Python Basic Review II: 제어 함수와 프로그래밍

AI빅데이터프로그래밍

Prof.Hyunsil Moon (hsmoon@kookmin.ac.kr)

1.
조건문

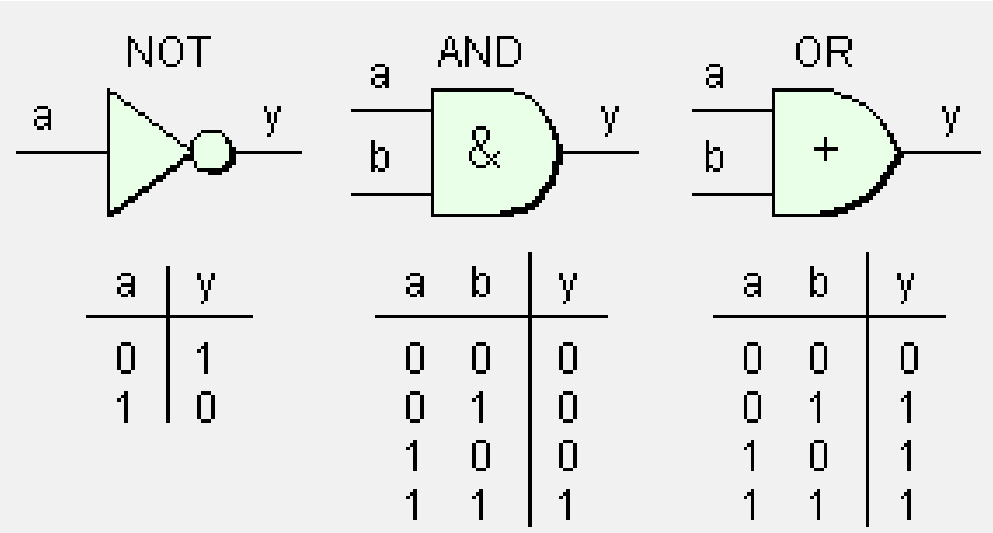
1 명제

- 1 + 3 = 4
- 12 – 4 = 7
- 남산타워는 서울에 있다.
- 모짜르트는 음악 천재이다.
- 영어책은 얇다

2 관계 연산자(relational operator) / 논리 연산자

연산자	의미
x == y	x와 y가 같다
x != y	x와 y가 다르다
x > y	x가 y보다 크다
x >= y	x가 y보다 크거나 같다
x < y	x가 y보다 작다
x <= y	x가 y보다 작거나 같다

연산자	의미
X and Y	X도 참이고 그리고 Y도 참이다
X or Y	X가 참이거나 또는 Y가 참이다
not X	X가 참이 아니다



a	y
0	1
1	0

a	b	y
0	0	0
0	1	0
1	0	0
1	1	1

a	b	y
0	0	0
0	1	1
1	0	1
1	1	1

3 조건문 : if

```
if x > 3:
    print('크다')
    y = x + 1
```

헤더(header)

스위트(suite)

- x 가 3 보다 클 때만 print('크다') 를 실행
- 조건문은 헤더와 스위트로 구성
  - 헤더는 if <조건>: 의 구조
  - 스위트는 조건이 참(True)일 때 실행할 코드로 들여쓰기 하여 표시

3 조건문 : if

```
if x>3:
        print('크다')
else:
        print('작다')
```

- x가 3 이하일 때는 '작다'를 출력
- else : if의 조건이 참이 아닐 때 실행
  - 들여쓰기에 주의 : else는 if와 줄을 맞춤

3 조건문 : if

```
if x>3:
        print('크다')
elif x==3:
        print('같다')
else:
        print('작다')
```

- elif : if의 조건이 참이 아닐 때 다른 조건을 확인
- elif는 여러 개 사용 가능

2.
반복문

1 반복문의 개념

- 반복문(loop)
  - 정해진 동작을 반복적으로 수행할 때 내리는 명령
- 반복문은 프로그래밍의 핵심
  - 반복 시작 조건
  - 종료 조건
  - 수행 명령
  - 들여쓰기와 블록(block)으로 구성

2 for 문

- 자료 구조에 포함된 값들에 대해 처리를 반복

```
xs = [17, 21, 35]
for x in xs:
        y= x% 3
        print(y)
```

- range() 함수의 사용
  - 일정 범위의 정수 수열을 생성
  - range(e) : 0에서 시작하여 e까지(e는 포함 X)
  - range(b, e) : b에서 시작하여 e까지(e는 포함 X)
  - range(b, e, s) : b에서 시작하여 e까지, s간격으로(e는 포함 X)

3 while 문

- 조건이 참(True)인 동안 반복
  - if문과 비슷한 구조
  - 무한 반복을 하지 않도록 주의!

```
x = 0
while x<3:
        print(x)
        x = x+1
```

4 반복문의 제어

- break
  - 반복문을 중단시킴
- for ... else ...
  - for문을 완료한 후 한가지 더 실행할 때 사용
  - break로 for문이 중단된 경우에는 else를 실행하지 않음
- continue
  - 진행을 중단하고 다음 반복으로 넘어감

3.
리스트 컴프리헨션

1 리스트 컴프리헨션(List Comprehension)

- 기존 리스트형을 사용하여 간단하게 새로운 리스트를 만드는 기법
  - 포괄형 리스트, 포함형 리스트, 지능형 리스트, 축약형 리스트
  - 반복문을 한줄에 사용

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

```
>>> result = [i for i in range(10)]
>>> result
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

2 리스트 컴프리헨션(List Comprehension)을 이용한 필터링

- if문과 함께 사용하여 코드를 축약

```
>>> result = []
>>> for i in range(10):
...     if i % 2 == 0:
...         result.append(i)
...
>>> result
[0, 2, 4, 6, 8]
```

```
>>> result = [i for i in range(10) if i % 2 == 0]
>>> result
[0, 2, 4, 6, 8]
```

- else문과 함께 사용도 가능
  - 조건을 만족하지 않을 때 else 뒤에 값을 할당

```
>>> result = [i if i % 2 == 0 else 10 for i in range(10)]
>>> result
[0, 10, 2, 10, 4, 10, 6, 10, 8, 10]
```

3 enumerate()와 zip()

- enumerate() 함수
  - 리스트 값에 인덱스를 붙여 출력
  - 딕셔너리 형태로 활용 : 인덱스를 키로 사용

```
>>> for i, v in enumerate(['tic', 'tac', 'toe']):    # 리스트에 있는 인덱스와 값을 언패킹
...     print(i, v)
...
0 tic
1 tac
2 toe
```

- zip() 함수
  - 1개 이상의 같은 인덱스에 있는 리스트 값을 병렬로 결합하는 함수

```
>>> alist = ['a1', 'a2', 'a3']
>>> blist = ['b1', 'b2', 'b3']
>>> for a, b in zip(alist, blist):                    # 병렬로 값을 추출
...     print(a, b)
...
a1 b1
a2 b2
a3 b3
```

Q&A

Thank you for your listening