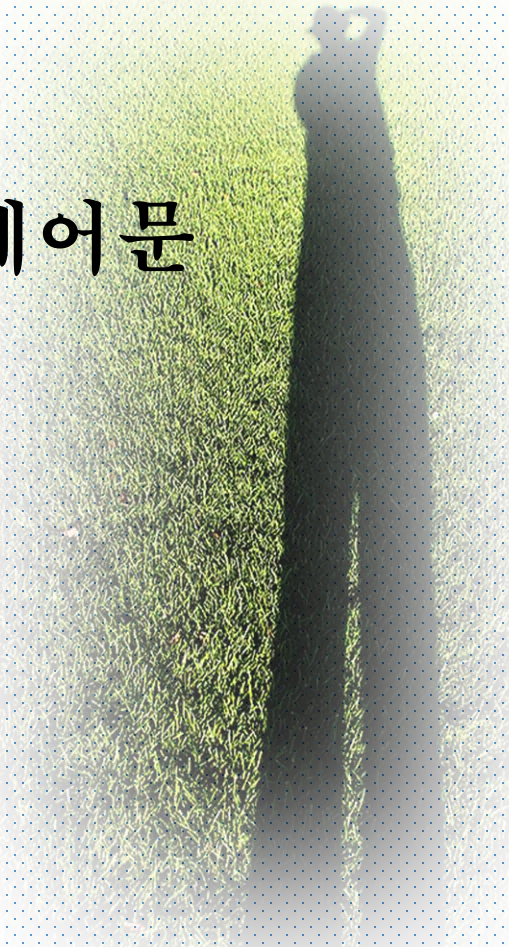
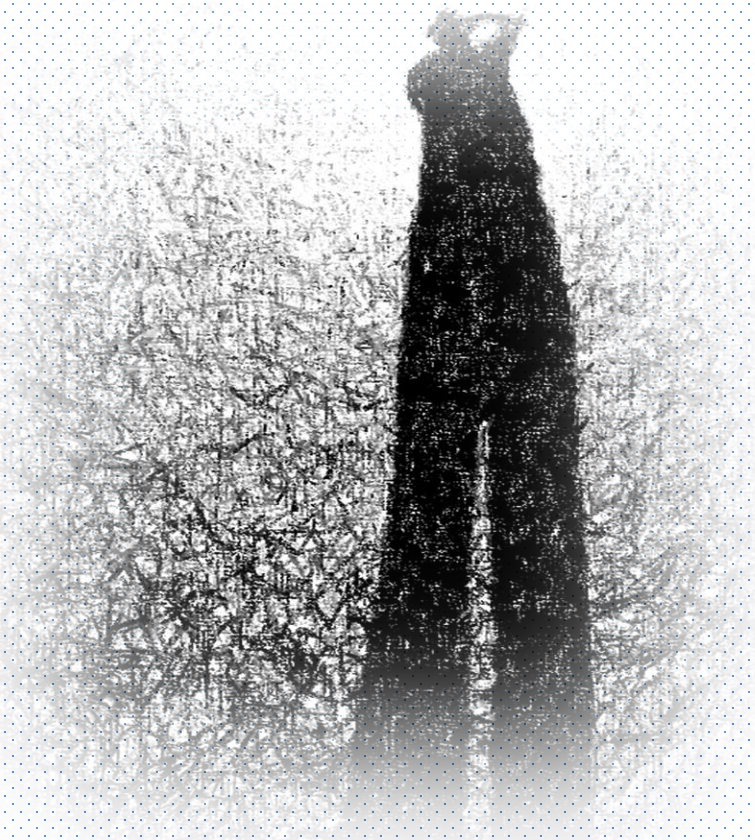
Python의 제어문

boolean
if
for
while



내용

1. 논리형
2. if
    1. 개요, 조건식에서 and, or의 연산 특성
    2. if,
    3. if-else
    4. if-elif-else
3. for
    1. for
    2. for-list,
    3. for-range,
    4. for-tuple,
    5. for-dictionary
4. while
    1. while
    2. while- break, continue
5. 중첩
6. 연습문제



제어문의 개념

- 프로그램은 기본적을 작성된 명령들의 순서대로 하나씩 순차실행 된다. 그러나 어느 조건에 따라 다른 처리를 하여야 한다면 명령의 실행 순서를 순차실행이 아니라 조건에 따라 해당되는 명령이 실행되게 하는 방법이 필요한다. 이러한 실행의 순서를 제어하기위 준비된 명령의 문장을 제어문이라 한다.
- 즉, 제어문은 프로그램의 실행 흐름의 제어하는 명령이다.
- 조건식과 논리형(Boolean)
  - 조건을 판단하기 위해서 조건식과 조건식의 연산결과를 표현해야 한다. 성년 여부를 판단하는 조건식과 연산결과는 다음과 같다.
    - 나이 = 10 #나이
    - 성년 = 나이>19 #조건식 '나이>19'의 연산결과 False가 Boolean형의 변수 '나이'에 저장된다.
- if 문
- for 반복문
  - for-list, for-tuple, for - dict
- while 반복문

1. 논리형(Boolean)

- 개요
  - 정수형,실수형,문자열형,리스트,튜플,사전형
  - 논리형(Boolean)은 참과(True)와 거짓(False)의 두 값을 갖는 자료형이다.
    - 참 : True, 0 또는 0.0이 아닌 모든 수
    - 거짓 : False, 0 또는 0.0
  - 논리형의 값은 표와 같은 연산자로 조건을 점검하는 데 사용된다.

```
>>> a=True
>>> type(a)
<class 'bool'>
>>> a
True
>>>
>>> 3==3
True
>>> 3!=3
False
>>> 3>3
False
>>> 3<3
False
>>> 3>=3
True
>>> 3<=3
True
>>>
```

연산자	연산자 의미
==	같다.
!=	다르다.
>	크다.
<	작다.
>=	크거나 같다.
<=	작거나 같다.

1. 논리형(Boolean)

- 조건식에서 and, or의 연산 특성
  - 조건식에서 and와 or의 연산결과는 True와 False만을 반환하는 것이 아니라 연산에 사용되는 수식들의 연산의 순서에 따라 다른 값을 반환한다.
    - A and B and C
      - A 가 False이면 False를, A가 True이고 B가 False이면 False 를, A,B 모두 True이면 C의 값을 반환한다.
    - A or B or C
      - A 가 True이면 A 를, A가 False이고 B가 B이면 True 를, A,B 모두 False이면 C의 값을 반환한다.
    - and와 or는 lazy evaluation을 사용하기때문에, 중간 단계에서 값이 결정되면 더 이상 evaluation을 하지 않는다.

```
>>> 3 and 4
4
>>> 3 and 0 and 5
0
>>> 0 or (3==4)
False
>>> (3 != 4) and 3
3
>>> (3!=3) and 5
False
```

```
>>> 3 or 4
3
>>> 0 or 0 or 0
0
>>> (3==3) or (4==3)
True
>>> 0 or (3==4)
False
>>>
```

2. if 문 : if, if~else

1. if 문
   - If 문은 조건문에 따른 처리문으로 구성된다. 예를 들어 삼성전자의 주식을 다음의 조건으로 매매하고 저 한다고 하자.
   - 삼성전자의 현재가가 41000원 이하로 떨이지면 매수하고
   - 삼성전자의 현재가가 47000원 이상으로 오르면 매도한다.

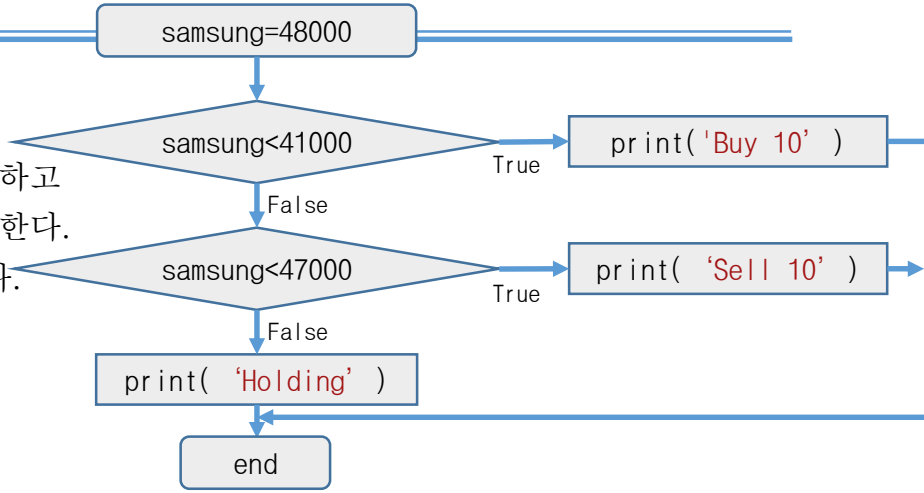
```
>>> samsung=40000
>>> if samsung<41000 :     # 조건식 (samsung<41000)을 연산하고 연산결과에 따라
...         print('Buy 10')     # 참(True)이면 들여쓰기한 문장 print('Buy 10')이 실행된다.
...                             # 거짓(False)이면 들여쓰기한 문장을 실행하지 않고 넘어간다.
Buy 10                          # 조건식(40000<41000)이 참(True)이 되어 print('Buy 10')이 실행된 결과이다.

>>>
```

2. if ~ else 문
   - 삼성전자의 현재가가 41000원 이하로 떨이지면 매수하고
   - 그 보다 높으면 보유고 있어야한다.

```
>>> samsung=45000
>>> if samsung<41000 :    # 조건식(samsung<41000) 을 연산하고 연산 결과에 따라
...         print('Buy 10')     # 참(True)이면 들여쓰기한 문장 print('Buy 10')이 실행된다.
...     else :                  # 거짓(False)이면 else : 다음 줄의 들여쓰기한 문장을 실행한다.
...         print('Holding')    # 거짓(True)이 되어 print('Holding') 이 실행된다.
...
Holding                         # 조건식(45000<41000)이 거짓(False)이 되어 print(Holding')이 실행된 결과이다.
>>>
```

2 if 문 : if~elif~else

3. if ~ elif ~ else 문
   - 삼성전자의 현재가가 41000원 이하로 떨이지면 매수하고
   - 삼성전자의 현재가가 47000원 이상으로 오르면 매도한다.
   - 아니면(41000보다 비싸고 47000보다 싸면)보유한다.




```
>>> samsung=48000
>>> if samsung<41000 :     # 논리 조건식(samsung<41000)을 연산하고 연산 결과가
...         print('Buy 10')      # 참(True)이면 들여쓰기한 문장 print('Buy 10')이 실행된다.
...     elif samsung>47000 :  # 다시 조건식(samsung>47000)을 연산하고 결과가
...         print('Sell 10')     # 참이면 print('Holding') 이 실행된다.
...     else :                   # 아니면(거짓이면) 다음 줄의 들여쓰기한 문장을 실행한다.
...         print('Holding ')    # 아니면 print('Holding') 이 실행된다.
...
Sell 10                          # 조건연산 (48000>47000) 이 참이므로 print('Sell 10')이 실행된 결과이다.
>>>
```

3. for : for – list, range

1. for –list, range
   - For 문은 반복적인 작업의 처리하는 문장이다.
     - 1부터 10까지 출력하는 프로그램의 예를 들어 설명해보자. 3 가지 방법이 가능하다.
       - 다른 요소들의 요소를 반복처리
       1. for 문으로
          1. Print(1);print(2)을 10번 반복 출력한다.
       2. for–list문으로
          1. 다른 요소를 [0,1,2,3,4,5,6,7,8,9,10]로 리스트를 만들고
          2. 반복작업 : 리스트에 있는 요소를 하나씩 꺼내어 반복 출력
       3. for–range문으로
          1. 다른 요소들을 [0,1,2,3,4,5,6,7,8,9,10]의 리스트 대신에 range(0,11)로 만들고
          2. 반복작업 : 리스트에 있는 요소를 하나씩 꺼내어 반복 출력
   - 반복처리는
     - 다른 요소들은 리스트에 담아 놓고
     - 하나씩 꺼내어 반복 처리한다.

```
>>> print(1)
1
>>> print(2)
2
>>> print(3)
3
>>> print(4)
4
>>> print(5)
5
>>> print(6)
6
>>> print(7)
7
>>> print(8)
8
>>> print(9)
9
>>> print(10)
10
>>>
```

```
>>> for i in [0,1,2,3,4,5,6,7,8,9,10] :
...         print(i)
...
0
1
2
3
4
5
6
7
8
9
10
>>>
```

```
>>> for i in range(0,11):
...         print(i)
...
0
1
2
3
4
5
6
7
8
9
10
>>>
```

3. for 문: for - tuple

2. for – tuple
   - 삼성전자, 네이버, 다음의 3 종목의 주식을 각각 10주씩 사는 코드를 작성한다
     - 다른 요소: 종목이름, 반복작업:요소를 10주씩 매수
     - 다른요소는 리스트에 담을 수도 있지만 튜플로도 가능하다.
     - stocks=('Samsung','Naver','Daum') #튜플
     - stocks=['Samsung','Naver','Daum'] #리스트

```
>>> stocks=('Samsung','Naver','Daum') # 튜플형 변수에 종목 대입
>>> for company in stocks:                # 튜플의 요소 반복 참조
...         print('Buy 10 stocks of {}'.format(company))
...
Buy 10 stocks of Samsung
Buy 10 stocks of Naver
Buy 10 stocks of Daum
>>>
```

   - 삼성,네이버,다음 주식을 각각 100,50,10주 사는 코드 ?
     - 다른 요소들
       - 종목이름과 매수수가 종목마다 다르므로 즉 두가지 정보를 담을 수 있는 자료구조 ?
         - 리스트,튜플 ?
         - 사전형 ?

3. for 문 : for- dictionary

3. for dictionary(사전)
   - 삼성전자,네이버,다음 주식을 각각 100,50,10주 사는 코드
     - 다른 요소들
       - (삼성전자,100),(네이버,50), (다음,10) => 사전형
       - stocks={'Samsung':100,'Naver':50,'Daum':10}
     - 반복참조
       - for company,num in stocks.items():
       - for company in stocks.keys():
         stocks[company], company

```
>>> stocks={'Samsung':100,'Naver':50,'Daum':10}
>>> for company,num in stocks.items():
...         print('Buy {} stocks of {}'.format(num,company))
...
Buy 100 stocks of Samsung
Buy 50 stocks of Naver
Buy 10 stocks of Daum
>>>
```

```
>>> stocks={'Samsung':100,'Naver':50,'Daum':10}
>>> for company in stocks.keys():
...         print('Buy {} stocks of {}'.format(stocks[company], company))
...
Buy 100 stocks of Samsung
Buy 50 stocks of Naver
Buy 10 stocks of Daum
>>>
```

4. while

- 반복적인 작업은 for 말고도 while 을 이용할 수 있다.
  - while : 반복 조건을 조건식으로 마음대로 프로그래머가 정의할 수 있다
  - for : 리스트의 요소 수 만큼 반복 횟수가 결정된다.
  - 1부터 10을 출력하는 코드 while로도 가능하다.

```
>>> i=0
>>> while i<=10 :
...         print(i)
...         i=i+1
...
0
1
2
3
4
5
6
7
8
9
10
>>>
```

```
>>> for i in range(0,11):
...         print(i)
...
0
1
2
3
4
5
6
7
8
9
10
>>>
```

- 삼선전자주식이 5일간 상한가를 치면 얼마가 될까?
  - =>while 문이 유리하다.

4. while

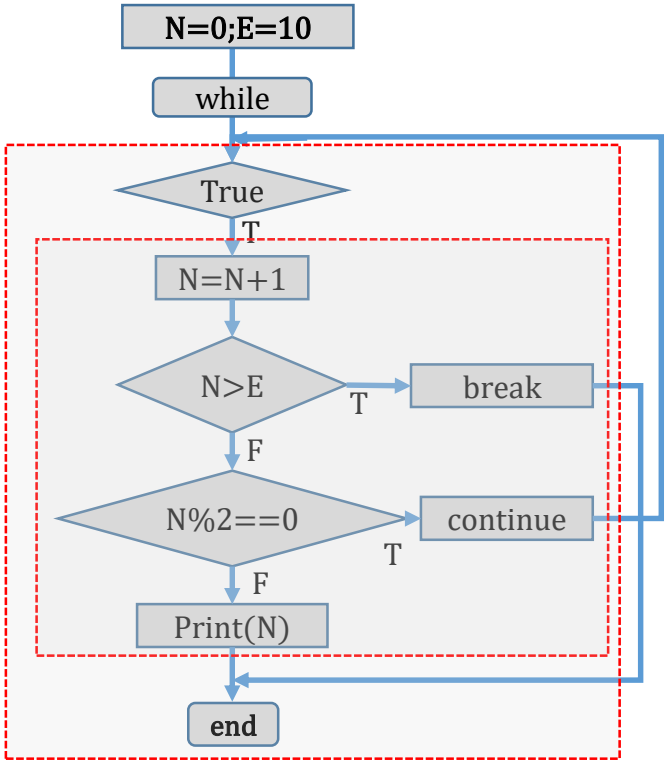
- 삼선전자주식이 5일간 상한가를 치면 얼마가 될까?
  - samsung=40000원이라면 다음 날에는 상한가가 15% 오른다고 가정하면
    samsung= samsung + samsung*0.15
    samsung= samsung + samsung*0.15
    …
  - 조건식의 연산결과가
    참이면 다음줄의 코드 블럭이 실행되고
    거짓이면 코드 블럭의 담으로 이동한다.
  - 정수형 변수가 0부터 4일때까지는 코드 블럭의 실행을 반복한다.

```
>>> samsung =40000
>>> day=0
>>>#조건식 day<5가 참이면 다음 들여쓰기 블럭의 코드가 실행된다.
>>> while day<5:
 ...          samsung = samsung+samsung*0.15
...          day=day+1
...          print(day,samsung)
...
1 46000.0
2 52900.0
3 60835.0
4 69960.25
5 80454.2875
>>>
```

4. while - break, continue

- While– break, continue
  - num부터 end 에 들어있는 수중에서 홀수를 출력하라

```
>>> num=0;end=10
>>> while True :
...     num+=1
...     if num>end:          #terminate process
...         Break            # block를 벗어난다.
...     if num % 2 == 0: #skip even number
...         Continue         # goto while문
...     print(num)           #print odd number
...
1
3
5
7
9
```



5. 연습문제

1. print('*',end='')은 *” 를 출력하고 다음과 같이 줄바꿈을 하지 않는다. print(‘*’ ,end= ‘’) 을 반복하는 방식으로 다음 8가지 패턴을 출력하는 프로그램을 작성하세요.

```
>>> print('*',end='')
*
>>> print('*',end='');print('*',end='')
**
```

A)

```
*****
```

B)

```
*****
*****
*****
*****
```

C)

```
*
**
***
****
*****
```

D)

```
*****
****
***
**
*
```

E)

```
    *
   **
  ***
 ****
*****
```

F)

```
*****
****
***
**
*
```

G)

```
    *
   ***
  *****
 *******
*********
```

H)

```
*********
*******
*****
***
*
```

2. 다음은 구구단 2단을 출력하는 프로그램이다. 리스트변수 dans=[2,3]에 대하여 즉 2단 3단을 다음과 같이 출력하는 프로그램을 작성하세요.

```
>>> for i in range(1,10) :
... print('2 x {} = {}'.format(i,i*2))
...
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
2 x 4 = 8
2 x 5 = 10
2 x 6 = 12
2 x 7 = 14
2 x 8 = 16
2 x 9 = 18
```

```
2 x 1 = 2       3 x 1 = 3
2 x 2 = 4       3 x 2 = 6
2 x 3 = 6       3 x 3 = 9
2 x 4 = 8       3 x 4 = 12
2 x 5 = 10      3 x 5 = 15
2 x 6 = 12      3 x 6 = 18
2 x 7 = 14      3 x 7 = 21
2 x 8 = 16      3 x 8 = 24
2 x 9 = 18      3 x 9 = 27
```