제어문

5

프로그램 코드는 라인 단위로 순차적으로 실행된다. 이런 흐름을 조건문, 반복문으로 제어하여 원하는 작업을 하게 하는데 이를 제어문 코드라 한다.

논리연산자

Boolean 자료형

파이썬의 기본 자료형은 정수형, 실수형, 문자열과 함께 Boolean이 있다. 불리언(자료형은 논리 자료형이라고도 하며, 참과 거짓을 나타내는데 쓰인다.

Boolean	내용
==	좌변과 우변이 같다
!=	좌변과 우변이 같지 않다
>	좌변이 우변보다 크다
<	좌변이 우변보다 작다
>=	좌변이 우변보다 크거나 같다
<=	좌변이 우변보다 작거나 같다

논리 연산자

```
In [1]: name="wolfpack"
        age=59

In [2]: name=="kim"
Out[2]: False

In [3]: age>50
Out[3]: True
```

```
In [*]: secret=3
        guess=int(input('Enter (1-9)?'))
```

- secret에 정수 3을 입력하고 input() 함수에 의해 콘솔에 키보드 입력 창이 열린다. 키보드로 입력 받은 값은 int() 함수에 의해 정수로 guess에 저장된다.

```
In [*]: secret=3
        guess=int(input('Enter (1-9)?'))
        Enter (1-9)? 7
```

- 키보드로부터 입력된 값 7은 guess에 저장된다.

```
In [9]: secret==guess
Out[9]: False
```

- guess 값과 secret 값을 비교하 불린 자료형은 False가 출력된다.

if 문

개념

프로그램 코드의 연산은 좌에서 우로 라인(행) 위에서 아래로 순서대로 실행된다. 이를 제어하는 것을 제어문이라 한다.

제어문 중 가장 사용 빈도가 높은 것은 논리 연산자를 사용하여 조건의 만족여부에 따라 샐행 흐름 규칙을 제언하는 if 문이다.

```
if 조건(논리연산자) :
    실행문장1
    실행문장2
    ...
```

- 조건문에는 if라는 키워드를 사용한다.
- if 다음에는 '조건'(논리연산자)이 나타난다. 참 true, 거짓 false 둘 중 하나
- if 문의 종료(끝)는 콜론(:)으로 표현한다.
- 조건이 참(True)이면 (반드시) 들여쓰기한 문장이 실행된다. 들여쓰기는 :을 입력하고 엔터를 하면 자동 들여쓰기가 된다.
- 조건이 만족되는 실행문은 여러 개인 경우 라인 단위로 입력한다. {}을 사용 할 필요는 없다.
- 들여쓰기를 하지 않은 실행문은 오류가 발생한다.

예제 1

```
In [ ]: secret=3 (1)
        guess=int(input('Enter (1-9)?')) (2)

In [ ]: if secret==guess: (3)
            print('good job') (4)
        if secret!=guess:
            print('Try again')
```

(1) secret에 정수 값 3이 입력된다.

(2) Enter (1-9)? 콘솔 화면에 (Jupyter는 바로 아래 공간) 출력되고 키보드 입력을 기다린다. 키보드 입력된 값은 guess에 저장된다.

(3) secret 값과 guess 값을 비교하여 True이면 : 아래 문장을 실행한다. False이면 : 아래 문장을 실행하지 않는다.

(4) (3)의 조건이 만족하면 실행되는 문장이다.

```
In [*]: secret=3
        guess=int(input('Enter (1-9)?'))

        Enter (1-9)? 4
```

- 코드를 실행하면 바로 아래 입력 창이 나타난다. 키보드 입력 전까지는 In[]안에 *가 표시되는데 이는 아직 실행 중이거나 외부 입력을 기다리는 상태이다.
- 키보드에서 4를 입력하고 Enter Key를 친다. 그러면 파이썬은 guess에 4를 입력하고 다음 라인으로 간다.

```
In [1]: secret=3
        guess=int(input('Enter (1-9)?'))

        Enter (1-9)?4

In [2]: if secret==guess:
            print('good job')
        if secret!=guess:
            print('Try again')

        Try again
```

- secret=3, guess=4가 저장되어 있어 같지 않으므로 두번째 조건이 성립하므로 두번째 문장인 “Try Again”이 콘솔 출력된다.

예제

```
In [31]: for x in range(1, 11):
             print('{0:2d} {1:3d} {2:4d}'.format(x, x*x, x*x*x))

          1   1    1
          2   4    8
          3   9   27
          4  16   64
          5  25  125
          6  36  216
          7  49  343
          8  64  512
          9  81  729
         10 100 1000
```

```
if 조건 :
    실행문1
else :
    실행문2
```

if 조건이 만족하면 : 아래 문장이 실행되고 만족하지 않으면 else : 아래 문장이 실행된다.

```
In [1]: secret=3
        guess=int(input('Enter (1-9)?'))

        Enter (1-9)?4
```

- 앞에서 if 문장을 두 번 사용되는 대신 else: 문장을 한 번 사용하면 된다.

```
In [*]: if secret==guess:
            print('good job')
        else:
            print('Try again')
            guess=int(input('Enter (1-9)?'))

        Enter (1-9)?
```

- secret=3, guess=4가 저장되어 있어 if 다음 조간이 만족하지 않으므로 : 다음 문장은 실행되지 않는다.
- 대신 else : 아래 문장이 실행되어 Try Again 이 출력된다. 그리고 그 다음 문장이 실행되어 출력된 Try Again은 사라지고 다음 input 문장이 콘솔에 나타난다.

예제

창세기 1장 1:4절에서 if 문을 사용하여 'day' 단어가 있는 경우 콘솔 출력하는 코드

```
In [1]: genesis='In the beginning God created the heavens
        if 'day' in genesis:
            print('today')
```

```
today
```

```
if 조건 :
        실행문1
elif 조건 :
        실행문2
else :
        실행문3
```

if 조건이 참이면 실행문 1, 거짓이면 elif 조건 참이면 실행문2, 거짓이면 실행문3을 실행한다.

```
In [4]: genesis='In the beginning God cre
        if 'wolfpack' in genesis:
            print('today')
        elif 'heavenss' in genesis:
            print('heavens')
        else :
            print('None')
```

```
None
```

pass 문 사용

pass 문은 아무것도 실행하지 않는다. 문법적으로 문장이 필요하지만, 프로그램이 특별히 할 일이 없을 때 사용할 수 있다.

```
In [5]: genesis='In the beginning God crea
        if 'wolfpack' in genesis:
            pass
        else :
            print('None')
```

```
None
```

```
In [2]: song = ['사랑', '했어요', '그', '때는', '몰랐지만']
        for i in range(len(song)):
            if song[i]=='그':
                pass
            else:
                print(i, song[i])
```

```
0 사랑
1 했어요
3 때는
4 몰랐지만
```

예제

쌍으로 된 데이터도 in 연속 옵션으로 가능하다. ds 타입은 list이다.

```
In [20]: ds = [(1,3), (3,5), (5,7)]
         for (first, second) in ds:
             print('곱하기', first*second)
```

```
곱하기 3
곱하기 15
곱하기 35
```

[표현식 for 항목 in 반복가능객체]

range(1,5,2)에 의해 x=[1, 3]가 들어 있다. 그러므로 for 문의 number 는 1, 3이 차례로 들어간다.

```
In [23]: x = range(1,5,2)
         result = [number * 3 for number in x]
         print(result)

         [3, 9]
```

[표현식 for 항목 in 반복가능객체 if 조건문]

반복 문 in에 if 조건문이 있어 조건에 맞는 값만 반복된다. x에는 3의 배수만 들어가게 된다.

```
In [25]: x = range(1,10)
         result = [number for number in x if number%3==0]
         print(result)

         [3, 6, 9]
```

for 항목1 in 반복가능객체1 if 조건문1
for 항목2 in 반복가능객체2 if 조건문2
...

for 문을 두 번 연속하여 사용 가능하다.

```
In [29]: result = [x*y for x in range(11,13)
                    for y in range(1,13)]
         print(result)

         [11, 22, 33, 44, 55, 66, 77, 88, 99, 110, 121, 132, 12, 24,
```

for 문

for 반복변수 in 리스트(또는 튜플, 문자열):
수행할 문장1
수행할 문장2

- 리스트에 지정된 순서대로 처음부터 끝까지 반복문장을 실행한다.
- 리스트에는 숫자(실수, 정수), 문자 모두 가능하고 숫자인 경우 크기 순으로 일련하여 넣을 필요는 없다.
- 반복문장에 반복변수를 넣지 않아도 된다.

```
In [9]: for i in [9,3.5,'wolfpack'] :
            print(i,"번입니다")

9 번입니다
3.5 번입니다
wolfpack 번입니다
```

반복변수 i에 9, 3.5, “wolfpack” 차례로 입력하면서 반복문장을 실행한다.

range(a,b) 사용

(a, b)는 모두 정수이여야 하며, a<b이어야 한다. a, a+1, …, (b-1) 정수 값이 차례로 들어간다. a부터 시작하지만 (b-1) 값이 종료 값이다.

```
In [27]: Rep=list(range(3,7))
         Rep

Out[27]: [3, 4, 5, 6]

In [28]: Rep=range(3,7)
         Rep

Out[28]: range(3, 7)
```

range()는 일련의 정수가 있으나 정수 값을 데이터로 사용하려면 list() 함수를 사용해야 한다.
list()를 사용하지 않으면 Rep에 range()가 저장된다.

예제

in 반복 리스트에는 문자, 숫자 모두 가능하다.

```
In [3]: count = ['one', 'two', 3, 4.5]
        for i in count:
            print(i)

one
two
3
4.5
```

if 조건문의 예와 동일하다. in 리스트에는 쌍으로 된 리스트 형태도 가능하다.

```
In [6]: count2 = [(1,2), (3,4), (5,6)]
        for (first, last) in count2:
            print(first + last)

        3
        7
        11
```

for 문에 range() 사용할 때는 list() 사용하지 않아도 된다.

```
In [21]: for i in range(3,7):
             print(i,"번입니다")

         3 번입니다
         4 번입니다
         5 번입니다
         6 번입니다
```

예제

```
In [8]: score = [95, 45, 67, 35, 70]

        num = 0
        for mark in score:
            num = num +1
            if mark >= 60:
                print("%d번 학생은 Pass." % num)
            else:
                print("%d번 학생은 Fail." % num)

        1번 학생은 Pass.
        2번 학생은 Fail.
        3번 학생은 Pass.
        4번 학생은 Fail.
        5번 학생은 Pass.
```

for 문과 Tuple 예제

```
In [32]: name=('wolfpack','kingkong','who')
         for student in name:
                 print('학생 이름은 %s이다.' %student)

         학생 이름은 wolfpack이다.
         학생 이름은 kingkong이다.
         학생 이름은 who이다.
```

for 문과 Dictionary 예제

```
In [42]: name_pt={'wolfpack':99,'kingkong':88,'who':77}
         for student in name_pt.keys():
             print('학생 %s 성적은 %s이다.' %(student, name_pt[student]))

         학생 wolfpack 성적은 99이다.
         학생 kingkong 성적은 88이다.
         학생 who 성적은 77이다.
```

for 문과 continue

if 조건 : - 조건이 만족하면 continue에 의해 print() 문 실행하지 않고 넘어간다.

```
In [18]: score = [95, 45, 67, 35, 70]

         num = 0
         for mark in score:
             num = num +1
             if mark < 60:
                 continue
             print("%d번 학생은 Pass." % num)
```

```
1번 학생은 Pass.
3번 학생은 Pass.
5번 학생은 Pass.
```

for 문과 range 함수

```
In [12]: sum = 0
         for i in range(1, 11):
             sum = sum + i
         print(sum)
```

```
55
```

```
sum += + i
```

```
In [19]: score = [95, 45, 67, 35, 70]
         for num in range(len(score)):
             if score[num] < 60:
                 continue
             print("%d번 학생 Pass입니다." % (num+1))
```

```
1번 학생 Pass입니다.
3번 학생 Pass입니다.
5번 학생 Pass입니다.
```

break 문 사용

if 조건 : else : 사용 시 조건에 range() 처럼 반복 체크해야 하는 경우 조건 만족여부에 따라 참 실행문, 거짓 실행문이 반복 실행된다.

만약 조건이 2부터 시작하여 10까지 숫자 중 3의 배수만 출력하고 싶다면…

```
In [15]: for x in range(2, 10):
             if x % 3 == 0:
                 print(x, '3의 배수')
             else:
                 pass
```

```
3 3의 배수
6 3의 배수
9 3의 배수
```

```
In [16]: for x in range(2, 10):
             if x % 3 == 0:
                 print(x, '3의 배수')
                 break
             else:
                 pass
```

```
3 3의 배수
```

continue 문 사용

if 조건 : else : 에서 else 문을 사용하는 대신 다음을 사용할 수 있다.

```
In [18]: for x in range(2, 10):
             if x % 3 == 0:
                 print(x, '3의 배수')
                 continue
             print(x, '3의 배수 X')
```

```
2 3의 배수 X
3 3의 배수
4 3의 배수 X
5 3의 배수 X
6 3의 배수
7 3의 배수 X
8 3의 배수 X
9 3의 배수
```

while 문

필요 이유

for문과 동일하게 반복문을 작성할 수 있다. for문은 반복 횟수가 미리 정해져 있거나 자료구조 (리스트, 튜플, 사전)를 사용할 수 있다.

반면 while 문은 반복해야 할 횟수가 특별히 정해지지 않고 어떤 조건을 충족하는 동안만 실행될 때 주로 사용한다.

```
while <조건문>:
    <수행할 문장1>
    <수행할 문장2>
```

10까지 정수 합을 구하는 코드이다. sum+=1 <=> sum=sum+1 과 같다. print 문을 들여 쓰기를 하지 않았으므로 while 문 내에서 실행되지 않고 while 문이 끝난 후 실행된다.

```
In [74]: sum=0;i=1
         while i<=10:
             sum+=i
             i+=1
         print('10까지 정수 합=%d'%sum)

         10까지 정수 합=55
```

while 문이 무한 루프를 돌면 CTRL+c 로 탈출 가능하다.

while 문과 if 문 사용, break, continue

break 사용하기

0~9까지 비번은 열번 안에 맞출 수 있다.

```
In [70]: import random as rn
         secret=rn.randint(0,9) (1)
         tri=0  (2)
         while tri<10:
             guess=int(input('Enter (0~9)?'))
             if secret==guess:(3)
                 print('Perfect!!!')
                 break (4)
             else:
                 tri+1
                 print('Fail !!!')

         Enter (0~9)?5
         Fail !!!
         Enter (0~9)?6
         Fail !!!
         Enter (0~9)?3
         Perfect!!!
```

(1) randint(0,9) 함수는 0~9 사이 정수 중 하나를 임의 추출하여 secret에 저장한다. 이 함수는 random 모듈에 있어 import 문을 사용한다.

(2) 시도 회수의 초기 값을 0으로 한다. 마지막 두 번째 행의 tri+1에 의해 1씩 증가

(3) input() 함수에 의해 사용자가 입력한 값이 guess에 저장되어 있어, 비번과 입력값이 같은지 체크한다.

(4) True인 경우 게임을 끝나기 위하여 break 문을 사용했다.

continue 사용하기

continue 문에 의해 if 조건이 만족하는 경우 while로 돌아간다.

```
In [76]: a = 0
         while a < 10:
             a = a + 1
             if a % 2 == 1: continue
             print(a)
```

```
2
4
6
8
10
```

1부터 100까지의 정수 중 3의 배수의 합을 구하시오.

```
In [78]: sum = 0; i=0
         while i <= 100:
             i = i + 1
             if i % 3 == 0: sum+=i
         print(sum)
```

```
1683
```

피보나치 수열

```
In [80]: def fib(n):    # Fibonacci series up to n
             """Print a Fibonacci series up to n."""
             a, b = 0, 1
             while a < n:
                 print(a, end=' ')
                 a, b = b, a+b
             print()
         fib(1000)
```

```
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
```

$$F_1 = F_2 = 1$$
$$F_n = F_{n-1} + F_{n-2} \qquad (n \in \{3, 4, \ldots\})$$