Input and Output
Console I/O

Contents

- 입력 함수 : input()
- 출력함수 : print()
- 입출력 함수 실습

input( ), print( ) 함수

```
name = input("이름이 무엇인가요? ")
print("만나서 반갑습니다. " + name +  " 님!")
```

```
이름이 무엇인가요? 포닉스
만나서 반갑습니다. 포닉스님!
```

input( ), print( ) 함수

```
x = input("첫 번째 정수: ")
y = input("두 번째 정수: ")
sum = x + y
print("합은 ", sum)
```

```
첫 번째 정수: 10
두 번째 정수: 20
합은  1020
```

문자열로 간주하여 두 문자를 합침

input( ), print( ) 함수

데이터 타입을 int로 변환하여 전달

```
x = int(input("첫 번째 정수: "))
y = int(input("두 번째 정수: "))
sum = x + y
print("합은 ", sum)
```

```
첫 번째 정수: 10
두 번째 정수: 20
합은  30
```

정수로 변환하여 정상적 결과 출력

입출력 함수 실습

```
File  Edit  Format  Run  Options  Window  Help
a=int(input("첫번째 숫자 입력:"))
b=int(input("두번째 숫자 입력:"))
result=a+b
print(a , "+" , b , "=" , result)
result=a-b
print(a , "-" , b , "=" , result)
result=a*b
print(a , "*" , b , "=" , result)
result=a/b
print(a , "/" , b , "=" , result)
```

```
File  Edit  Shell  Debug  Options
Python 3.5.2 (v3.5.2:4de
Type "copyright", "credi
>>>
==========================
첫번째 숫자 입력:10
두번째 숫자 입력:20
10 + 20 = 30
10 - 20 = -10
10 * 20 = 200
10 / 20 = 0.5
>>>
```

print( ) 함수

- 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

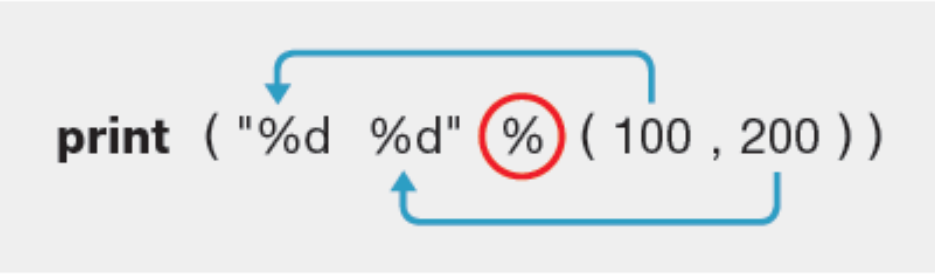
print( ) 함수

- 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 번째 행에는 %d가 두 개인데, 숫자는 하나(100)밖에 나오지 않음



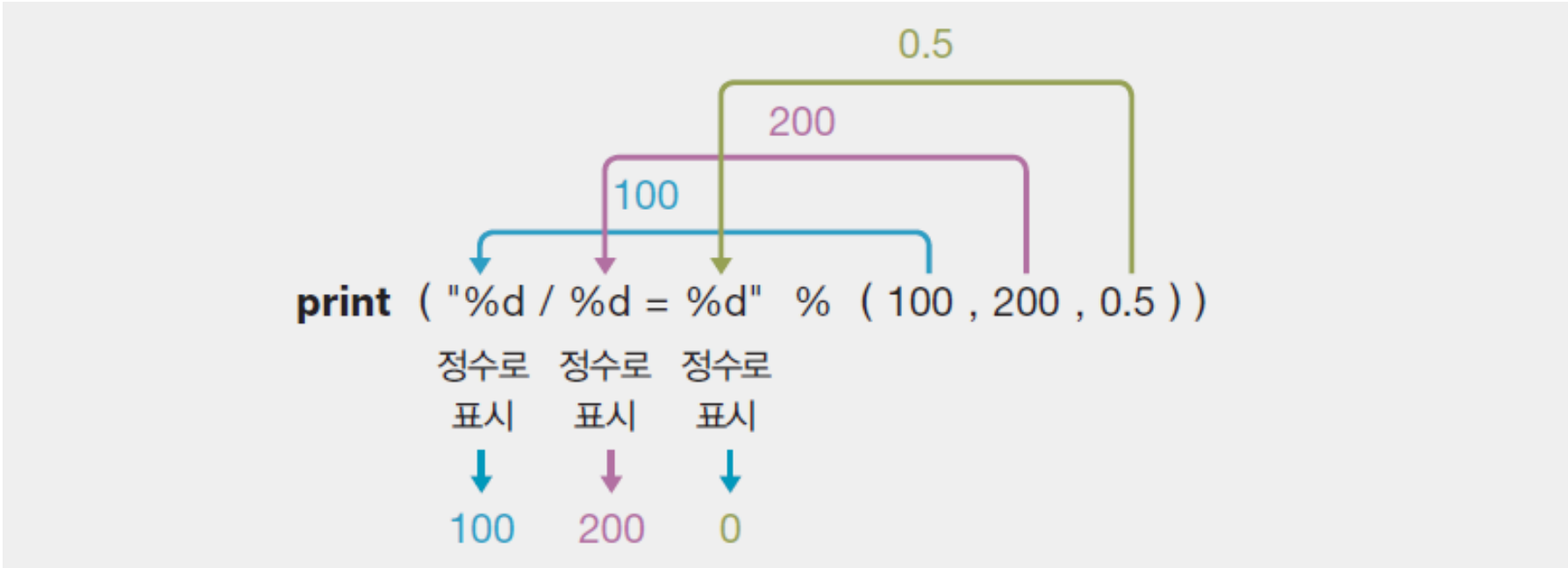
print( ) 함수

- 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



print( ) 함수

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 지정

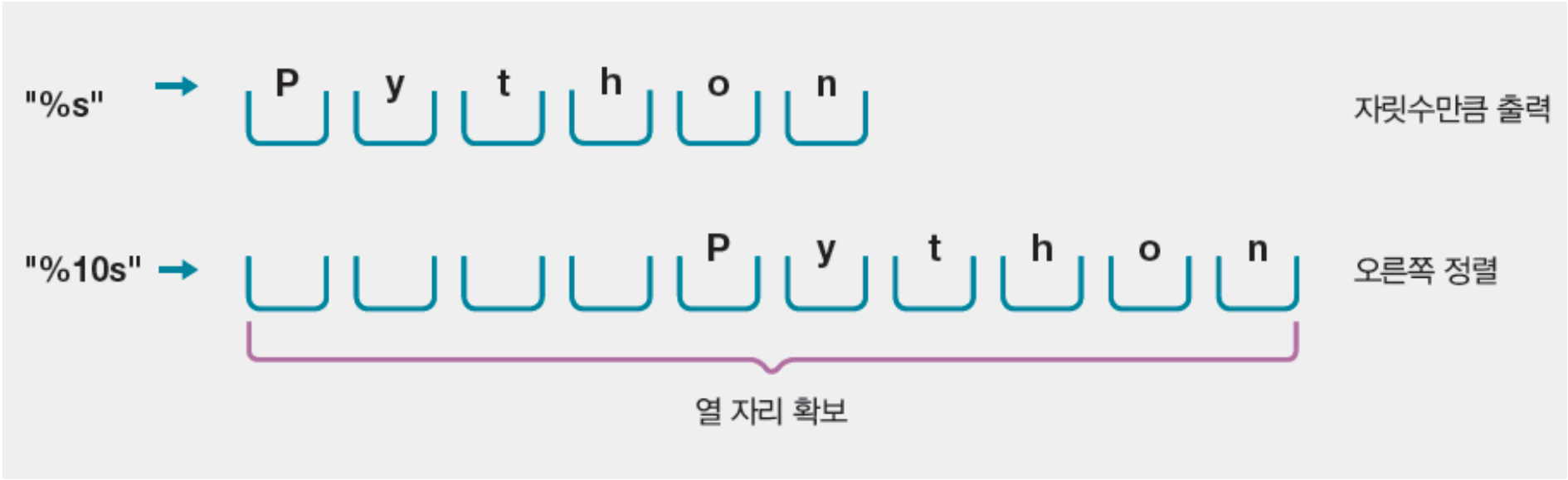
- 정수형 데이터 서식 지정

print( )의 서식 지정

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

print( )의 서식 지정

- 문자열 형 데이터 서식 지정



print( )의 서식 지정

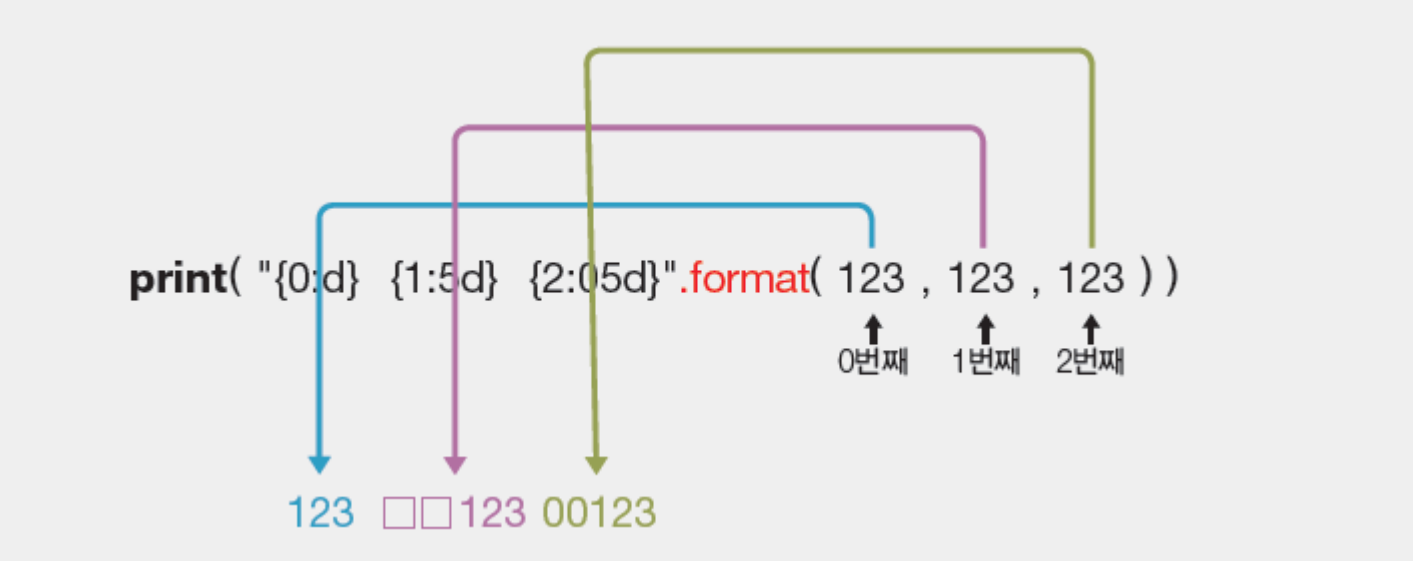
- **format( ) 함수의 사용**

```
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.



print( )의 서식 지정

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

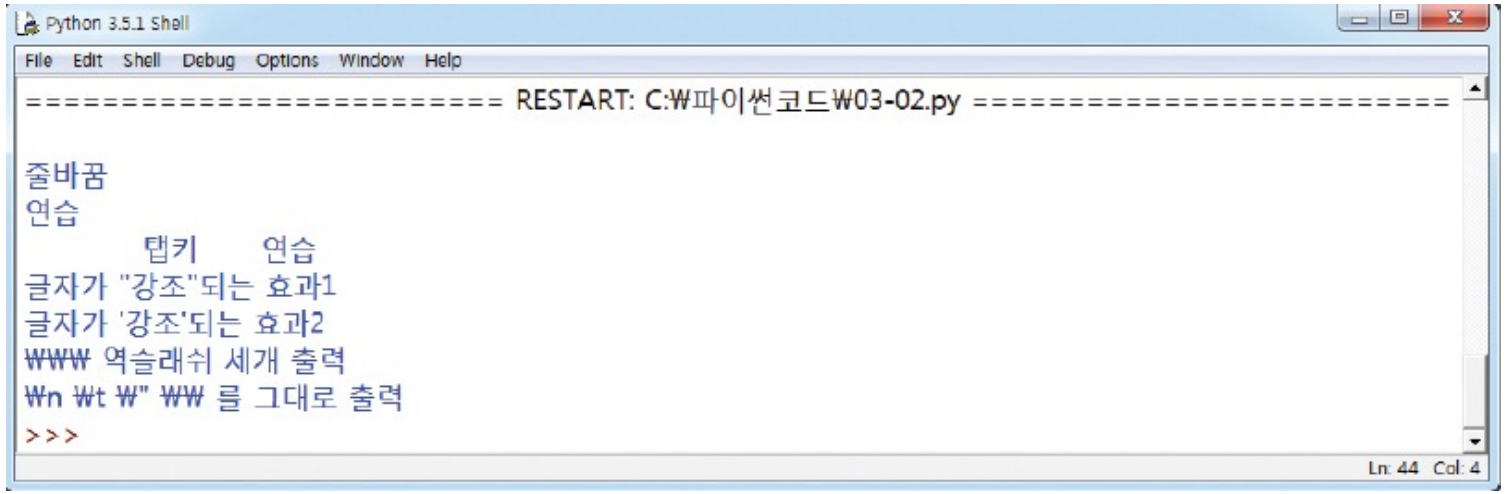
```
print("한행입니다. 또 한행입니다")
print("한행입니다. \n또 한행입니다")
```

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 지정

- 이스케이프 문자 활용

```
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")
```



리스트란?

- 리스트(list)는 여러 개의 데이터가 저장되어 있는 장소

```
리스트이름 = [값1, 값2, 값3]
```

```
scores  = [ 32, 56, 64, 72, 12, 37, 98, 77, 59, 69]
```

- 리스트는 여러 개의 데이터를 하나의 이름으로 관리할 수 있는 데이터 구조
- 서로 다른 데이터 타입의 데이터를 하나의 리스트이름으로 관리 가능

리스트

- 여러 정보를 하나로 묶어서 저장하고 관리할 수 있게 하는 데이터 구조
- 선언

```
리스트이름 = [값1, 값2, 값3]
```

  : []안에 쉼표로 구분하여 정보를 적어 주면 됨

- 문자열을 원소로 가지는 예제

```
>>> fruit = ["banana", "apple", "cherry"]
```

- 숫자를 원소로 가지는 예제

```
>>> score = [70, 99, 25, 100]
```

- Empty list

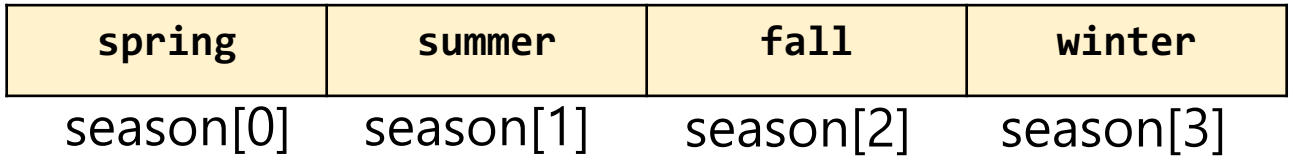
```
>>> empty_list = []
```

리스트 원소에 접근하기

- 인덱스

: 원소가 배열된 순서를 나타냄. (0번부터 시작)

```
>>> season = ['spring', 'summer', 'fall', 'winter']
```



- 인덱스를 이용하여 원소에 접근할 수 있음

```
>>> season[1]
```

‘summer’

리스트와 연산자

- ‘in’ & ‘not in’

in	list의 element 인가를 결정하는 연산자
not in	list의 element가 아닌 element를 결정하는 연산자

```
>>> fruit =  ['banana', 'apple', 'orange']
>>> 'apple' in fruit
True
>>> 'cherry' in fruit
False
```

리스트 순회하기

```
scores = [ 32, 56, 64, 72, 12, 37, 98, 77, 59, 69]

for element  in scores:
      print(element, end=' ')
```

```
32 56 64 72 12 37 98 77 59 69
```

리스트 사용 가능 함수

함수	설명	사용법
append( )	리스트 제일 뒤에 항목을 추가한다.	리스트이름.append(값)
pop( )	리스트 제일 뒤의 항목을 빼내고, 빼낸 항목은 삭제한다.	리스트이름.pop( )
sort( )	리스트의 항목을 정렬한다.	리스트이름.sort( )
reverse( )	리스트 항목의 순서를 역순으로 만든다.	리스트이름.reverse( )
index( )	지정한 값을 찾아서 그 위치를 반환한다.	리스트이름.index(찾을 값)
insert( )	지정된 위치에 값을 삽입한다.	리스트이름.insert(위치, 값)
remove( )	리스트에서 지정한 값을 제거한다. 단 지정한 값이 여러 개일 경우 첫 번째 값만 지운다.	리스트이름.remove(지울 값)
extend( )	리스트 뒤에 리스트를 추가한다. 리스트의 더하기(+) 연산과 동일한 기능을 한다.	리스트이름.extend(리스트)
count( )	리스트에서 찾을 값의 개수를 센다.	리스트이름.count(찾을 값)
del( )	리스트에서 해당 위치의 항목을 삭제한다.	del(리스트이름[위치])
len( )	리스트에 포함된 전체 항목의 개수를 센다.	len(리스트이름)

예제: 성적 처리 프로그램

- 학생들의 성적을 처리하는 프로그램 작성
  - 사용자로부터 성적을 입력 받아서 리스트에 저장
  - 성적의 평균을 구하고 80점 이상 성적을 받은 학생의 숫자를 계산하여 출력

```
성적을 입력하세요: 89
성적을 입력하세요: 89
성적을 입력하세요: 90
성적을 입력하세요: 55
성적을 입력하세요: 77
성적 평균은 80.0 입니다.
80점 이상 성적을 받은 학생은 3 명입니다.
```

성적 처리 프로그램

```
STUDENTS = 5

scores = []
scoreSum = 0

for i in range(STUDENTS):
    value = int(input("성적을 입력하세요: "))
    scores.append(value) #리스트에 값을 추가
    scoreSum += value

scoreAvg = scoreSum / len(scores)
highScoreStudents = 0
for i in range(len(scores)):
    if scores[i] >= 80:
        highScoreStudents += 1

print("성적 평균은", scoreAvg, "입니다.")
print("80점 이상 성적을 받은 학생은", highScoreStudents, "명입니다.")
```

예제: 문자열 처리 프로그램

- 리스트는 문자열도 저장 가능
- 학생들의 이름을 저장하였다가 출력하는 프로그램을 작성

```
학생 이름을 입력하세요: 포닉스
학생 이름을 입력하세요: 넙죽이
학생 이름을 입력하세요: 클로바
학생 이름을 입력하세요: 독수리
학생 이름을 입력하세요:
학생 이름:
포닉스, 넙죽이, 클로바, 독수리,
```

문자열 처리 예제

```
Names = []
while True:
    name = input("학생 이름을 입력하세요: ")
    if name == "" :
        break
    Names.append(name)

print("학생 이름:")
for name in Names:
    print(name, end=", ")
```

```
학생 이름을 입력하세요: 포닉스
학생 이름을 입력하세요: 넙죽이
학생 이름을 입력하세요: 클로바
학생 이름을 입력하세요: 독수리
학생 이름을 입력하세요:
학생 이름:
포닉스, 넙죽이, 클로바, 독수리,
```