제어문

Update : 2018.09

Contents

- 비교 연산자
- 논리 연산자
- 제어문
- if
- if.. Else..
- if 조건문과 in List / not in List
- pass 키워드
- elif
- while 문
- break
- continue
- for문
- for문과 range()
- for 문과 append() -> 구구단

비교 연산자

비교연산자	설명
x < y	x가 y보다 작다
x > y	x가 y보다 크다
x == y	x와 y가 같다
x != y	x와 y가 같지 않다
x >= y	x가 y보다 크거나 같다
x <= y	x가 y보다 작거나 같다

```
>>> x = 3
>>> y = 2
>>> x > y
True
>>> x < y
False
>>> x == y
False
>>> x != y
True
```

논리 연산자

논리 연산자 : and, or, not

연산자	설명
x or y	x와 y 둘중에 하나만 참이면 참이다
x and y	x와 y 모두 참이어야 참이다
not x	x가 거짓이면 참이다

제 어 문

- 제어문이란?

  - 조건문, 반복문

  - 프로그램의 흐름을 제어해서 효율적으로 이용하기 위한 명령문

  - if, if~else, if~elif~

  - for ~

  - while ~

If 문

들여쓰기(Indentation)

```
if 조건문:
    수행할 문장1
    수행할 문장2
    수행할 문장3
```

들여쓰기 오류

```
if 조건문:
    수행할 문장1
수행할 문장2
    수행할 문장3
```

X

```
>>>
money = 2000
if money:
        print('택시를')
    print(' 타고')
print('가자')


>>>
print(' 타고')
                ^
IndentationError: unindent does
not match any outer indentation
level
```

If 문

None, '' 가능

```
money = 0
if money:
    print('택시를')
    print(' 타고')
print('가자')
```

```
money = 2000
if money:
    print('택시를')
    print(' 타고')
print('가자')
```

True 조건이 되려면?

: 0, None, '' 값 외. 음수도 가능.

If … else 문

돈이 있으면 택시를 타고,
돈이 없으면 걸어 간다

if문의 기본구조

```
if 조건문:
    수행할 문장1
    수행할 문장2
    ...
else:
    수행할 문장A
    수행할 문장B
    ...
```

```
>>> money = 1
>>> if money:
...     print("택시를 타고 가라")
... else:
...     print("걸어 가라")
...

>>> 택시를 타고 가라
```

If … else 문

만약 3000원 이상의 돈을 가지고 있으면 택시를 타고 그렇지 않으면 걸어 가라

```
>>> money = 2000
>>> if money >= 3000:
...     print("택시를 타고 가라")
... else:
...     print("걸어가라")
...
걸어가라
```

If … else 문

돈이 3000원 이상 있거나 카드가 있다면 택시를 타고 그렇지 않으면 걸어 가라

```
>>> money = 2000
>>> card = 1  # True
>>> if money >= 3000 or card:
...     print("택시를 타고 가라")
... else:
...     print("걸어가라")
...
택시를 타고 가라
```

If … else 문

a 값이 홀수인지 짝수인지 표시하여라.

```
>>>
a = 3
# if a%2 != 0 :
# if a%2 == 1 :
if a%2 :
    print('홀수')
else:
    print('짝수')
```

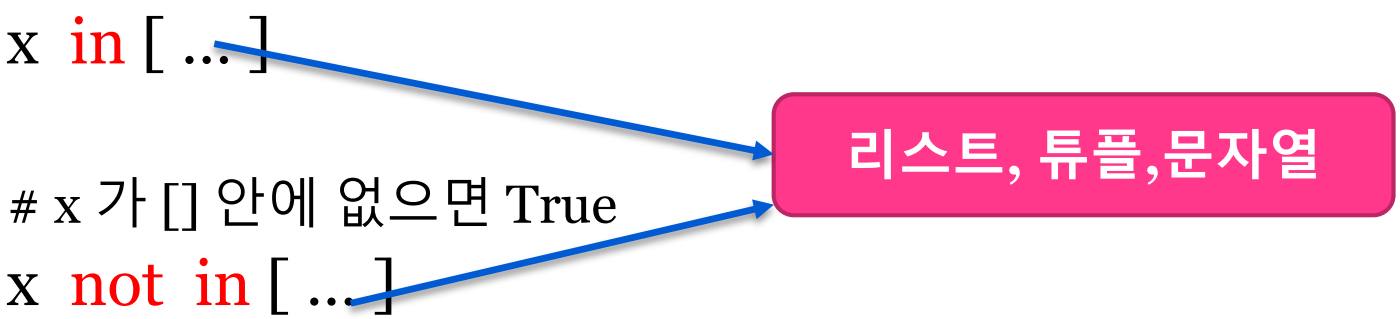
If 문 – in List / not in List

기본 구조

```
# x 가 [] 안에 있으면 True
x  in [ ... ]

# x 가 [] 안에 없으면 True
x  not  in [ ... ]
```

```
>>> 1 in [1, 2, 3]
True
>>> 1 not in [1, 2, 3]
False
```



If 문 – in List / not in List

만약 주머니에 돈이 있으면 택시를 타고, 없으면 걸어 가라

```
>>> pocket = ['paper', 'cellphone', 'money']
>>> if 'money' in pocket:
...     print("택시를 타고 가라")
... else:
...     print("걸어가라")
...
택시를 타고 가라
>>>
```

elif 문

다중 조건 판단 elif 구조

```
if 조건문:
    수행할 문장1
    수행할 문장2
    ...
elif 조건문:
    수행할 문장a
    수행할 문장b
    ...

else:
    수행할 문장a
    수행할 문장b
    ...
```

```
>>> pocket = ['paper', 'cellphone']
>>> card = 1
>>> if 'money' in pocket:
...     print("택시를 타고가라")
... elif card:
...     print("택시를 타고가라")
... else:
...     print("걸어가라")
...
택시를 타고가라
```

다른언어의 switch case문과 비슷한 구조

elif 문

음수인지 양수인지 0인지 알려주세요.

```
>>>
a = -1
if a<0:
    print('음수입니다.')
elif a>0:
    print('양수입니다.')
else:
    print('0 입니다.')
```

```
>>>
a = -1
 if a<0:
    print('음수입니다.')
if a>0:
    print('양수입니다.')
if a==0:
    print('0 입니다.')
```

```
>>>
if a<0:
    print('음수입니다.')
else:
    if a>0:
        print('양수입니다.')
    else:
        print('0 입니다.')
```

elif 문

```
myWay = ['bus', 'subway', 'taxi']
myStatus = 0

if 'my Car' in myWay:
    print('내 차를 타고 간다.')
elif myStatus:
    print('버스를 타고 간다.')
elif 'subway' not in myWay:
    print('지하철을 타고 간다. ')
elif 'school bus' not in myWay:
    print ('스쿨 버스를 타고 간다.')
else:
    print('걸어간다.')
```

If 문과 pass 키워드

조건문에서 아무 일도 하지 않게 설정하고 싶다면?

```
if 조건문:
    pass
```

```
>>> pocket = ['paper', 'money', 'cellphone']
>>> if 'money' in pocket:
...     pass
... else:
...     print("카드를 꺼내라")
...
```

while 문

while문의 기본 구조

```
while <조건문>:
    <수행할 문장1>
    <수행할 문장2>
    <수행할 문장3>
    ...
```

```
<초기값>
while <조건문>:
    <수행할 문장>
        …
    <False값이 되는 조건문>
```

열 번 찍어 안 넘어 가는 나무 없다

```
>>> treeHit = 0
>>> while treeHit < 10:
...     treeHit = treeHit +1
...     print("나무를 %d번 찍었습니다." % treeHit)
...     if treeHit == 10:
...         print("나무 넘어갑니다.")
...
나무를 1번 찍었습니다.
나무를 2번 찍었습니다.
나무를 3번 찍었습니다.
…
나무 넘어갑니다.
```

while 문

무한 루프

```
While True:
    <수행할 문장1>
    <수행할 문장2>
    <수행할 문장3>
    ...
```

열 번 찍어 안 넘어 가는 나무 없다

```
treeHit = 0
while True:
    treeHit += 1
    print('나무를 %d번 찍었습니다.' % treeHit)
```

나무를 1번 찍었습니다.

나무를 2번 찍었습니다.

나무를 3번 찍었습니다.

...

break 문

break
- 제어문의 블록을 탈출할 때 사용한다.

```
>>> coffee = 10
>>> money = 300
>>> while money:
...     print("돈을 받았으니 커피를 줍니다.")
...     coffee = coffee -1
...     print("남은 커피의 양은 %d개입니다." % coffee)
...     if not coffee:
...         print("커피가 다 떨어졌습니다. 판매를 중지합니다.")
...         break
...
```

break 문

break
- 제어문의 블록을 탈출할 때 사용한다.

```
m = 100
n = 10
while m:
        n = n -1
        print("현재 %d 입니다" %n)

        if not n:
                print("n의 값은 0 입니다")
                break #while문을 빠져나간다.
```

break 문

break
- 제어문의 블록을 탈출할 때 사용한다.

```
# 1에서 부터 10까지의 정수중 홀수를 출력하는 코드
i = 0
while i < 10:
        i = i+1
        if i % 2 == 0 : continue
        #continue문은 while문의 맨처음 (i<10)으로 돌아가게하는 명령문
        print(i)
```

continue문

continue
: 실행문을 실행하지 않고 다음 명령문을 실행한다.

```
>>> a = 0
>>> while a < 10:
...     a = a+1
...     if a % 2 == 0: continue
...     print(a)
...
1
3
5
7
9
```

for 문

for문의 기본 구조

```
for 변수 in 리스트(또는 튜플, 문자열):
    수행할 문장1
    수행할 문장2
    ...
```

```
>>> test_list = ['one', 'two', 'three']
>>> for i in test_list:
...     print(i)
...
one
two
three
```

for 문

60점이 넘으면 합격이고 그렇지 않으면 불합격

```
marks = [90, 25, 67, 45, 80]

number = 0
for mark in marks:
    number = number +1
    if mark >= 60:
        print("%d번 학생은 합격입니다." % number)
    else:
        print("%d번 학생은 불합격입니다." % number)
```

for 문

?

다양한 for문의 사용
: 튜플 이용

```
aa = [('a', 'b'), ('c', 'd'),
('e', 'f')]
for (i, j) in aa:
        print(i+j)
```

ab
cd
ef

for 문

다양한 for문의 사용
: 튜플 이용

?

```
# 2차원 리스트
>>> a = [(1,2), (3,4), (5,6)]
>>> for (first, last) in a:
...     print(first + last)
...
3
7
11
```

```
# 3차원 리스트
myList = [(1,2,3), (5,6,7)]
for (n1, n2, n3) in myList:
    print(n1+n2+n3)



6
18
```

for 문

for문과 continue

```
marks = [90, 25, 67, 45, 80]

number = 0
for mark in marks:
    number = number +1
    if mark < 60: continue
    print("%d번 학생 축하합니다. 합격입니다. " % number)
```

for 문과 range()

for와 함께 자주 사용하는 range함수

```
for 변수 in range(시작값, 끝값-1, 증감치):
    수행할 문장1
    수행할 문장2
    ...
```

- range 함수는 두개, 세개의 숫자를 이용하는 함수이다.
  숫자가 두개인 경우 1씩 증가하는 숫자의 리스트를 반환한다.
  숫자가 세개인 경우 세번 째 숫자는 증가치를 의미한다.

  사용예>
range(1,5)
---> 리스트 [1,2,3,4] 반환한다.
range(1,5,2) ---> [1,3]
//2씩 증가 시킨다.

```
>>> sum = 0
>>> for i in range(1, 11):
...     sum = sum + i
...
>>> print(sum)
55
```

```
for i in range(0,21,2):
    if i == 0:
        pass
    else:
        print(i,'\t',end='')

>>>  2  4  6  8  10  12  14  16  18  20
```

for 문과 range()

구구단

```
>>> for i in range(2,10):
...     for j in range(1, 10):
...         print(i*j, end=" ")
...     print('')
...
2 4 6 8 10 12 14 16 18
3 6 9 12 15 18 21 24 27
4 8 12 16 20 24 28 32 36
5 10 15 20 25 30 35 40 45
6 12 18 24 30 36 42 48 54
7 14 21 28 35 42 49 56 63
8 16 24 32 40 48 56 64 72
9 18 27 36 45 54 63 72 81
```

```
# 문제
# 0~24 한줄에 5개씩

0 1 2 3 4
5 6 7 8 9
10 11 12 13 14
15 16 17 18 19
20 21 22 23 24
```

```
for i in range(0,25,1):
    print(i,' ', end='')
    if (i+1)%5==0:
        print()
```

for 문과 append() -> 구구단

구구단
: 리스트안에 for문을 이용하여 프로그램 해보기

```
aa = [1,2,3,4,5,6,7,8,9]

gugudan_2 = []

for i in aa:
        gugudan_2.append(i*2)
```

for 문 과 리스트 내포

리스트 내포(List comprehension)

```
[listItem  Command for listItem in range(startNum, endNum)]
```

리스트 내포(List comprehension)의 기본적인 구조

1) [표현식 for 항목 in 반복가능한 객체 if 조건]
   위에서 if조건은 생략이 가능하다.

2) [표현식 for 항목1 in 반복가능한객체1 if 조건1
        for 항목2 in 반복가능한객체2 if 조건2

```
#구구단의 결과를 저장하는 리스트 구현
gugudan = [i*j for i in range(2,10) for j in range(1,10) ]
print(gugudan)
```

for 문 과 리스트 내포

리스트 내포(List comprehension)

```
[listItem  Command for listItem in range(startNum, endNum)]
```

```
# 3의 배수를 출력하여 리스트로 구성하여라. (5개)
mylist = [num*3 for num in range(1,6) ]
print(mylist)
```

```
# 구구단 값을 리스트로 구성하여라
print( [ x*y for x in range( 2, 10 ) for y in range( 1, 10 ) ] )
```

```
# 5단의 결과값 중에서 짝수만 리스트에 추가하는 예
gugudan_5 = [i*5 for i in aa if i %2 == 0]
print (gugudan_5)
```