컴퓨팅 사고력을 키우는 SW 교육

파이썬

Chapter 02
자료형 및 연산자

Section 01 print()의 서식

print( )의 서식 (1)

- **서식을 지원하는 print( ) 함수 사용법**
  - 서식은 앞에 %가 붙음. %d는 정수(Decimal)를 의미.

```
print("안녕하세요?")
```

➔ 안녕하세요

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

➔ 글자 100(일영영)

➔ 숫자 100

```
print("100+100")
print("%d" % (100+100) )
```

➔ 글자 100+100

➔ 숫자 200

print( )의 서식 (2)

- 서식의 개수와 % 뒤에 나오는 숫자(또는 문자)의 개수가 같아야 함

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

➔ 오류발생

첫 번째 행에는 %d가 하나밖에 없는데 숫자는 두 개(100, 200)가 나왔고, 두 번 째 행에는 %d가 두 개인데, 숫자는 하나(100)밖에 나오지 않음

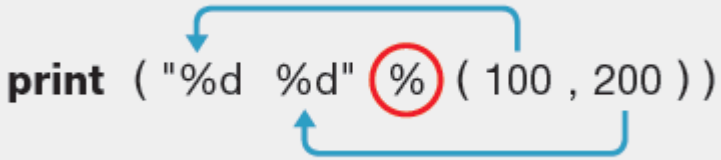


그림 3-3
서식과 숫자의 대응

print( )의 서식 (3)

- **정수(%d) 외에 자주 사용되는 서식**

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

➔ 100/200=0.5 가 아닌 100/200=0 이 나옴.

세 번째 숫자 0.5는 실수(소수점이 있는 수)이지만 보여주는 방식이 정수임

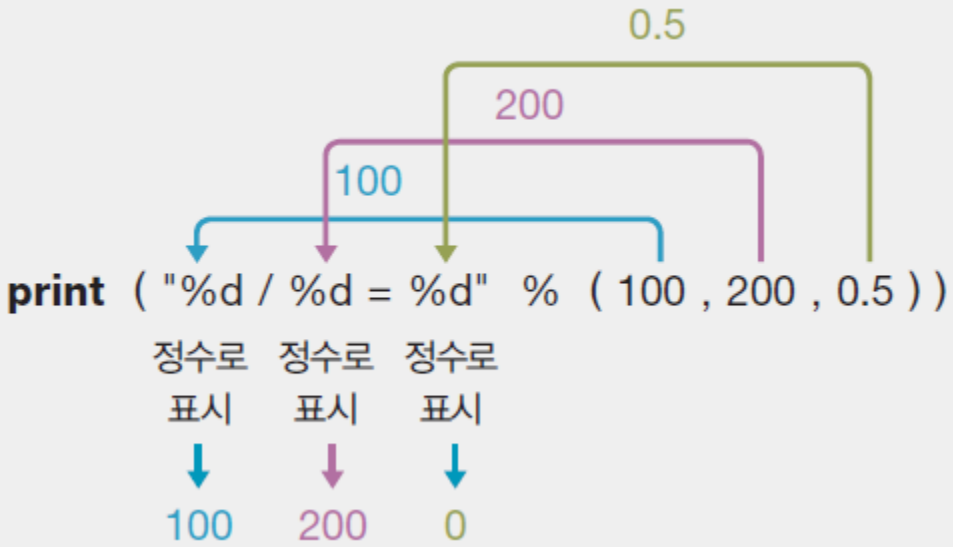


그림 3-4
서식과 숫자의 불일치 상황

print( )의 서식 (4)

표 3-1
print( )에서 사용하는 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"	문자 한 글자
%s	"안녕", "abcdefg", "a"	한 글자 이상의 문자열

- 세 번째 %d 대신에 %f로 수정

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

print( )의 서식 (5)

- **서식 출력 연습**

소스코드 3-1
(파일명 : 03-01.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")
```

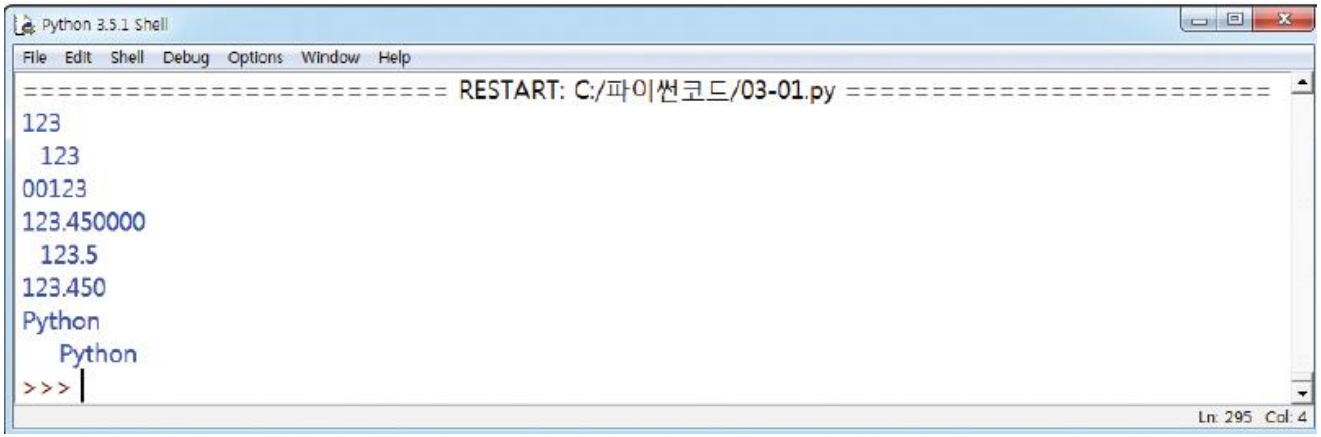


그림 3-5
실행 결과

print( )의 서식 (6)

- 정수형 데이터 서식 지정

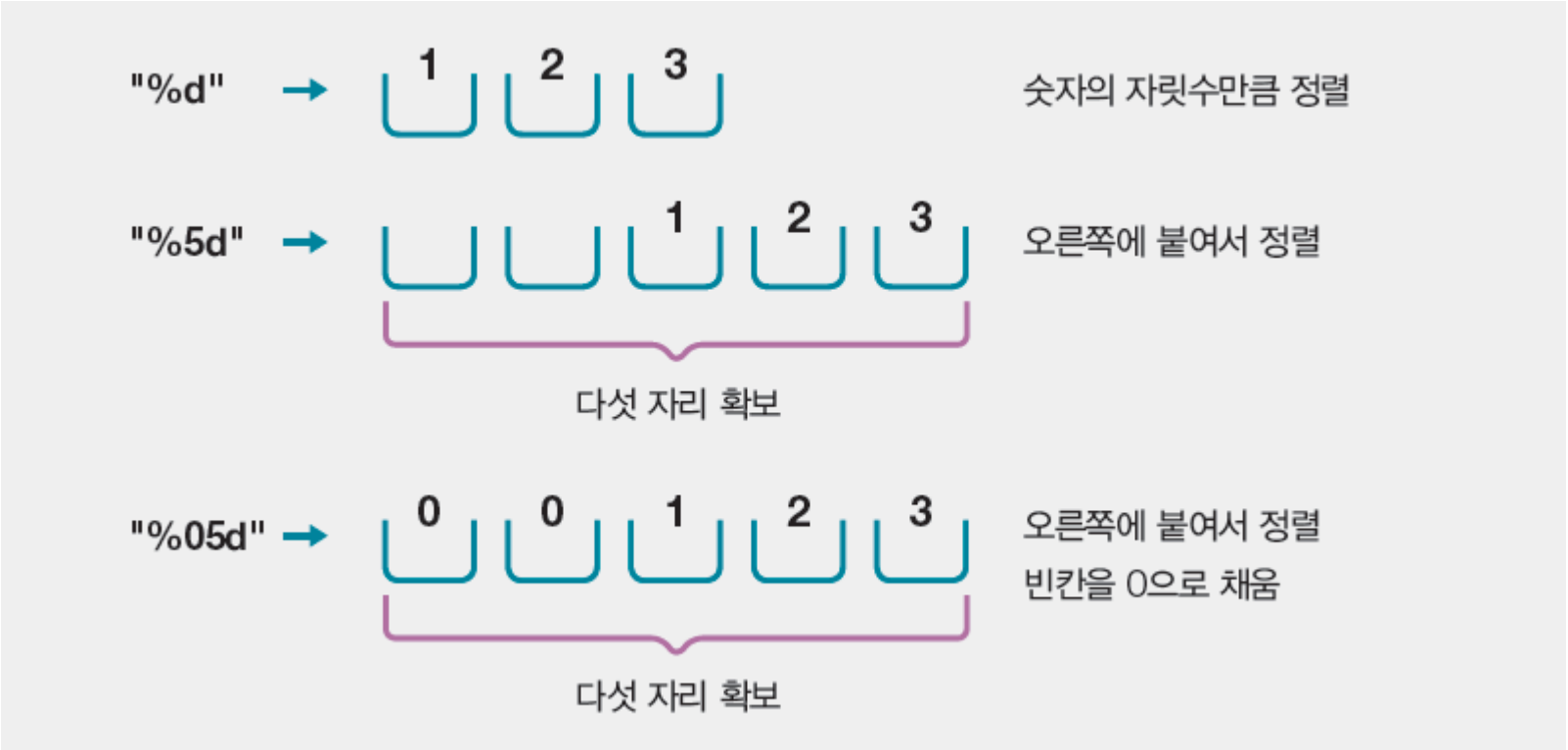


그림 3-6
정수형 데이터 서식 지정

print( )의 서식 (7)

- 실수 형 데이터의 서식 지정
  - 두 번째 %7.1f는 소수점 을 포함한 전체 자리인 일곱 자리를 확보하고 소수점 아래는 한 자리만 차지한다는 의미

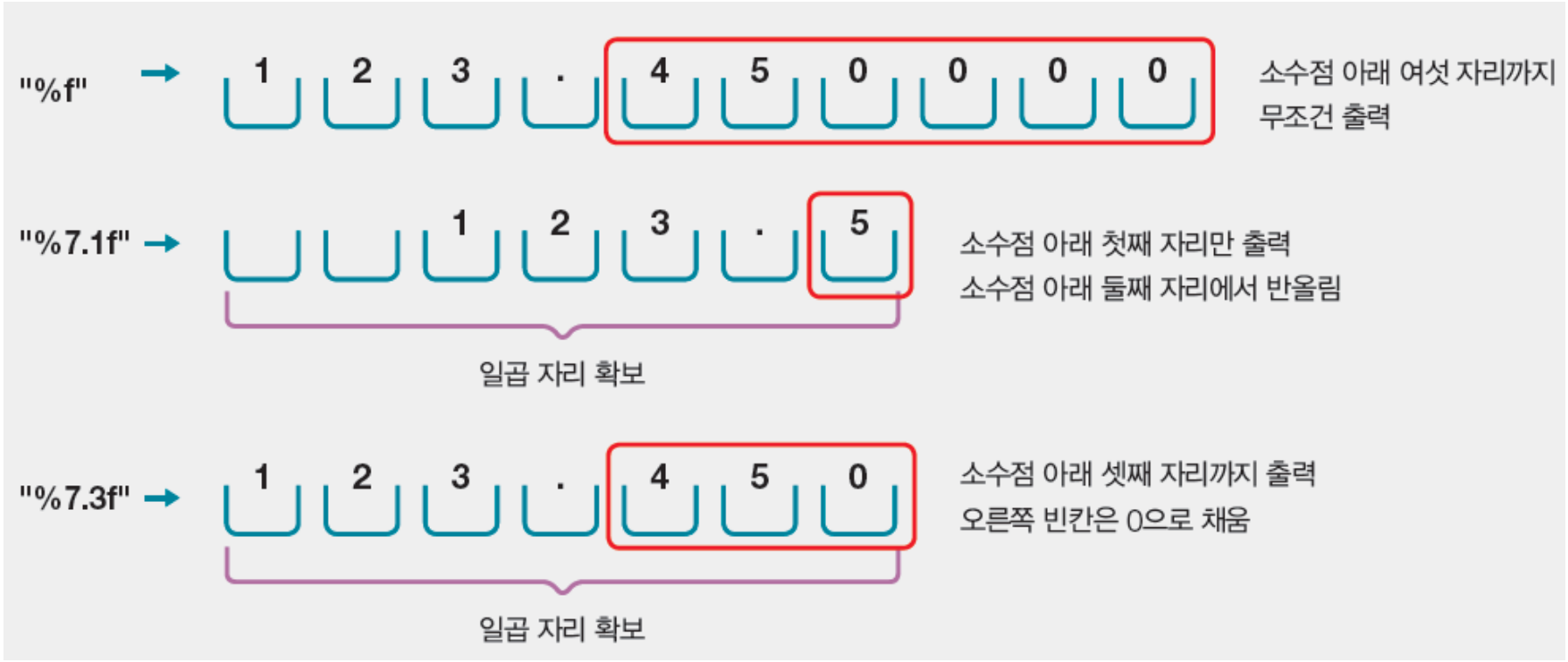


그림 3-7
실수형 데이터 서식 지정

print( )의 서식 (8)

- 문자열 형 데이터 서식 지정

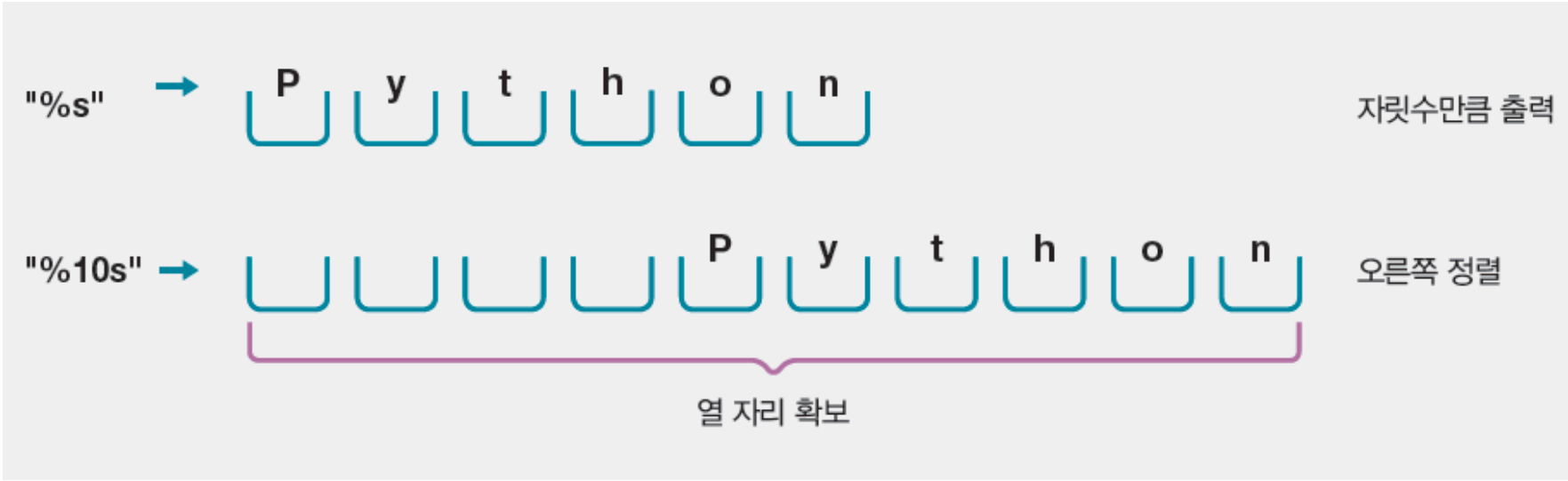


그림 3-8
문자열형 데이터 서식 지정

print( )의 서식 (9)

- format( ) 함수의 사용

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

➔ 두 행은 동일 결과를 출력.

두 번째 행에서 { } 안의 0, 1, 2는 format( ) 안의 0번째, 1번째, 2번째 값에 대응한다는 의미. %d 에서 %를 떼고 d로 표시.

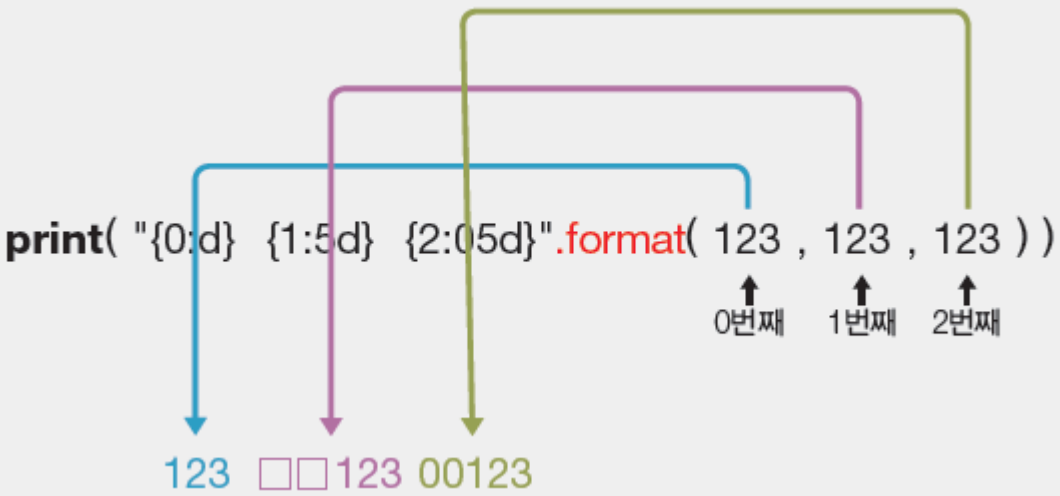


그림 3-9
format( ) 함수의 사용

print( )의 서식 (10)

- **다양한 이스케이프 문자**
  - print( )문은 내용을 출력한 후에 한 행을 넘겨줌

```
print("한행입니다. 또 한행입니다")
print("한행입니다. \n또 한행입니다")
```

표 3-2
이스케이프 문자

이스케이프 문자	역할	비고
\n	새로운 줄로 이동	Enter 키를 누른 효과
\t	다음 탭으로 이동	Tab 키를 누른 효과
\b	뒤로 한 칸 이동	Backspace 키를 누른 효과
\\	\ 출력	
\‘	‘ 출력	
\“	“ 출력	

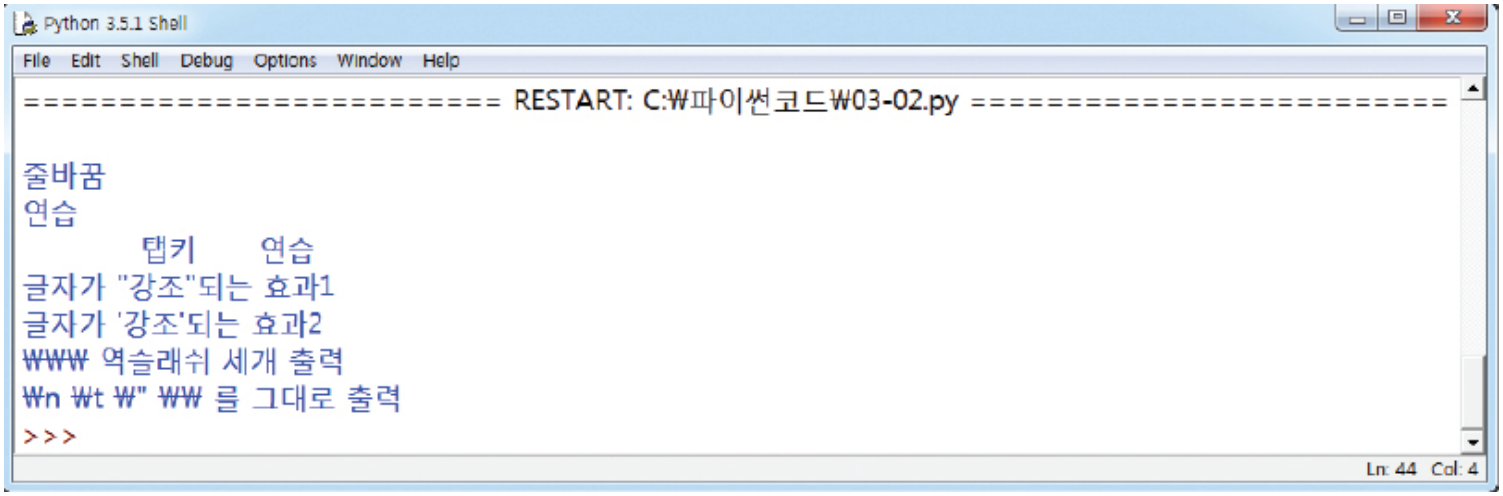
print( )의 서식 (11)

- 이스케이프 문자 활용

소스코드 3-2
(파일명 : 03-02.py)

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

그림 3-10
실행 결과



print( )의 서식 (12)

- **별표 출력 프로그램 완성**

소스코드 3-3

(파일명 : 03-03.py)

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

Section 02 변수

변수를 확실히 이해합시다(1)

- **변수의 선언**
  - 변수는 어떤 값을 저장하기 위한 메모리 공간. '그릇'이라고 생각함
  - 가장 많이 사용하는 변수

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

그림 3-11
변수 종류의 개념

변수를 확실히 이해합시다(2)

- type( ) 함수는 변수의 종류를 확인하는 함수

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

출력 결과

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

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

```
myVar=100   → 정수형 변수를 생성함 (=국그릇 생성)
type(myVar) → <class 'int'>가 출력됨
myVar=100.0 → 이 순간에 실수형 변수로 변경됨 (=밥그릇으로 변경됨)
type(myVar) → <class 'float'>가 출력됨
```

변수를 확실히 이해합시다(11)

변수명

- 문자, 숫자, 밑줄(_)로 구성됩니다.
  숫자는 처음에 나올 수 없습니다.
- 대소문자를 구분합니다.
- 예약어 사용 불가.

- ex.

```
>>> friend = 10
>>> Friend = 1
>>> friend
10
>>> Friend
1
```

Section 03 비트, 바이트와 진수

비트와 바이트, 진수에 대해 알아봅시다(6)

- 파이썬에서는 2진수를 입력하려면 앞에 0b 또는 0B를 붙여줌. 출력 시 10진수로 나옴.

```
0b10010011
```

출력 결과

```
147
```

- 또는 int('숫자' , 진수)를 사용하면 10진수로 변환

```
int('10010011', 2)
```

출력 결과

```
147
```

- 16진수는 앞에 0x 또는 0X를 붙여주거나 int() 함수를 사용해 10진수로 변환.

```
0x93 ; int('93', 16)
```

출력 결과

```
147
147
```

비트와 바이트, 진수에 대해 알아봅시다(9)

- hex(숫자), oct(숫자), bin(숫자) 함수
  - hex()는 16진수로, oct()는 8진수로, bin()은 2진수로 결과 출력

```
bin(13) ; bin(0x13) ; bin(0xC5F7)
```

출력 결과

```
'0b1101'
'0b10011'
'0b1100010111110111'
```

비트와 바이트, 진수에 대해 알아봅시다(10)

- **진수 변환 프로그램 완성**
  - 조건식이 참일 경우에 if문 아래가 수행

```
if 조건식 :
  참일 때 수행
```

소스코드 3-4
(파일명 : 03-04.py)

```
1  sel=int(input("입력진수 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))
```

Section 04 기본 자료형

기본 자료형 (1) - integer

정수 데이터 형식

- 정수형은 소수점이 없는 데이터임. (기본적인 정수형 100, -123, 0 등)
- 파이썬은 변수의 선언이 없으며 변수에 값을 넣는 순간 변수의 데이터 형식이 결정됨.

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

출력 결과

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

- int는 기본적인 정수 데이터 형식이며 파이썬 버전 3.5에서는 int 크기의 제한이 없음

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

출력 결과

```
1000000 ~~~ 00000
```

기본 자료형 (2)

- 16진수는 0x나 0X로, 8진수는 0o나 0O(숫자+알파벳 오)로, 2진수는 0b나 0B로 표현

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

출력 결과

```
255 63 15
```

기본 자료형 (3) - float

실수 데이터 형식

- 실수형은 3.14, -2.7처럼 소수점이 있는 데이터

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

출력 결과

```
3.14 314000.0
```

- 사칙연산인 +, -, *, /를 수행

```
a=10; b=20
print(a+b, a-b, a*b, a/b)
```

출력 결과

```
30 -10 200 0.5
```

기본 자료형 (4)

- 제곱을 의미하는 **, 나머지를 구하는 %, 나눈 후에 소수점을 버리는 // 연산자 사용

```
a,b=9,2
print(a**b, a%b, a//b)
```

출력 결과

```
81 1 4
```

기본 자료형 (5) - Boolean

- **불 데이터 형식**
  - 참(True)이나 거짓(False)만 저장
  - 불형은 단독으로 사용하기보다는 if 조건문 이나 while 반복문 등과 함께 주로 사용

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

출력 결과

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

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

출력 결과

```
True False
```

기본 자료형 (6) - String

■ 문자열 데이터 형식

- 문자열은 양쪽을 큰따옴표(“”)나 작은따옴표(‘’)로 감싸야 함

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

출력 결과

```
'파이썬 만세'
파이썬 만세
<class 'str'>
```

```
"작은따옴표는 ' 모양입니다."
'큰따옴표는 " 모양입니다.'
```

출력 결과

```
"작은따옴표는 ' 모양입니다."
'큰따옴표는 " 모양입니다.'
```

기본 자료형 (7) - String

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

출력 결과

```
이건 큰따옴표 " 모양. 이건 작은따옴표 ' 모양.
```

```
a='파이썬 \n만세'
print(a)
```

출력 결과

```
파이썬
만세
```

```
a="""파이썬
만세"""
a
print(a)
```

출력 결과

```
'파이썬\n만세'
파이썬
만세
```

기본 자료형 (8) - String

+, * 연산자

```
>>> 'py' 'thon'    # ( == 'py'+'thon')
'python'
>>> 'py' * 3
'pypypy'
```

인덱싱 & 슬라이싱

```
>>> 'python'[0]
'p'
>>> 'python'[5]
'n'
>>> 'python'[1:4]
'yth'
>>> 'python'[-2:]
'on'
```

p	y	t	h	o	n	
0	1	2	3	4	5	6
-6	-5	-4	-3	-2	-1	

기본 자료형 (9) - String

```
>>> a[:]
'python'
>>> a[::2]
'pto'
```

Section 05 연산자

연산자 - 산술 연산자(1)

- **산술 연산자**

표 4-1
산술 연산자의 종류

산술 연산자	설명	사용 예	예 설명
=	대입 연산자	a = 3	정수 3을 a에 대입
+	더하기	a = 5 + 3	5와 3을 더한 값을 a에 대입
−	빼기	a = 5 − 3	5에서 3을 뺀 값을 a에 대입
*	곱하기	a = 5 * 3	5와 3을 곱한 값을 a에 대입
/	나누기	a = 5 / 3	5를 3으로 나눈 값을 a에 대입
//	나누기(몫)	a = 5 // 3	5를 3으로 나눈 뒤 소수점을 버리고 a에 대입
%	나머지 값	a = 5 % 3	5를 3으로 나눈 뒤 나머지 값을 a에 대입
**	제곱	a = 5 ** 3	5의 3제곱을 a에 대입

```
a = 5; b = 3
print(a + b, a - b, a * b, a / b, a // b, a % b, a ** b)
```

출력 결과

```
8 2 15 1.6666666666666667 1 2 125
```

연산자 - 산술 연산자(2)

우선순위

- 산술 연산자의 우선순위는 괄호가 가장 우선, 곱셈(또는 나눗셈)이 그 다음, 덧셈(또는 뺄셈)이 가장 마지막으로 수행.
- 덧셈(또는 뺄셈)끼리 나오거나 곱셈(또는 나눗셈)끼리 나오면 왼쪽에서 오른쪽으로 계산이 진행됨

```
a,b,c = 2,3,4
print(a + b - c, a + b * c, a * b / c)
```

출력 결과

```
1 14 1.5
```

연산자 - 산술 연산자(3)

문자열과 숫자의 상호 변환

- 문자열이 int() 함수에 의해서 정수로, float() 함수에 의해서 실수로 변경

```
s1, s2, s3 = "100", "100.123", "99999999999999999999999999"
print(int(s1) + 1, float(s2) + 1, int(s3) + 1)
```

출력 결과

```
101 101.123 100000000000000000000000000
```

- 숫자를 문자열로 변환하기 위해서는 str() 함수를 사용

```
a = 100; b = 100.123
str(a) + '1'; str(b) + '1'
```

출력 결과

```
'1001'
'100.1231'
```

연산자 - 산술 연산자(4)

- **대입 연산자**

표 4-2
대입 연산자의 종류

대입 연산자	사용 예	예 설명
+=	a += 3	a = a + 3과 동일
-=	a -= 3	a = a - 3과 동일
*=	a *= 3	a = a * 3과 동일
/=	a /= 3	a = a / 3과 동일
//=	a //= 3	a = a // 3과 동일
%=	a %= 3	a = a % 3과 동일
**=	a **= 3	a = a ** 3과 동일

연산자 - 산술 연산자(5)

- a는 10에서 시작하여 프로그램이 진행될수록 값이 누적됨

```
a = 10
a += 5; print(a)
a -= 5; print(a)
a *= 5; print(a)
a /= 5; print(a)
a //= 5; print(a)
a %= 5; print(a)
a **= 5; print(a)
```

출력 결과

```
15 10 50 10.0 2.0 2.0 32.0
```

연산자 - 산술 연산자(6)

- **동전 교환 프로그램 완성**

소스코드 4-1
(파일명 : 04-01.py)

```
## 변수 선언 부분
money, c500, c100, c50, c10 = 0,0,0,0,0

## 메인(main) 코드 부분
money = int(input("교환할 돈은 얼마 ? "))

c500 = money // 500
money %= 500

c100 = money // 100
money %= 100

c50 = money // 50
money %= 50

c10 = money // 10
money %= 10

```

연산자 - 산술 연산자(7)

```
19  print("\n 오백원짜리 ==> %d 개 " % c500)
20  print(" 백원짜리   ==> %d 개 "% c100)
21  print(" 오십원짜리 ==> %d 개 "% c50)
22  print(" 십원짜리   ==> %d 개 "% c10)
23  print(" 바꾸지 못한 잔돈 ==> %d 원 \n"% money)
```

- 2행 : 교환할 돈(money)과 500원, 100원, 50원, 10원짜리 동전 개수를 저장할 변수 초기화
- 7행 : 입력한 값을 500으로 나눈 몫, 즉 500원짜리 동전의 개수를 구하고, 8행에서 다시 money에 500으로 나눈 후의 나머지 값을 저장.

  결국 입력한 money의 값을 최대한 많이 500원짜리로 바꾸고 500원 미만의 나머지 돈을 다시 money에 저장함

연산자 - 관계 연산자 (1)

- **관계 연산자**
  - 어떤 것이 큰지, 작은지, 같은지를 비교하는 것, 결과는 참(True)이나 거짓(False)
  - 주로 조건문(if)이나 반복문(for, while)에서 사용

그림 4-3
관계 연산자의 기본 개념

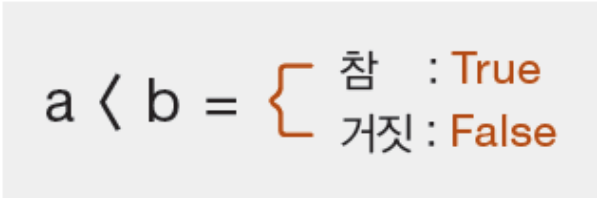


표 4-3
관계 연산자의 종류

관계 연산자	의미	설명
==	같다	두 값이 동일하면 참
!=	같지 않다	두 값이 다르면 참
〉	크다	왼쪽이 크면 참
〈	작다	왼쪽이 작으면 참
〉=	크거나 같다	왼쪽이 크거나 같으면 참
〈=	작거나 같다	왼쪽이 작거나 같으면 참

연산자 - 관계 연산자 (2)

- 관계 연산자 예

```
a,b = 100,200
print(a == b, a != b, a > b, a < b, a >= b, a <= b)
```

출력 결과

```
False True False True False True
```

- a와 b를 비교하기 위한 관계 연산자 ==를 사용시 =을 하나만 쓰는 경우 → 오류발생

  a =b는 b의 값을 a에 대입하라는 의미이지 관계 연산자가 아님

```
print(a = b)
```

연산자 - 논리 연산자 (1)

■ 논리 연산자

표 4-4
논리 연산자의 종류

논리 연산자	의미	설명	사용 예
and	~이고, 그리고(AND)	둘 다 참이어야 참	(a 〉 100) and (a 〈 200)
or	~이거나, 또는(OR)	둘 중 하나만 참이어도 참	(a = = 100) or (a = = 200)
not	~아니다, 부정(NOT)	참이면 거짓, 거짓이면 참	not(a 〈 100)

표 4-5
진리표

입력값		A 그리고 B	A 또는 B	A가 아니다
A	B	(A and B)	(A or B)	(!A)
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

연산자 - 논리 연산자 (2)

- 논리 연산자 예

```
a = 99
(a > 100) and (a < 200)
(a > 100) or (a < 200)
not(a == 100)
```

출력 결과

```
False True True
```

연산자 - 연산자 우선순위(1)

표 4-7 연산자 우선순위

우선순위	연산자	설명
1	( ) [ ] { }	괄호, 리스트, 딕셔너리, 세트 등
2	**	지수
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

Thank You