7
x: 8
6 번 반복

x: 1
x: 3
x: 5
x: 7
x: 9
5 번 반복

x: 10
x: 9
x: 8
x: 7
x: 6
x: 5
x: 4
x: 3
x: 2
x: 1
10 번 반복
```

In [7]:
```
# 7. 리스트와 range 함수를 함께 사용한 for문의 이해를 위해
# 다음 코드를 실행하여 결과를 확인해보세요.

items = ["item1", "item2", "item3", "item4"] # items 리스트 선언

print(len(items), "만큼 반복") # 리스트 길이 출력
for x in range(len(items)): # 리스트 길이가 4이기 때문에 0 ~ 3 까지 4개를 x 변수에 대입
    print(x, "번째 반복.", x, "번째 데이터:", items[x]) # x를 출력
```

```
4 만큼 반복
0 번째 반복. 0 번째 데이터: item1
1 번째 반복. 1 번째 데이터: item2
2 번째 반복. 2 번째 데이터: item3
3 번째 반복. 3 번째 데이터: item4
```

In [8]:
```
# 8. 중첩 for문의 이해를 위해 다음 코드를 실행하여 결과를 확인해보세요.

for dan in range(1, 10): # 1 ~ 9 까지 9개를 dan 변수에 대입
    for num in range(1, 10): # 1 ~ 9 까지 9개를 num 변수에 대입
        print("{} * {} = {}" .format(dan, num, dan * num))
```

```
        # dan, num, dan * num의 값들을 출력
    print() # 공백 출력
```

```
1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
1 * 4 = 4
1 * 5 = 5
1 * 6 = 6
1 * 7 = 7
1 * 8 = 8
1 * 9 = 9

2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18

3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27

4 * 1 = 4
4 * 2 = 8
4 * 3 = 12
4 * 4 = 16
4 * 5 = 20
4 * 6 = 24
4 * 7 = 28
4 * 8 = 32
4 * 9 = 36

5 * 1 = 5
5 * 2 = 10
5 * 3 = 15
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
5 * 7 = 35
5 * 8 = 40
5 * 9 = 45

6 * 1 = 6
6 * 2 = 12
6 * 3 = 18
6 * 4 = 24
6 * 5 = 30
6 * 6 = 36
6 * 7 = 42
6 * 8 = 48
6 * 9 = 54
```

```
7 * 1 = 7
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
7 * 7 = 49
7 * 8 = 56
7 * 9 = 63

8 * 1 = 8
8 * 2 = 16
8 * 3 = 24
8 * 4 = 32
8 * 5 = 40
8 * 6 = 48
8 * 7 = 56
8 * 8 = 64
8 * 9 = 72

9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
```

In [19]:

```python
# 9. 상하 대칭한 직각삼각형 형태로 별 찍는 프로그램입니다.
# 다음과 같은 결과가 나오도록 빈칸을 채워 실행하세요.

x = int(input("줄 수 입력 > ")) # 정수를 입력 받아서 x에 대입
for i in range(x): # 0 ~ x 만큼 i 변수에 대입
    for j in range(0, x-i): # 0 ~ (x-i) 만큼 j 변수에 대입
        print("*", end=" ") # 별과 공백 출력
    print() # 공백 출력
```

```
줄 수 입력 > 15
* * * * * * * * * * * * * * *
* * * * * * * * * * * * * *
* * * * * * * * * * * * *
* * * * * * * * * * * *
* * * * * * * * * * *
* * * * * * * * * *
* * * * * * * * *
* * * * * * * *
* * * * * * *
* * * * * *
* * * * *
* * * *
* * *
* *
*
```