

03

CHAPTER

변수와 데이터형

학습목표

- `print()` 함수로 서식에 맞추어 출력하는 방법을 익힌다.
- 변수의 사용법을 익힌다.
- 비트, 바이트 같은 데이터 표현 방법과 진수 변환 방법을 익힌다.
- 정수, 실수, 문자열 등 기본 데이터형을 알아본다.
- `print()` 함수로 별표를 다이아몬드 모양으로 출력하는 프로그램을 만든다.
- 입력한 숫자의 진수를 변환하는 프로그램을 만든다.

SECTION 01 이 장에서 만들 프로그램

SECTION 02 `print()` 함수를 사용한 다양한 출력

SECTION 03 변수의 선언과 사용

SECTION 04 데이터 표현 단위와 진수 변환

SECTION 05 기본 데이터형

요약

연습문제

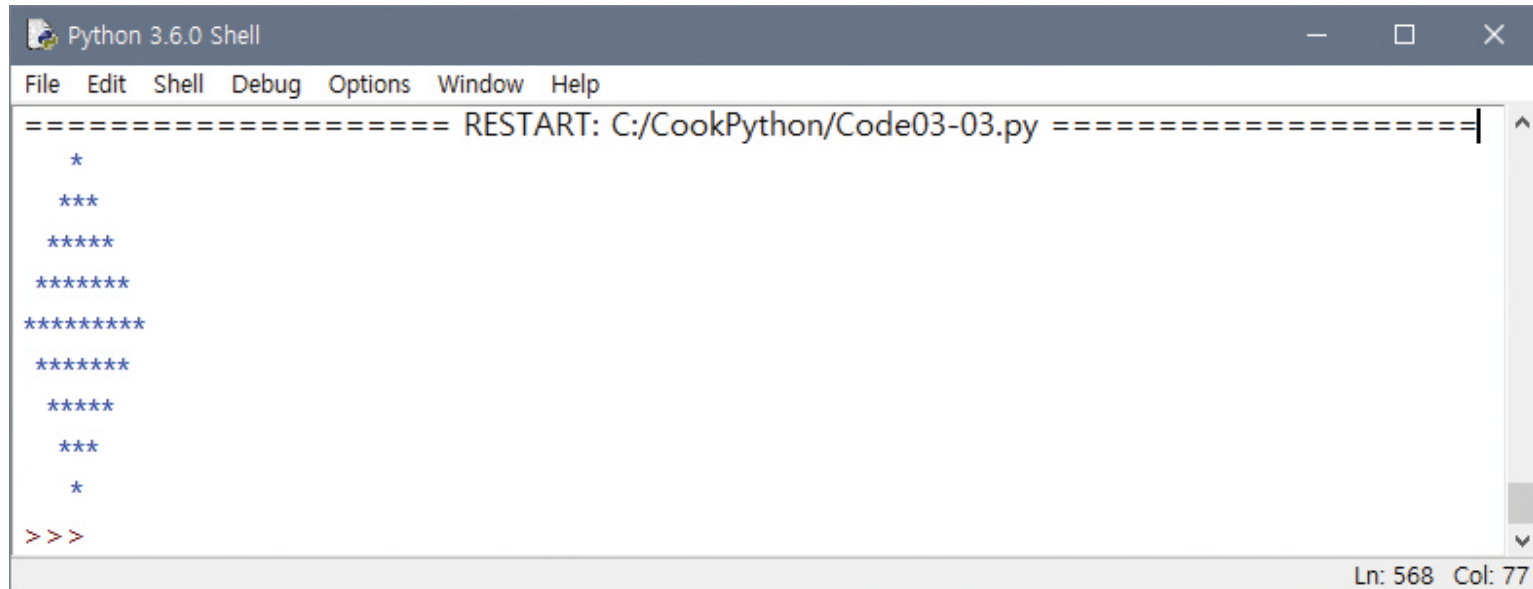
응용예제



Section01 이 장에서 만들 프로그램

■ [프로그램1] 다이아몬드 모양 출력

- 다이아몬드 모양의 별표를 출력



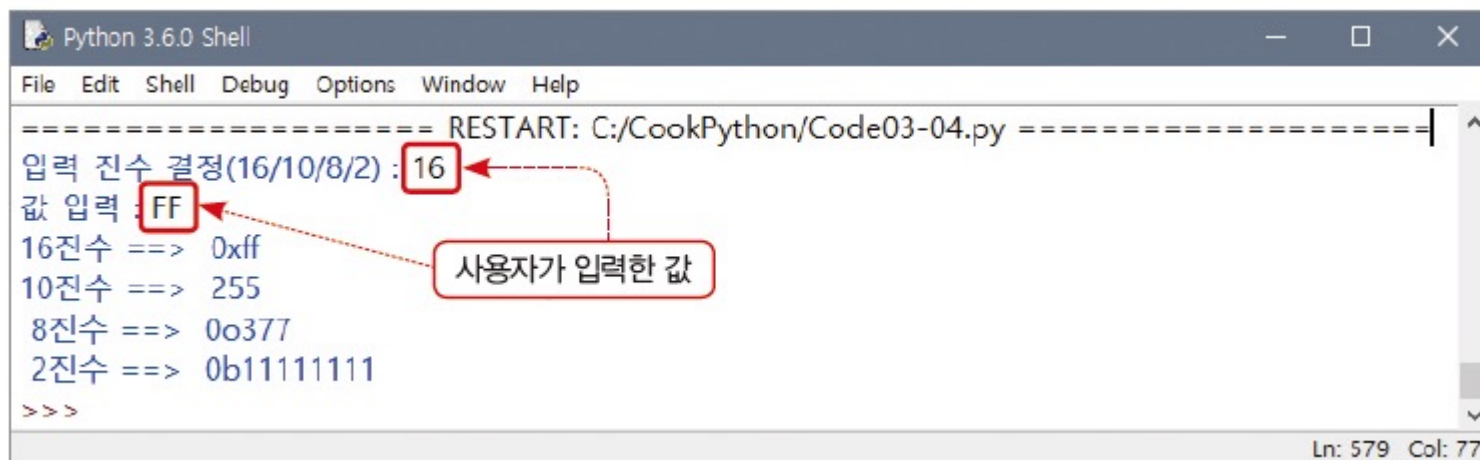
The screenshot shows a Python 3.6.0 Shell window with a menu bar (File, Edit, Shell, Debug, Options, Window, Help) and a status bar (Ln: 568 Col: 77). The main text area displays the output of a program, which is a diamond shape made of asterisks. The output is preceded by a separator line: "===== RESTART: C:/CookPython/Code03-03.py =====". The diamond shape is composed of 11 lines of asterisks, with the number of asterisks per line increasing from 1 to 5 and then decreasing back to 1. The prompt ">>>" is visible at the bottom left of the text area.

```
===== RESTART: C:/CookPython/Code03-03.py =====  
  *  
 ***  
*****  
*****  
*****  
*****  
*****  
*****  
  *  
 >>>
```

Section01 이 장에서 만들 프로그램

■ [프로그램2] 진수 변환

- 숫자를 세는 방법인 2진수, 8진수, 10진수, 16진수 등을 선택하고 값을 입력해 해당 진수별 숫자를 출력



```
Python 3.6.0 Shell
File Edit Shell Debug Options Window Help
===== RESTART: C:/CookPython/Code03-04.py =====
입력 진수 결정(16/10/8/2) : 16
값 입력 : FF
16진수 ==> 0xff
10진수 ==> 255
8진수 ==> 0o377
2진수 ==> 0b11111111
>>>
```

사용자가 입력한 값

Section02 print() 함수를 사용한 출력 : formatted 출력

■ print() 함수의 서식 출력

```
print("안녕하세요?")
```

결과는 '안녕하세요?'이다

```
❶ print("100")
```

```
❷ print("%d" % 100)
```

❶의 결과로 나온 100은 숫자 100(백)이 아닌 문자 100(일영영)이다.
" " 안의 내용이 문자든 숫자든 무조건 문자로 취급한다.
❷의 결과로 나온 100은 숫자 100(백)을 의미한다

```
❸ print("100 + 100")
```

```
❹ print("%d" % (100 + 100))
```

❸은 100+100이 출력되고,
❹는 숫자 100과 숫자 100을 더한 결과인 숫자 200을 출력한다.

```
❺ print("%d" % (100, 200))
```

```
❻ print("%d %d" % (100))
```

❺는 %d가 하나밖에 없는데 숫자가 2개이고,
❻은 %d가 2개인데 숫자는 하나라 서로 짝이 맞지 않다.
❻은 단순히 %d를 하나 삭제하면 되지만 ❺는 숫자 2개를 출력하려면 %d가 2개 필요하므로 [그림 3-1]과 같이 수정한다.

```
print( "%d %d" %d ( 100 , 200 ) )
```

그림 3-1 서식과 숫자의 대응

Section02 print() 함수를 사용한 다양한 출력

■ print() 함수를 사용한 다양한 출력

```
print("%d / %d = %d" % (100, 200, 0.5))
```

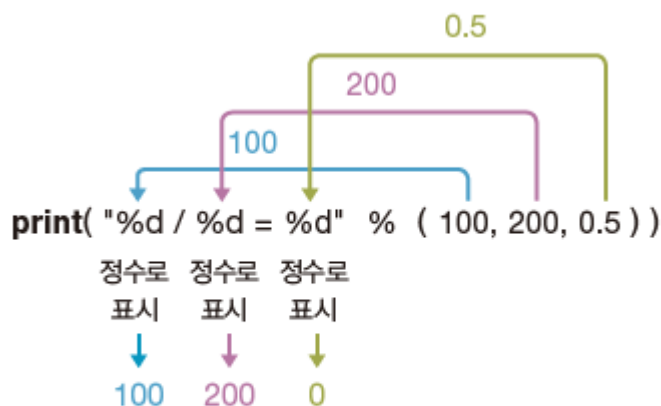
 결과는 100/200=0이다.

그림 3-2 서식과 숫자의 불일치 상황

표 3-1 print() 함수에서 사용할 수 있는 서식

서식	값의 예	설명
%d, %x, %o	10, 100, 1234	정수(10진수, 16진수, 8진수)
%f	0.5, 1.0, 3.14	실수(소수점이 붙은 수)
%c	"b", "한"	한글자
%s	"안녕", "abcdefg", "a"	두 글자 이상인 문자열

따라서 코드를 다음과 같이 수정

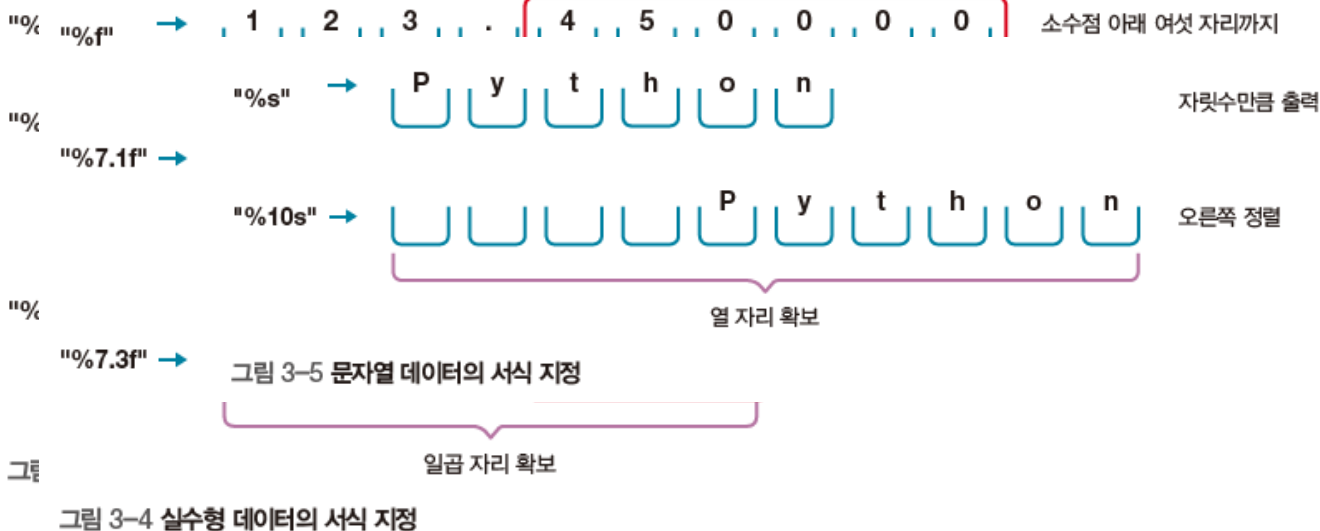
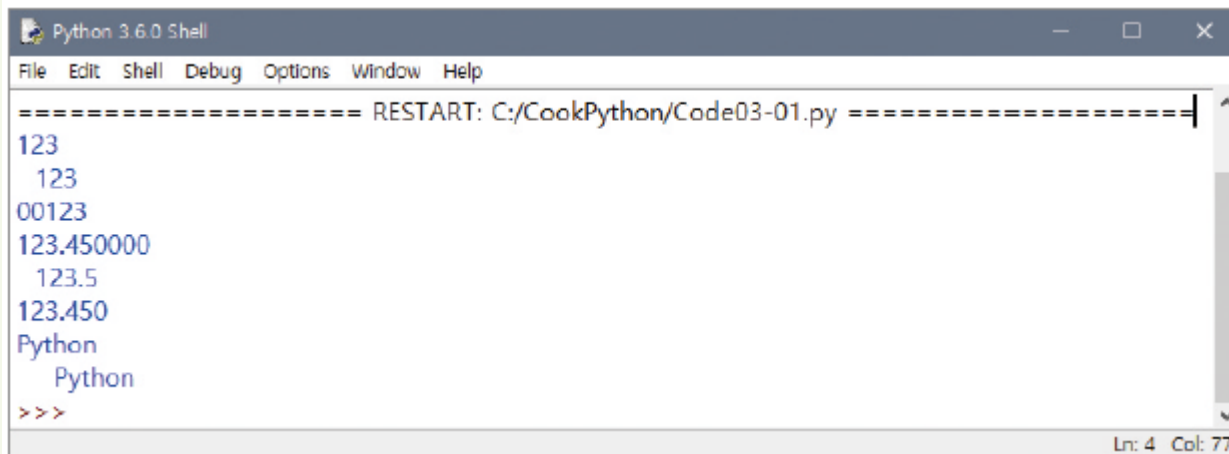
```
print("%d / %d = %5.1f" % (100, 200, 0.5))
```

Section02 print() 함수를 사용한 다양한 출력

■ print() 함수를 사용한 깔끔한 출력

Code03-01.py

```
1 print("%d" % 123)
2 print("%5d" % 123)
3 print("%05d" % 123)
4
5 print("%f" % 123.45)
6 print("%7.1f" % 123.45)
7 print("%7.3f" % 123.45)
8
9 print("%s" % "Python")
10 print("%10s" % "Python")
```



Section02 print() 함수를 사용한 다양한 출력

■ print() 함수를 사용한 깔끔한 출력

- format() 함수와 { }를 함께 사용해 서식 지정 : 함수 위치 인자(Positonal Arguments)

```
print("%d %5d %05d" % (123, 123, 123))  
print("{0:d} {1:5d} {2:05d}".format(123, 123, 123))
```

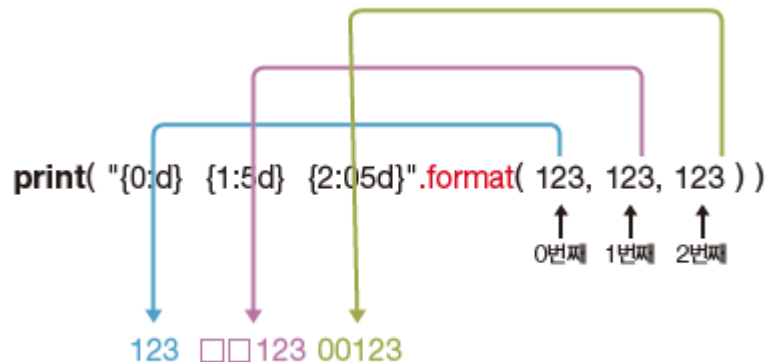


그림 3-6 format() 함수의 사용

- .format을 사용해 출력 순서 지정

```
print("{2:d} {1:d} {0:d}".format(100, 200, 300))
```

- 강제 행 넘기기는 'wn'을 사용

```
print("한 행입니다. 또 한 행입니다.")  
print("한 행입니다. \n또 한 행입니다.")
```

실습 : 결과 값은 ?

```
print("%d %d %d" % (100, 200, 300))
```

```
print("x = %d y = %5d z = %05d" % (100, 200, 300))
```

함수 위치 인자

```
print("{0:d} {1:5d} {2:05d}".format(100, 200, 300))
```

```
print("{1:5d} {0:d} {2:05d}".format(100, 200, 300))
```

함수 키워드 인자

```
print("{a}, {b}, {c}".format(a=100, b=200, c=300))
```


Section02 print() 함수를 사용한 다양한 출력

■ print() 함수를 사용한 깔끔한 출력

표 3-2 이스케이프 문자

이스케이프 문자	역할	설명
\n	새로운 줄로 이동	<input type="button" value="Enter"/> 를 누른 효과
\t	다음 탭으로 이동	<input type="button" value="Tab"/> 을 누른 효과
\b	뒤로 한 칸 이동	<input type="button" value="Backspace"/> 를 누른 효과
\\	\ 출력	
\'	' 출력	
\"	" 출력	

Code03-02.py

```
1 print("\n줄바꿈\n연습 ")
2 print("\t탭키\t연습")
3 print("글자가 \"강조\"되는 효과1")
4 print("글자가 \'강조\'되는 효과2")
5 print("\\\\\\\\ 역슬래시 세 개 출력")
6 print(r"\n \t \" \"를 그대로 출력")
```



```
Python 3.6.0 Shell
File Edit Shell Debug Options Window Help
===== RESTART: C:/Co

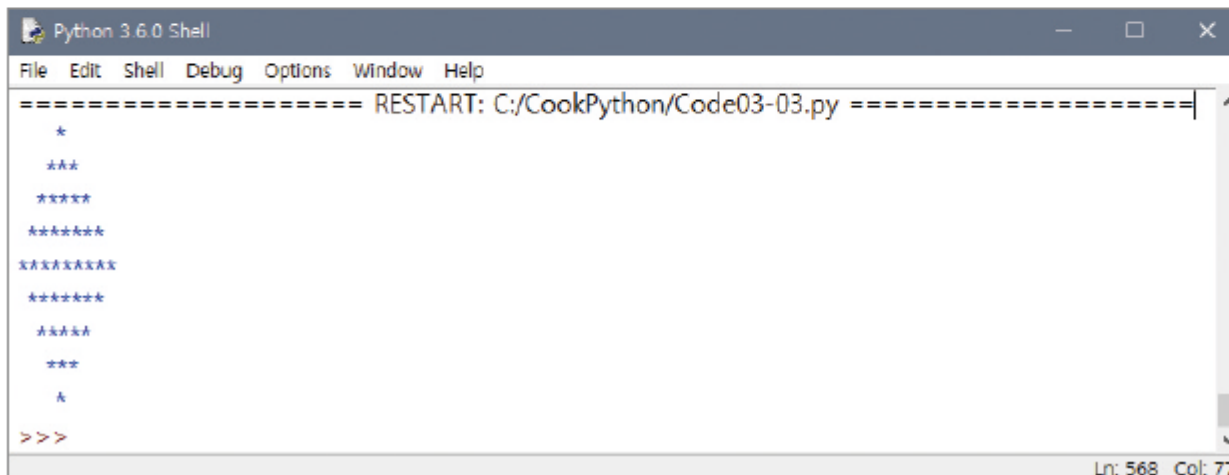
줄바꿈
연습
        탭키    연습
글자가 "강조"되는 효과1
글자가 '강조'되는 효과2
\\\\ 역슬래시 세 개 출력
\n \t \" \"를 그대로 출력
>>>
```

Section02 print() 함수를 사용한 다양한 출력

■ [프로그램 1]의 완성

Code03-03.py

```
1 print("  *  ")
2 print(" *** ")
3 print(" ***** ")
4 print(" ******* ")
5 print("*****")
6 print(" ******* ")
7 print(" ***** ")
8 print("  *** ")
9 print("   *   ")
```



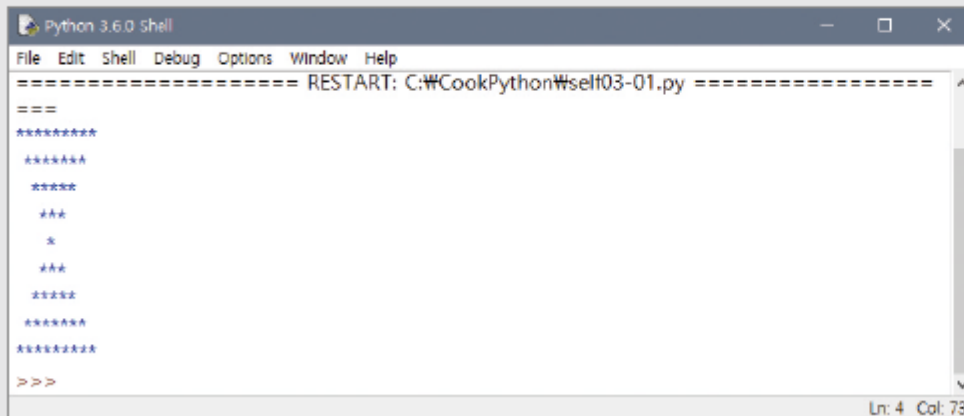
```
Python 3.6.0 Shell
File Edit Shell Debug Options Window Help
===== RESTART: C:/CookPython/Code03-03.py =====
 *
 ***
 *****
 *******
*****
 *******
 *****
  ***
   *
>>>
```

Ln: 568 Col: 77

Section02 print() 함수를 사용한 다양한 출력

SELF STUDY 3-1

별표가 출력되도록 print() 문을 작성해 보자.



```
Python 3.6.0 Shell
File Edit Shell Debug Options Window Help
===== RESTART: C:\CookPython\self03-01.py =====
===
*****
*****
*****
***
*
***
*****
*****
*****
>>>
```

Ln: 4 Col: 73

Section03 변수의 선언과 사용

■ 변수의 선언

- 변수는 어떠한 값을 저장하는 메모리 공간(그릇)
- 변수 선언은 그릇을 준비하는 것
- 파이썬은 C/C++, 자바 등과는 달리 변수를 선언하지 않아도 되지만 긴 코드를 작성할 때는 사용될 변수를 미리 계획적으로 준비하는 것이 더 효율적

```
boolVar = True
intVar = 0
floatVar = 0.0
strVar = ""
```

TIP • 이 구문은 다음과 같이 표현해도 된다.

```
boolVar, intVar, floatVar, strVar = True, 0, 0.0, ""
```

- 가장 많이 사용하는 변수는 불형(Boolean, True 또는 False 저장), 정수형, 실수형, 문자열



그림 3-7 변수의 종류

Section03 변수의 선언과 사용

- Type() 함수를 사용하면 변수가 bool(불형), int(정수), float(실수), str(문자열)형으로 생성된 것을 확인할 수 있음

```
type(boolVar), type(intVar), type(floatVar), type(strVar)
```

출력 결과

```
(<class 'bool'>, <class 'int'>, <class 'float'>, <class 'str'>)
```

■ 변수명 규칙

- 대·소문자를 구분한다(myVar와 MyVar는 다른 변수).
- 문자, 숫자, 언더바(_)를 포함할 수 있다. 하지만 숫자로 시작하면 안 된다(var2(O), _var(O), var_2(O), 2Var(X)).
- 예약어는 변수명으로 쓰면 안 된다. 파이썬의 예약어는 True, False, None, and, or, not, break, continue, return, if, else, elif, for, while, except, finally, gloval, import, try 등이다.

Section03 변수의 선언과 사용

■ 변수의 사용(1)

- 변수는 값을 담으면(대입하면) 사용 가능. 변수에 있던 기존 값은 없어지고 새로 입력한 값으로 변경됨

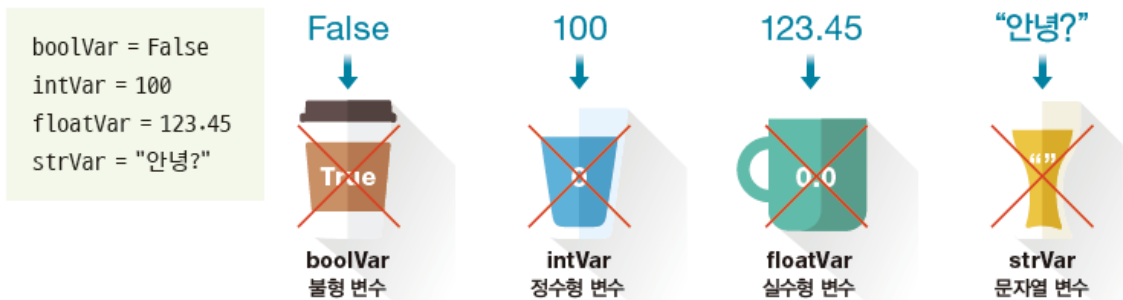


그림 3-8 변수에 값을 대입해 새로운 값으로 변경된 상태

- 변수에는 변수의 값을 넣을 수도 있고, 계산 결과를 넣을 수도 있음

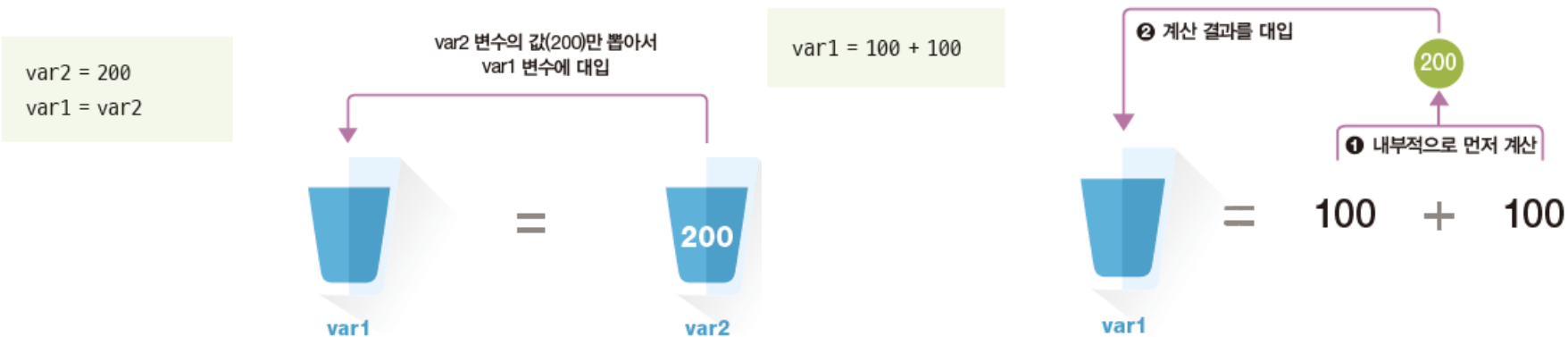


그림 3-9 변수에 변수를 대입하는 방식

그림 3-10 숫자끼리 연산한 결과를 대입하는 방식

Section03 변수의 선언과 사용

■ 변수의 사용(2)

- 변수에는 숫자와 변수의 연산을 넣을 수도 있음

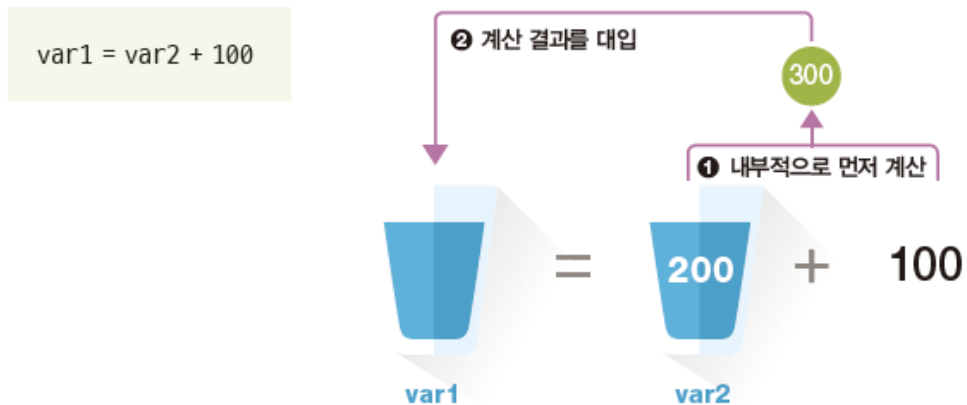


그림 3-11 변수와 숫자를 연산한 결과를 대입하는 방식

Section03 변수의 선언과 사용

■ 변수의 사용(3)

- 변수에 연속된 값을 대입하는 방식

```
var1 = var2 = var3 = var4 = 100
```

또는

```
var4 = 100  
var3 = var4  
var2 = var3  
var1 = var2
```

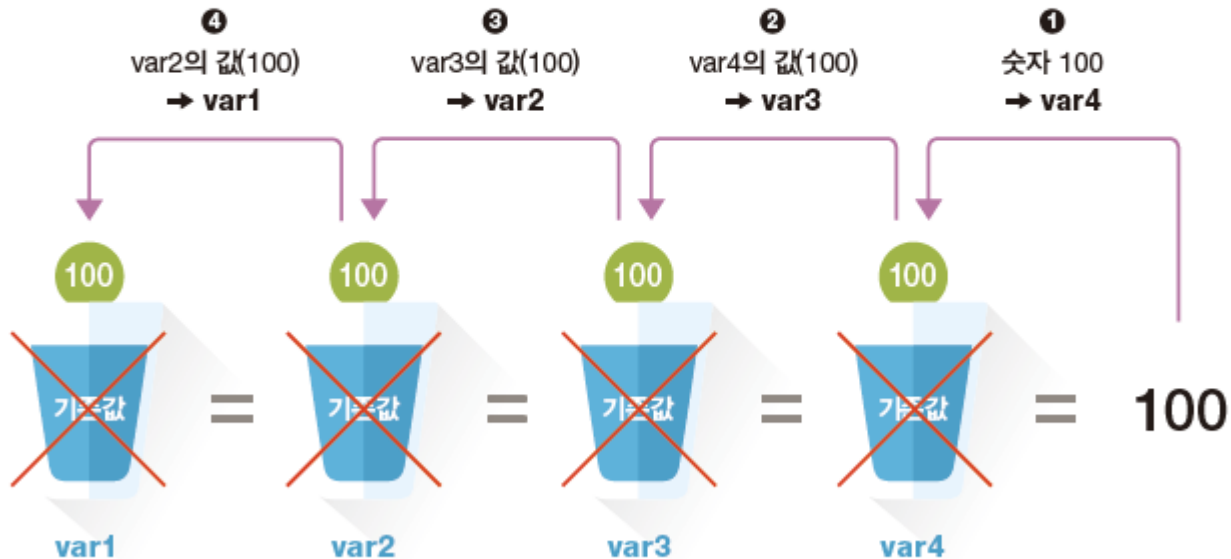


그림 3-12 연속된 값을 대입하는 방식

■ 변수의 사용(4)

- 변수에 연산 결과를 자신의 값으로 다시 대입하는 방식

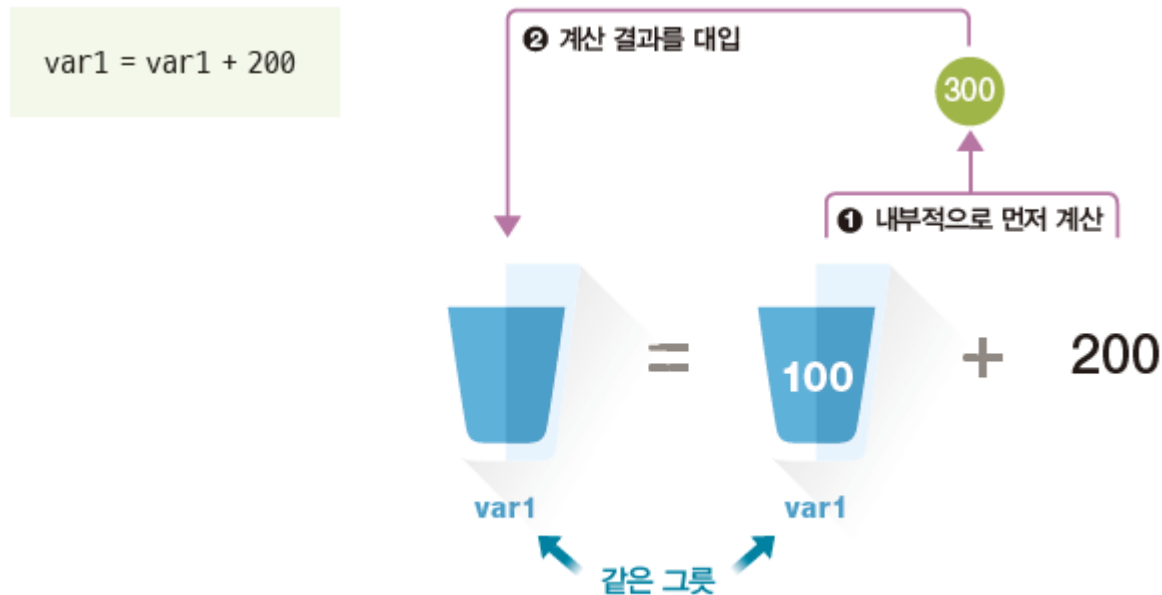


그림 3-13 연산 결과를 자신의 값에 다시 대입하는 방식

Section03 변수의 선언과 사용

■ 변수의 사용(5)

- 파이썬에서 변수의 데이터 형식은 값을 넣는 순간마다 변경될 수 있는 유연한 구조

```
myVar = 100          # 정수형 변수를 생성(국 그릇 생성)
type(myVar)          # <class 'int'>가 출력
myVar = 100.0        # 이 순간에 실수형 변수로 변경(밥 그릇으로 변경)
type(myVar)          # <class 'float'>가 출력
```

- 대입 연산자의 왼쪽에는 무조건 변수만 올 수 있고, 오른쪽에는 무엇이든(값, 변수, 수식, 함수 등) 올 수 있음



그림 3-14 왼쪽에 값을 넣을 그릇이 없음



그림 3-15 왼쪽에 값을 넣을 그릇이 있음

■ 비트와 바이트

- 컴퓨터에서 표현할 수 있는 제일 작은 단위는 비트(Bit)
- 비트 8개가 모이면 바이트(Byte)

■ 비트

- 비트는 0과 1만 존재하므로 1비트로는 두 가지를 표현 가능

전기 스위치								
의미	꺼짐 , 꺼짐		꺼짐 , 켜짐		켜짐 , 꺼짐		켜짐 , 켜짐	
2진수	00		01		10		11	
10진수	0		1		2		3	

그림 3-16 전기 스위치 2개와 2진수, 10진수의 비교

n 개의 전기 스위치로 표현할 수 있는 가짓수 = 2^n

■ 비트

표 3-3 10진수, 2진수, 16진수 변환표

10진수(0~9)	2진수(0~1)	16진수(0~F)
00	0000	0
01	0001	1
02	0010	2
03	0011	3
04	0100	4
05	0101	5
06	0110	6
07	0111	7
08	1000	8
09	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

■ 바이트

표 3-4 비트와 바이트 크기에 따른 숫자의 범위

비트 수	바이트 수	표현 개수	2진수	10진수	16진수
1		$2^2=2$	0~1	0~1	0~1
2		$2^4=4$	0~11	0~3	0~3
4		$2^4=16$	0~1111	0~15	0~F
8	1	$2^8=256$	0~11111111	0~255	0~FF
16	2	$2^{16}=65536$	0~11111111 11111111	0~65535	0~FFFF
32	4	$2^{32}=\text{약 } 42\text{억}$	0~...	0~약 42억	0~FFFF FFFF
64	8	$2^{64}=\text{약 } 1800\text{경}$	0~... ..	0~약 1800경	0~... ..

Section04 데이터 표현 단위와 진수 변환

■ 진수 변환

2진수

1	0	0	1		0	0	1	1
×	×	×	×		×	×	×	×
2^7	2^6	2^5	2^4		2^3	2^2	2^1	2^0
128	0	0	16		0	0	2	1

↑ + ↑

10진수

147

그림 3-17 2진수를 10진수로 변환하는 방법

2진수

1	0	0	1		0	0	1	1
×	×	×	×		×	×	×	×
2^3	2^2	2^1	2^0		2^3	2^2	2^1	2^0
8	+	0	+	0	+	0	+	1

↓ ↓

16진수

9		3
×		×
16^1		16^0
144		3

↑ + ↑

10진수

147

그림 3-18 2진수를 16진수로 변환한 후 10진수로 변환하는 방법

Section04 데이터 표현 단위와 진수 변환

■ 진수 변환

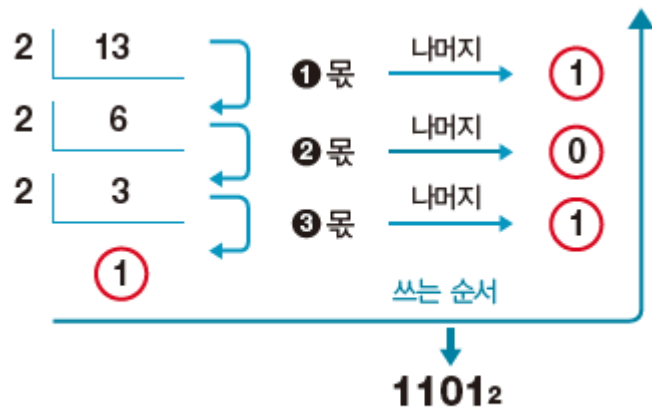


그림 3-19 10진수를 2진수로 변환하는 방법

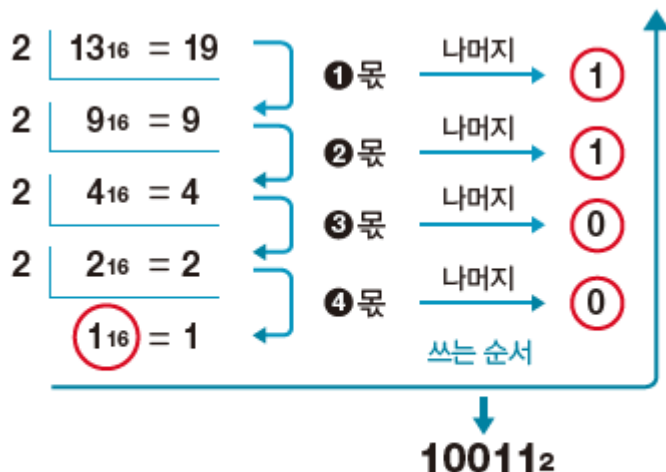


그림 3-20 16진수를 2진수로 변환하는 방법

표 3-5 16진수, 2진수 변환표

16진수	2진수	16진수	2진수
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

Section04 데이터 표현 단위와 진수 변환

■ 진수 변환

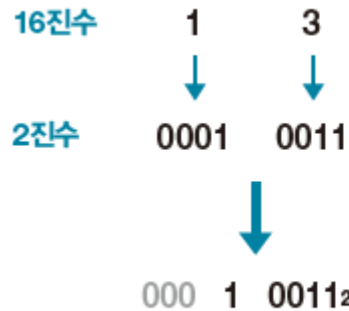


그림 3-21 16진수를 2진수로 변환하는 예 1

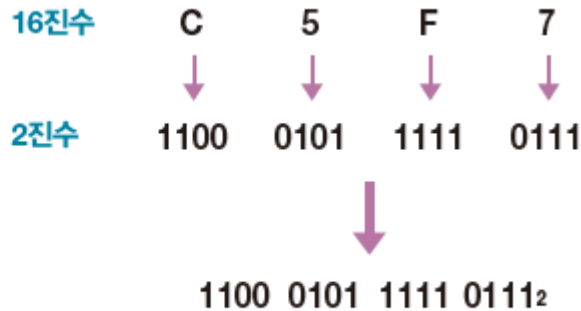


그림 3-22 16진수를 2진수로 변환하는 예 2

10진수 8진수 16진수 → 2진수: ϕb
bin(11); bin(0b11); bin(0x11)
oct(11); oct(0b11); oct(0x11) → 8진수: ϕo
hex(11); hex(0b11); hex(0x11) → 16진수: ϕx

출력 결과

'0b1011' '0b1001' '0b10001'
'0o13' '0o3' '0o21'
'0xb' '0x3' '0x9'

TIP • 16진수 C5F7₁₆을 10진수로 변환하면 50679이지만, 굳이 10진수로 계산할 필요는 없다.

➤ 참고 : <https://bit.ly/3i4eTQe>

Section04 데이터 표현 단위와 진수 변환

■ [프로그램 2]의 완성

Code03-04.py

```
1 sel = int(input("입력 진수 결정(16/10/8/2) : "))
2 num = input("값 입력 : ")
3
4 if sel == 16 :
5     num10 = int(num, 16)
6 if sel == 10 :
7     num10 = int(num, 10)
8 if sel == 8 :
9     num10 = int(num, 8)
10 if sel == 2 :
11     num10 = int(num, 2)
12
13 print("16진수 ==> ", hex(num10))
14 print("10진수 ==> ", num10)
15 print(" 8진수 ==> ", oct(num10))
16 print(" 2진수 ==> ", bin(num10))
```

```
Python 3.6.0 Shell
File Edit Shell Debug Options Window Help
===== RESTART: C:/CookPython/Code03-04.py =====
입력 진수 결정(16/10/8/2) : 16
값 입력 FF
16진수 ==> 0xff
10진수 ==> 255
 8진수 ==> 0o377
 2진수 ==> 0b11111111
>>>
```

사용자가 입력한 값

Ln: 579 Col: 77

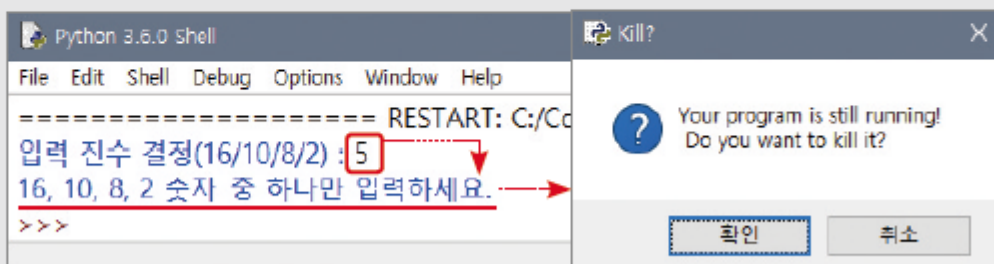
Section04 데이터 표현 단위와 진수 변환

SELF STUDY 3-2

Code03-04.py는 16, 10, 8, 2 이외의 숫자를 입력하면 오류가 발생한다. 코드를 수정해서 16, 10, 8, 2 외의 숫자를 입력하면 '16, 10, 8, 2 숫자 중 하나만 입력하세요'라는 메시지를 출력하고 프로그램을 종료하도록 if 문을 추가해 보자.

힌트1 같지 않다는 !=로 비교한다.

힌트2 여러 조건이 동시에 참이어야 하는 관계 연산자는 and를 사용한다.



■ 숫자형(정수형과 실수형)(1)

```
a = 123  
type(a)
```

변수에 값을 넣는 순간에 변수의 데이터형이 결정된다

출력 결과

```
<class 'int'>
```

```
a = 100 ** 100  
print(a)
```

int의 크기에는 제한이 없다.

출력 결과

```
10000000~000000
```

■ 숫자형(정수형과 실수형)(2)

```
a = 0xFF  
b = 0o77  
c = 0b1111  
print(a, b, c)
```

정수형에는 16진수, 8진수, 2진수도 사용할 수 있다.

출력 결과

255 63 15

```
a = 3.14  
b = 3.14e5  
print(a, b)
```

실수형은 3.14, -2.7처럼 소수점이 있는 데이터이다.
또 3.14e5처럼 표현할 수도 있다. 3.14e5는
3.14*10⁵을 의미한다.

출력 결과

3.14 314000.0

■ 숫자형(정수형과 실수형)(3)

```
a = 10; b = 20
```

```
print(a + b, a - b, a * b, a / b)
```

 정수 및 실수 데이터형은 사칙 연산 +, -, *, /를 수행할 수 있다.

출력 결과

```
30 -10 200 0.5
```

```
a, b = 9, 2
```

```
print(a ** b, a % b, a // b)
```

제곱을 의미하는 **, 나머지를 구하는 %,
나눈 후에 소수점을 버리는 // 연산자도 사용할 수 있다.

출력 결과

```
81 1 4
```

■ 불형

```
a = True  
type(a)
```

불(Bool)형은 참(True)이나 거짓(False)만 저장할 수 있다.

출력 결과

```
<class 'bool'>
```

```
a = (100 == 100)  
b = (10 > 100)  
print(a, b)
```

불형은 비교의 결과를 참이나 거짓으로 저장하는 데 사용될 수도 있다.

출력 결과

```
True False
```

■ 문자열(1)

```
a = "파이썬 만세"  
a  
print(a)  
type(a)
```

문자열을 'abc', "파이썬 만세", "1" 등 문자집합을 의미한다.
문자열은 양쪽을 큰따옴표(")나 작은따옴표(')로 감싸야 한다.

출력 결과

```
'파이썬 만세'  
파이썬 만세  
<class 'str'>
```

```
"작은따옴표는 ' 모양이다."  
'큰따옴표는 " 모양이다.'
```

문자열 중간에 작은따옴표나 큰따옴표를 출력하고 싶다면
다른 따옴표로 묶어 주면 된다.

출력 결과

```
"작은따옴표는 ' 모양이다."  
'큰따옴표는 " 모양이다.'
```

■ 문자열(2)

```
a = "이건 큰따옴표 \" 모양."  
b = '이건 작은따옴표 \' 모양.'  
print(a, b)
```

역슬래시(\) 뒤에 큰따옴표나 작은따옴표를 사용해도
글자로 인식한다.

출력 결과

이건 큰따옴표 " 모양. 이건 작은따옴표 ' 모양.

```
a = '파이썬 \n만세'  
print(a)
```

문자열을 여러 줄로 넣으려면
중간에 \n을 포함시키면 된다.

출력 결과

파이썬
만세

```
a = """파이썬  
만세"""  
a  
print(a)
```

작은따옴표나 큰따옴표 3개를
연속해서 묶어도 된다.

출력 결과

'파이썬\n만세'
파이썬
만세

Section05 기본 데이터형

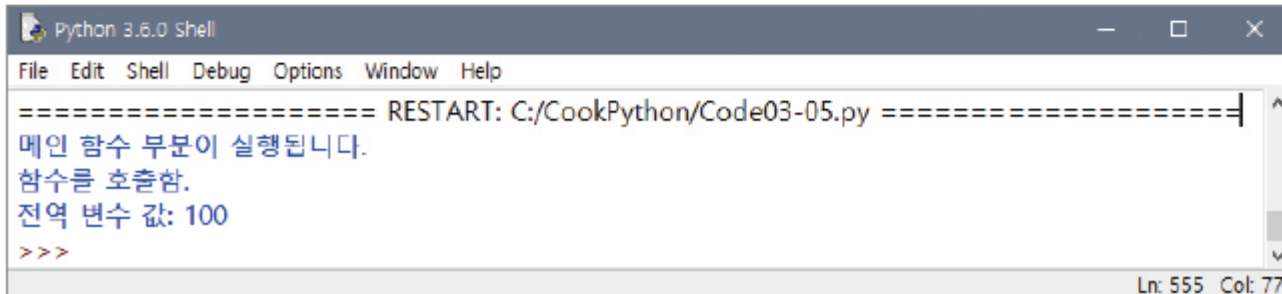
Code03-05.py

```
1  ## 함수 선언 부분 ##
2  def myFunc() :
3      print('함수를 호출함.')
4
5  ## 전역 변수 선언 부분 ##
6  gVar = 100
7
8  ## 메인 코드 부분 ##
9  if __name__ == '__main__' :
10     print('메인 함수 부분이 실행됩니다.')
11     myFunc()
12     print('전역 변수 값:', gVar)
```

```
def main() :
    print('메인 함수 부분이 실행됩니다.')
    myFunc()
    print('전역 변수 값:', gVar)

if __name__ == '__main__' :
    main()
```

파이썬 내장변수



```
Python 3.6.0 Shell
File Edit Shell Debug Options Window Help
===== RESTART: C:/CookPython/Code03-05.py =====
메인 함수 부분이 실행됩니다.
함수를 호출함.
전역 변수 값: 100
>>>
Ln: 555 Col: 77
```



Thank You
