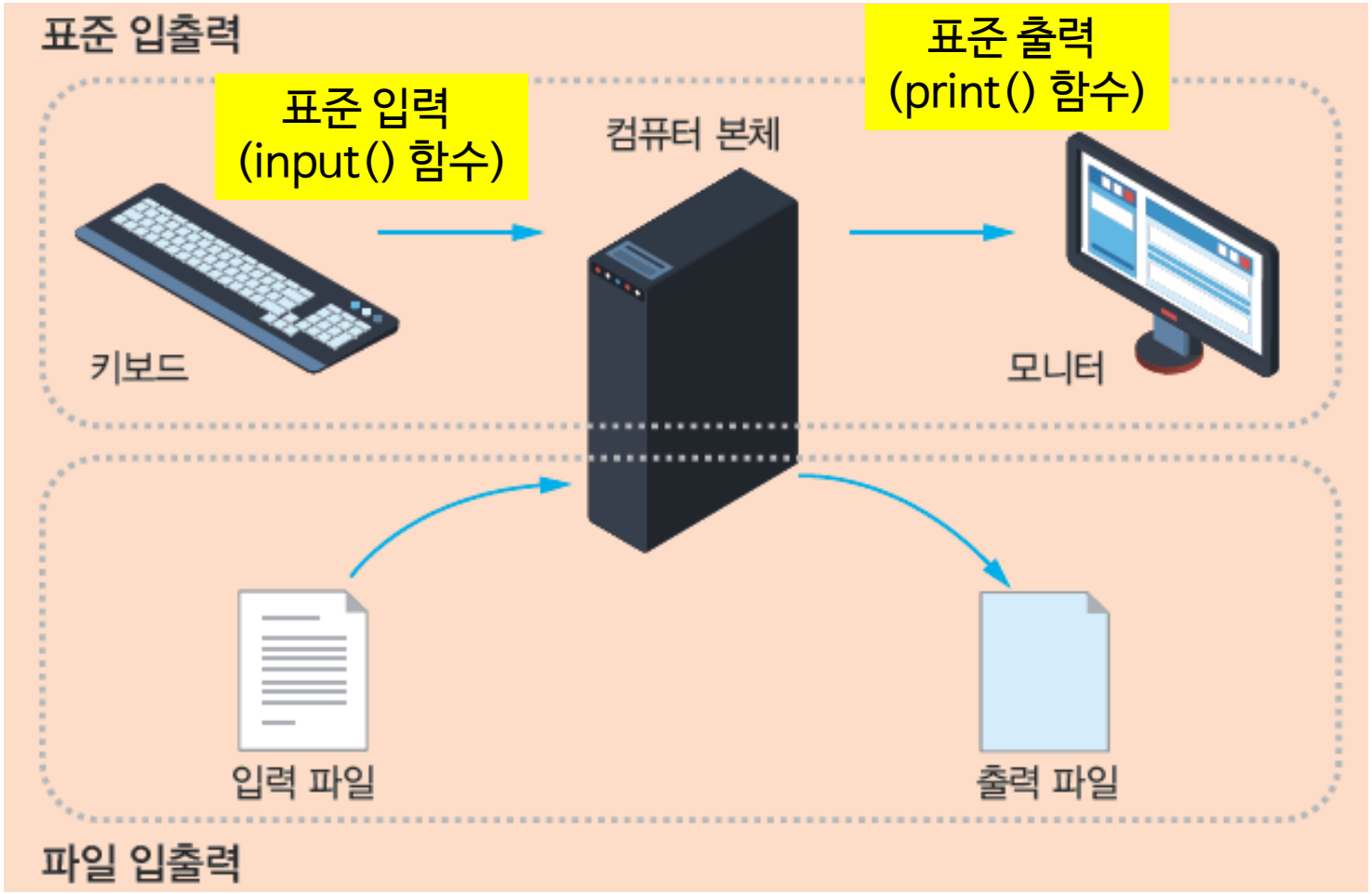
5. 표준 입출력문

1. print() 함수를 이용하여 표준 출력하기
2. print() 함수에 % 기호를 이용하여 포맷팅하기
3. 문자열 format() 메소드를 이용하여 출력하기
4. input() 함수를 이용하여 입력하기
5. 정리

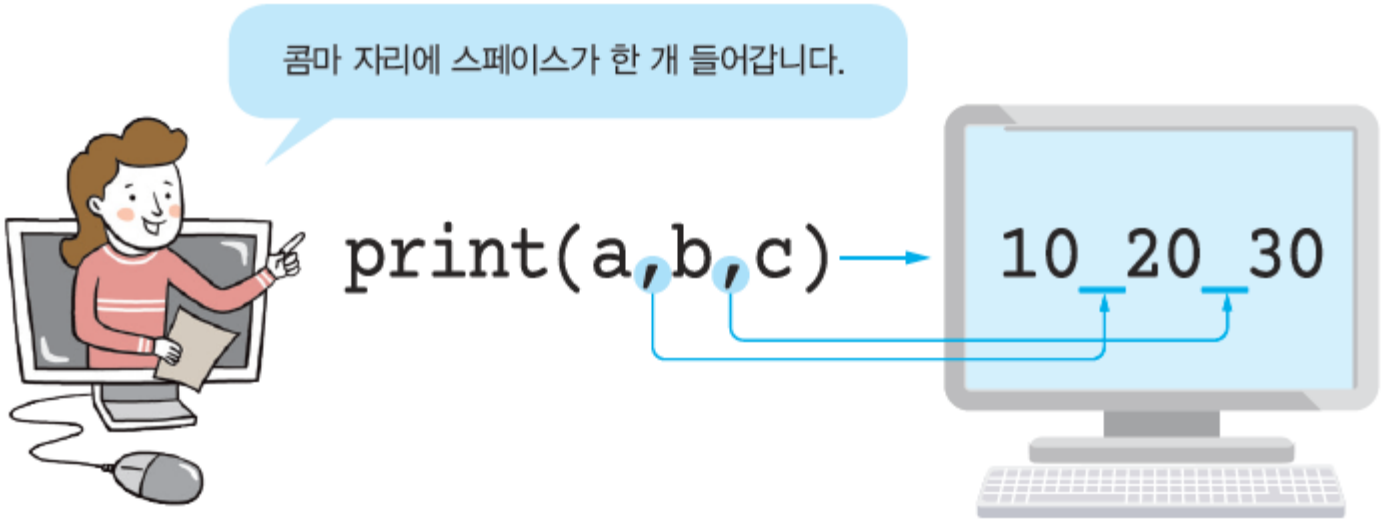
표준 입출력과 파일 입출력



1. print() 함수를 이용하여 표준 출력하기

◆ 정수 출력하기

- print() 함수에 변수를 넣으면 그 변수에 저장된 값이 출력됨.
- 여러 변수의 값들을 출력하고 싶다면 콤마로 분리해서 적는데, 이 때 콤마 자리에 스페이스가 하나 들어감.



1. print() 함수를 이용하여 표준 출력하기

◆ 정수 출력하기

```
>>> print(50, 100)          # print( ) 함수의 괄호에 데이터를 그냥 써도 됨.
50 100
>>> x = 5
>>> print(x, 10)            # 변수와 숫자를 섞어서 써도 됨.
5 10
>>> a = 5; b = 10
>>> print(a+b)              # a와 b를 더한 결과를 출력함.
15
>>> print(a+100)            # a에 100을 더한 결과를 출력함.
105
>>> print(b**a)             # b^a 결과를 출력함.
100000
```

1. print() 함수를 이용하여 표준 출력하기

◆ 실수 출력하기

```
>>> a = 3.5; b = 7.0; c = 1.235
>>> print(a, b, c)
3.5 7.0 1.235
```

스페이스 있음

◆ 복소수 출력하기

```
>>> x = 4+5j; y = 10J; z = 7+1j
>>> print(x, y, z)
(4+5j) 10j (7+1j)
```

1. print() 함수를 이용하여 표준 출력하기

◆ 부울 출력하기

```
>>> t = True; s = False
>>> print(t, s)
True False
```

```
>>> print(True)
True
>>> print(False)
False
```

◆ 리스트 출력하기

```
>>> L = [80, 'python', 90, 3.5, True]      # 다양한 데이터를 저장한 리스트.
>>> print(L)                  # 리스트를 그대로 출력함.
[80, 'python', 90, 3.5, True]
```

1. print() 함수를 이용하여 표준 출력하기

◆ 튜플 출력하기

```
>>> T = ('Alice', 'David', 'Paul')      # 문자열로 구성된 튜플임.
>>> print(T)                            # 튜플을 통째로 그대로 출력함.
('Alice', 'David', 'Paul')
```

◆ 집합 출력하기

```
>>> S = {4, 9, 10, 2, 3, 4, 7, 9, 4, 2, 5}
>>> print(S)     # 중복 데이터는 한 번만 출력되고 저장된 순서대로 출력되지 않을 수 있음.
{2, 3, 4, 5, 7, 9, 10}
```

1. print() 함수를 이용하여 표준 출력하기

◆ 사전 출력하기

```
>>> D = {'name':'Alice', 'age':10, 'grade':3}
>>> print(D)
{'name': 'Alice', 'age': 10, 'grade': 3}
```

◆ 문자열 출력하기

- 따옴표 안에 있는 문자열을 그대로 출력함.

```
>>> print('hello world')
hello world
>>> print("hello world")
hello world
```

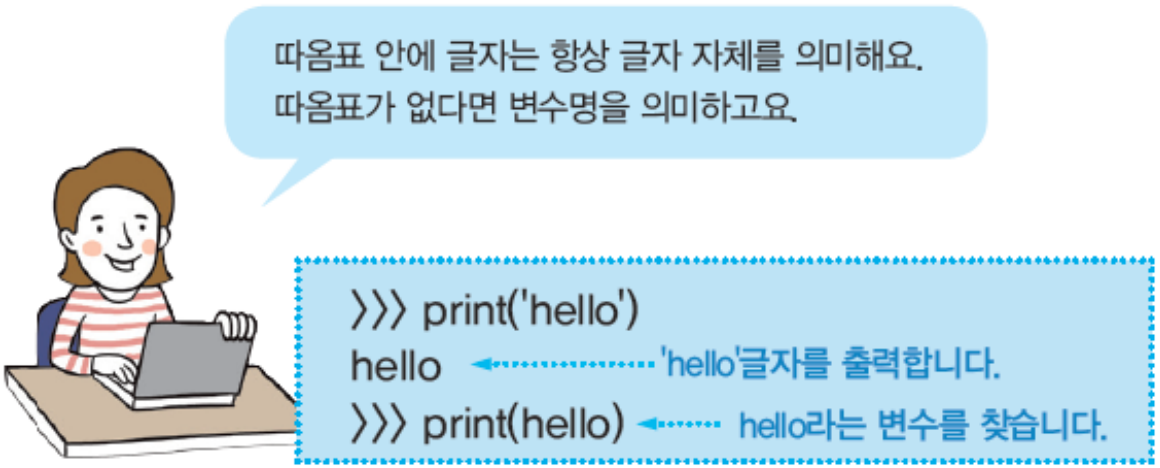
```
>>> print('''hello world''')
hello world
>>> print("""hello world""")
hello world
```

1. print() 함수를 이용하여 표준 출력하기

◆ 문자열 출력하기

- 따옴표가 있으면 문자열이고, 없으면 변수임.

```
>>> x = 10
>>> print(x, 'x')          # 변수 x의 값과 문자열 'x'를 출력함.
10 x
```



1. print() 함수를 이용하여 표준 출력하기

◆ 문자열 출력하기

- 문자열 연결 기호 '+' 사용하기

```
>>> print('hello' + 'world')    # 'hello'다음에 바로 'world'를 연결함.
helloworld
>>> print('hello', 'world')      # 콤마 자리에 스페이스가 하나 출력됨.
hello world
>>> print('hello' + ' ' + 'world')   # 'hello'다음에 ' ', 그리고 'world'를 연결하여 출력함.
hello world
```

1. print() 함수를 이용하여 표준 출력하기

◆ 문자열 출력하기

- 문자열과 숫자를 '+'로 연결할 때는 에러가 발생함.

```
>>> print('hello' + 100)                # 문자열 'hello'에 정수 100을 연결하려고 함.
Traceback (most recent call last):
  File "<pyshell#42>", line 1, in <module>
    print('hello' + 100)
TypeError: must be str, not int         # '문자열 +' 다음에는 문자열(str)이 와야 함.
```

```
>>> print(100 + 'hello')               # 정수 100에 문자열 'hello'를 더하려고 함.
Traceback (most recent call last):
  File "<pyshell#21>", line 1, in <module>
    print(100 + 'hello')
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

1. print() 함수를 이용하여 표준 출력하기

◆ IDLE 프롬프트에서 간단히 출력하기

- IDLE에서는 print() 함수를 사용하지 않아도 변수명만 넣으면 변수 값이 출력됨.
- 이 때 문자열 데이터는 따옴표를 같이 출력함.

```
>>> m = 5              # m은 정수형 변수.
>>> n = '5'            # n은 문자열 변수.
```

```
>>> print(m)
5
>>> print(n)      # print( ) 함수를 사용하면 n이 정수인지 문자열인지 알 수가 없음.
5
```

```
>>> m                 # m은 정수로 저장되어 있기 때문에 그냥 숫자 5가 출력됨.
5
>>> n                 # n은 숫자 5를 문자열로 저장하기 때문에 따옴표를 붙여서 출력함.
'5'
```

1. print() 함수를 이용하여 표준 출력하기

◆ 특수 문자 출력하기

```
>>> print('hello \
world')
hello world
>>> print('hello \'world\'')
hello 'world'
>>> print("hello \"world\"")
hello "world"
```

다음 줄에 계속 이어짐.

```
>>> print('hello \\ world')
hello \ world
>>> print('hello \n world')
hello
 world
>>> print('hello \t world')
hello	world
```

\를 출력하고 싶을 때

[Enter]

탭키

1. print() 함수를 이용하여 표준 출력하기

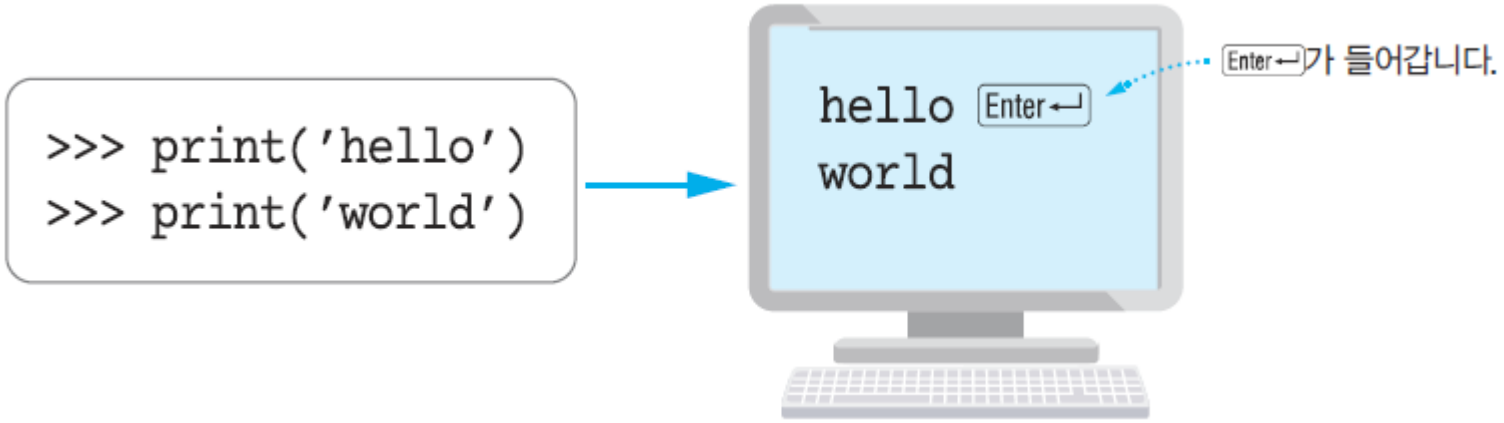
◆ print() 함수 내에 end, sep 매개 변수 사용하기

코드

```
print('hello') # 'hello'출력하고 Enter↵키를 누르는 효과가 있음.
print('world') # 'world'출력하고 Enter↵키를 누르는 효과가 있음.
print('Let\'s learn python.')
```

결과

```
hello
world
Let's learn python.
```



1. print() 함수를 이용하여 표준 출력하기

- print() 함수 내에 end, sep 매개 변수 사용하기
  - 'end'를 이용하면 print() 함수에 기본으로 되어 있는 'Enter'를 없앨 수 있음.

코드

```
print('hello', end=' *** ')
print('world')
```

[Enter] 대신에 ***를 출력함.

end= 옆에 넣는 문자가 [Enter]를 대신함.

결과

```
hello *** world
```

코드

```
print('hello', end='#$%*')
print('world', end='@!')
print('python programming')
```

결과

```
hello#$%*world@!python programming
```

1. print() 함수를 이용하여 표준 출력하기

- print() 함수 내에 end, sep 매개 변수 사용하기
  - 'sep'를 이용하면 print() 함수에 콤마 자리에 스페이스 대신에 다른 문자를 넣을 수 있음.

코드

```
a = 10
b = 20
c = 30
print(a, b, c)
print(a, b, c, sep = '@@')   # 콤마 자리에 '@@'를 넣음.
```

sep= 옆에 넣는 문자가 콤마 자리에 오는 스페이스를 대신함.

결과

```
10 20 30
10@@20@@30
```

1. print() 함수를 이용하여 표준 출력하기

- print() 함수 내에 end, sep 매개 변수 사용하기

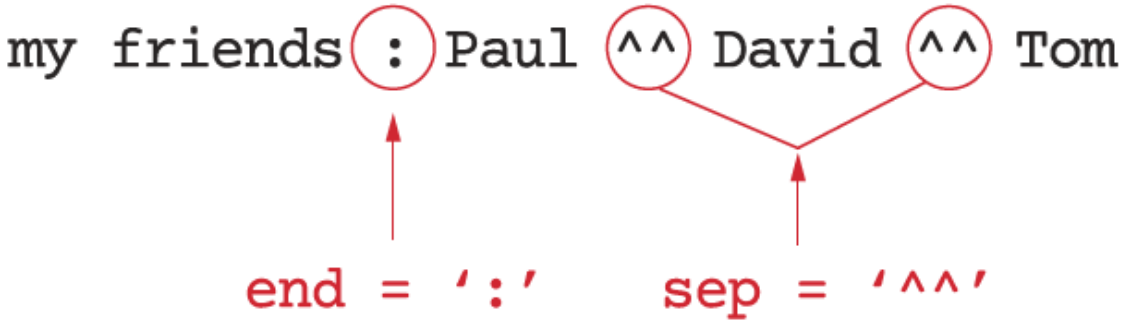
코드

```
friend1 = 'Paul' ; friend2 = 'Cindy' ; friend3 = 'Tom'

print('my friends ', end=':')
print(friend1, friend2, friend3, sep=' ^^ ')
```

결과

```
my friends :Paul ^^ Cindy ^^ Tom
```



2. print() 함수에 % 기호를 이용하여 포맷팅하기

◆ 포맷팅 (formatting)

name	score	goods	price
alice	90	eraser	$ 1.50
Paul	100	cup	$ 5.25
Jennifer	77	staple	$ 9.99
Jimmy	88	folder	$ 2.10

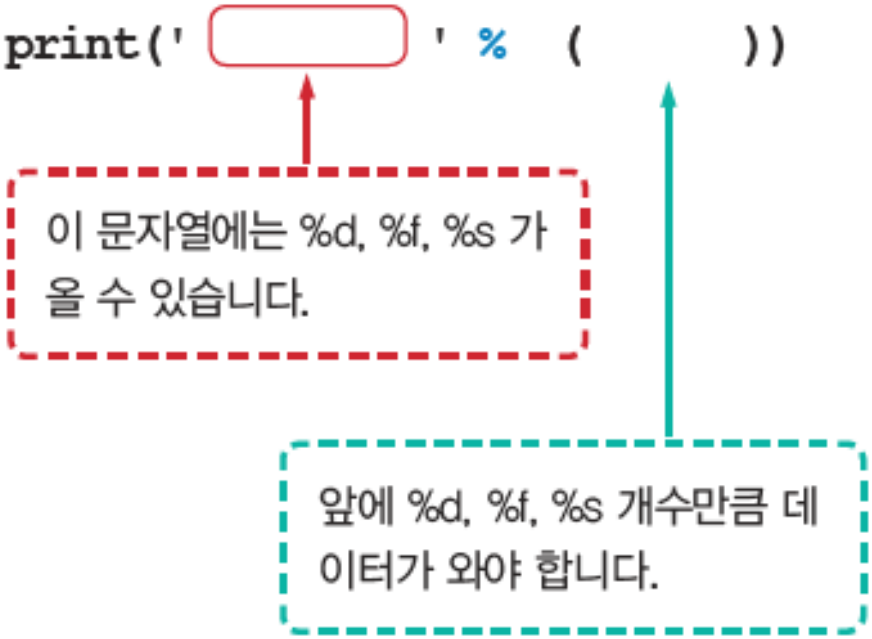
① % 기호 이용하기

② 문자열 자료형의 format() 메소드 이용하기

③ format() 함수 이용하기

2. print() 함수에 % 기호를 이용하여 포맷팅하기

- print() 함수 안에 '…' % ( ) 형태 이용함



%d 정수

%f 실수

%s 문자열

2. print() 함수에 % 기호를 이용하여 포맷팅하기

```
>>> a = 345; b = 5.3678; c ='apple'
>>> print('%d and %f and %s' % (a, b, c))
345 and 5.367800 and apple
>>> print('%s, %f, %d' % (c, b, a))
apple, 5.367800, 345
```

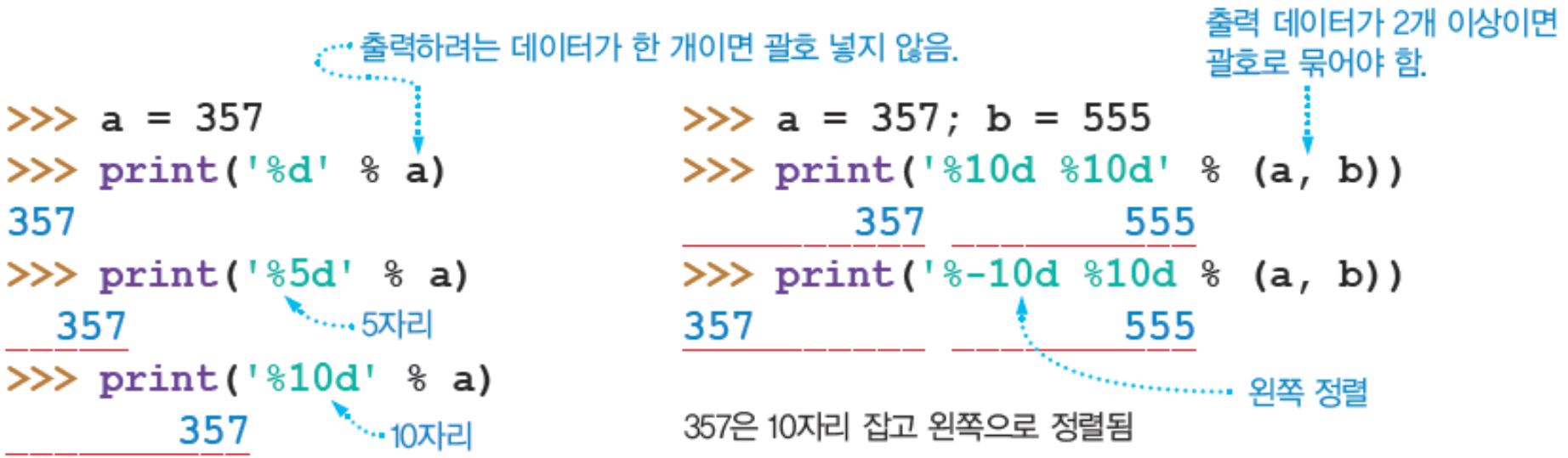
%f는 소수점 아래 여섯자리까지 출력함.

```
>>> a = 345;b = 5.3678; c = 'apple'
>>> print('%d and %f and %s' %(a,b,c)
345 and 5.367800 and apple
>>> print('%s,%f,%d' %(c,b,a))
apple, 5.367800, 345
```

```
print('%d and %f and %s' % (a, b, c))
```

2. print() 함수에 % 기호를 이용하여 포맷팅하기

- %d를 이용하여 정수 포맷팅하기



2. print() 함수에 % 기호를 이용하여 포맷팅하기

- %f를 이용하여 정수 포맷팅하기

```
>>> x = 35.756; y = 234.63452
>>> print('x = %10.5f and y = %10.3f' % (x, y))
x =   35.75600 and y =    234.635
>>> print('x = %10.2f and y = %10.2f' % (x, y))
x =       35.76 and y =       234.63
>>> print('x = %-10.3f and y = %-10.1f' % (x, y))
x = 35.756      and y = 234.6
>>> print('x = %5.3f and y = %0.2f' % (x, y))
x = 35.756 and y = 234.63
```

35.756 출력하려면 최소 6자리 필요함.
5보다 큰 자리가 필요하므로 5는 무시함.

2. print() 함수에 % 기호를 이용하여 포맷팅하기

- %s를 이용하여 정수 포맷팅하기

```
>>> title = 'nudge'
>>> author = 'R. H. Tahler'
>>> print('The book %s is written by %s.' % (title, author))
The book nudge is written by R. H. Tahler.
>>> print('The book %10s is written by %20s.' % (title, author))
The book _____nudge is written by _______R. H. Tahler.
>>> print('The book %-10s is written by %20s.' % (title, author))
The book nudge_____ is written by _______R. H. Tahler.
```

3. 문자열 format() 메소드를 이용하여 출력하기

◆ format() 메소드에 정수 포맷 넣기

```
>>> x = 'I got { } score in the test'
>>> x.format(90)
'I got 90 score in the test'
```

```
'I got { } 90 score in the test'.format(90)
```

```
>>> x = 'I got {:5d} score in the test'
>>> x.format(90)
'I got    90 score in the test'
```

```
'I got {:5d} 90 score in the test'.format(90)
```

```
'I got    90 score in the test'
```

```
>>> x = 'I got {:10d} score in the test'
>>> x.format(90)
'I got         90 score in the test'
```

```
'I got {:10d} 90 score in the test'.format(90)
```

```
'I got         90 score in the test'
```

3. 문자열 format() 메소드를 이용하여 출력하기

◆ format() 메소드에 정수 포맷 넣기

<	왼쪽 정렬
>	오른쪽 정렬
^	가운데 정렬
=	숫자의 부호(+,-)를 가장 왼쪽에 정렬

```
>>> x = 'I got {:>10d} score in the test'
>>> x.format(90)
'I got         90 score in the test'
```

'I got __________90 score in the test'

```
>>> x = 'I got {:<10d} score in the test'
>>> x.format(90)
'I got 90         score in the test'
```

'I got 90__________ score in the test'

3. 문자열 format() 메소드를 이용하여 출력하기

- format() 메소드에 정수 포맷 넣기

>>> x = 'I got {:^10d} score in the test'
>>> x.format(90)
'I got 90 score in the test'
	'I got ____90____ score in the test'

>>> x = 'I got {:=10d} score in the test'
>>> x.format(-90)
'I got - 90 score in the test'
	'I got -___90____ score in the test'

3. 문자열 format() 메소드를 이용하여 출력하기

- format() 메소드에 실수 포맷 넣기

>>> x = 'I got {:10.3f} score in the test'
>>> x.format(85.78294)
'I got 85.783 score in the test'
	'I got 85.783 score in the test'

>>> x = 'I got {:>10.3f} score in the test'
>>> x.format(85.78294)
'I got 85.783 score in the test'
	'I got 85.783 score in the test'

>>> x = 'I got {:<10.3f} score in the test'
>>> x.format(85.78294)
'I got 85.783 score in the test'
	'I got 85.783 score in the test'

>>> x = 'I got {:^10.3f} score in the test'
>>> x.format(85.78294)
'I got 85.783 score in the test'
	'I got 85.783 score in the test'

>>> x = 'I got {:=10.3f} score in the test'
>>> x.format(-85.78294)
'I got - 85.783 score in the test'
	'I got - 85.783 score in the test'

3. 문자열 format() 메소드를 이용하여 출력하기

- format() 메소드에 문자열 포맷 넣기

>>> x = 'I have a { } pen'
>>> x.format('green')
'I have a green pen'
	'I have a green pen'

>>> x = 'I have a {:10} pen'
>>> x.format('green')
'I have a green pen'
	'I have a green_____ pen'

>>> x = 'I have a {:>10} pen'
>>> x.format('blue')
'I have a blue pen'
	'I have a _____blue pen'

>>> x = 'I have a {:<10} pen'
>>> x.format('red')
'I have a red pen'
	'I have a red_____ pen'

>>> x = 'I have a {:^10} pen'
>>> x.format('red')
'I have a red pen'
	'I have a ___red____ pen'

3. 문자열 format() 메소드를 이용하여 출력하기

- format() 메소드에 문자열 포맷 넣기

```
>>> x = 'I have a {:*<10} pen'
>>> x.format('red')
'I have a red******* pen'
```
빈 공간을 '*'으로 채움 → 'I have a red******* pen'

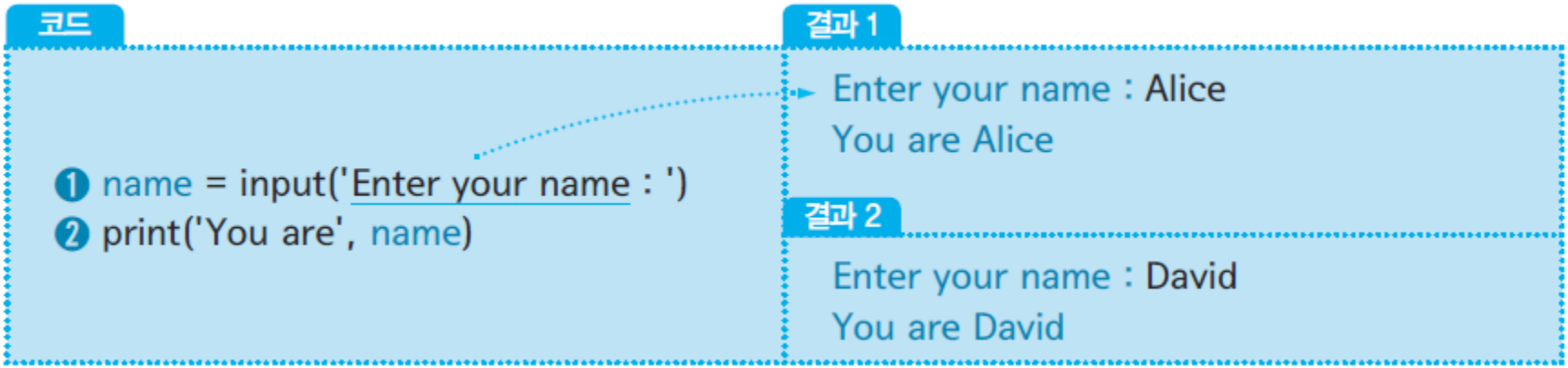
```
>>> x = 'I have a {:*^10} pen'
>>> x.format('red')
'I have a ***red**** pen'
```
→ 'I have a ***red**** pen'

```
>>> x = 'I have a {:?>10} pen'
>>> x.format('blue')
'I have a ??????blue pen'
```
빈 공간을 '?'으로 채움 → 'I have a ??????blue pen'

4. input() 함수를 이용하여 입력하기

◆ 키보드로부터 입력받기

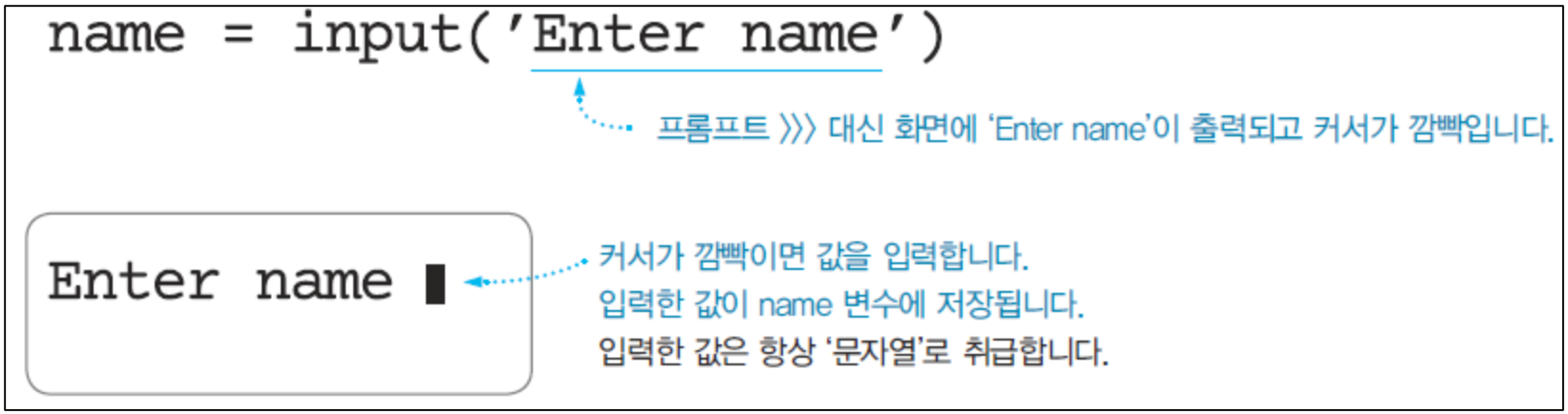
- input() 함수를 이용하면 사용자가 데이터를 입력하여 메모리 변수에 저장할 수 있음.



4. input() 함수를 이용하여 입력하기

◆ 키보드로부터 입력받기

- 입력받은 데이터는 항상 '문자열'로 취급함.



4. input() 함수를 이용하여 입력하기

◆ 키보드로부터 입력받기

>>> name = input('Enter name : ')
Enter name : Alice
>>> print(name)
Alice
>>> name
'Alice' ← name은 문자열 자료형
	>>> number = input('Enter number : ')
	Enter number : 5
	>>> print(number)
	5
	>>> number
	'5' ← number는 문자열 자료형

>>> value = input('Enter value : ')
Enter value : 35.78
>>> print(value)
35.78
>>> value
'35.78' ← value는 문자열 자료형
	>>> boolValue = input('Enter bool value : ')
	Enter bool value : True
	>>> print(boolValue)
	True
	>>> boolValue
	'True' ← boolValue는 문자열 자료형

4. input() 함수를 이용하여 입력하기

◆ int(), float() 함수 이용하기

```
>>> number = input('Enter number : ')
Enter number : 5
>>> type(number)
<class 'str'>
>>> number
'5'
>>> number + 10     # 문자열 + 정수이기 때문에 TypeError가 발생함.
Traceback (most recent call last):
  File "<pyshell#112>", line 1, in <module>
    number + 10
TypeError: must be str, not int
>>> int(number) + 10
15
```

4. input() 함수를 이용하여 입력하기

- int(), float() 함수 이용하기

```
>>> number = input('Enter number : ')
Enter number : 5
>>> number = int(number)
>>> number
5
```

한 줄로 표현할 수 있다

```
>>> number = int(input('Enter number : '))
Enter number : 5
>>> number
5
```

4. input() 함수를 이용하여 입력하기

- int(), float() 함수 이용하기

```
>>> number = float(input('Enter number : '))
Enter number : 5.5
>>> type(number)
<class 'float'>
```

- input() 함수 괄호에 아무 것도 넣지 않기

코드

```
sentence = input()
print(sentence)
```

결과

```
Hello world  <-- 사용자가 입력한 거예요.
Hello world  <-- 컴퓨터가 출력한 거예요.
```

5. 정리

- print() 내장 함수 사용법을 자세히 학습하였음.
- print() 함수에 sep, end 매개 변수 사용하기
- print()함수를 이용해서 포맷팅하기
- 문자열 메소드 중에서 format() 메소드를 이용하여 포맷 출력하기
- input() 함수를 이용하여 키보드로부터 입력을 받아들이기