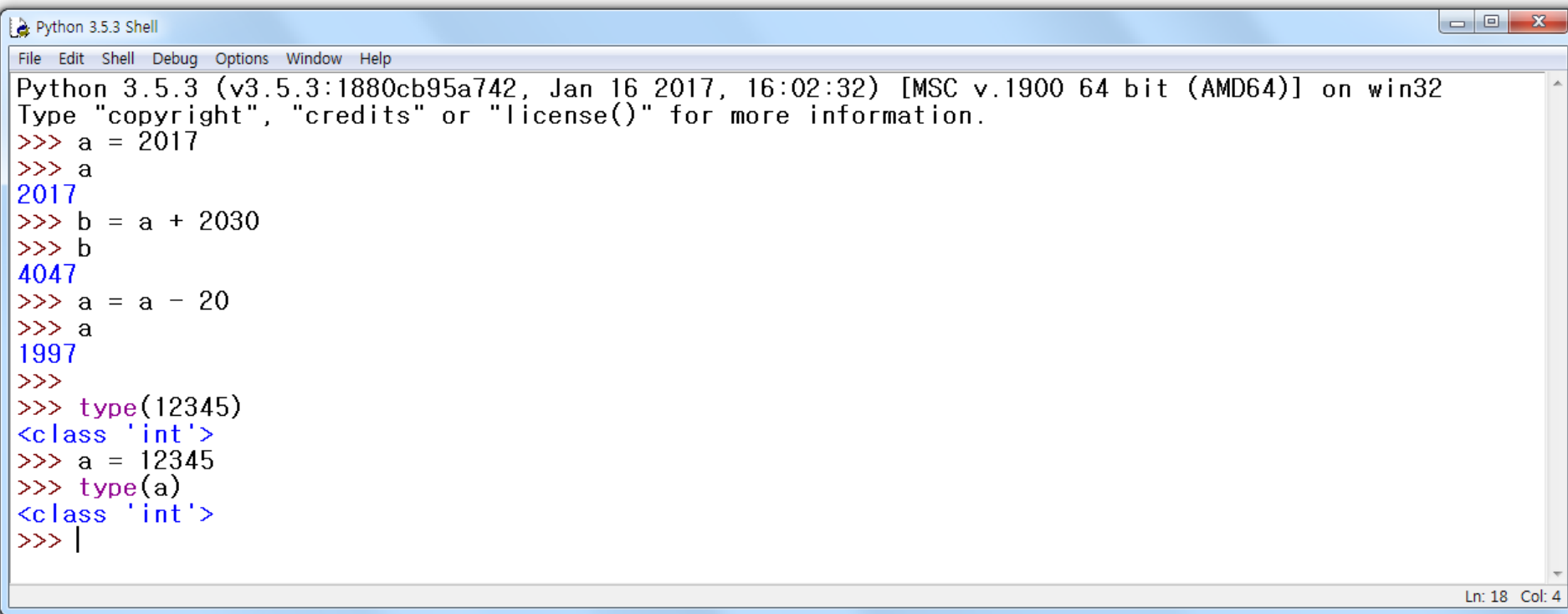


데이터

변수

- 데이터의 저장과 참조를 위해 할당된 메모리 공간
- 단 하나의 값을 저장할 수 있는 공간



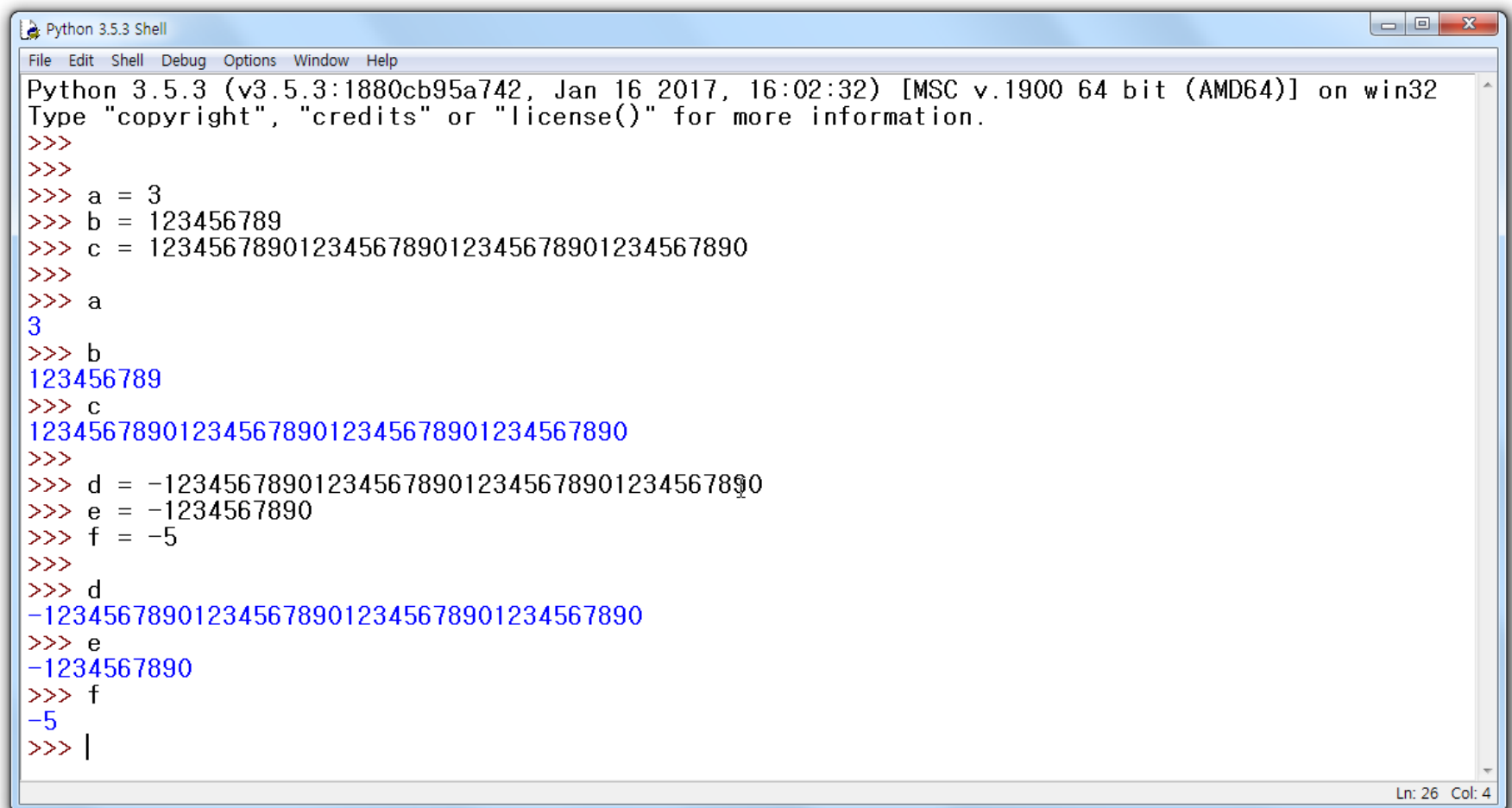
```
Python 3.5.3 Shell
File Edit Shell Debug Options Window Help
Python 3.5.3 (v3.5.3:1880cb95a742, Jan 16 2017, 16:02:32) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> a = 2017
>>> a
2017
>>> b = a + 2030
>>> b
4047
>>> a = a - 20
>>> a
1997
>>>
>>> type(12345)
<class 'int'>
>>> a = 12345
>>> type(a)
<class 'int'>
>>> |
```

Ln: 18 Col: 4

- 동적 형식 언어(Dynamic typed language) / 정적 형식 언어(Static typed language)

수 다루기 - 정수

- 메모리가 허용하는 한, 무한대의 정수 처리

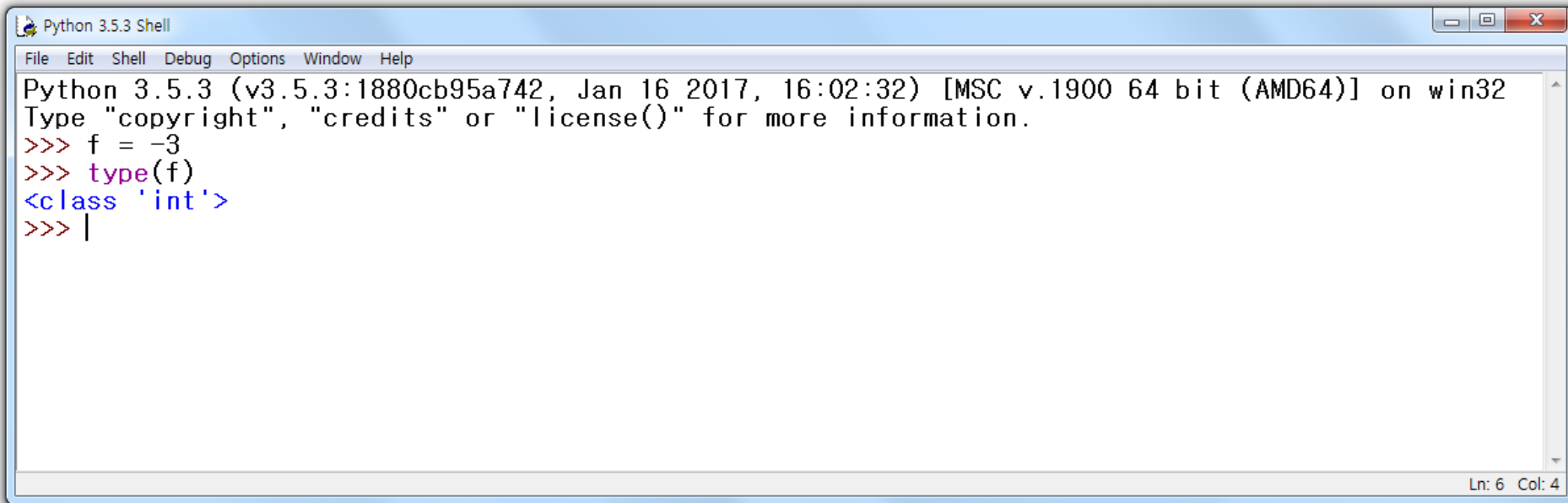


```
Python 3.5.3 Shell
File Edit Shell Debug Options Window Help
Python 3.5.3 (v3.5.3:1880cb95a742, Jan 16 2017, 16:02:32) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
>>>
>>> a = 3
>>> b = 123456789
>>> c = 1234567890123456789012345678901234567890
>>>
>>> a
3
>>> b
123456789
>>> c
1234567890123456789012345678901234567890
>>>
>>> d = -1234567890123456789012345678901234567890
>>> e = -1234567890
>>> f = -5
>>>
>>> d
-1234567890123456789012345678901234567890
>>> e
-1234567890
>>> f
-5
>>> |
```

Ln: 26 Col: 4

수 다루기 - 정수

- 파이썬은 코드가 실행될 때 변수의 형식을 결정



```
Python 3.5.3 Shell
File Edit Shell Debug Options Window Help
Python 3.5.3 (v3.5.3:1880cb95a742, Jan 16 2017, 16:02:32) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> f = -3
>>> type(f)
<class 'int'>
>>> |
```

Ln: 6 Col: 4

정수의 사칙 연산

| 연산자 | 기호 |
|---------|----|
| 더하기 | + |
| 빼기 | - |
| 곱하기 | * |
| 나누기 | / |
| 나눗셈 몫 | // |
| 나눗셈 나머지 | % |

산술 연산

```
Python 3.5.3 Shell
File Edit Shell Debug Options Window Help
Python 3.5.3 (v3.5.3:1880cb95a742, Jan 16 2017, 16:02:32) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
>>> a = 3 + 4
>>> a
7
>>>
>>> b = 7 - 10
>>> b
-3
>>>
>>> c = 7 * -3
>>> c
-21
>>>
>>> d = 30 // 7
>>> d
4
>>>
>>> e = 30 % 7
>>> e
2
```

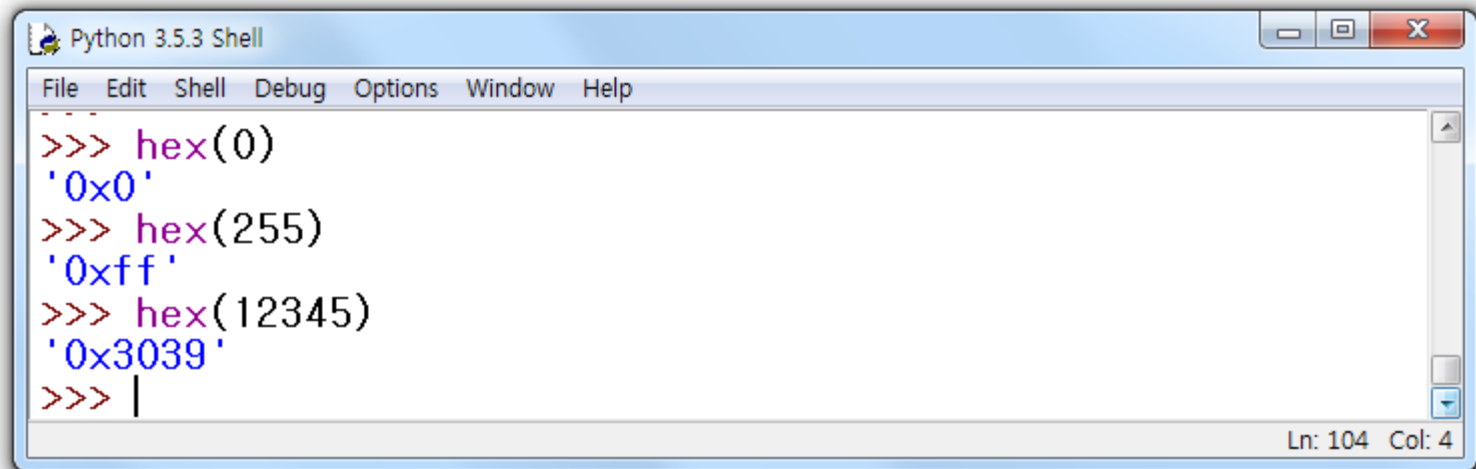
```
Python 3.5.3 Shell
File Edit Shell Debug Options Window Help
>>>
>>> f = 22 / 7
>>> f
3.142857142857143
>>> type(f)
<class 'float'>
>>> |
```

Ln: 91 Col: 4

10진수/2진수/16진수

| 10진수 | 2진수 | 16진수 |
|------|------|------|
| 0 | 0000 | 0 |
| 1 | 0001 | 1 |
| 2 | 0010 | 2 |
| 3 | 0011 | 3 |
| 4 | 0100 | 4 |
| 5 | 0101 | 5 |
| 6 | 0110 | 6 |
| 7 | 0111 | 7 |
| 8 | 1000 | 8 |
| 9 | 1001 | 9 |
| 10 | 1010 | A |
| 11 | 1011 | B |
| 12 | 1100 | C |
| 13 | 1101 | D |
| 14 | 1110 | E |
| 15 | 1111 | F |

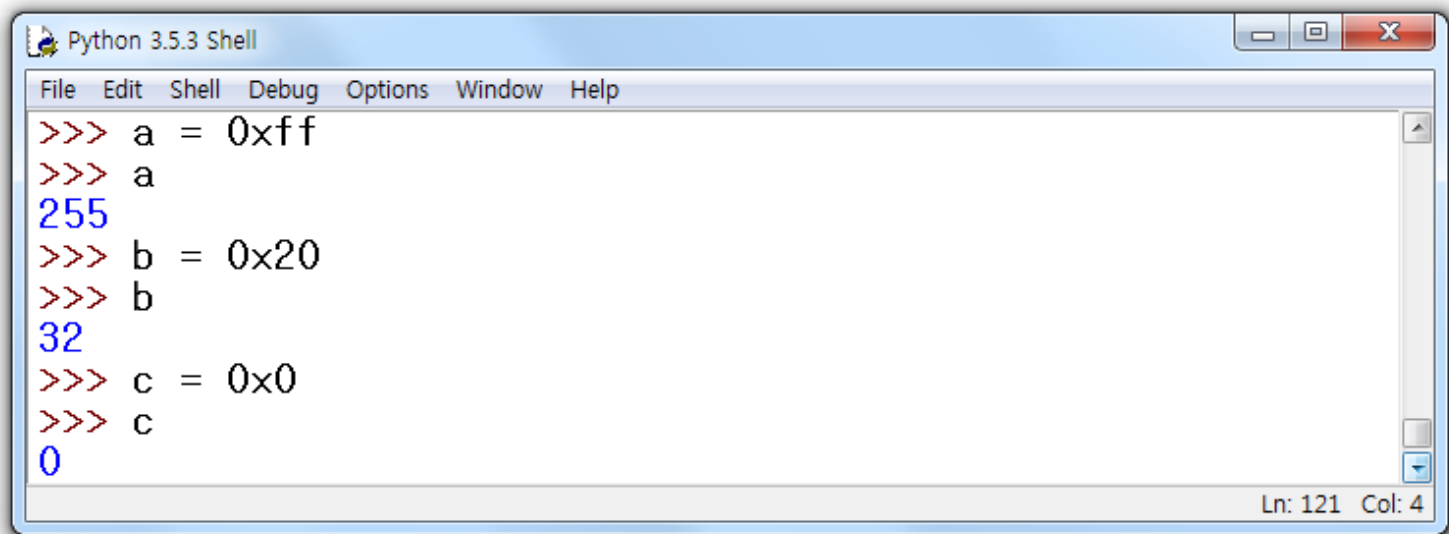
10진수 -> 16진수 변환

A screenshot of a Python 3.5.3 Shell window. The window has a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main text area shows the following code and output:

```
>>> hex(0)
'0x0'
>>> hex(255)
'0xff'
>>> hex(12345)
'0x3039'
>>> |
```

The status bar at the bottom right indicates 'Ln: 104 Col: 4'.

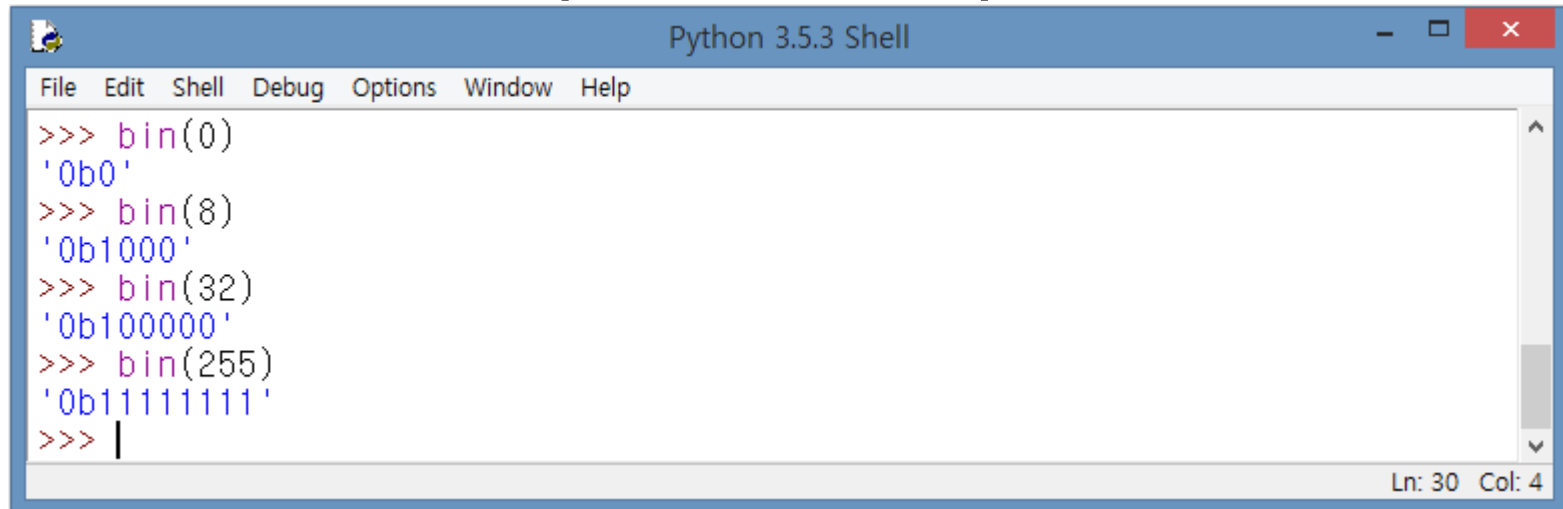
16진수 -> 10진수 변환

A screenshot of a Python 3.5.3 Shell window. The window has a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main text area shows the following code and output:

```
>>> a = 0xff
>>> a
255
>>> b = 0x20
>>> b
32
>>> c = 0x0
>>> c
0
```

The status bar at the bottom right indicates 'Ln: 121 Col: 4'.

10진수 -> 2진수 변환

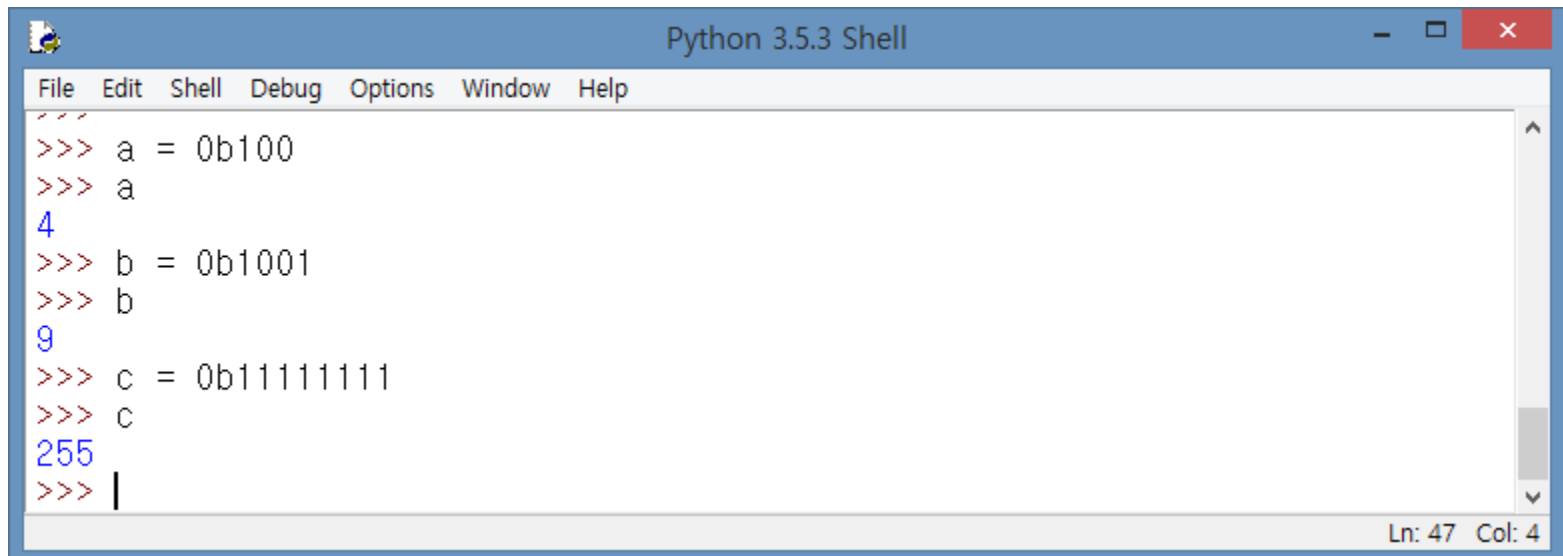


A screenshot of a Python 3.5.3 Shell window. The window has a blue title bar with the text 'Python 3.5.3 Shell' and standard window controls. Below the title bar is a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main text area shows the following code and output:

```
>>> bin(0)
'0b0'
>>> bin(8)
'0b1000'
>>> bin(32)
'0b100000'
>>> bin(255)
'0b11111111'
>>> |
```

The status bar at the bottom right shows 'Ln: 30 Col: 4'.

2진수 -> 10진수 변환

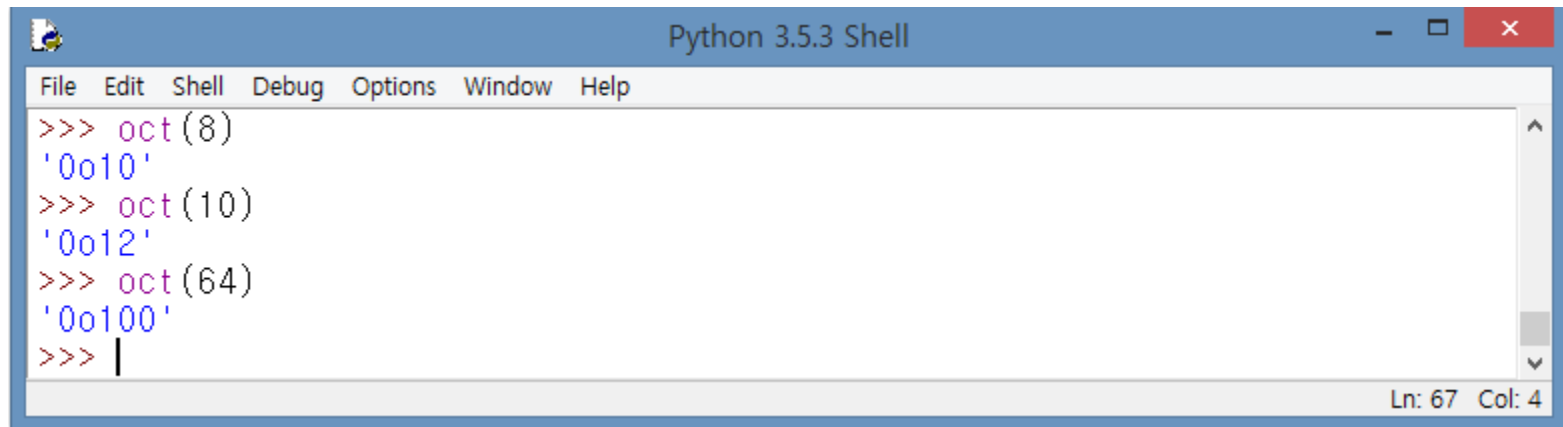


A screenshot of a Python 3.5.3 Shell window. The window has a blue title bar with the text 'Python 3.5.3 Shell' and standard window controls. Below the title bar is a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main text area shows the following code and output:

```
>>> a = 0b100
>>> a
4
>>> b = 0b1001
>>> b
9
>>> c = 0b11111111
>>> c
255
>>> |
```

The status bar at the bottom right shows 'Ln: 47 Col: 4'.

10진수 -> 8진수 변환

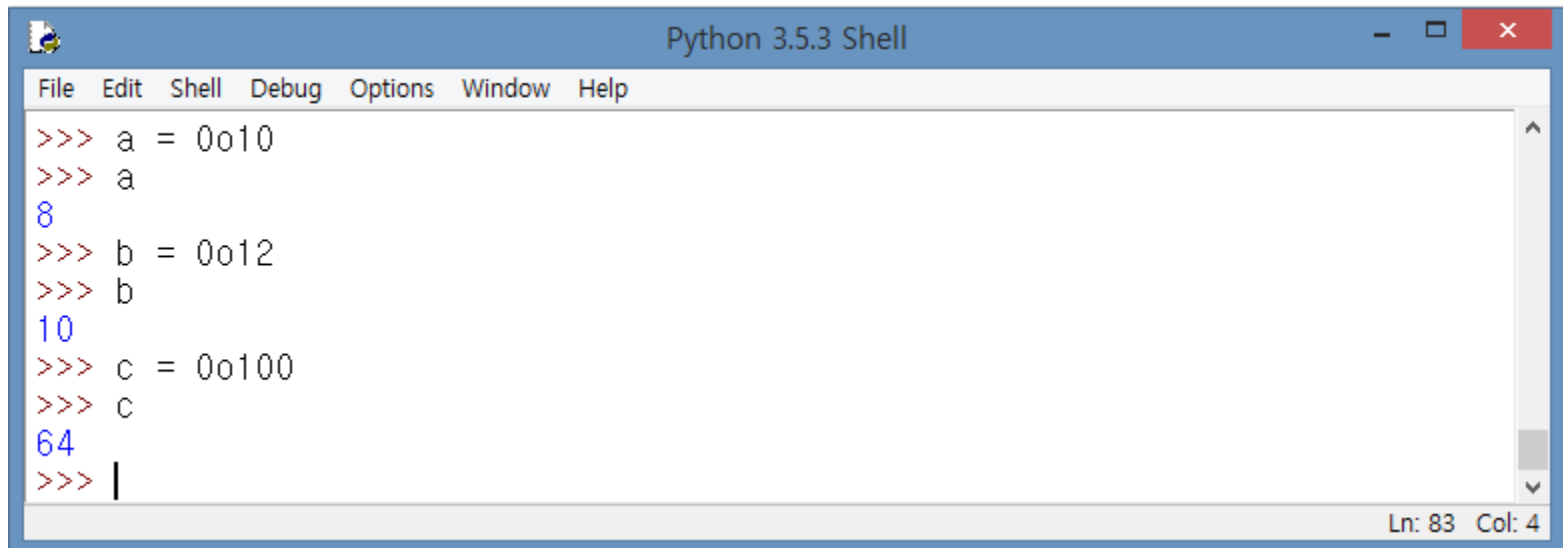
A screenshot of a Python 3.5.3 Shell window. The window has a blue title bar with the text "Python 3.5.3 Shell" and standard window controls. Below the title bar is a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main text area shows the following code:

```
>>> oct(8)
'0o10'
>>> oct(10)
'0o12'
>>> oct(64)
'0o100'
>>> |
```

 The status bar at the bottom right shows "Ln: 67 Col: 4".

```
Python 3.5.3 Shell
File Edit Shell Debug Options Window Help
>>> oct(8)
'0o10'
>>> oct(10)
'0o12'
>>> oct(64)
'0o100'
>>> |
Ln: 67 Col: 4
```

8진수 -> 10진수 변환

A screenshot of a Python 3.5.3 Shell window. The window has a blue title bar with the text "Python 3.5.3 Shell" and standard window controls. Below the title bar is a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main text area shows the following code:

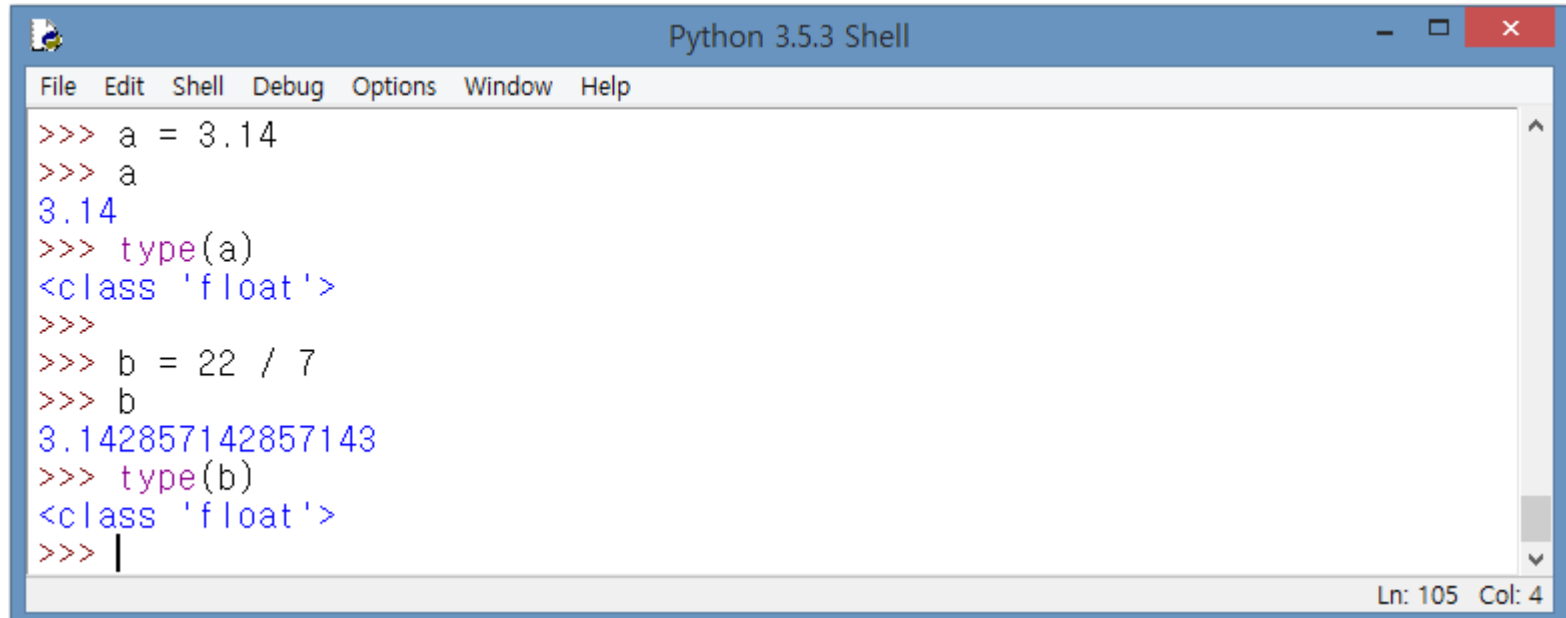
```
>>> a = 0o10
>>> a
8
>>> b = 0o12
>>> b
10
>>> c = 0o100
>>> c
64
>>> |
```

 The status bar at the bottom right shows "Ln: 83 Col: 4".

```
Python 3.5.3 Shell
File Edit Shell Debug Options Window Help
>>> a = 0o10
>>> a
8
>>> b = 0o12
>>> b
10
>>> c = 0o100
>>> c
64
>>> |
Ln: 83 Col: 4
```

수 다루기 - 실수

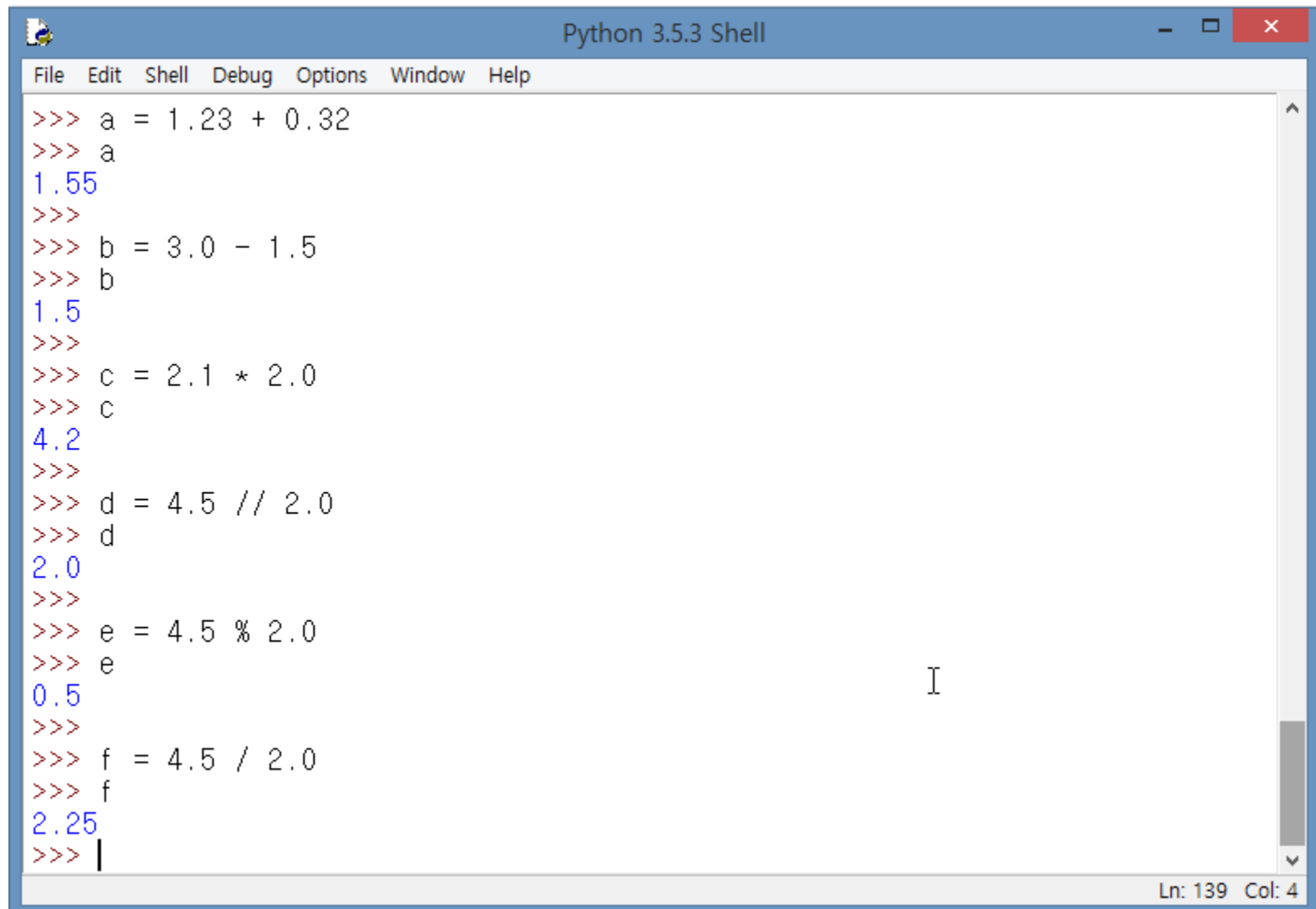
- 파이썬에서는 실수 지원을 위해 부동 소수형을 제공
- 부동 소수형은 소수점을 움직여 소수를 표현하는 자료형
- 부동 소수형은 8byte만을 이용해서 수를 표현
- 한정된 범위의 수만 표현 가능
- 디지털 방식으로 소수를 표현해야 하므로 정밀도에 한계



```
Python 3.5.3 Shell
File Edit Shell Debug Options Window Help
>>> a = 3.14
>>> a
3.14
>>> type(a)
<class 'float'>
>>>
>>> b = 22 / 7
>>> b
3.142857142857143
>>> type(b)
<class 'float'>
>>> |
```

Ln: 105 Col: 4

부동 소수형의 사칙 연산



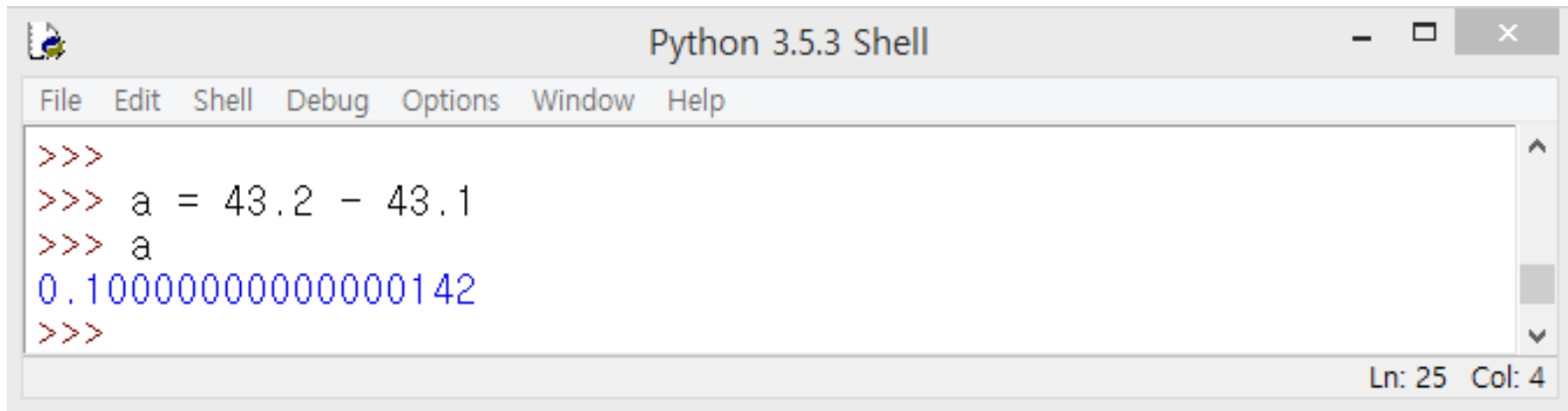
A screenshot of a Python 3.5.3 Shell window. The window has a blue title bar with the text "Python 3.5.3 Shell" and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Window", and "Help". The main area is a text editor with a white background, showing a series of Python commands and their outputs. The commands are: `a = 1.23 + 0.32`, `a`, `b = 3.0 - 1.5`, `b`, `c = 2.1 * 2.0`, `c`, `d = 4.5 // 2.0`, `d`, `e = 4.5 % 2.0`, `e`, `f = 4.5 / 2.0`, and `f`. The outputs are: `1.55`, `1.5`, `4.2`, `2.0`, `0.5`, and `2.25`. The cursor is at the end of the last line. The status bar at the bottom right shows "Ln: 139 Col: 4".

```
>>> a = 1.23 + 0.32
>>> a
1.55
>>>
>>> b = 3.0 - 1.5
>>> b
1.5
>>>
>>> c = 2.1 * 2.0
>>> c
4.2
>>>
>>> d = 4.5 // 2.0
>>> d
2.0
>>>
>>> e = 4.5 % 2.0
>>> e
0.5
>>>
>>> f = 4.5 / 2.0
>>> f
2.25
>>> |
```

Ln: 139 Col: 4

부동 소수형 사용시 주의할 점

- IEEE 754 표준을 따름
- 부동 소수형은 정밀도의 한계를 가지고 있음

A screenshot of a Python 3.5.3 Shell window. The window has a title bar with the text 'Python 3.5.3 Shell' and standard window controls. Below the title bar is a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main text area contains the following code:

```
>>>  
>>> a = 43.2 - 43.1  
>>> a  
0.1000000000000000142  
>>>
```

The output '0.1000000000000000142' is displayed in blue text. At the bottom right of the window, the status bar shows 'Ln: 25 Col: 4'.

Python 3.6.0 Shell

File Edit Shell Debug Options Window Help

```
>>> dir(str)
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'capitalize', 'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isdecimal', 'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']
```

```
>>>
```

```
>>> help(str.strip)
```

Help on method_descriptor:

```
strip(...)
```

S.strip([chars]) -> str

Return a copy of the string S with leading and trailing
whitespace removed.

If chars is given and not None, remove characters in chars instead.

```
>>>
```

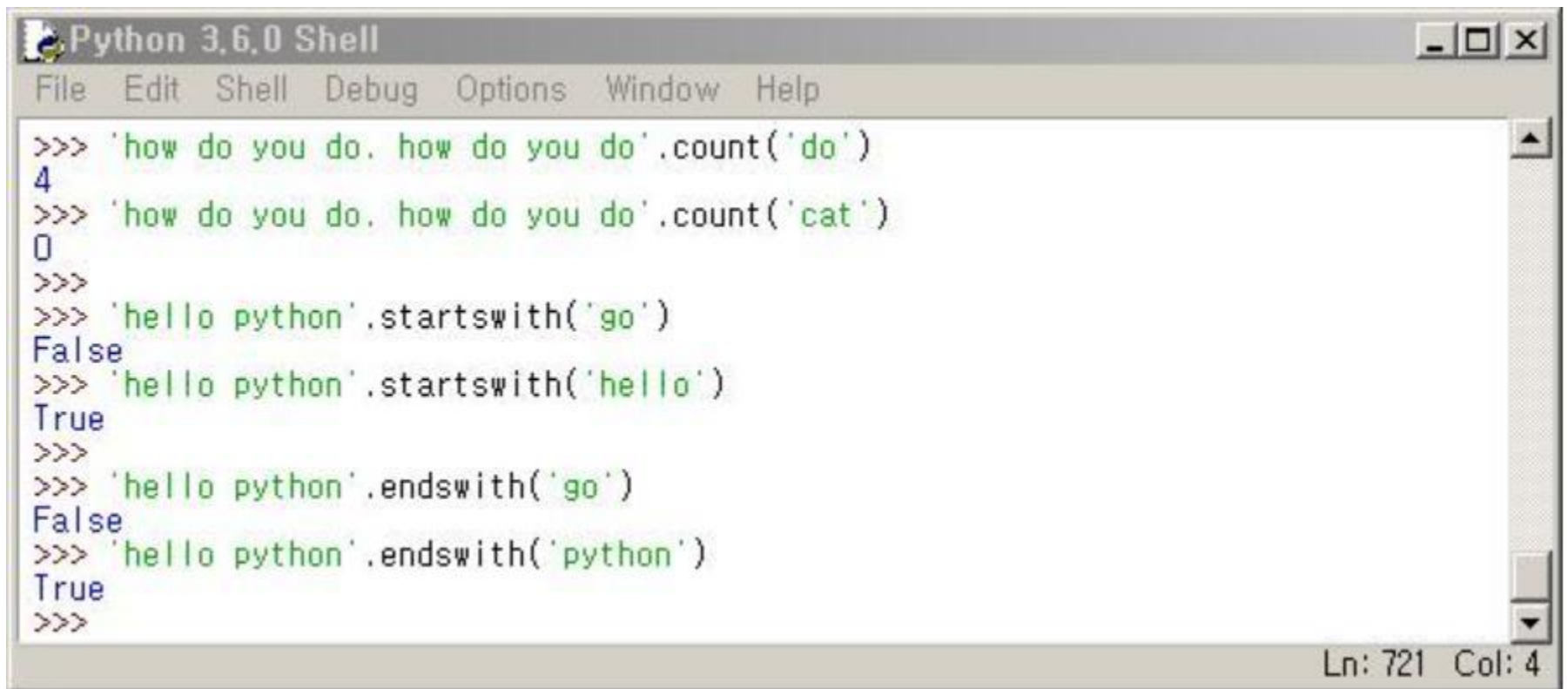
Ln: 40 Col: 4

문자열

- **문자열 확인**

- **count()** - 특정 단어(문자열)의 수를 구함 (없으면 0을 반환)
 - 문자열의 글자수는 len() 함수를 사용하여 구한다.
- **startswith()** - 특정 단어로 시작하는지 확인
- **endswith()** - 특정 단어로 끝나는지 확인
- **find()** - 특정 단어를 찾아 인덱스를 리턴 (없으면 -1을 리턴)
- **rfind()** - 뒤에서부터 특정 단어를 찾아 인덱스를 리턴
 - ❖ in, not in을 사용하면 특정 단어가 있는지 없는지 확인 가능 (True, False)
- **index()** - find()와 동일하지만 특정 단어 없을 때 예외를 발생시킴
- **rindex()** - rfind()와 동일하지만 특정 단어 없을 때 예외를 발생시킴

문자열

A screenshot of a Python 3.6.0 Shell window. The window has a title bar with the text "Python 3.6.0 Shell" and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with the following items: File, Edit, Shell, Debug, Options, Window, and Help. The main area of the window contains a series of Python commands and their outputs, all in green text. The commands are: 1. '>>> 'how do you do. how do you do'.count('do')' followed by the output '4'. 2. '>>> 'how do you do. how do you do'.count('cat')' followed by the output '0'. 3. '>>> 'hello python'.startswith('go')' followed by the output 'False'. 4. '>>> 'hello python'.startswith('hello')' followed by the output 'True'. 5. '>>> 'hello python'.endswith('go')' followed by the output 'False'. 6. '>>> 'hello python'.endswith('python')' followed by the output 'True'. 7. '>>>' followed by a blank line. At the bottom right of the window, there is a status bar showing "Ln: 721 Col: 4".

```
Python 3.6.0 Shell
File Edit Shell Debug Options Window Help
>>> 'how do you do. how do you do'.count('do')
4
>>> 'how do you do. how do you do'.count('cat')
0
>>>
>>> 'hello python'.startswith('go')
False
>>> 'hello python'.startswith('hello')
True
>>>
>>> 'hello python'.endswith('go')
False
>>> 'hello python'.endswith('python')
True
>>>
Ln: 721 Col: 4
```



```
>>> 'I am a boy. I am a student'.find('am')
2
>>> 'I am a boy. I am a student'.find('you')
-1
>>>
>>> 'I am a boy. I am a student'.rfind('am')
14
>>> 'I am a boy. I am a student'.rfind('you')
-1
>>>
>>> 'am' in 'I am a boy. I am a student'
True
>>> 'you' in 'I am a boy. I am a student'
False
>>>
>>> 'I am a boy. I am a student'.index('am')
2
>>> 'I am a boy. I am a student'.index('you')
Traceback (most recent call last):
  File "<pyshell#426>", line 1, in <module>
    'I am a boy. I am a student'.index('you')
ValueError: substring not found
>>>
>>> 'I am a boy. I am a student'.rindex('am')
14
>>> 'I am a boy. I am a student'.rindex('you')
Traceback (most recent call last):
  File "<pyshell#429>", line 1, in <module>
    'I am a boy. I am a student'.rindex('you')
ValueError: substring not found
>>>
```

문자열

- **문자열 변환(변경)**

- **upper()** - 대문자로 변경
- **lower()** - 소문자로 변경
- **swapcase()** - 대문자는 소문자로, 소문자는 대문자로 변경
- **capitalize()** - 첫 문자를 대문자로 변경
- **title()** - 각 단어의 첫 글자를 대문자로 변경
- **strip()** - 문자열 양쪽 끝을 자른다. 제거할 문자를 인자로 전달 (디폴트는 공백)
- **lstrip()** - 문자열 왼쪽을 자름
- **rstrip()** - 문자열 오른쪽을 자름
- **replace()** - 문자열 특정 부분을 변경 (대체)
- **format()** - 틀(포맷)을 만들어 놓고 문자열을 생성
- **join()** - 리스트 같은 iterable 인자를 전달하여 문자열로 연결

```
Python 3.6.0 Shell
File Edit Shell Debug Options Window Help

>>> 'Hello Python'.upper()
'HELLO PYTHON'
>>> 'Check it OUT'.lower()
'check it out'
>>> 'Hello Python'.swapcase()
'hELLO pYTHON'
>>> 'hello python!'.capitalize()
'Hello python!'
>>> 'hello python'.title()
'Hello Python'
>>>
```

```
Python 3.6.0 Shell
File Edit Shell Debug Options Window Help

>>> ' Oh my god! '.strip()
'Oh my god!'
>>> '#$%#@#$^Hello Python*&^$%^$%^@'.strip('!@#$$%^&*()')
'Hello Python'
>>>
>>> ' Oh my god! '.lstrip()
'Oh my god!'
>>> 'TEL NUMBER 010-1234-5678 '.lstrip('NUMBERTEL ')
'010-1234-5678'
>>>
>>> ' Oh my god! '.rstrip()
'Oh my god!'
>>> 'TEL NUMBER 010-1234-5678 '.rstrip('1234567890- ')
'TEL NUMBER'
>>>
```

Ln: 564 Col: 4

문자열

A screenshot of a Python 3.6.0 Shell window. The window has a title bar with the text 'Python 3.6.0 Shell' and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with the following items: File, Edit, Shell, Debug, Options, Window, and Help. The main area of the window contains a series of Python code snippets and their outputs. The code is as follows:

```
>>> 'good job!'.replace('good', 'bad')
'bad job!'
>>>
>>> '{} is a {}'.format('Jack', 'cook')
'Jack is a cook'
>>>
>>> '&'.join(['c++', 'python', 'php'])
'c++&python&php'
>>> '\n'.join(['c++', 'python', 'php'])
'c++\npython\nphp'
>>> print('\n'.join(['c++', 'python', 'php']))
c++
python
php
>>>
```

The output of the last command is displayed on three separate lines. At the bottom right of the window, the status bar shows 'Ln: 581 Col: 4'.

```
Python 3.6.0 Shell
File Edit Shell Debug Options Window Help
>>> 'good job!'.replace('good', 'bad')
'bad job!'
>>>
>>> '{} is a {}'.format('Jack', 'cook')
'Jack is a cook'
>>>
>>> '&'.join(['c++', 'python', 'php'])
'c++&python&php'
>>> '\n'.join(['c++', 'python', 'php'])
'c++\npython\nphp'
>>> print('\n'.join(['c++', 'python', 'php']))
c++
python
php
>>>
Ln: 581 Col: 4
```

문자열

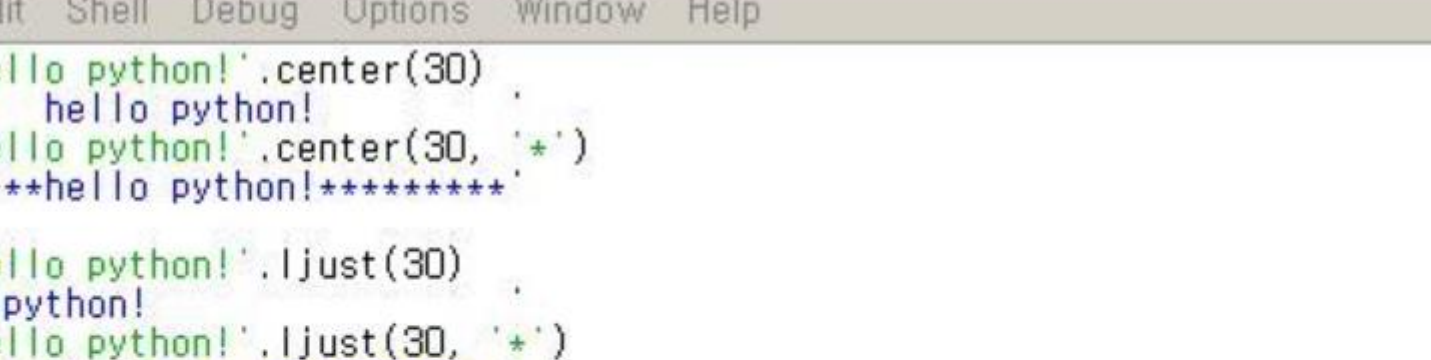
- **문자열 정렬**

- **center()** - 문자열 가운데 정렬. (인자로 넓이를 지정, 채울 문자 선택 가능)
- **ljust()** - 문자열 왼쪽 정렬
- **rjust()** - 문자열 오른쪽 정렬

- **문자열 분리(나누기)**

- **partition()** - 전달한 문자로 문자열을 나눔(분리), 결과는 튜플(구분자도 포함)
- **rpartition()** - 뒤에서 부터 전달한 인자로 문자열을 나눔
- **split()** - 전달한 문자로 문자열을 나눔, 결과는 리스트(구분자 포함 안됨)
- **rsplit()** - 뒤에서 부터 전달한 문자로 문자열을 나눔
- **splitlines()** - 라인 단위로 문자열을 나눔

문자열



The screenshot shows a Python 3.6.0 Shell window with a menu bar (File, Edit, Shell, Debug, Options, Window, Help) and a command prompt. The following code is entered and executed:

```
>>> 'hello python!'.center(30)
      hello python!
>>> 'hello python!'.center(30, '*')
*****hello python!*****
>>>
>>> 'hello python!'.ljust(30)
hello python!
>>> 'hello python!'.ljust(30, '*')
hello python!*****
>>>
>>> 'hello python!'.rjust(30)
      hello python!
>>> 'hello python!'.rjust(30, '*')
*****hello python!
>>>
>>> '1234'.zfill(30)
'000000000000000000000000000000001234'
>>>
```

The status bar at the bottom right indicates "Ln: 604 Col: 4".

문자열

A screenshot of a Python 3.6.0 Shell window. The window has a title bar with the text 'Python 3.6.0 Shell' and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with the following items: File, Edit, Shell, Debug, Options, Window, and Help. The main area of the window contains a series of Python commands and their outputs, demonstrating string methods. The commands are: 1. '>>> '010-1234-5678'.partition('-')' which outputs ('010', '-', '1234-5678'). 2. '>>> '010-1234-5678'.rpartition('-')' which outputs ('010-1234', '-', '5678'). 3. '>>> '010-1234-5678'.split('-')' which outputs ['010', '1234', '5678']. 4. '>>> '010-1234-5678'.split('-', 1)' which outputs ['010', '1234-5678']. 5. '>>> '010-1234-5678'.rsplit('-')' which outputs ['010', '1234', '5678']. 6. '>>> '010-1234-5678'.rsplit('-', 1)' which outputs ['010-1234', '5678']. 7. '>>> '''Hello Python. Nice to meet you! Good bye~~~'''.splitlines()' which outputs ['Hello Python.', 'Nice to meet you!', 'Good bye~~~']. The status bar at the bottom right of the window shows 'Ln: 648 Col: 4'.

```
Python 3.6.0 Shell
File Edit Shell Debug Options Window Help
>>> '010-1234-5678'.partition('-')
('010', '-', '1234-5678')
>>> '010-1234-5678'.rpartition('-')
('010-1234', '-', '5678')
>>>
>>> '010-1234-5678'.split('-')
['010', '1234', '5678']
>>> '010-1234-5678'.split('-', 1)
['010', '1234-5678']
>>>
>>> '010-1234-5678'.rsplit('-')
['010', '1234', '5678']
>>> '010-1234-5678'.rsplit('-', 1)
['010-1234', '5678']
>>>
>>> """Hello Python.
Nice to meet you!
Good bye~~~"""
['Hello Python.', 'Nice to meet you!', 'Good bye~~~']
>>>
```

Ln: 648 Col: 4

문자열

- 문자열 종류 판단

- **isalnum()** - 알파벳 또는 숫자인가?
- **isalpha()** - 알파벳인가?
- **isdecimal()** - 숫자(decimal, 10진수)인가?
- **isdigit()** - 숫자(digit, 10진수)인가?
- **isidentifier()** - 식별자로 사용 가능한가?
- **islower()** - 소문자인가?
- **isnumeric()** - 숫자인가?
- **isspace()** - 공백인가?
- **istitle()** - title 형식인가? (단어마다 첫 글자가 대문자인가?)
- **isupper()** - 대문자인가?


```
Python 3.6.0 Shell
File Edit Shell Debug Options Window Help
>>> 'year2017'.isalnum()
True
>>> 'year2017!!!'.isalnum()
False
>>>
>>> 'numberOne'.isalpha()
True
>>> 'number1'.isalpha()
False
>>>
>>> 'one'.isdecimal()
False
>>> '12345'.isdecimal()
True
>>>
>>> 'book'.isdigit()
False
>>> '1004'.isdigit()
True
>>>
>>> 'book'.isidentifier()
True
>>> 'age@'.isidentifier()
False
>>>
>>> 'Hello Python'.islower()
False
>>> 'hello python'.islower()
True
>>>
>>> '1234'.isnumeric()
True
>>> 'age20'.isnumeric()
False
>>>
>>> 'good job'.isspace()
False
>>> ' '.isspace()
True
>>>
>>> 'hello python'.istitle()
False
>>> 'Hello python'.istitle()
False
>>> 'Hello Python'.istitle()
True
>>>
>>> 'Super Man'.isupper()
False
>>> 'SUPER MAN'.isupper()
True
>>> |
```

비트 연산자

| 비트연산자 | 설명 | 의미 |
|-------|------------|------------------|
| & | 비트 논리곱 | 둘 다 1일때 -> 1 |
| | 비트 논리합 | 둘 중 하나만 1 -> 1 |
| ^ | 비트 배타적 논리합 | 둘이 같으면 0, 다르면 1 |
| ~ | 비트 부정 | 1은 0으로, 0은 1로 변경 |
| << | 비트 왼쪽이동 | 비트를 왼쪽으로 이동 |
| >> | 비트 오른쪽이동 | 비트를 오른쪽으로 이동 |