

새내기 파이썬



3장 수식과 연산자





학습 목표

1. 나머지 연산자를 이용하여 짝수와 홀수를 구분할 수 있나요?
2. 총매출 계산과 세금 계산을 도와주는 간단한 계산기를 작성할 수 있나요?
3. 복리 이자를 계산하는 프로그램을 작성할 수 있나요?
4. 거스름돈을 다양한 동전으로 계산해주는 프로그램을 만들 수 있나요?



3장에서 만들 프로그램

산수 퀴즈에 오신 것을 환영합니다.

$2 + 5 = 7$

True

$7 - 6 = 1$

True

$2 ** 3 = 8$

True

$3.0 / 1.5 = 2.0$

True

물건값을 입력하시오: 750

받은 금액: 1000

거스름돈은 아래와 같습니다.

500원=0개 100원=2개 10원=5개 1원=0개



산술 연산자

연산자	기호	사용예	결과값
덧셈	+	$7 + 4$	11
뺄셈	-	$7 - 4$	3
곱셈	*	$7 * 4$	28
정수 나눗셈	//	$7 // 4$	1
실수 나눗셈	/	$7 / 4$	1.75
나머지	%	$7 \% 4$	3

```
>>> 7 / 4  
1.75
```



나머지 연산자

```
x = int(input('피젯수: '))  
y = int(input('젯수: '))  
  
q = x // y  
r = x % y  
print(f'{x}을 {y}로 나눈 몫={q} ')  
print(f'{x}을 {y}로 나눈 나머지={r} ')
```

```
피젯수: 10  
젯수: 3  
10을 3로 나눈 몫=3  
10을 3로 나눈 나머지=1
```



나머지 연산자

```
x = 10  
y = 3  
quotient, remainder = divmod(x, y)  
print(quotient, remainder)
```

이 함수로도 몫과 나머지를
계산할 수 있다.

3 1

```
today = 0  
print( (today + 10) % 7 )
```

3

오늘이 일요일이다. 오늘로부터
10일 후는 무슨 요일일까?



나머지 연산자

```
>>> 2 ** 7  
128
```

2의 7승이 계산된다.



Lab: 원리금 계산 프로그램

- 원금 a , 이자율 r , n 년 후에 원리금 합계는 $b = a(1+r)^n$ 이 된다.

```
a = 1000          # 원금
r = 0.05          # 이자율
n = 10            # 기간
result = a*(1+r)**n  # 원리금 합계

print("원리금 합계=", result)
```

원리금 합계= 1628.894626777442



참고사항



NOTE

부동소수점수를 사용할 때는 계산이 부정확할 수도 있음을 알아야 한다. 예를 들어서 파이썬에서 다음과 같은 수식을 계산해보자.

```
>>> 1.2-1.0  
0.19999999999999996
```

놀랍게도 **0.2**가 나오지 않는다. 일반 사람들은 이것을 파이썬의 버그로 생각할 수도 있다. 하지만 아니다. 이것은 컴퓨터 내부에서 실수를 나타낼 때, **2진법**을 사용하고 제한된 개수의 비트를 사용하기 때문에 어떤 실수는 이진수로 정확하게 표현할 수 없는 것이다. **10진법**에서 **1/3**이 **0.33333...**으로 계산되는 것이나 마찬가지이다.



정가점거 중간점검

1. $10\%6$ 의 값은 무엇인가?
2. 나눗셈 연산인 $10//6$ 의 값은 얼마인가?
3. 다음의 할당문에서 무엇이 잘못되었는가?

$$3 = x$$

4. 10의 3제곱값을 계산하는 문장을 작성해보자.





할당 연산

```
x = 1           # 변수 x에 1을 할당한다.  
value = 3.0     # 변수 value에 3.0을 할당한다.  
x = (1/2)+3     # 변수 x에 수식의 결과를 할당한다
```

```
x = y = z = 0
```

```
x, y, z = 10, 20, 30
```

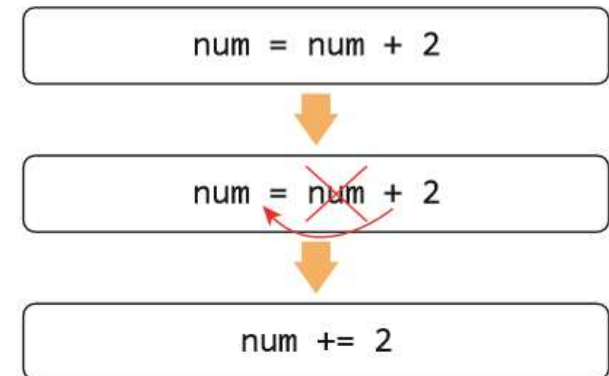
10은 x에, 20은 y에, 30은 z에
할당된다. 파이썬의 독특한 할당
연산이다. 나중에 학습하는 튜플을
이용한다.



복합 할당 연산자

- 복합 할당 연산자(**compound operator**)란 +=처럼 대입 연산자와 다른 연산자를 합쳐 놓은 연산자이다.

복합 연산자	의미
$x += y$	$x = x + y$
$x -= y$	$x = x - y$
$x *= y$	$x = x * y$
$x /= y$	$x = x / y$
$x \% = y$	$x = x \% y$





복합 연산자

```
>>> x = 1000
```

```
>>> x += 2
```

x는 1002가 된다.

```
>>> x -= 2
```

x는 다시 1000이 된다.

제일 많이 사용하는 복합 연산자
문장은 `x += 1`이다.



Exmample: 하루 매출 계산 프로그램

판매된 우유의 개수: 3

판매된 콜라의 개수: 2

판매된 김밥의 개수: 5

오늘 총 매출은 29500원입니다.





Example: 하루 매출 계산 프로그램

```
total_sales = 0
milk_count = int(input("판매된 우유의 개수: "))
cola_count = int(input("판매된 콜라의 개수: "))
krice_count = int(input("판매된 김밥의 개수: "))

total_sales += milk_count*2000
total_sales += cola_count*3000
total_sales += krice_count*3500

print(f"\n오늘 총 매출은 {total_sales}원입니다.")
```





정가점검 중간점검

1. 대입 연산자의 왼쪽에 올 수 있는 것은 무엇인가?
2. 등호(=)가 수학에서의 의미와 다른 점은 무엇인가?
3. 복합 대입 연산자 $x *= y$ 의 의미를 설명하라.
4. 다음의 할당문이 실행된 후의 변수 a, b, c 의 값은?

$$a = b = c = 100$$

1. 현재 x 는 1이고 y 는 2라고 하자. 다음의 할당문이 실행된 후의 변수 x, y 의 값은?

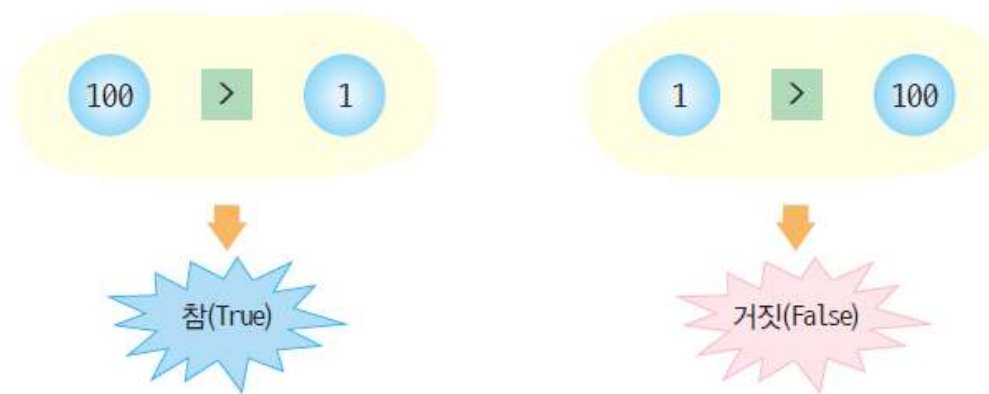
$$x, y = y, x$$





관계 연산자

- 관계 연산자(relational operator)는 두 개의 피연산자를 비교하는데 사용된다. 예를 들면 “변수 x가 변수 y보다 큰지”를 따지는데 사용된다.





관계 연산자

- 관계 연산자(**relational operator**)는 두 개의 피연산자를 비교하는데 사용된다. 예를 들면 “변수 x 가 변수 y 보다 큰지”를 따지는데 사용된다.

표 3.2 관계 연산자

연산	의미	수학적 표기
$x == y$	x 와 y 가 같은가?	$=$
$x != y$	x 와 y 가 다른가?	$\neq$
$x > y$	x 가 y 보다 큰가?	$>$
$x < y$	x 가 y 보다 작은가?	$<$
$x >= y$	x 가 y 보다 크거나 같은가?	$\geq$
$x <= y$	x 가 y 보다 작거나 같은가?	$\leq$



부울 변수

```
radius = 100  
flag = (radius > 32)  
print(flag)
```

True

```
expensive = price > 20000      # expensive가 부울 변수이다.  
if expensive :                 # 관계 수식 대신에 부울 변수가 들어가도 된다.  
    shipping_cost = 0  
else :  
    shipping_cost = 3000
```

조건문은 4장에서 학습한다.
price가 20000이상이면
shipping_cost는 0이 된다.



문자열

```
s1 = "Audrey Hepburn"  
s2 = "Audrey Hepburn"  
print(s1 == s2)
```

True

```
s1 = "Audrey Hepburn"  
s2 = "Grace Kelly"  
print(s1 < s2)
```

True





실수와 실수의 비교

```
from math import sqrt

n = sqrt(3.0)
if n*n == 3.0 :
    print("sqrt(3.0)*sqrt(3.0)은 3.0과 같다. ")
else :
    print("sqrt(3.0)*sqrt(3.0)은 3.0과 같지 않다. ")
```

sqrt(3.0)*sqrt(3.0)은 3.0과 같지 않다.

실수 계산은 오차가 있을 수 있다

```
if abs(n*n - 3.0) < 0.00001 :
    print("sqrt(3.0)*sqrt(3.0)은 3.0과 같다. ")
```

sqrt(3.0)*sqrt(3.0)은 3.0과 같다.

실수를 비교할 때는 값의 차이가 작으면 같다고 생각해야 한다.



정가점거 중난점거

1. 관계 수식의 결과로 생성될 수 있는 값은 무엇인가?
2. 실수와 실수를 비교할 때 주의해야 할 점은 무엇인가?





비트 연산자

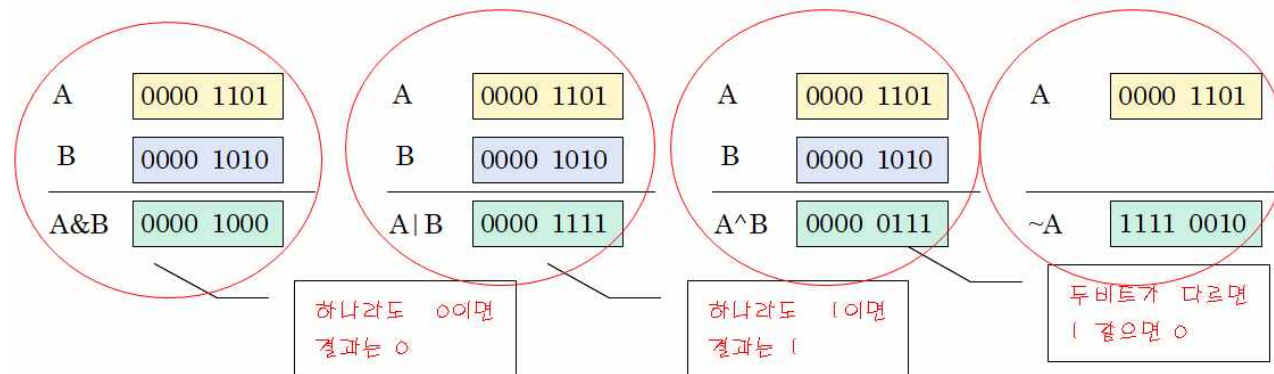
- 파이썬에서는 정수를 이루고 있는 각각의 비트를 가지고 작업할 수 있는 연산자가 제공된다. 예를 들어서, 정수의 특정한 위치에 있는 비트를 추출할 수 있다. 비트 연산자에는 다음과 같은 것들이 있다.

연산자	의미	예
~	비트 NOT	0을 1로, 1을 0으로 바꾼다.
&	비트 AND	두개의 비트가 1인 경우에만 1이 된다.
^	비트 XOR	0이면 1로, 1이면 0로 바꾼다.
	비트 OR	두개의 비트 중에서 하나만 1이면, 1이 된다.



비트 연산자

- 파이썬에서는 정수를 이루고 있는 각각의 비트를 가지고 작업할 수 있는 연산자가 제공된다. 예를 들어서, 정수의 특정한 위치에 있는 비트를 추출할 수 있다. 비트 연산자에는 다음과 같은 것들이 있다.





비트 연산자

```
a = 0b00001101
b = 0b00001010
print( a&b, a|b, a^b )
print( bin(a&b), bin(a|b), bin(a^b) )
```

```
8 15 7
0b1000 0b1111 0b111
```

bin()은 정수를 받아서 2진수 형태로
만들어주는 함수이다.

```
a = 0b00001101
print(~a&255, bin(~a&255))
```

```
242 0b11110010
```

비트 AND 연산자가 없으면 음수로
인식한다.



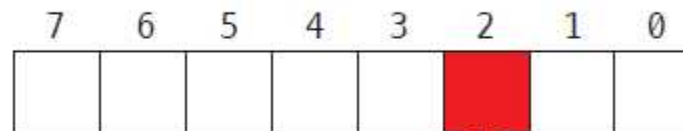
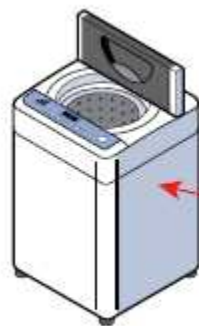
-
- 7 6 5 4 3 2 1 0
- 문이 열려있으면 1



Example: 비트 연산자

```
status = 0b01101110;  
print( "문열림 상태=" , ((status & 0b00000100)!=0) )
```

문열림 상태= True

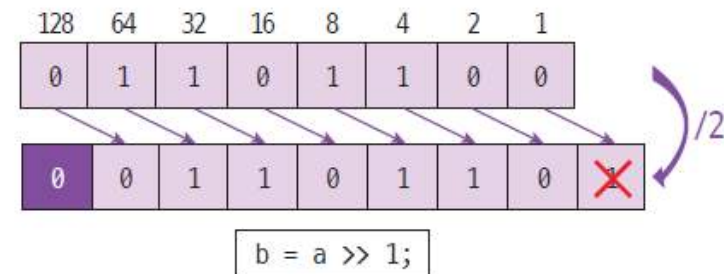
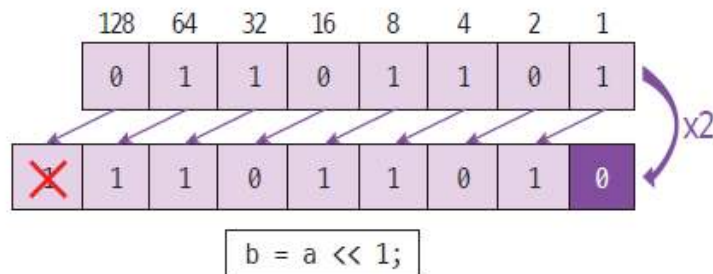


문이 열려있으면 1



비트 이동 연산자

- 부호 이동 연산자인 << 연산자는 비트를 왼쪽으로 이동한다. >> 연산자는 반대로 비트를 오른쪽으로 이동한다.



`a = 0b01101101` # a는 109이다.
`print( a<<1, a>>1)`
`print( bin(a<<1), bin(a>>1))`

218 54
0b11011010 0b110110

`a<<1`은 `a`를 2배로 만든다.
`a>>1`은 `a`를 $\frac{1}{2}$ 배로 만든다.



정가점거 중난점

1. 비트 **AND** 연산자를 나타내는 기호는?
2. $0b00011 \ll 1$ 의 결과를 예측해보라.





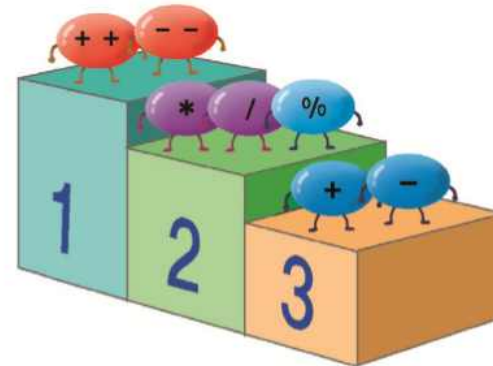
연산자의 우선 순위

$$x + \underbrace{y * z}_{\textcircled{1}}_{\textcircled{2}}$$

$$\underbrace{(x + y) * z}_{\textcircled{1}}_{\textcircled{2}}$$

파이썬의 산술 연산자의 경우, 다음과 같은 우선 순위를 가진다.

- ① 지수 **
- ② 곱셈 *, 나눗셈 /, 나머지 %
- ③ 덧셈 +과 뺄셈 -





괄호의 사용

```
>>> 1 + 2 * 3
```

```
>>> 4 - 40 - 3  
-39
```

```
>>> 10 + 20 / 2  
20.0
```

```
>>> (10 + 20) / 2  
15.0
```



우선 순위표

연산자	설명
**	지수 연산자
~, +, -	단항 연산자
*, /, %, //	곱셈, 나눗셈, 나머지 연산자
+, -	덧셈, 뺄셈
>>, <<	비트 이동 연산자
&	비트 AND 연산자
^,	비트 XOR 연산자, 비트 OR 연산자

우선순위가 기억나지 않으면
괄호 (...) 를 사용한다.



Example: 평균 성적 계산 프로그램

- 평균 성적을 계산하는 프로그램을 작성해보자. 사용자에게 각 과목의 성적을 입력받는다. 평균을 계산할 때 연산자의 우선순위에 주의하자.

국어 성적: 90
수학 성적: 95
영어 성적: 93
평균 성적은 92.67점입니다.

```
score1 = int(input("국어 성적: "))  
score2 = int(input("수학 성적: "))  
score3 = int(input("영어 성적: "))  
avg = ( score1 + score2 + score3 ) / 3.0  
print(f"\n평균 성적은 {avg:.02f}점입니다.")
```

avg는 소수점 2번째
자리까지만 표시한다.



정가점거 중간점거

1. *와 ** 중에서 어떤 연산자가 우선순위가 높은가?
2. 우선순위가 생각나지 않으면 어떻게 하는 것이 좋은가?
3. + 연산자와 = 연산자 중에서 어떤 연산자가 우선순위가 높은가?





타입 변환

- 정수와 부동소수점수를 동시에 사용하여 수식을 만들면 파이썬은 자동으로 정수를 부동소수점수로 변환한다. 이것을 자동적인 타입 변환이라고 한다.

```
>>> 3 * 1.23          # 이것은 3.0 * 1.23과 같다.
3.69

>>> x = 3.14
>>> int(x)
3
>>> y = 3
>>> float(y)
3.0
```



- `round()` 함수는 실수를 반올림하기 위하여 사용한다. 예를 들면 다음과 같다

```
>>> x = 1.723456
>>> round(x)
2
```

```
>>> x = 1.723456
>>> round(x, 2)
1.72
```

```
>>> x = 1.7
>>> round(x)
2
>>> x
1.7
```

X를 소수점 2번째 자리까지
반올림한다.



Example: 부가 가치세 계산 프로그램

- 하나의 예로 물건의 값의 7.5%가 부가세라고 하자. 물건값이 12345 원일 때, 부가세를 소수점 2번째 자리까지 계산하는 프로그램을 작성해보자.

```
price = 12345  
tax = price * 0.075  
tax = round(tax, 2)  
print(tax)
```

925.88



정가점거 중난점

1. int형 변수 `x`를 float형으로 형변환하는 문장을 써보라.
2. 하나의 수식에 정수와 부동소수점수가 섞여 있으면 어떻게 되는가?





Lab: 산수 퀴즈

- 0부터 9까지의 숫자를 이용하여서 간단한 산수 퀴즈를 출제하는 프로그램을 만들어보자.

산수 퀴즈에 오신 것을 환영합니다.

$2 + 5 = 7$

True

$7 - 6 = 1$

True

$2 ** 3 = 8$

True

$3.0 / 1.5 = 2.0$

True



Sol: 산수 퀴즈

```
print("산수 퀴즈에 오신 것을 환영합니다.\n")
```

```
ans = int(input("2 + 5 = "))
```

```
print(ans==2+5)
```

```
ans = int(input("7 - 6 = "))
```

```
print(ans==7-6)
```

```
ans = int(input("2 ** 3 = "))
```

```
print(ans==2**3)
```

```
ans = float(input("3.0 / 1.5 = "))
```

```
print(ans==3.0/1.5)
```



Lab: 답답형 문제 채점 프로그램

- 몇 개의 답답형 문제를 출제하고 사용자가 대답한 답안을 채점하는 시스템을 만들어보자.

가장 쉬운 프로그래밍 언어는? 파이썬

True

거듭제곱을 계산하는 연산자는? **

True

파이썬에서 출력시에 사용하는 함수이름은? printf

False

점수 = 2



Sol: 단답형 문제 채점 프로그램

```
score = 0

ans = input("가장 쉬운 프로그래밍 언어는? ")
check = (ans=="파이썬")
print(check)
score += int(check)

ans = input("거듭제곱을 계산하는 연산자는? ")
check = (ans=="**")
print(check)
score += int(check)

ans = input("파이썬에서 출력시에 사용하는 함수이름은? ")
check = (ans=="print")
print(check)
score += int(check)

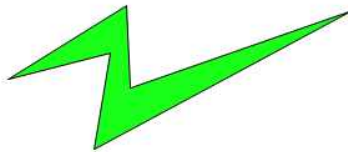
print(f"점수 = {score}")
```



Lab: 명왕성까지의 시간 계산하기

- 지구에서 명왕성까지의 평균 거리는 약 48억km라고 한다. 빛의 속도로 가면 시간이 얼마나 걸리는 지를 계산해보자.

4 시간 26 분





Sol: 명왕성까지의 시간 계산하기

- 지구에서 명왕성까지의 평균 거리는 약 48억km라고 한다. 빛의 속도로 가면 시간이 얼마나 걸리는 지를 계산해보자.

```
##
#           이 프로그램은 명왕성까지 빛이 가는 시간을 계산한다.
#
speed = 300000.0                # 빛의 속도
distance = 4800000000.0         # 거리
secs = distance / speed         # 걸리는 시간, 단위는 초
secs = int(secs)                # 부동소수점수->정수 변환
time = secs // 3600             # 초를 시간으로 변환, //은 정수 나눗셈
minute = (secs % 3600) // 60    # 남은 초를 분으로 변환
print(time, "시간", minute, "분")
```



Lab: 상점 계산기 만들기

- 상점에서 필요한 계산기 프로그램을 만들어보자. 할인이나 거스름돈 계산 등의 기능이 있어야 한다.

상품의 가격: 10000

상품의 개수: 3

할인율(%): 10

받은 금액: 50000

거스름돈: 23000





Sol: 상점 계산기 만들기

```
price = int(input("상품의 가격: "))  
amount = int(input("상품의 개수: "))  
disRate = int(input("할인율(%): ")) / 100.0
```

```
print()  
payment = int(input("받은 금액: "))  
total = price * amount  
change = payment - (total - total * disRate)
```

결과를 출력한다.

```
print(f"거스름돈: {int(change)}")
```

change를 정수형으로 변환



Lab: 복리 계산

- 1626년에 아메리카 인디언들이 뉴욕의 맨하탄섬을 단돈 60길더(약 24달러)에 탐험가 **Peter Minuit**에게 팔았다고 한다. 382년 정도 경과한 맨하탄 땅값은 약 600억달러라고 한다.
- 하지만 만약 인디언이 24달러를 은행의 정기예금에 입금해두었다면 어떻게 되었을까? 예금 금리는 복리로 6%라고 가정하자. 그리고 382년이 지난 후에는 원리금을 계산하여 보자.





Solution

```
init_money = 24  
interest = 0.06  
years = 382  
print(init_money*(1+interest)**years)
```

```
111442737812.28842
```



Lab: 자동판매기 프로그램

- 자동 판매기를 시뮬레이션하는 프로그램을 작성하여 보자.

물건값을 입력하시오: 750

1000원 지폐개수: 1

500원 동전개수: 0

100원 동전개수: 0

500원 = 0 100원 = 2 10원 = 5 1원 = 0





Solution

```
##
#           이 프로그램은 자판기에서 거스름돈을 계산한다.
#
itemPrice = int(input("물건값을 입력하시오: "))
note = int(input("1000원 지폐개수: "))
coin500 = int(input("500원 동전개수: "))
coin100 = int(input("100원 동전개수: "))

change = note*1000 + coin500*500 + coin100*100 - itemPrice

# 거스름돈(500원 동전 개수)을 계산한다.
nCoin500 = change//500
change = change%500

# 거스름돈(100원 동전 개수)을 계산한다.
nCoin100 = change//100
change = change%100

# 거스름돈(10원 동전 개수)을 계산한다.
nCoin10 = change//10
change = change%10

# 거스름돈(1원 동전 개수)을 계산한다.
nCoin1 = change

print("500원=", nCoin500, "100원=", nCoin100, "10원=", nCoin10, "1원=", nCoin1)
```



Mini Project: 상점 계산기 최종 버전

상품의 가격: 9000

상품의 개수: 3

세금(10%): 2700

봉사료(5%): 1350

전체 가격 = 31050

받은 금액: 50000

거스름돈: 18950

10000원 지폐의 개수: 1

5000원 지폐의 개수: 1

1000원 지폐의 개수: 3

500원 동전의 개수: 1

100원 동전의 개수: 4

50원 동전의 개수: 1



이번장에서 배운 것

- 파이썬에서는 덧셈, 뺄셈, 곱셈, 나눗셈을 위하여 $+$, $-$, $*$, $/$ 기호를 사용한다.
- 지수 연산자는 $**$ 이다.
- 나눗셈에서 정수로 몫을 계산하려면 $//$ 연산자를 사용한다.
- 나눗셈에서 나머지를 계산하려면 $%$ 연산자를 사용한다.
- 우선순위가 높은 연산자가 먼저 계산된다.
- $*$ 와 $/$ 가 $+$ 와 $-$ 보다 우선순위가 높다.
- 연산자의 우선 순서를 변경하려면 괄호를 사용한다.





Q & A

