파이썬 프로그래밍

파이썬의 각종 연산자

한국기술교육대학교
온라인평생교육원

▣ 산술 연산자

연산자	우선순위	설명	결합순서
+, -	1	단항 연산자	-
**	2	지수 연산자	왼쪽 <- 오른쪽
*,/,%,//	3	곱하기,나누기,나머지,몫	왼쪽 -> 오른쪽
+, -	4	더하기,빼기	왼쪽 -> 오른쪽

```
print 2 ** 3
print 5 % 2
print -5 % 2
```

```
8
1
1
```

- 산술 연산자, 관계 연산자, 논리 연산자
- 산술 연산자: 더하기, 빼기, 곱하기, 나누기와 같은 연산자
- 단항 연산자: 피 연산자가 두 개가 아닌 하나인 연산자
- 단항 연산자는 가장 우선순위가 높아서 먼저 수행
- 지수 연산자: **
- 결합순서: 오른쪽 → 왼쪽
- %: 나머지를 구해주는 연산자
- 나머지가 나올 수 있는 수: 0과 1

```
print 3 + 5
print 3 + 5.0 # 정수 + 실수의 결과는 실수
```

```
8
8.0
```

▣ 산술 연산자

```
print 5 / 2.0   # 정수 / 실수의 결과는 실수
print 5 / 2
```

```
2.5
2
```

- 정수/정수 = 정수

```
a = 5 / 3
b = 5 % 3

print a, b

print divmod(5,3)
```

```
1 2
(1, 2)
```

- 5/3=1.66666…… = 1
- divmod: 내장함수
- divmod: 몫과 나머지를 tuple 형태로 돌려주는 함수

```
print 5 / 3
print 5 // 3
```

```
1
1
```

- //: %에 대응되는 연산자 = 몫을 구해주는 연산자
- / = 소수점 이하를 버리는 것. // = 몫을 구해서 출력

▣ 산술 연산자

단항 연산자(-)의 우선순위가 이항 연산자(/)의 우선순위보다 높다

```
print -7/4    # -7을 4로 나눈다
print -(7/4)
```

```
-2
-1
```

- -1.666666..... 보다 더 작은 정수를 찾는 것 = -2
- 1.66666..... 보다 더 작은 정수 중에서 가장 큰수 = 1

```
print 2 + 3 * 4
print (2 + 3) * 4
```

```
14
20
```

```
print 4 / 2 * 2
```

```
4
```

** 연산자의 결합순서는 오른쪽에서 왼쪽

```
print 2 ** 3 ** 2
print (2 ** 3) ** 2
```

```
512 6
4
```

- 지수연산자의 결합순서: 오른쪽 → 왼쪽

파이썬 프로그래밍

파이썬의 각종 연산자

한국기술교육대학교
온라인평생교육원

▣ 관계 연산자

관계 연산자: 객체가 지닌 값의 크기(대소)를 비교하여 True 또는 False를 반환함

```
print 6 == 9
print 6 != 9
print 1 > 3
print 4 <= 5

a = 5
b = 10
print a < b
```

```
False
True
False
True
True
```

- 관계 연산자: 객체가 지닌 값의 크기(대소)를 비교하여 ture, false를 반환
- ==: 두 개의 객체가 동일한지를 판단하는 연산자
- 양쪽에 있는 두 객체의 값이 동일해야 ture
- !=: 두 개의 객체 값이 달라야 true
- 1이 3보다 커야하는데 그렇지 않으니 false
- 4가 5보다 작거나 같으므로 true

```
a = 5
b = 10
print 0 < a < b
```

```
True
```

- 0<a<b 가 옳은 문법으로 true가 나옴

▣ 관계 연산자

```
a = 5
b = 10
print 0 < a and a < b
```

```
True
```

- 0<a<b = 0<a and <b
- and : 논리 연산자

```
• 문자열, 튜플, 리스트의 관계 연산 비교는 일반 사전 순서
• 사전에서 앞에 나오는 값이 작은 값으로 평가됨
```

```
print 'abcd' > 'abd'
print (1, 2, 4) < (2, 1, 0)
print [1, 3, 2] == [1, 2, 3]
```

```
False
True
False
```

- 문자열과 문자열을 비교
- tuple과 tuple을 비교
- list와 list 비교
- 내용은 1,2,3으로 같지만 순서가 달라서 동일하지 않음
- 내용이 동일하므로 ture
- 사전 순서로 대소관계 비교
- c가 d보다 사전순서가 앞서므로 abcd<abd
- 4개문자> 3개문자지만, abcd가 더 작으므로 false
- 1보다 2가 더 뒤에 나오는 값으로 2가 tuple이 더 큼
- 동일한 자료형을 가지고 비교할 때는 '사전 순서'

▣ 관계 연산자

• 서로 다른 자료형간의 크기 관계
  - 숫자 < 사전 < 리스트 < 문자열 < 튜플

```
print 999999999999999999999L < 'abc'
print {3:2} < [1,2,3] < (1,2,3)
```

```
True
True
```

- 자료형이 다르면 자료형 자체가 가지고 있는 크기관계를 비교
- 사전 <list(리스트) <tuple(튜플) = ture

```
L = [1,2,3, 'abc', 'a', 'z', (1,2,3), [1,2,3], {1:2}, ['abc']]
L.sort()
print L
```

```
[1, 2, 3, {1: 2}, [1, 2, 3], ['abc'], 'a', 'abc', 'z', (1, 2, 3)]
```

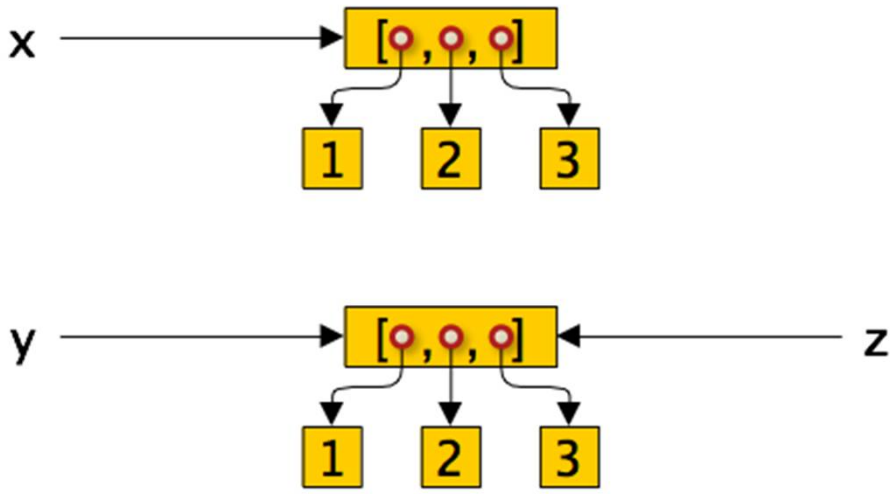
- 숫자 < 문자열
- 문자열 '사전순서'대로 비교

▣ 관계 연산자

```
x = [1,2,3]
y = [1,2,3]
z = y

print x == y
print x == z
print x is y
print x is z
print y is z
```

```
True
True
False
False
True
```



- y가 지니고 있는 레퍼런스 값을 z에 할당하는 것
- is: x의 식별자와 y의 식별자를 비교 = 같은 객체인지 비교
- x, y는 서로 다른 객체 가리킴 = x의 식별자, y의 식별자 다름
- id: 식별자를 돌려주는 내장함수

파이썬 프로그래밍

파이썬의 각종 연산자

한국기술교육대학교
온라인평생교육원

▣ 논리 연산자

• 피연산자의 값으로 진리값인 True 또는 False을 취하여 논리 적인 계산을 수행하는 연산자
 - and
 - or
 - not

•[주의] 논리 연산자 자체가 값을 반환하지는 않는다.
 - 논리 연산을 따라 최종적으로 평가(Evaluation)되어진 값이 반환된다.

```
a = 20
b = 30
print a > 10 and b < 50
```

```
True
```

- and, or, not → 값을 절대 반환하지 않음
- a>10 → true
- 앞 and 뒤: 앞도 true 뒤도 true여야 true
- 앞 and 뒤: 앞이 true고 뒤는 false면 false
- and가 리턴한 것이 아니라 b=50을 확인한 결과가 리턴
- 50 >b가 true이기 때문에 true값이 리턴된 것
- or 연산자는 evaluate 하지 않음
- 앞 or 뒤: 앞이 true 이면 뒤쪽은 볼 것도 없이 true

▣ 논리 연산자

• 진리값에 해당하는 True와 False는 다른 사칙연산자를 만나면 다음과 같이 평가됨
- True: 1
- False: 0

```
print True + 1
print False + 1
print False * 75
print True * 75
```

```
2
1
0
75
```

- true가 1이니까 1을 출력
- false가 0 → 0+1 =1이 출력
- false가 0 → 0*75 = 0이 출력
- true가 1 → 1* 75 = 75 출력

▣ 논리 연산자

• bool() 내장 함수를 이용해서 수치 값을 진리 값으로 교환 가능

```
print bool(0) # 정수 0은 거짓
print bool(1)
print bool(100)
print bool(-100)
print
print bool(0.0) # 실수 0.0은 거짓
print bool(0.1)
```

```
False
True
True
True

False
True
```

- bool(불): 하나 받는 객체를 evaluate하여 true 또는 false로 반환
- 객체는 대부분 true지만 false로 변환되는 객체들이 존재
- 정수값 0은 bool(불) 내장함수에서 false로 evaluate 됨
- 0.0이 아닌 나머지 모든 실수 = true

▣ 논리 연산자

• 값이 없는 빈 객체나 None 객체는 False로 평가됨

```
print bool('abc')
print bool('')
print
print bool([]) # 공 리스트는 거짓
print bool([1,2,3])
print
print bool(()) # 공 튜플은 거짓
print bool((1,2,3))
print
print bool({}) # 공 사전은 거짓
print bool({1:2})
print
print bool(None) # None 객체는 거짓
```

```
True
False

False
True

False
True

False
True

False
```

- 일반적인 문자열(ex. abc) → true
- 공백조차 없는 비어있는 문자열 → false
- 비어있는 tuple, 비어있는 list, 비어있는 사전 → false
- 내용이 하나라도 있으면 → true
- none 객체 → false

▣ 논리 연산자

```
print 1 and 1
print 1 and 0
print 0 or 0
print 1 or 0
print
print [] or 1 # [] 거짓
print [] or () # [], () 거짓
print [] and 1 # [] 거짓이므로 1은 참조할 필요 없음
```

```
1
0
0
1

1
()
[]
```

- 앞 and 뒤 = 앞, 뒤 모두 evaluate한 결과
- 앞 or 뒤: 앞이 false면 뒤쪽도 확인하여 false면 false
- 1 or 0 = 앞이 true 이므로 뒤 확인 생략 = 1 출력
- 비어있는 tuple = false
- false or false = false
- 앞 and 뒤: 앞이 false면 뒤도 안보고 무조건 false

▣ 논리 연산자

```
print 1 and 2
print 1 or 2
print
print [[]] or 1 # [[]] 참으로 간주
print [{}] or 1 # [{}] 참으로 간주
print '' or 1 # 빈 문자열('')은 거짓
```

```
2
1

[[]]
[{}]
1
```

- [[ ]]: 공 list를 가지고 있는 list = 바깥 list는 공list가 아님 = true
- [{ }]: 공 사전을 가지고 있는 list = true
- or 연산자가 true를 반환하지 않고 뒤를 evaluate 한 결과 출력

```
print not(True)
print not(1 and 2)
print not('' or 1)
```

```
False
False
False
```

- not 뒤: 뒤가 false면 true, true면 false로 바꿔 출력
- 연산자 우선순위에 의존보다 적절한 괄호 활용이 필요