

# 스크립트 프로그래밍

## 03 수학함수, 문자열 및 객체

2016 2학기 (02분반)

강승우

# 학습 목표

- math 모듈에 포함된 함수를 사용하여 수학적 문제를 해결할 수 있다(3.2절)
- 문자열과 문자를 표현하고 처리할 수 있다(3.3~3.4절)
  - ASCII와 유니코드를 사용하여 문자를 인코딩할 수 있다(3.3.1~3.3.2절)
  - ord 함수를 사용하여 문자의 숫자 코드를 구하고 chr 함수로 숫자 코드를 문자로 변환할 수 있다(3.3.3절)
  - 이스케이프 시퀀스를 사용하여 특수 문자를 표현할 수 있다(3.3.4절).
  - end 인자와 함께 print 함수를 호출할 수 있다(3.3.5절).
  - str 함수를 사용하여 숫자를 문자열로 변환할 수 있다(3.3.6절).
  - 문자열 연결에 + 연산자를 사용할 수 있다(3.3.7절).
  - 키보드로부터 문자열을 읽을 수 있다(3.3.8절).
- 객체와 메소드의 개념을 설명할 수 있다(3.5절).
- format 함수를 사용하여 숫자와 문자열의 서식을 지정할 수 있다(3.6절).

# 공통 파이썬 함수

- 함수
  - 특정 작업을 수행하는 명령문의 집합
- (함수) 라이브러리
  - 프로그래밍 과정에서 자주 사용되는 함수를 정의하고 기능별로 구분하여 모아놓은 함수 그룹
- 파이썬 내장 함수
  - eval, input, print 등 (앞 장에서 이미 사용)
  - 파이썬 인터프리터에서 언제든지 사용 가능
    - 모듈 import 필요 없음

# 파이썬 내장함수

```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> max(2, 3, 4) # 최대값 반환
4
>>> min(2, 3, 4) # 최소값 반환
2
>>> round(3.51) # 가장 가까운 정수로 근사
4
>>> round(3.4)
3
>>> abs(-3) # 절대값 반환
3
>>> pow(2,3) # 2**3
8
>>> |
```

Ln: 15 Col: 4

# 파이썬 내장함수

```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> round(5.466, 2) # 소수점 이하 셋째 자리를 반올림하여 둘째자리로 표현한 실수 반환
5.47
>>> round(5.463, 2)
5.46
>>> round(5.46537, 4)
5.4654
>>> round(5.46532, 4)
5.4653
>>> |
```

Ln: 11 Col: 4

# 파이썬 수학 함수 – math 모듈

함수	설명	예
<code>fabs(x)</code>	x의 절대값을 실수로 반환한다.	<code>fabs(-2) = 2.0</code>
<code>ceil(x)</code>	x의 가장 가까운 정수로 올림하여 반환한다.	<code>ceil(2.1) = 3</code> <code>ceil(-2.1) = -2</code>
<code>floor(x)</code>	x의 가장 가까운 정수로 버림하여 반환한다.	<code>floor(2.1) = 2</code> <code>floor(-2.1) = -3</code>
<code>exp(x)</code>	x의 지수함수( $e^x$ ) 값을 반환한다.	<code>exp(1) = 2.71828</code>
<code>log(x)</code>	x의 자연로그 값을 반환한다.	<code>log(2.71828) = 1.0</code>
<code>log(x, base)</code>	특정 밑을 갖는 x의 로그 값을 반환한다.	<code>log(100, 10) = 2.0</code>
<code>sqrt(x)</code>	x의 제곱근을 반환한다.	<code>sqrt(4.0) = 2</code>
<code>sin(x)</code>	x의 사인 값을 반환한다. x는 라디안 각도로 표현된다.	<code>sin(3.14159 / 2) = 1</code> <code>sin(3.14159) = 0</code>
<code>asin(x)</code>	사인의 역함수에 대한 라디안 각도를 반환한다.	<code>asin(1.0) = 1.57</code> <code>asin(0.5) = 0.523599</code>
<code>cos(x)</code>	x의 코사인 값을 반환한다. x는 라디안 각도로 표현된다.	<code>cos(3.14159 / 2) = 0</code> <code>cos(3.14159) = -1</code>
<code>acos(x)</code>	코사인 역함수에 대한 라디안 각도를 반환한다.	<code>acos(1.0) = 0</code> <code>acos(0.5) = 1.0472</code>
<code>tan(x)</code>	x의 탄젠트 값을 반환한다. x는 라디안 각도로 표현된다.	<code>tan(3.14159 / 4) = 1</code> <code>tan(0.0) = 0</code>
<code>degrees(x)</code>	라디안 각도 x를 도 단위로 변환한다.	<code>degrees(1.57) = 90</code>
<code>radians(x)</code>	도 x를 라디안 각도 단위로 변환한다.	<code>radians(90) = 1.57</code>

# 파이썬 수학 함수 – math 모듈

## MathFunctions.py

```
import math # 수학 함수를 사용하기 위해 math 모듈을 임포트한다.
```

```
# 수학 함수를 테스트한다.
```

```
print("exp(1.0) =", math.exp(1))
```

```
print("log(3.78) =", math.log(math.e))
```

```
print("log10(10, 10) =", math.log(10, 10))
```

```
print("sqrt(4.0) =", math.sqrt(4.0))
```

```
# 삼각함수를 테스트한다.
```

```
print("sin(PI / 2) =", math.sin(math.pi / 2))
```

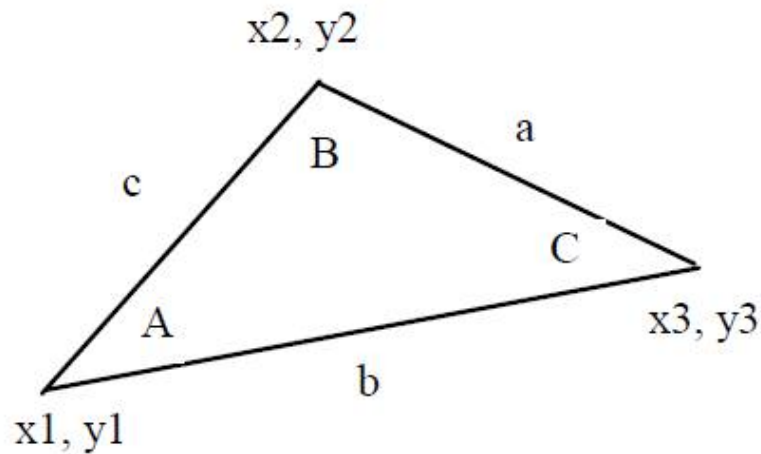
```
print("cos(PI / 2) =", math.cos(math.pi / 2))
```

```
print("tan(PI / 2) =", math.tan(math.pi / 2))
```

```
print("degrees(1.57) =", math.degrees(1.57))
```

```
print("radians(90) =", math.radians(90))
```

# 삼각형의 내각 계산



$$A = \arccos((a * a - b * b - c * c) / (-2 * b * c))$$

$$B = \arccos((b * b - a * a - c * c) / (-2 * a * c))$$

$$C = \arccos((c * c - b * b - a * a) / (-2 * a * b))$$

- 위 공식에 따르면 삼각형의 세 변의 길이를 알면 내각을 계산할 수 있음
- 삼각형의 세 변의 길이,  $a$ ,  $b$ ,  $c$ 는 어떻게 계산할까?



# 삼각형의 내각 계산

## ComputeAngles.py

```
import math

x1, y1, x2, y2, x3, y3 = eval(input("세 점을 입력하세요: "))

a = math.sqrt((x2 - x3) * (x2 - x3) + (y2 - y3) * (y2 - y3))
b = math.sqrt((x1 - x3) * (x1 - x3) + (y1 - y3) * (y1 - y3))
c = math.sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2))

A = math.degrees(math.acos((a * a - b * b - c * c) / (-2 * b * c)))
B = math.degrees(math.acos((b * b - a * a - c * c) / (-2 * a * c)))
C = math.degrees(math.acos((c * c - b * b - a * a) / (-2 * a * b)))

print("세 내각은 ", round(A * 100) / 100.0,
      round(B * 100) / 100.0, round(C * 100) / 100.0, "입니다.")
```

# 문자열 및 문자

- 문자열
  - 연속된 문자를 의미
  - 파이썬에서는 문자에 대한 별도의 데이터 타입이 없음  
→ 한 문자의 문자열이 곧 문자를 나타낸다
- 반드시 작은따옴표(' ')나 큰따옴표(" ")의 쌍으로 둘러 쌓여진다
- 표기 방식
  - 작은따옴표: 한 문자, 공백 문자열
  - 큰따옴표: 두 문자 이상의 문자열

# 문자열 및 문자

```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> letter = 'A'
>>> letter
'A'
>>> letter = "A"
>>> letter
'A'
>>> numChar = '4'
>>> numChar
'4'
>>> numChar = "4"
>>> numChar
'4'
>>> message = "좋은 아침"
>>> message
'좋은 아침'
>>>
```

Ln: 20 Col: 4

# ASCII 코드와 유니코드

- 문자 인코딩 (character encoding)
  - 문자를 2진 표현으로 변환하는 것
- 문자 인코딩 방법
  - 인코딩 체계 (encoding scheme)에 의해 결정
- 예
  - ASCII (American Standard Code for Information Interchange): 모든 대문자, 소문자, 숫자, 구두 기호와 제어 문자를 표현하기 위한 7비트 인코딩 체계 (0부터 127까지 숫자를 사용하여 문자 표현)
  - 유니코드(Unicode): 전 세계 문자를 표현하기 위한 인코딩 체계
    - ASCII는 유니코드의 일부분
    - 전 세계 다양한 언어로 작성된 텍스트의 교환, 처리 및 출력을 지원하기 위해 유니코드 컨소시엄에서 제안
    - \u로 시작, \u0000부터 \uFFFF까지 4자리 16진수로 표현

# ASCII 코드 (10진수)

TABLE B.1 ASCII Character Set in the Decimal Index

	0	1	2	3	4	5	6	7	8	9
0	nul	soh	stx	etx	eot	enq	ack	bel	bs	ht
1	nl	vt	ff	cr	so	si	dle	dcl	dc2	dc3
2	dc4	nak	syn	etb	can	em	sub	esc	fs	gs
3	rs	us	sp	!	"	#	\$	%	&	'
4	(	)	*	+	,	-	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[	\	]	^	_	`	a	b	c
10	d	e	f	g	h	i	j	k	l	m
11	n	o	p	q	r	s	t	u	v	w
12	x	y	z	{		}	~	del		

# ord와 chr 함수

- ord(ch)
    - 문자 ch에 대한 ASCII 코드를 반환하는 함수
  - chr(code)
    - ASCII 코드 code에 해당하는 문자를 반환하는 함수
- 
- 대소문자 변환 프로그램을 작성하려면 어떻게 할 수 있을까?

# ord와 chr 함수



```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help

>>>
>>> ch = 'a'
>>> ord(ch)
97
>>> chr(98)|
'b'
>>> ord('a') - ord('A')
32
>>> ord('d') - ord('D')
32
>>> chr(ord('h') - 32)
'H'
>>> chr(ord('Z') + 32)
'z'
>>>
```

Ln: 39 Col: 11

# 이스케이프 시퀀스

- 문자열에서 특수 문자를 표현하기 위한 표기법
- \로 시작하는 문자, 특수 문자 등
- \b: 백스페이스
- \n: 개행, 행바꿈, end-of-line
- \t: 탭
- \\: 역슬래시
- \': 단일 인용부호
- \": 이중 인용부호
- \r: 같은 라인의 맨 앞의 위치에 커서를 이동



# 줄바꿈 없이 출력하기

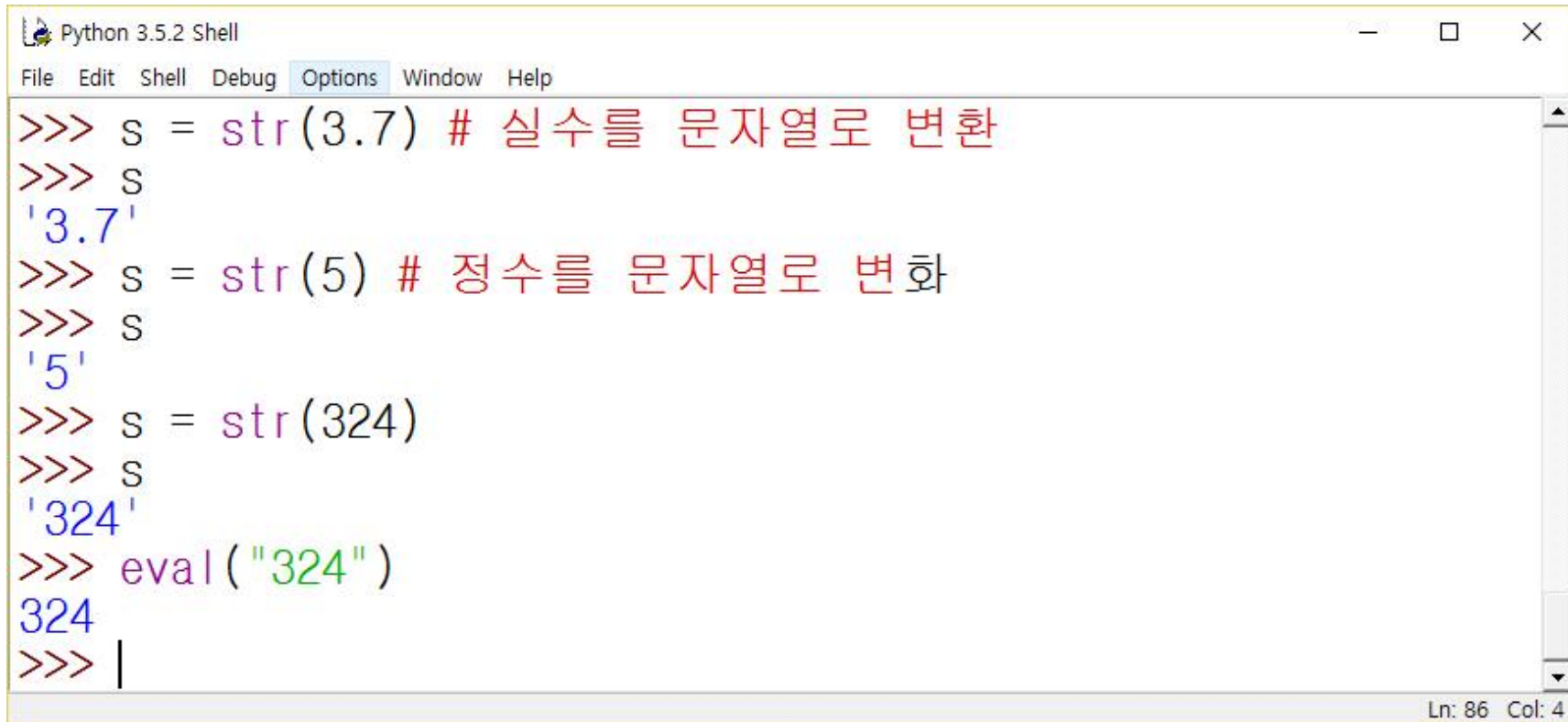
- print 함수
  - 자동으로 줄바꿈(\n) 삽입
- print(string, end="ending")
  - 문자열 string 출력 후 줄바꿈 대신 맨 마지막에 “ending” 문자열 출력

```
print("AAA", end = ' ')\nprint("BBB", end = " ")\nprint("CCC", end = '***')\nprint("DDD", end = '***')
```

AAA BBBCCC\*\*\*DDD\*\*\*

# str 함수

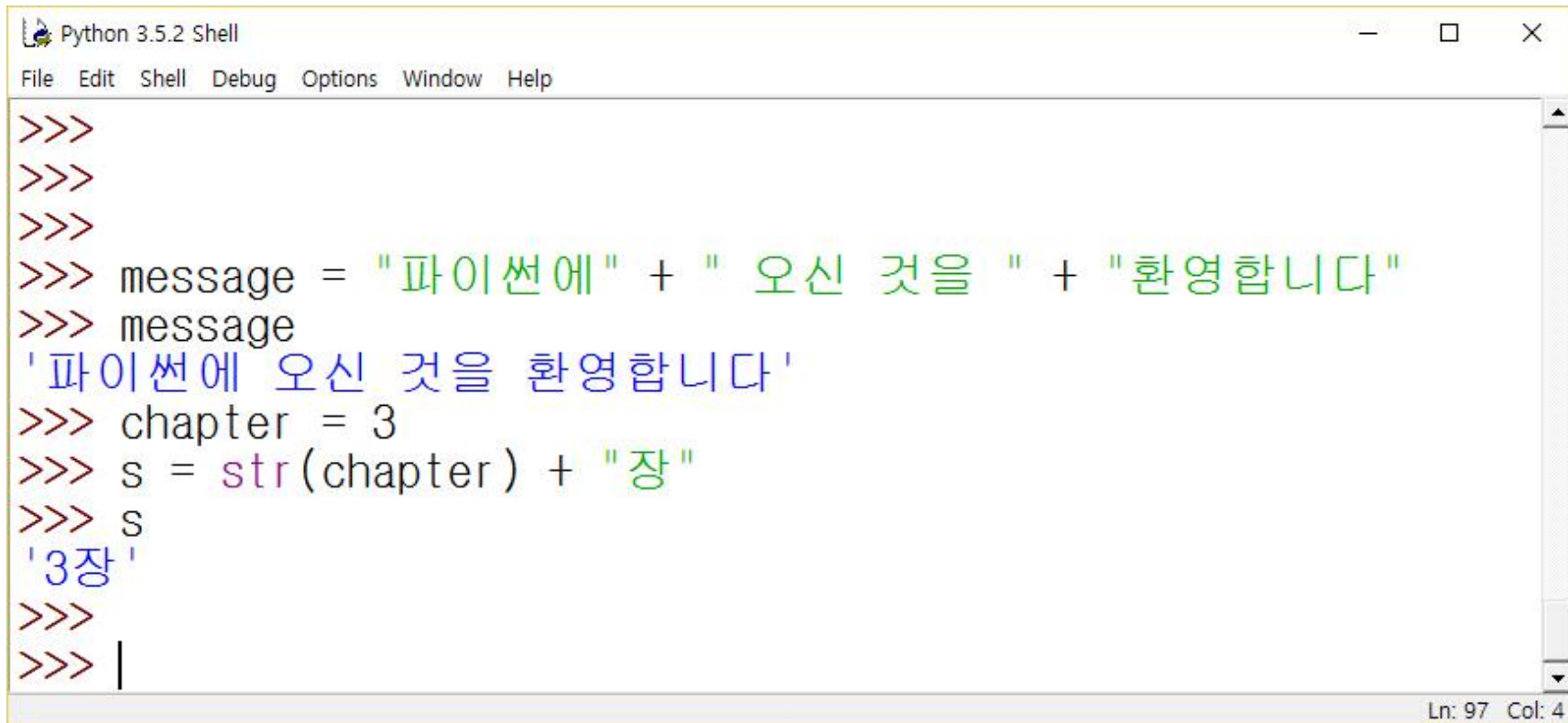
- 숫자를 문자로 변환

A screenshot of a Python 3.5.2 Shell window. The window has a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main area shows a series of Python commands and their outputs. The commands are: 1. s = str(3.7) # 실수를 문자열로 변환, followed by s, which outputs '3.7'. 2. s = str(5) # 정수를 문자열로 변환, followed by s, which outputs '5'. 3. s = str(324), followed by s, which outputs '324'. 4. eval("324"), which outputs 324. The prompt is currently at the start of a new line. The status bar at the bottom right shows 'Ln: 86 Col: 4'.

```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
>>> s = str(3.7) # 실수를 문자열로 변환
>>> s
'3.7'
>>> s = str(5) # 정수를 문자열로 변환
>>> s
'5'
>>> s = str(324)
>>> s
'324'
>>> eval("324")
324
>>> |
Ln: 86 Col: 4
```

# 문자열 연결 연산자, +

- 두 문자열을 연결하는데 사용

A screenshot of a Python 3.5.2 Shell window. The window has a title bar with the text 'Python 3.5.2 Shell' and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main area of the window contains a series of Python commands and their outputs. The commands are: three empty lines, followed by 'message = "파이썬에" + " 오신 것을 " + "환영합니다"', then 'message' (outputting '파이썬에 오신 것을 환영합니다'), then 'chapter = 3', then 's = str(chapter) + "장"', then 's' (outputting '3장'), and finally three more empty lines with a cursor at the end of the last line. The status bar at the bottom right shows 'Ln: 97 Col: 4'.

```
>>>
>>>
>>>
>>> message = "파이썬에" + " 오신 것을 " + "환영합니다"
>>> message
'파이썬에 오신 것을 환영합니다'
>>> chapter = 3
>>> s = str(chapter) + "장"
>>> s
'3장'
>>>
>>> |
```

# 숫자 및 문자 서식 지정하기

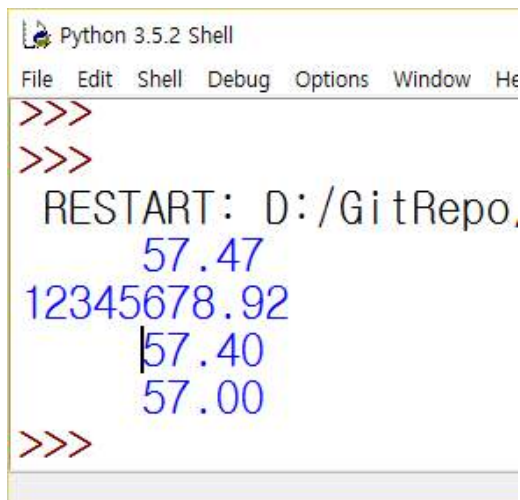
- format 함수
  - 숫자 혹은 문자를 특정 서식에 맞춰 출력하기 위해 사용
- format(항목, 형식 지정자)
  - 항목: 숫자 혹은 문자열 값
  - 형식 지정자: 항목이 어떤 형식으로 표현되어야 하는지 지정하는 문자열

# 부동소수점 숫자 서식 지정

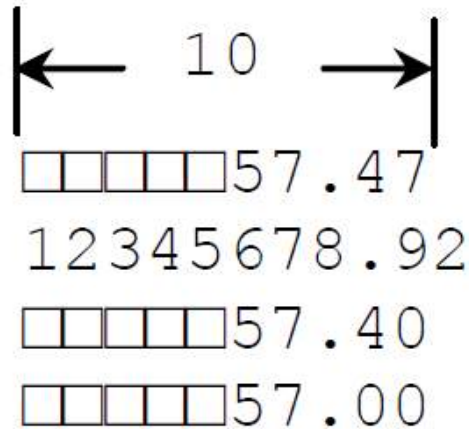
- 부동소수점 숫자(실수값)의 서식 지정
  - 형식 지정자: width.precisionf
  - width: 출력 문자열의 폭 지정
  - precision: 소수점 이하 자릿수 지정
  - f: 부동소수점 숫자를 위한 서식 설정을 위한 변환 코드
- 10.2f
  - 폭: 10
  - 소수점 이하 자릿수: 2
  - 변환 코드: f
- width(폭 지정자)는 생략 가능: 기본값 0 (필요한 만큼 폭의 크기가 자동 설정)

# 부동소수점 숫자 서식 지정

```
print(format(57.467657, "10.2f"))  
print(format(12345678.923, "10.2f"))  
print(format(57.4, "10.2f"))  
print(format(57, "10.2f"))
```



```
Python 3.5.2 Shell  
File Edit Shell Debug Options Window Help  
>>>  
>>>  
RESTART: D:/GitRepo.  
57.47  
12345678.92  
57.40  
57.00  
>>>
```



# 과학적 표기법 서식 지정

- 부동소수점 숫자 서식을 위한 형식 지정자와 유사
- 변환 코드 f를 e로 변경

```
print(format(57.467657, "10.2e"))  
print(format(0.0033923, "10.2e"))  
print(format(57.4, "10.2e"))  
print(format(57, "10.2e"))
```

← 10 →

□□5.75e+01

□□3.39e-03

□□5.74e+01

□□5.70e+01

# 백분율 서식 지정

- 숫자를 백분율로 서식화
- 변환 코드로 % 사용

```
print(format(0.53457, "10.2%"))  
print(format(0.0033923, "10.2%"))  
print(format(7.4, "10.2%"))  
print(format(57, "10.2%"))
```




Diagram illustrating the width of the formatted output. A double-headed arrow indicates a width of 10 characters. Below the arrow, four examples of formatted numbers are shown, each with a width of 10 characters:

- 53.46%
- 0.34%
- 740.00%
- 5700.00%



# 서식 정렬

- 기본적인 숫자 서식
  - 오른쪽 정렬
- < 기호
  - 형식 지정자에 < 기호 삽입
  - 지정된 폭 내에서 항목을 왼쪽 정렬

```
print(format(57.467657, "10.2f"))  
print(format(57.467657, "<10.2f"))
```

10

□□□□□57.47

57.47

# 정수 서식 지정

- 변환 코드 d, x, o, b 사용
  - 10진수, 16진수, 8진수, 2진수로 서식화
- 부동소수점 형식 지정자의 precision 부분이 없음

```
print(format(59832, "10d"))  
print(format(59832, "<10d"))  
print(format(59832, "10x"))  
print(format(59832, "<10x"))
```

The diagram illustrates the formatting of the integer 59832. At the top, a horizontal line with arrows at both ends is labeled '10', indicating a total width of 10 characters. Below this, four rows show the number 59832 formatted with different alignment and padding options:

- Row 1: Five empty boxes followed by 59832, representing right-aligned padding.
- Row 2: The number 59832 with no padding.
- Row 3: Five empty boxes followed by e9b8, representing right-aligned padding in hexadecimal.
- Row 4: The hexadecimal string e9b8 with no padding.

# 문자열 서식 지정

- 변환 코드 s 사용
- 문자열은 기본적으로 왼쪽 정렬
- 오른쪽 정렬
  - > 기호 사용
- 문자열이 지정한 폭보다 길면, 폭은 자동으로 문자열에 맞게 확장

```
print(format("파이썬에 오신 것을 환영합니다.", "20s"))  
print(format("파이썬에 오신 것을 환영합니다.", "<20s"))  
print(format("파이썬에 오신 것을 환영합니다.", ">20s"))
```

← 20 →

파이썬에 오신 것을 환영합니다.

파이썬에 오신 것을 환영합니다.

□□□ 파이썬에 오신 것을 환영합니다.

# 객체와 메소드의 개념

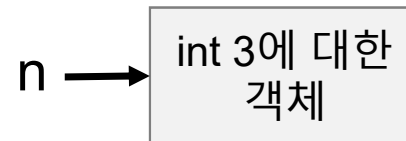
- 파이썬에서 모든 데이터는 객체이다
  - 숫자, 문자열
- 각 객체는 id와 type을 갖는다
- id
  - 프로그램이 실행될 때 파이썬에 의해 자동으로 할당되는 중복되지 않는 정수값
  - 프로그램이 실행되는 동안 객체의 id는 변하지 않음
  - 프로그램이 실행될 때마다 다른 id 값을 할당할 수 있음
- type
  - 객체의 타입은 객체의 값에 따라 파이썬에 의해 결정됨
  - 클래스에 의해 정의됨 (파이썬에서 타입과 클래스는 동의어)
    - 문자열 클래스: str
    - 정수: int
    - 실수: float

# 객체의 type과 id

```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
>>> n = 3
>>> id(n)
493508912
>>> type(n)
<class 'int'>
>>> f = 3.0
>>> id(f)
53230528
>>> type(f)
<class 'float'>
>>> s = "welcome"
>>> id(s)
59374272
>>> type(s)
<class 'str'>
>>> |
```

n = 3

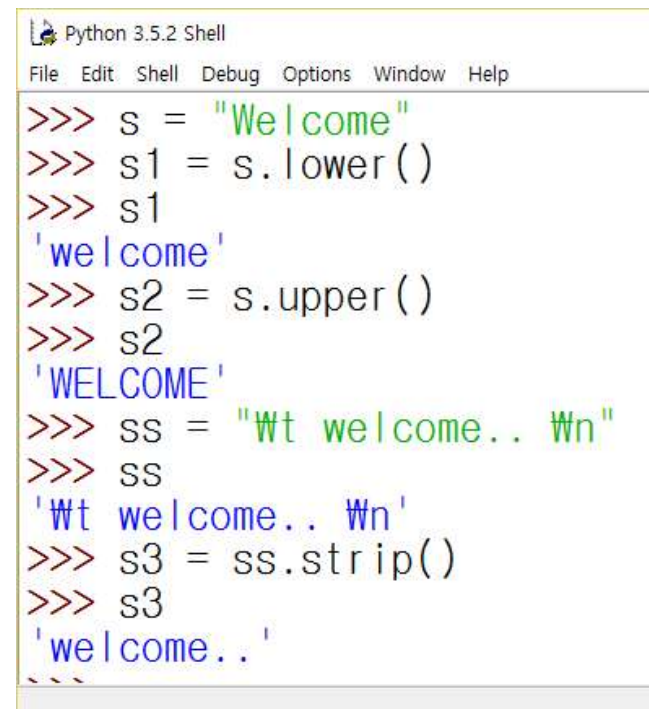
id: 493508912



- n = 3 명령문 → 변수 n에 정수 3을 할당
- 변수 n에 의해 참조되는 int 객체에 3을 저장 (n은 정수값 3을 저장하는 int 객체를 참조하는 변수)

# 메소드

- 객체에 대하여 수행할 수 있는 연산을 정의한 함수
- 메소드의 예
  - 문자열 타입(클래스)은 lower(), upper() 메소드를 가지고 있음
  - lower()
    - 해당 문자열의 소문자 문자열을 반환
  - upper()
    - 해당 문자열의 대문자 문자열을 반환
- 메소드 호출
  - object.method()

A screenshot of a Python 3.5.2 Shell window. The window has a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The shell displays a series of Python commands and their outputs. The commands are: s = "Welcome", s1 = s.lower(), s1, s2 = s.upper(), s2, ss = "Wt welcome.. Wn", ss, s3 = ss.strip(), and s3. The outputs are: 'welcome', 'WELCOME', and 'welcome..' respectively. The text is color-coded: strings in green, variables in blue, and method calls in red.

```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
>>> s = "Welcome"
>>> s1 = s.lower()
>>> s1
'welcome'
>>> s2 = s.upper()
>>> s2
'WELCOME'
>>> ss = "Wt welcome.. Wn"
>>> ss
'Wt welcome.. Wn'
>>> s3 = ss.strip()
>>> s3
'welcome..'
>>>
```