



# 파이썬 프로그래밍

# Chapter 03

## 연산자



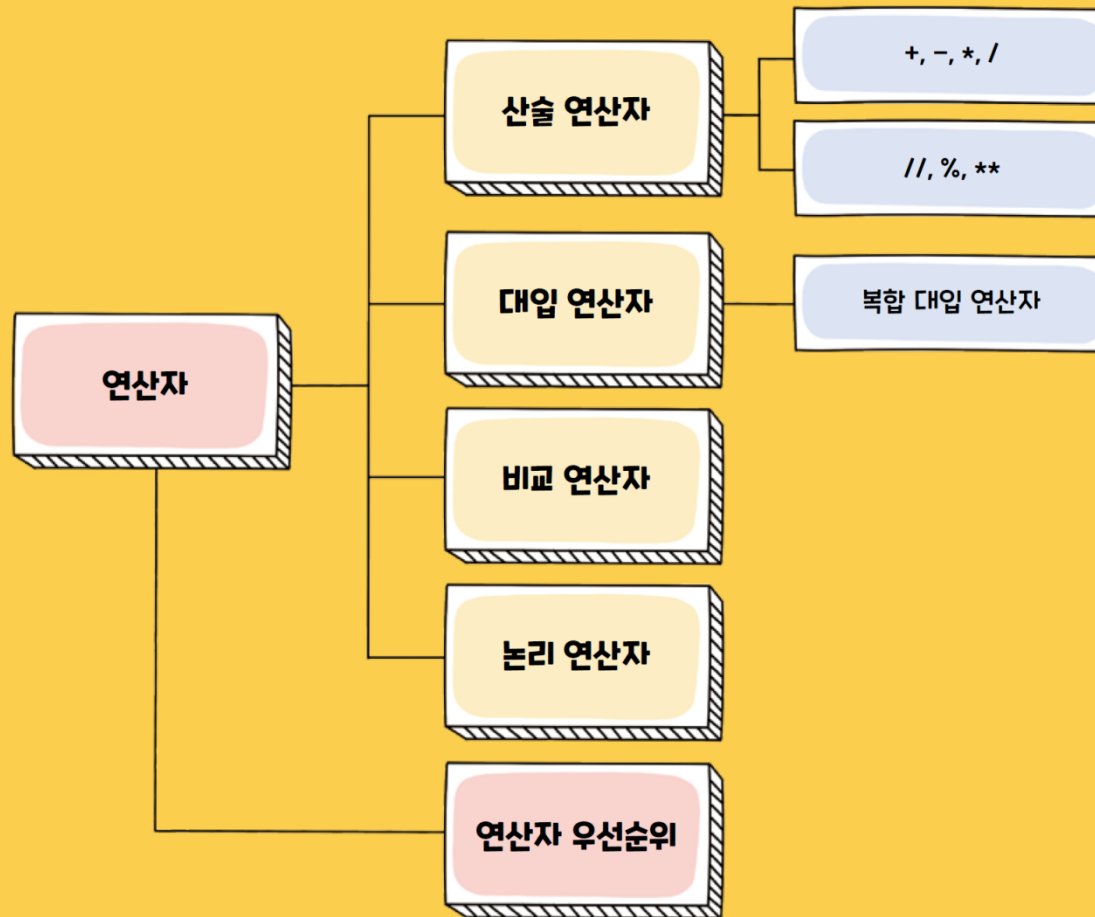
# 목차

1. 산술 연산자
2. 대입 연산자
3. 비교 연산자와 논리 연산자
4. 연산자의 우선순위

[실전 예제] 거북이를 그리는 펜의 변화

[실전 예제] 입력한 값만큼 거북이 움직이기

## Preview



# 학습목표

- 연산자의 개념과 종류를 익힙니다.
- 주요 연산자인 산술, 대입, 비교, 논리 연산자를 이해합니다.
- 연산자의 우선순위를 이해하고 실습합니다.

# Section 01

## 산술 연산자



## ■ 사칙 연산자

- 파이썬에서 사용되는 기본적인 산술 연산자

표 3-1 기본 산술 연산자

연산자	사용 예	설명
+	num = 4+3	4와 3을 더한 값을 num에 대입
-	num = 4-3	4와 3을 뺀 값을 num에 대입
*	num = 4*3	4와 3을 곱한 값을 num에 대입
/	num = 4/3	4를 3으로 나눈 값을 num에 대입

- 더하기 · 빼기 · 곱하기 · 나누기

- n1 = 200, n2 = 150

```
>>> res = n1 + n2
>>> print(res)
350
```

```
>>> res = n1 - n2
>>> print(res)
50
```

```
>>> res = n1 * n2
>>> print(res)
30000
```

```
>>> res = n1 / n2
>>> print(res)
1.33333333333
```



## ■ 산술 연산자의 우선순위

- 여러 개의 연산자가 동시에 계산에 나올 때 어떤 연산자가 먼저 계산되는지 정확히 파악해야 계산의 실수가 발생하지 않음
- 더하기와 빼기가 동시에 나오는 경우

```
>>> a, b, c = 3, 4, 5
>>> print(a + b - c)
2
>>> print(a - c + b)
2
>>> print(-c + a + b)
2
```

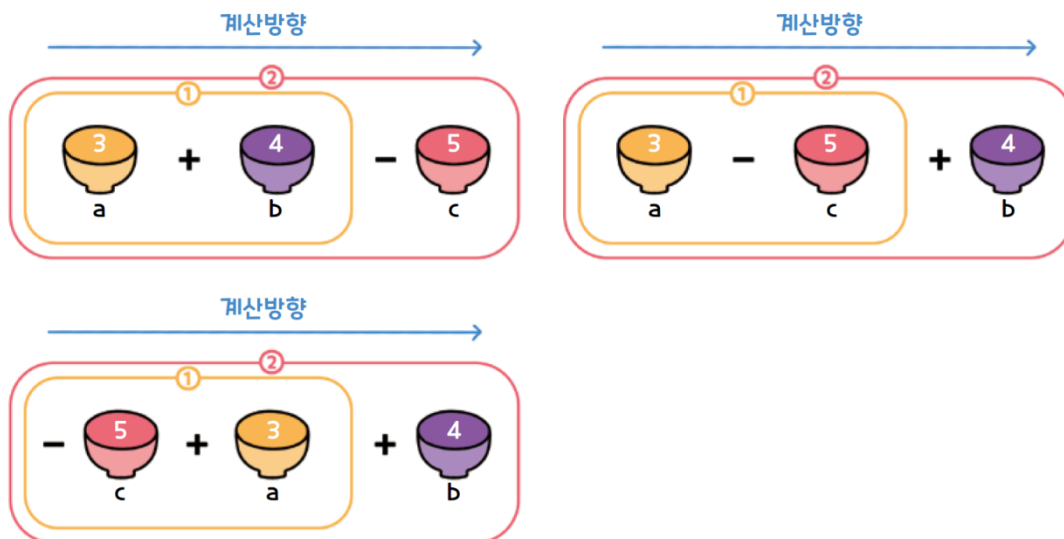


그림 3-1 더하기, 빼기가 여러 개 나올 때의 계산 순서





## ■ 산술 연산자의 우선순위

- 나누기와 곱하기가 동시에 나오는 경우

```
>>> a, b, c = 2, 4, 6
>>> print(a / b * c)
3.0
>>> print(a * c / b)
3.0
>>> print(c / a * b)
3.0
```

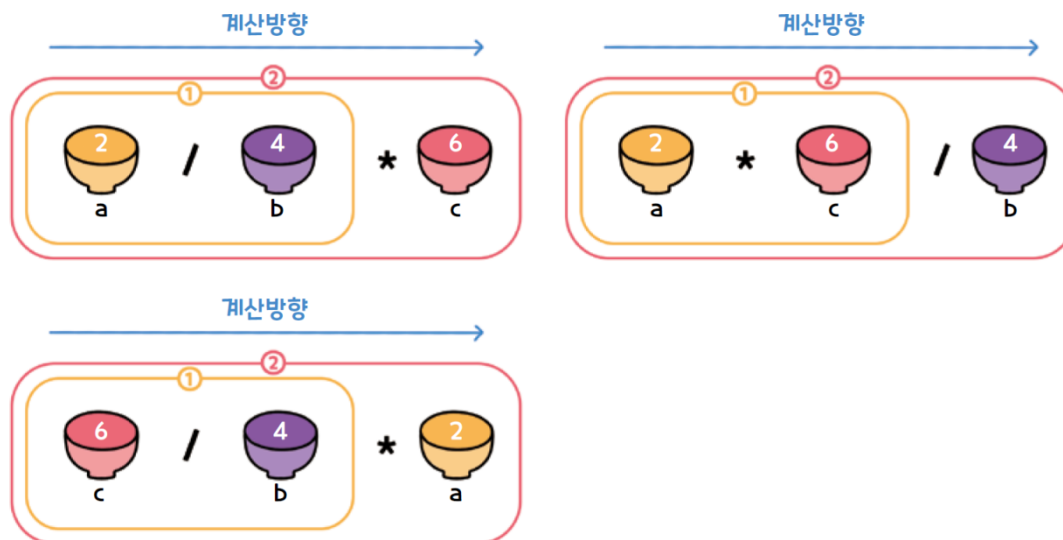


그림 3-2 곱하기, 나누기가 여러 개 나올 때의 계산 순서



## ■ 산술 연산자의 우선순위

- 더하기와 곱하기가 동시에 나오는 경우

```
>>> a, b, c = 3, 4, 5  
>>> print(a * b + c)  
17  
>>> print(c + a * b)  
17
```

- 앞에서부터 순서대로 계산하면 결과가 다르게 나옴

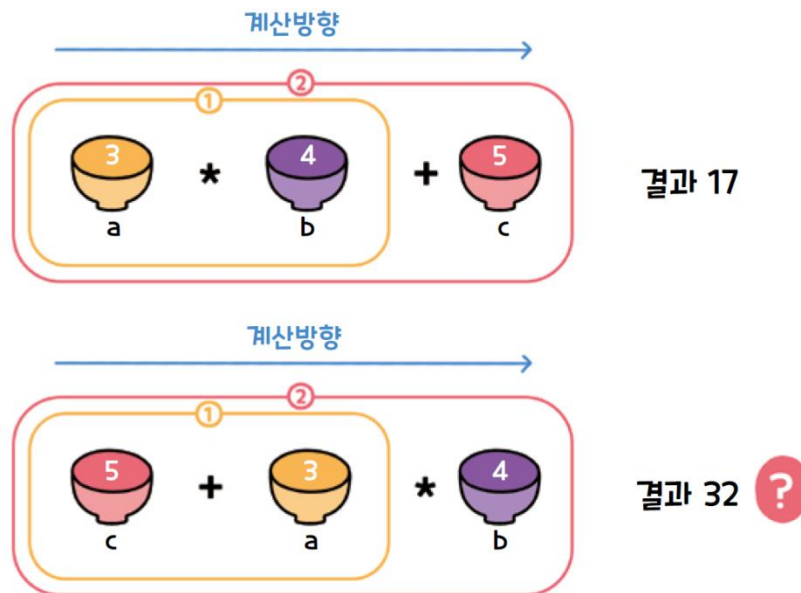


그림 3-3 더하기와 곱하기가 함께 나올 때의 계산 순서 1



## ■ 산술 연산자의 우선순위

- 더하기와 곱하기가 동시에 나오는 경우
  - → 더하기(빼기)와 곱하기(나누기)가 함께 섞여서 나오면, 곱하기(나누기)가 먼저 계산됨

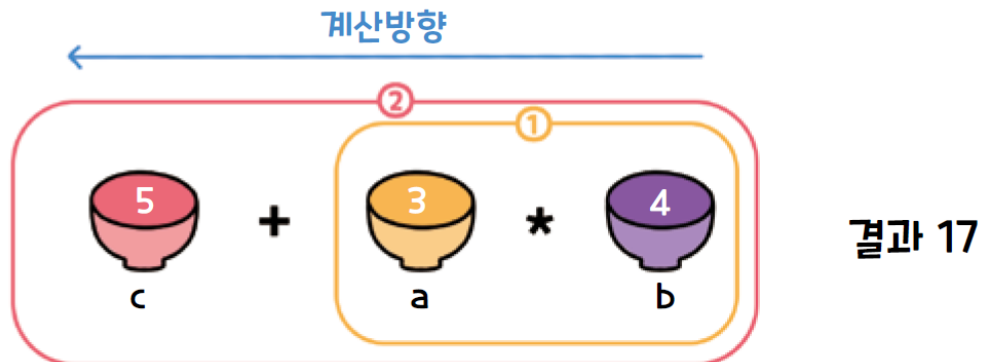


그림 3-4 더하기와 곱하기가 함께 나올 때의 계산 순서 2



## ■ 그 외 연산자

표 3-2 몫, 나머지, 제곱 연산자

연산자	의미	사용 예	설명
//	나누기(몫)	num = 5//3	5를 3으로 나눈 뒤 소수점을 버리고 num에 대입
%	나머지 값	num = 5%3	5를 3으로 나눈 뒤 나머지 값을 num에 대입
**	제곱	num = 5**3	5의 3제곱을 num에 대입

### ■ 몫과 나머지 연산자

```
>>> q = 5 // 3
>>> r = 5 % 3
>>> print(q, r)
1 2
```

### ■ 제곱 연산자

```
>>> num = 5 ** 3
>>> print(num)
125
```



## ■ 입력된 숫자의 나머지와 몫을 계산하기

### [코드 3-1]

```
num1 = int(input("나뉘지는 수 ==> "))
num2 = int(input("나누는 수 ==> "))
q = num1 // num2
r = num1 % num2
print(num1, '을(를)', num2, '(으)로 나눈 몫은 ', q, '입니다.')
print(num1, '을(를)', num2, '(으)로 나눈 나머지는 ', r, '입니다.')
```

### [실행결과]

```
나뉘지는 수 ==> 25
나누는 수 ==> 10
25 을(를) 10 (으)로 나눈 몫은 2 입니다.
25 을(를) 10 (으)로 나눈 나머지는 5 입니다.
```

사용자 입력

# 몫과 나머지 연산자, 제공 연산자



## 확인문제

다음 빈칸에 들어갈 단어를 채우시오.

몫 연산자는 , 나머지 연산자는 , 제공 연산자는  이다.

정답

Click!

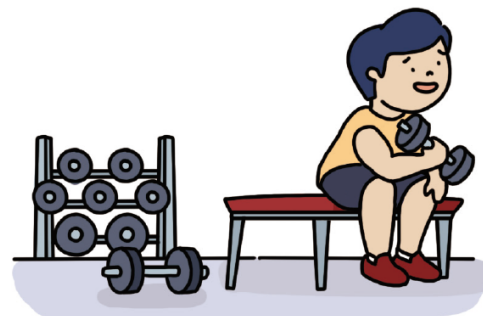
한빛 헬스장은 미국에서 수입한 덤벨만 사용해 덤벨 무게가 킬로그램(kg)이 아닌 파운드(lb)로만 표시되어 있습니다. 내가 드는 덤벨이 몇 킬로그램인지 알기 위해서 어떻게 해야 할까요?

**파운드(lb)와 킬로그램(kg)을 상호 변환하는 프로그램을 만들어 봅시다.**

## [조건]

1 파운드(lb) = 0.453592 킬로그램(kg)

1 킬로그램(kg) = 2.204623 파운드(lb)



## 실행 결과

파운드(lb)를 입력하세요 : 15 ● 사용자 입력

15 파운드(lb)는 6.80387 킬로그램(kg)입니다

킬로그램(kg)을 입력하세요 : 15 ● 사용자 입력

15 킬로그램(kg)은 33.06934 파운드(lb)입니다

1. lab03-01.py 파일을 만들고, 파운드 단위의 값을 입력받아 킬로그램으로 변환하기

```
pound = int(input("파운드(lb)를 입력하세요 : "))  
kg = pound * 0.453592  
print( pound, "파운드(lb)는", kg, "킬로그램(kb)입니다")
```

2. 이번에는 킬로그램 단위의 값을 입력받아 파운드로 변환하기

```
kg = int(input("킬로그램(kg)을 입력하세요 : "))  
pound = kg * 2.204623  
print( kg, "킬로그램(kg)은", pound, " 파운드(lb)입니다")
```

3. <Ctrl> + <S>를 눌러서 변경된 내용을 저장하고, <F5>를 눌러 실행 결과 확인하기



# Section 02

## 대입 연산자



## ■ 대입 연산자

- 오른쪽의 값이나 계산 결과를 왼쪽으로 대입하라는 의미로, '=' 연산자가 가장 기본적인 대입 연산자

```
num = 100  
num = 100 * 200  
num = int("100") + int("200")
```

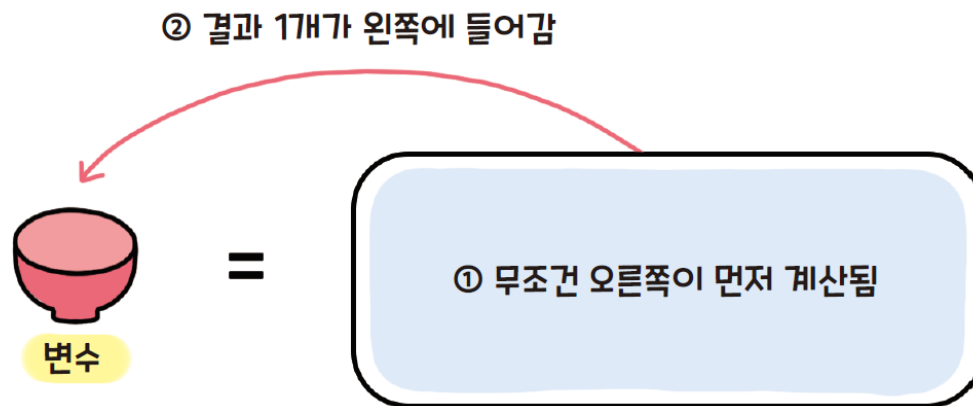
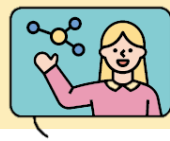


그림 3-5 한 개의 대입 연산자



## ■ 여러 개의 대입 연산자

- 콤마(,)로 분리해서 왼쪽에 변수가 2개 이상 나올 수도 있음
- 그런 경우에는 오른쪽도 반드시 콤마로 분리된 2개의 숫자, 수식, 문자열 등이 와야 함.

```
cnum1, num2 = 100, 200  
num1, num2 = 100*200, 100+200  
num1, num2 = int("100"), 100//5
```

③ 결과가 왼쪽에 각각 들어감

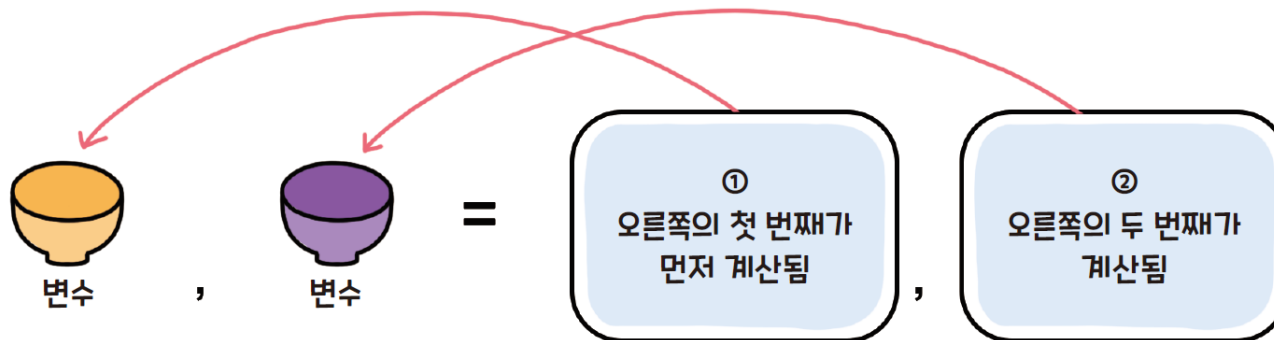


그림 3-6 여러 개의 대입 연산자(=)



## ■ 여러 개의 대입 연산자

- 3개 이상도 한꺼번에 나올 수 있지만 '='을 기준으로 왼쪽과 오른쪽의 개수가 같아야 함
- 비정상적인 대입 방식

```
num1, num2, num3 = 100, 200
```

```
num1, num2 = 100
```

```
num1 = 100, 200 <- - 실행하면 오류는 발생하지 않는 특수한 대입법
```



## 확인문제

1. 다음 빈칸에 들어갈 단어를 채우시오.

대입 연산자는 기호로  (을)를 사용하는데, 오른쪽이 먼저 계산된 후에 왼쪽 변수에 대입하라는 연산자이다.

2. 다음 보기 중에서 오류가 발생하는 것을 고르시오.

- ①  $n1 = 100$
- ②  $n1, n2 = 100, 200$
- ③  $n1, n2, n3 = 100, 200, 300$
- ④  $n1, n2 = 100$

## 정답

Click!



## ■ 복합 연산자의 역할

- 변수에 값을 변경한 후에 다시 자신에게 대입함

```
>>> num1 = 100  
>>> num1 = num1 + 200  
>>> print(num1)  
300
```

- 이때 `num1 = num1 + 200`과 `num += 200`은 같은 코드임

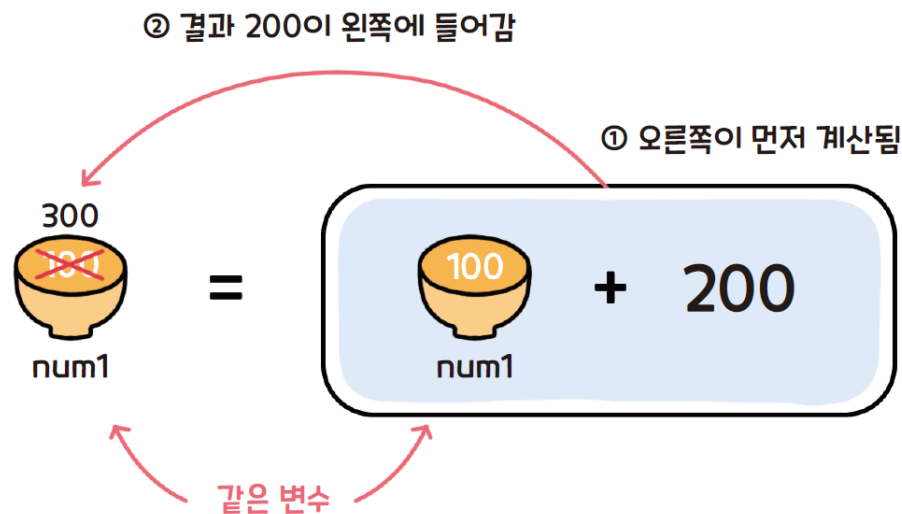


그림 3-7 자신과 연산 후 다시 자신에게 대입하기



## ■ 복합 대입 연산자

표 3-3 복합 대입 연산자

연산자	사용 예	설명
<code>+=</code>	<code>num += 3</code>	<code>num = num + 3</code> 과 동일
<code>-=</code>	<code>num -= 3</code>	<code>num = num - 3</code> 과 동일
<code>*=</code>	<code>num *= 3</code>	<code>num = num * 3</code> 과 동일
<code>/=</code>	<code>num /= 3</code>	<code>num = num / 3</code> 과 동일
<code>//=</code>	<code>num //= 3</code>	<code>num = num // 3</code> 과 동일
<code>%=</code>	<code>num %= 3</code>	<code>num = num % 3</code> 과 동일
<code>**=</code>	<code>num **= 3</code>	<code>num = num ** 3</code> 과 동일



## ■ 복합 대입 연산자 연습하기

```
>>> num = 20
>>> num += 3 ; print(num)
23
>>> num -= 3 ; print(num)
20
>>> num *= 3 ; print(num)
60
>>> num /= 3 ; print(num)
20.0
>>> num //= 3 ; print(num)
6.0
>>> num %= 3 ; print(num)
0.0
>>> num **= 3 ; print(num)
0.0
```



편의점에서는 물품을 본사에서 구입하면 물품값을 지불하고, 물품이 손님에게 판매되면 물품값을 받습니다. 이때 편의점에서는 본사에서 구입한 물품의 가격에 일부 이익을 붙여서 손님에게 판매합니다.

오늘 구입 또는 판매한 물건의 총 매출을 계산하는 프로그램을 만들어 봅시다.

	캔 커피	삼각김밥	바나나 우유	도시락	콜라	새우깡
구입 가격	500	900	800	3500	700	1000
판매 가격	1800	1400	1800	4000	1500	2000

#### [구매/판매 내역]

- 삼각김밥(900원) 10개 구입
- 바나나맛 우유(1800원) 2개 판매
- 도시락(3500원) 5개 구입
- 도시락(4000원) 4개 판매
- 콜라(1500원) 1개 판매
- 새우깡(2000원) 4개 판매
- 캔커피(1800원) 5개 판매



#### 실행 결과

오늘 총 매출액은 11600 원입니다

1. lab03-02.py 파일을 만들고, 총 매출액 변수 total을 0으로 초기화하기

```
total = 0
```

2. 구입한 물품은 구입 가격에 개수를 곱한 후 총 매출액에서 빼기

```
total -= 900*10  
total -= 3500*5
```

3. 판매한 물품은 판매 가격에 개수를 곱한 후, 총 매출액에 더하기

```
total += 1800*2  
total += 4000*4  
total += 1500  
total += 2000*4  
total += 1800*5
```

4. 마지막으로 총 매출액을 출력하기

```
print("오늘 총 매출액은 ", total, "원입니다")
```

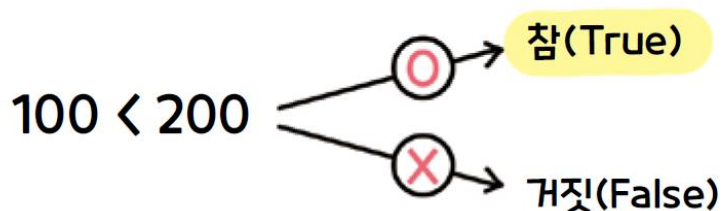
# Section 03

## 비교 연산자와 논리 연산자

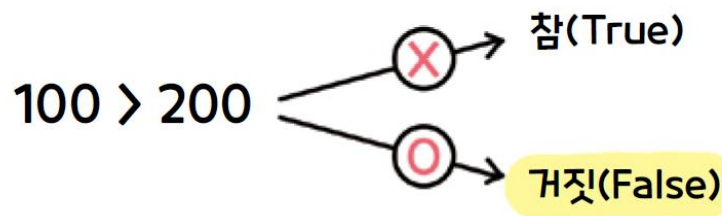


## ■ 비교 연산자

- 어떤 것이 큰지, 작은지, 같은지를 비교하는 연산자
- 결과는 참을 의미하는 True와 거짓을 의미하는 False로 표시함
- 비교 연산자를 단독으로 사용하는 경우는 거의 없음
  - 조건문이나 반복문과 함께 사용함



(a) 100은 200 보다 작다: 참(True)



(b) 100은 200 보다 크다: 거짓(False)

그림 3-8 비교 연산자의 기본 개념



## ■ 비교 연산자의 활용

- 변수에 들어 있는 값을 주로 사용함

① 시험 점수를 입력하세요.

② 시험 점수가 70 이상인가요?



그림 3-9 입력 점수에 따른 비교 연산자의 사용

### [코드 3-2]

```
score = int(input("필기 시험점수를 입력하세요 ==>"))  
print(score >= 70)
```

### [실행결과]

필기 시험점수를 입력하세요 ==> 80 ● ————— 사용자 입력  
True



## ■ 비교 연산자의 종류

표 3-4 비교 연산자

연산자	의미	설명
==	같다	두 값이 동일하면 참(True)
!=	같지 않다	두 값이 다르면 참(True)
>	크다	왼쪽이 크면 참(True)
<	작다	왼쪽이 작으면 참(True)
>=	크거나 같다	왼쪽이 크거나 같으면 참(True)
<=	작거나 같다	왼쪽이 작거나 같으면 참(True)



## ■ 비교 연산자의 예제

```
>>> n1 = 100
>>> n2 = 200
>>> print(n1 == n2 , n1 != n2)
False True
>>> print(n1 > n2 , n1 < n2)
False True
>>> print(n1 >= n2 , n1 <= n1)
False True
```

### 하나 더 알기 ✓

#### 비교 연산자(==)와 대입 연산자(=)

n1과 n2가 같은지 확인하는 비교 연산자는 ==입니다. 그런데, 가끔 수학의 같다(=) 연산자와 헷갈려 =을 하나만 쓰는 경우가 있습니다.

```
>>> print(n1 = n2)
```

오류 발생

이 경우 오류가 발생합니다. 'n1 = n2'는 n2의 값을 n1에 대입하라는 의미이지, 비교 연산자가 아니기 때문입니다.





## ■ 논리 연산자

- 비교 연산자가 여러 번 필요할 때 사용함
- 만약 num 변수의 값이 10과 20 사이에 있어야 한다(10과 20은 제외)면
  - 조건1) num은 10보다 커야 함.
  - 조건2) num은 20보다 작아야 함.

```
(num > 10) and (num < 20)
```

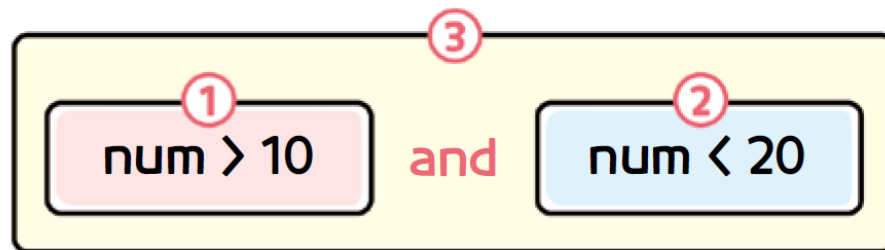


그림 3-10 and 연산





## ■ 논리 연산자의 종류

표 3-5 논리 연산자

연산자	의미	설명	사용 예
and	그리고(AND)	둘 다 참이어야 참	(num > 10) and (num < 20)
or	또는(OR)	둘 중 하나만 참이어도 참	(num == 10) or (num == 20)
not	부정(NOT)	참이면 거짓, 거짓이면 참	not(num < 100)

## ■ 논리 연산자의 예제

```
>>> num = 99
>>> (num > 100) and (num < 200)
False
>>> (num == 99) or (num == 100)
True
>>> not(num == 100)
True
```



## 확인문제

1. 다음 빈칸에 들어갈 단어를 채우시오.

비교 연산자 중에 같다는  이고, 같지 않다는  이다.

2. 다음 빈칸에 들어갈 단어를 채우시오.

둘 다 참이어야 참이 되는 논리 연산자는  이고, 둘 중에 하나만 참이어도 참이 되는 논리 연산자는  이다.

## 정답

Click!

# Section 04

## 연산자의 우선순위



## ■ 연산자의 우선순위

- 연산자가 동시에 나올 경우 어떤 연산자를 먼저 계산할지 결정되어 있는 순서

표 3-6 연산자의 우선순위

연산자 우선순위	연산자	설명
1	() [] {}	괄호, 리스트, 딕셔너리, 세트 등
2	**	제곱
3	+ - ~	단항 연산자
4	* / % //	산술 연산자
5	+ -	산술 연산자
6	<< >>	비트 시프트 연산자
7	&	비트 논리곱
8	^	비트 배타적 논리합
9		비트 논리합
10	< > >= <=	관계 연산자
11	== !=	동등 연산자
12	= %= /= //= -= += *= **=	대입 연산자
13	not	논리 연산자
14	and	논리 연산자
15	or	논리 연산자
16	if ~ else	비교식

여기서 +와 -는 더하기와 빼기가 아니라, 숫자나 변수 앞에 붙는 플러스, 마이너스 부호를 의미합니다.

# 연산자의 우선순위란?



## 확인문제

다음 항목 중에서 연산자 우선순위가 가장 높은 것과 가장 낮은 것을 고르시오.

덧셈(+), 곱셈(\*), 제곱(\*\*), 대입(=), 대괄호([])

정답

Click!



## 하나 더 알기 Y

### 파이썬 주석

주석(Remark)은 코드에 설명을 달 때 사용합니다. 사실 주석은 있어도 실행되지 않기 때문에 프로그램과 관련이 없어 보입니다. 하지만 주석 없이 몇 백 또는 그 이상의 행으로 작성된 소스 코드를 이해하기는 상당히 어렵습니다. 그래서 실무에서는 주석을 많이 달도록 권장하고 있습니다.

#### ■ 한 줄 주석

먼저 주석에는 # 기호가 있는데, #은 한 줄 주석을 의미합니다. #이 나오면 그 이후부터는 주석이 되며 실행 결과에 영향을 미치지 않습니다.

#### 코드 3-3

ex03-03.py

```
01 # print("여기는 주석~")
02 print("안녕~ 파이썬!")      # 여기부터 주석임
```

안녕~ 파이썬

1행은 #이 제일 앞에 나왔으므로, 그 줄 전체가 주석이 되어서 실행되지 않습니다. 2행은 `print()` 이후에 #이 나왔으므로 #이 나온 부분 이후부터 주석이 됩니다.

#### ■ 여러 줄의 주석

여러 줄을 주석으로 처리하려면 큰따옴표 또는 작은따옴표가 연속으로 3개씩 나오면 됩니다. 다음은 큰따옴표 사이의 모든 행이 주석이 됩니다.

#### 코드 3-4

ex03-04.py

```
01 """
02 이 줄은 주석임
03 이 줄도 주석임
04 """
```

실행 결과 없음



한빛 대학교 1학년인 난생이는 기말고사 성적표를 받았습니다.  
다음 성적표를 참고하여 난생이의 평균 학점을 구해봅시다.

## 실행 결과

평균 학점 : 3.83

과목(이수학점)	성적
파이썬(3)	B(3.5)
모바일(2)	A0(4.0)
엑셀(1)	A(4.5)

1. lab03-03.py 파일을 만들고, 파이썬, 모바일, 엑셀 변수에 각각의 이수학점을 넣어 선언하기

```
python = 3  
mobile = 2  
excel = 1
```

2. A, A0, B 학점에 대응하는 점수를 넣어 선언하기

```
A = 4.5  
A0 = 4.0  
B = 3.5
```

3. 과목과 해당하는 점수를 곱하여, 전체 학점 점수로 나누기

- 이때 연산자의 우선순위를 주의하여 괄호로 묶어줌

```
avg = ((python * B) + (mobile * A0) + (excel * A)) / (python  
+ mobile + excel)
```

4. 평균 학점을 출력하고, <F5>를 눌러서 결과를 확인하기

```
print("평균 학점 : ", avg)
```



# [실전 예제] 거북이를 그리는 펜의 변화

# 실전 예제 거북이를 그리는 펜의 변화

## [문제]

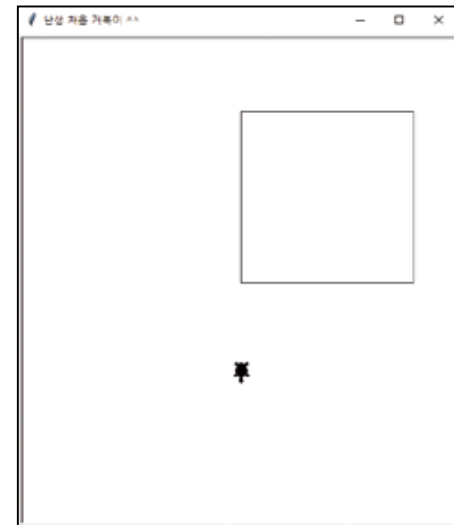
거북이가 선을 긋지 않으면서 이동해 보자.

- `turtle.penup()`: 펜을 드는 함수, 선을 그리지 않음
- `turtle.pendown()`: 펜을 내리는 함수, 다시 선을 그림

## [해결]

1. `turtle.penup()` 함수로 펜을 들고 `turtle.forward()`로 이동시키면 선을 그리지 않고 거북이가 펜을 들고 아래쪽으로 100만큼 이동

```
>>> turtle.penup( )  
>>> turtle.forward(100)
```



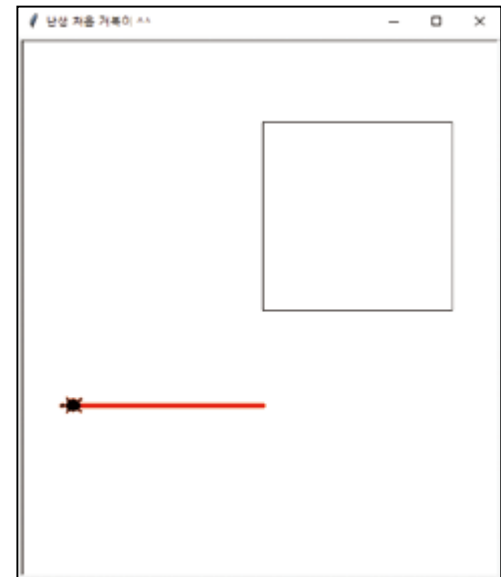
## 실전 예제 거북이를 그리는 펜의 변화

2. 이번에는 펜을 다시 내리고, 오른쪽으로 머리를 회전시킨 후에 선 긋기.  
선을 긋기 전에 펜의 두께와 펜의 색상도 변경하기

- `turtle.pensize( )`: 펜의 두께를 미리 설정. 펜 두께는 1~10 사이로 지정
- `turtle.pencolor( )`: 펜의 색상을 미리 설정.

색상은 red, green, blue 등 다양하게 설정할 수 있음

```
>>> turtle.pensize(5)
>>> turtle.pencolor("red")
>>> turtle.right(90)
>>> turtle.pendown( )
>>> turtle.forward(200)
```



# [실전 예제] 입력한 값만큼 거북이 움직이기

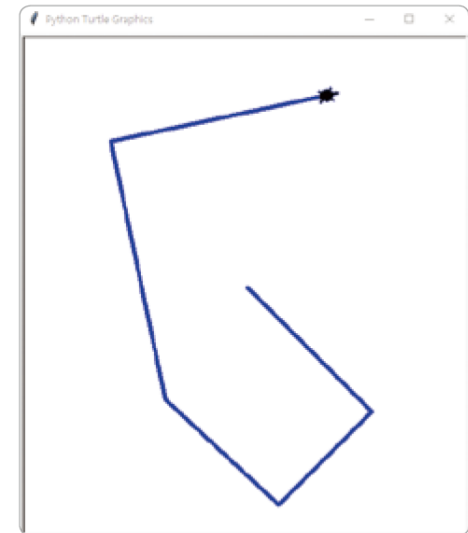
# 실전 예제 입력한 값만큼 거북이 움직이기

## [문제]

사용자가 입력한 거리와 각도만큼 거북이가 이동하는 코드를 작성해 보자. 이때 이동하는 횟수는 무한 반복함

거북이의 회전 각도 : 45  
거북이가 이동 거리 : 200  
...생략...  
거북이의 회전 각도 : 45  
거북이가 이동 거리 : 250

사용자 입력



# 실전 예제 입력한 값만큼 거북이 움직이기

## 1. 무한 반복을 하는 코드

- 주의할 점: while True: 아래 행들은 4칸의 들여쓰기가 되어 있어야 함.
- 만약 들여쓰기가 되어 있지 않으면, 그 부분은 반복의 대상이 아님

```
while True :  
    pass  <-- 이 부분을 무한 반복함
```

- 무한 반복을 멈추려면 <Ctrl> + <C>를 누르기

## 실전 예제 입력한 값만큼 거북이 움직이기

2. 사용자가 각도와 거리를 입력하면, 거북이가 그 각도와 거리로 계속 이동하는 코드를 작성하기

```
import turtle

turtle.shape("turtle")
turtle.pensize(5)          # 펜의 두께: 5
turtle.pencolor("blue")    # 펜의 색상: 파란색

while True :
    angle = int(input("거북이의 회전 각도 : "))
    distance = int(input("거북이의 이동 거리 : "))

    turtle.right(angle)
    turtle.forward(distance)
```

# Thank you!