

# 2D 게임 프로그래밍

- 반드시 카메라를 ON 하고 !
- 입장 이름은 "학번 이름"으로 설정 !
- 미리 수업 git 서버에서 자료를 Pull 해서 준비 !

# Lecture #1. 파이썬 기초 (1)

2D 게임 프로그래밍

이대현 교수

# 파이썬

---

- 1991년, Guido van Rossum 이 개발
- VM 기반 인터프리터 언어
- 최신 버전 3.9.2 (2021년 2월 기준)



# 특징

---

## ■ 다양한 프로그래밍 패러다임 제공

- 구조적 프로그래밍
- 절차지향 프로그래밍
- 객체지향 프로그래밍
- 함수형 프로그래밍

## ■ 동적 자료형

## ■ 풍부한 기본 라이브러리 함수

# 장점

---

- 쉽다.
- 간결하다.
- 빠르게 개발할 수 있다.

# 파이썬의 인기

■ 지난 30년간 지속적으로 순위가 상승됨.

Programming Language	2021	2016	2011	2006	2001	1996	1991	1986
C	1	2	2	2	1	1	1	1
Java	2	1	1	1	3	28	-	-
Python	3	5	6	7	24	15	-	-
C++	4	3	3	3	2	2	2	8
C#	5	4	5	6	9	-	-	-
JavaScript	6	7	10	9	6	30	-	-
PHP	7	6	4	4	19	-	-	-
R	8	14	39	-	-	-	-	-
SQL	9	-	-	-	-	-	-	-
Go	10	57	16	-	-	-	-	-
Perl	14	9	7	5	4	3	-	-
Lisp	29	24	13	13	17	5	3	2
Ada	33	23	21	15	15	6	9	3

프로그래밍 언어 순위 변화 ( Source : <https://www.tiobe.com/tiobe-index/> )

# 단점

---

- 느리다.
- 너무 유연하다.
- 진입 장벽이 낮다.

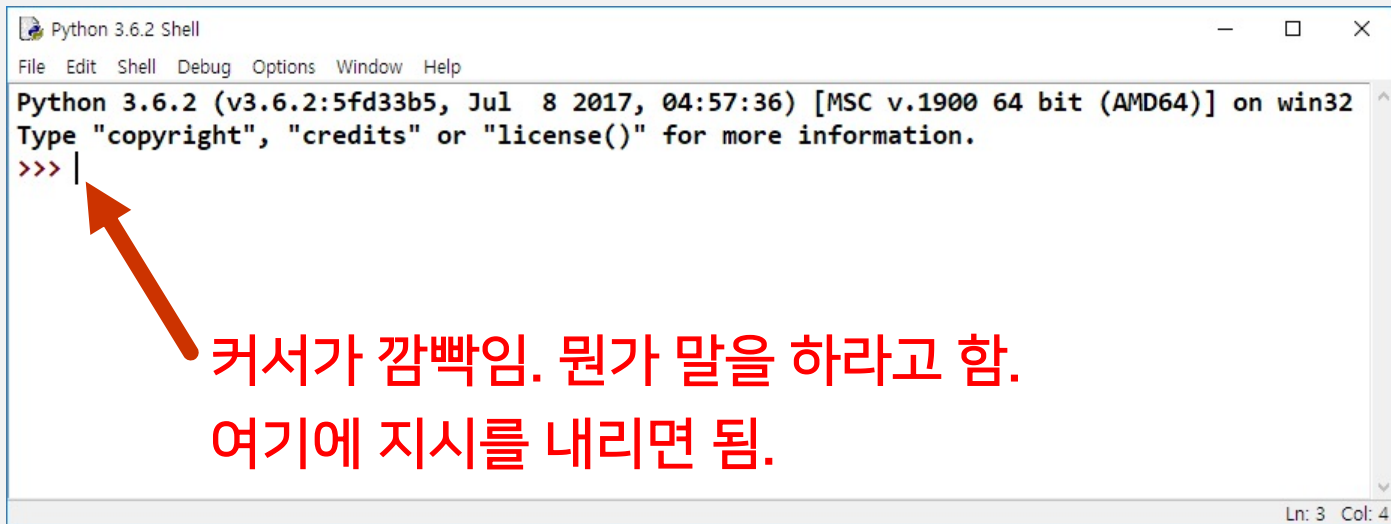
## Python Key Words

<b>False</b>	<b>class</b>	<b>return</b>	<b>is</b>	<b>finally</b>
<b>None</b>	<b>if</b>	<b>for</b>	<b>lambda</b>	<b>continue</b>
<b>True</b>	<b>def</b>	<b>from</b>	<b>while</b>	<b>nonlocal</b>
<b>and</b>	<b>del</b>	<b>global</b>	<b>not</b>	<b>with</b>
<b>as</b>	<b>elif</b>	<b>try</b>	<b>or</b>	<b>yield</b>
<b>assert</b>	<b>else</b>	<b>import</b>	<b>pass</b>	
<b>break</b>	<b>except</b>	<b>in</b>	<b>raise</b>	



# IDLE 실행 화면

- Python 언어로 지시하면, 이를 해석해서 일을 함.



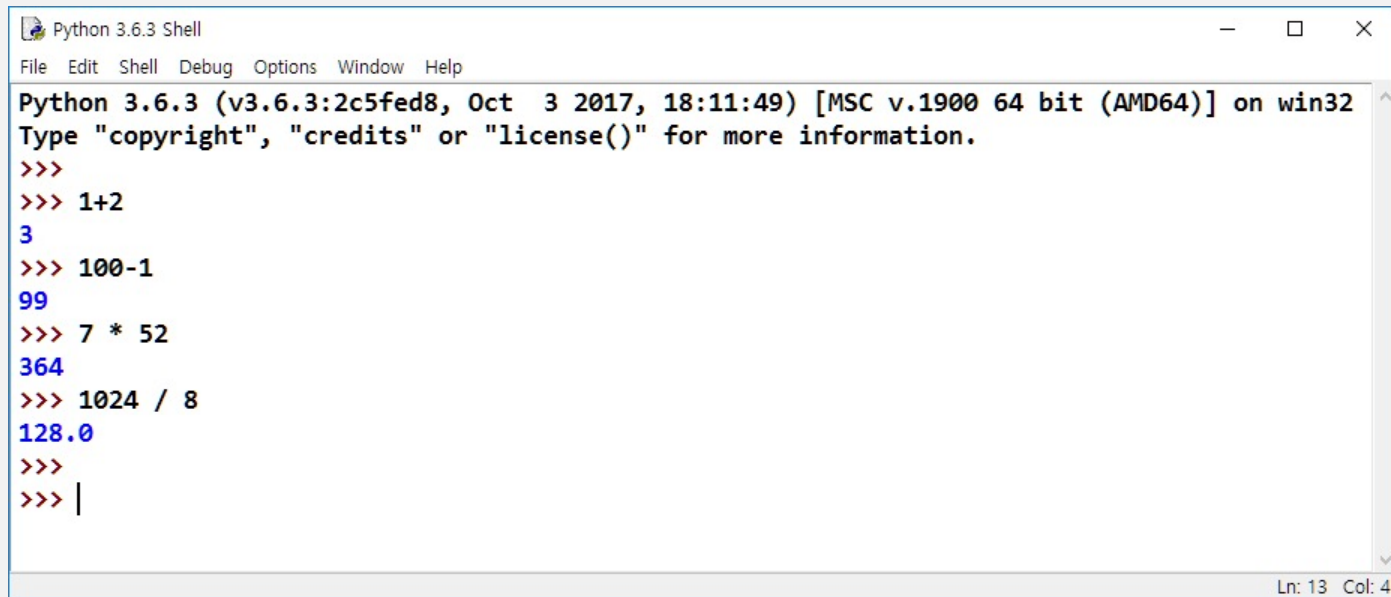
```
Python 3.6.2 Shell
File Edit Shell Debug Options Window Help
Python 3.6.2 (v3.6.2:5fd33b5, Jul 8 2017, 04:57:36) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> |
```

커서가 깜빡임. 뭔가 말을 하라고 함.  
여기에 지시를 내리면 됨.

Ln: 3 Col: 4

# 계산을 시켜보자.

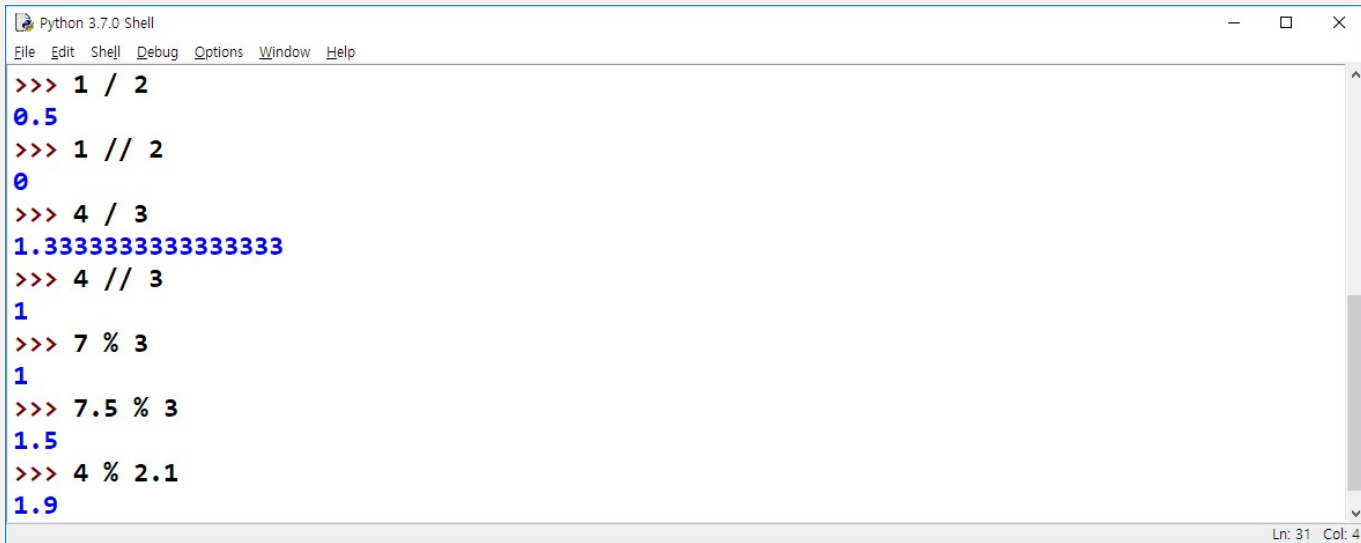
---



```
Python 3.6.3 Shell
File Edit Shell Debug Options Window Help
Python 3.6.3 (v3.6.3:2c5fed8, Oct 3 2017, 18:11:49) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
>>> 1+2
3
>>> 100-1
99
>>> 7 * 52
364
>>> 1024 / 8
128.0
>>>
>>> |
```

Ln: 13 Col: 4

# 나누기와 나머지 연산



```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
>>> 1 / 2
0.5
>>> 1 // 2
0
>>> 4 / 3
1.3333333333333333
>>> 4 // 3
1
>>> 7 % 3
1
>>> 7.5 % 3
1.5
>>> 4 % 2.1
1.9
Ln: 31 Col: 4
```

# 원의 넓이를 구해보자. 반지름이 3미터 이면?

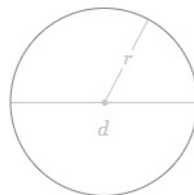
Circle

Solve for area ▾

$$A = \pi r^2$$

$r$  Radius

Enter value



```
Python 3.6.3 Shell
File Edit Shell Debug Options Window Help
>>>
>>>
>>>
>>>
>>>
>>> 3.141592653589793 * (3 * 3)
28.274333882308138
>>>
>>> 3.141592653589793 * 3 ** 2
28.274333882308138
>>>
>>> |
```

28.274328 평방미터

Ln: 77 Col: 4

# IDLE 에디팅 팁

---

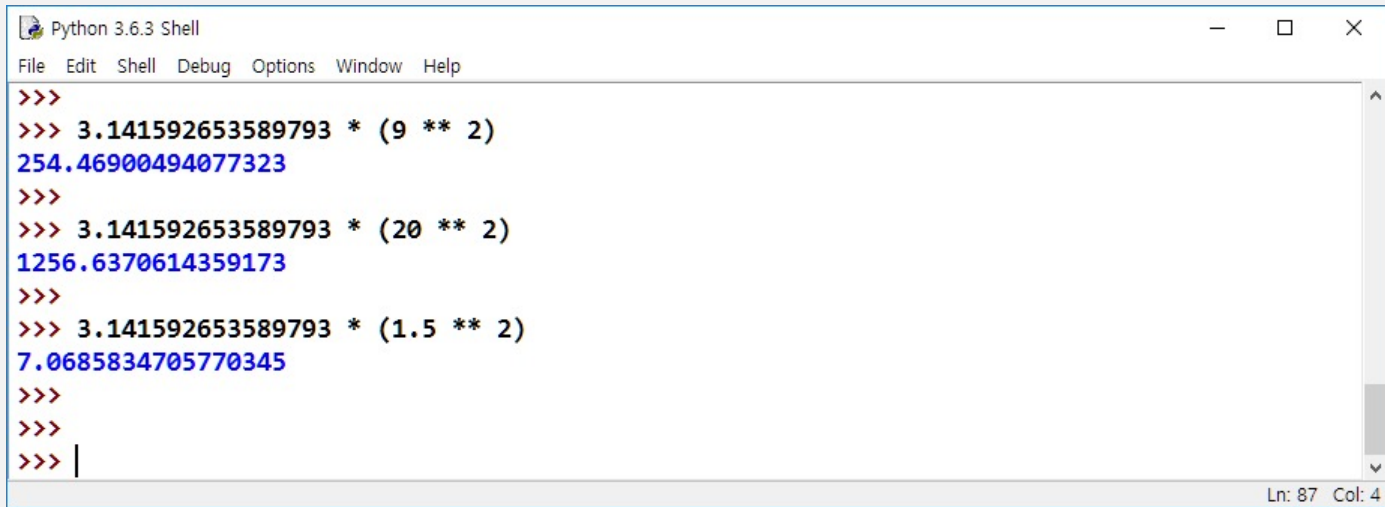
- 라인 단위로의 입력이 기본
- 이미 입력한 라인은 편집이 불가능
- 입력한 라인들은 내부 버퍼에 저장되어 있음.
- Alt + p 와 Alt + n 을 이용해서, 앞서 입력했던 라인들을 꺼내올 수 있음.

# 연산 기호

---

연산자	연산
+	덧셈
-	뺄셈
*	곱셈
/ 과 //	나눗셈
**	제곱
%	나머지

# 반지름이 9미터이면? 20 미터면? 1.5미터면?



```
Python 3.6.3 Shell
File Edit Shell Debug Options Window Help
>>>
>>> 3.141592653589793 * (9 ** 2)
254.46900494077323
>>>
>>> 3.141592653589793 * (20 ** 2)
1256.6370614359173
>>>
>>> 3.141592653589793 * (1.5 ** 2)
7.0685834705770345
>>>
>>>
>>> |
```

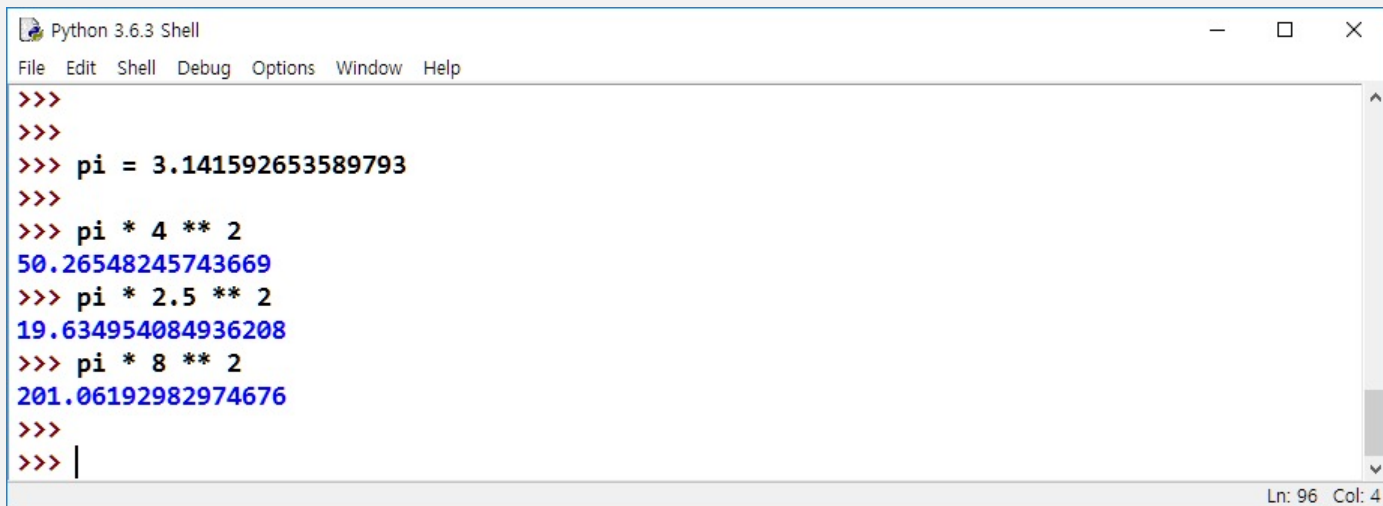
Ln: 87 Col: 4

슬슬 귀찮아지기 시작한다.

3.141592... 를 어디엔가 기록해놓고, 이걸 재사용하면 좋을 것 같은데...

# 변수(variable)

- 변수: 값을 저장해놓는 컴퓨터 메모리 안의 공간
- 변수는 해당되는 이름이 있다. 프로그래머가 이름을 지어야 함.
- 이름은 영문자와 숫자를 조합해서 씀. 단, 파이썬의 기본 단어는 쓰면 안됨.
- 사실, 변수의 값은 맘대로 언제든지 바꿀 수 있다.

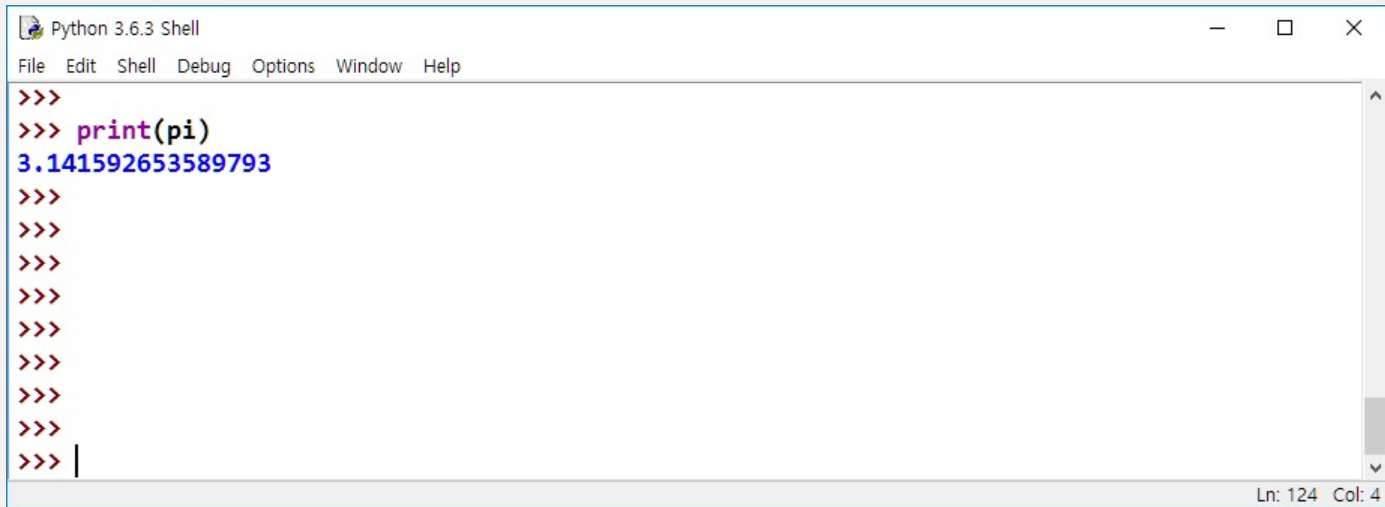
A screenshot of a Python 3.6.3 Shell window. The window has a title bar with the text 'Python 3.6.3 Shell' and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main area is a text editor showing a Python script. The script starts with three empty lines, followed by the assignment 'pi = 3.141592653589793'. Then, there are three lines of calculations: 'pi \* 4 \*\* 2' resulting in '50.26548245743669', 'pi \* 2.5 \*\* 2' resulting in '19.634954084936208', and 'pi \* 8 \*\* 2' resulting in '201.06192982974676'. The script ends with two empty lines and a cursor on the third line. The status bar at the bottom right shows 'Ln: 96 Col: 4'.

```
>>>
>>>
>>> pi = 3.141592653589793
>>>
>>> pi * 4 ** 2
50.26548245743669
>>> pi * 2.5 ** 2
19.634954084936208
>>> pi * 8 ** 2
201.06192982974676
>>>
>>> |
```



# print 함수를 이용하면, 변수의 값을 볼 수 있다.

---



A screenshot of a Python 3.6.3 Shell window. The window has a title bar with the text "Python 3.6.3 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 is a text editor with a white background. It contains the following text:  
>>>  
>>> print(pi)  
3.141592653589793  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>> |  
The output "3.141592653589793" is displayed in a blue font, while the prompt and code are in a red font. A vertical scrollbar is visible on the right side of the text area. At the bottom right of the window, the status bar shows "Ln: 124 Col: 4".

# 반지름 변수 r과 면적 변수 area 를 사용한 면적 계산



```
Python 3.6.3 Shell
File Edit Shell Debug Options Window Help
>>>
>>>
>>>
>>> r = 7
>>>
>>> area = pi * r ** 2
>>>
>>> print(area)
153.93804002589985
>>>
>>>
>>>
Ln: 133 Col: 4
```

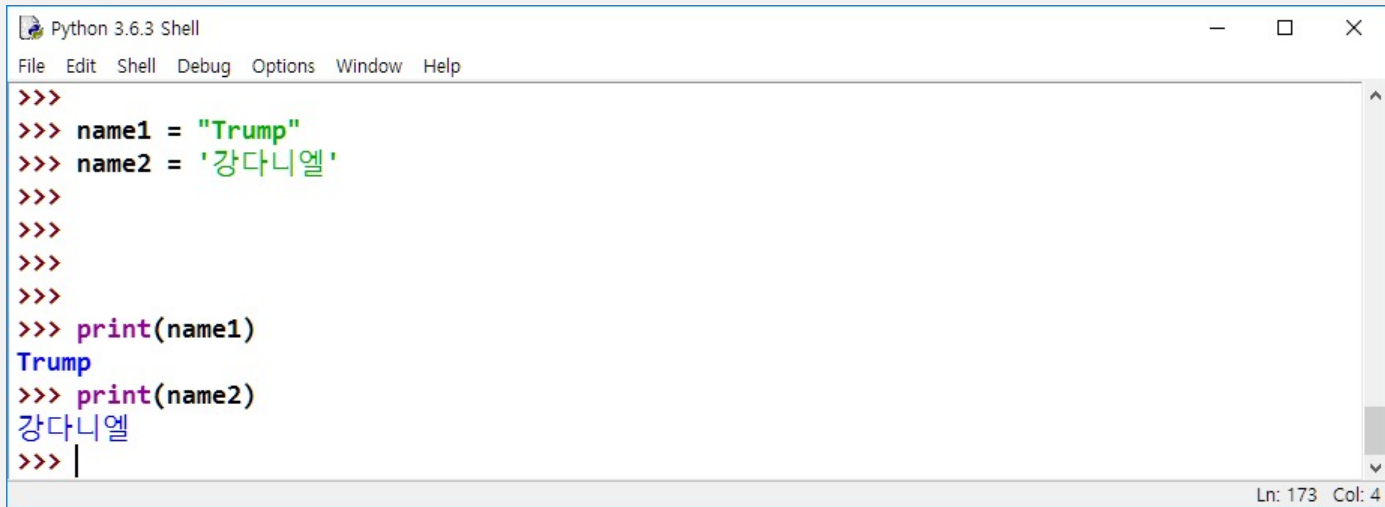
# 변수값을 바꿔서 사용 : r 에 11을 대입

A screenshot of a Python 3.6.3 Shell window. The window has a title bar with the text 'Python 3.6.3 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 of the window contains a series of command-line prompts and code. The first three prompts are empty. The fourth prompt is followed by the code 'r = 11'. The next two prompts are empty. The seventh prompt is followed by the code 'area = pi \* r \*\* 2'. The next two prompts are empty. The tenth prompt is followed by the code 'print(area)'. The output of this command is '380.132711084365'. The bottom status bar shows 'Ln: 139 Col: 4'.

```
>>>  
>>> r = 11  
>>>  
>>> area = pi * r ** 2  
>>>  
>>> print(area)  
380.132711084365  
>>>  
>>>  
>>>  
>>>  
>>>
```

Alt + p

# 변수에는 문자열(string)을 담을 수 있다.

A screenshot of a Python 3.6.3 Shell window. The window has a title bar with the text 'Python 3.6.3 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 of the window contains a Python REPL session. The prompt '>>>' is followed by two lines of code: 'name1 = "Trump"' and 'name2 = '강다니엘''. There are three blank lines, then 'print(name1)' is entered, followed by the output 'Trump' on the next line. Then 'print(name2)' is entered, followed by the output '강다니엘' on the next line. The prompt '>>>' is followed by a vertical bar '|'. The status bar at the bottom right shows 'Ln: 173 Col: 4'.

```
>>>
>>> name1 = "Trump"
>>> name2 = '강다니엘'
>>>
>>>
>>>
>>>
>>> print(name1)
Trump
>>> print(name2)
강다니엘
>>> |
```

문자열은 문자들의 집합임. 여러 개의 문자들을 나열한 것. 큰따옴표 또는 작은 따옴표로 감쌌. 한글, 영어 상관없음.

# 기본 자료형(Type)

---

- 변수에는 다양한 종류의 정보를 담을 수 있음.


order = 4    정수형 int

pi = 3.141592    실수형 float

name1 = "Trump"    문자열형 str  
name2 = 'Daehyun'

result = True    참거짓형 bool

# 자료형 파악 type



```
Python 3.6.3 Shell
File Edit Shell Debug Options Window Help
>>>
>>> order = 4
>>> pi = 3.141592
>>> name = "Trump"
>>>
>>> type(order)
<class 'int'>
>>> type(pi)
<class 'float'>
>>> type(name)
<class 'str'>
>>>
```

Ln: 199 Col: 4

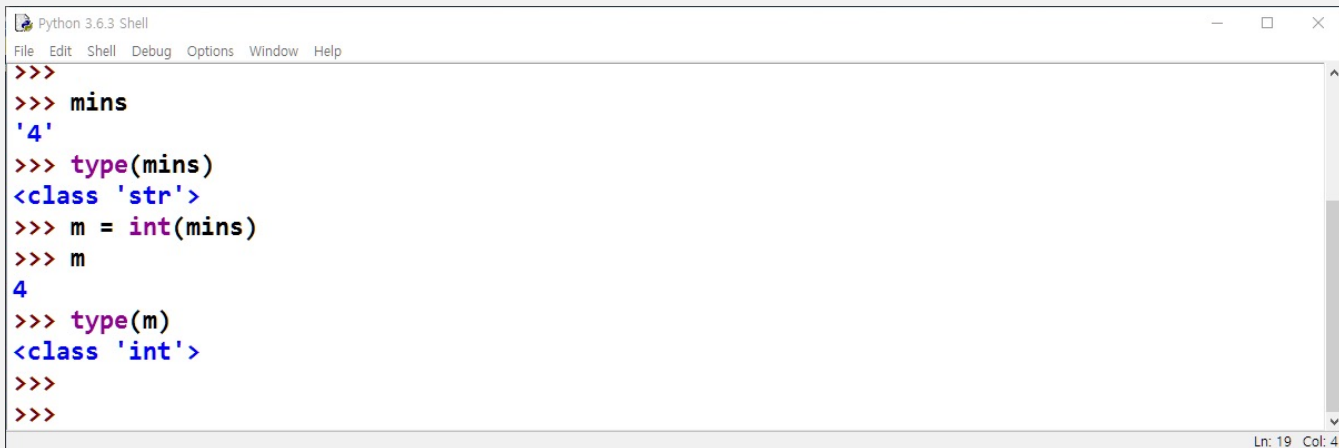
## 사용자로부터 입력 받기

- input 함수를 이용함.
- 사용자가 입력한 정보가 "문자열"로 되어 넘어옴.

[illegible]

# 자료형 변환

- mins의 값은 4가 아니고, '4'임. 즉, 정수가 아니고 문자열임.
- 이것을 정수로 바꾸기 위해서는 int() 라는 함수를 사용함.

A screenshot of a Python 3.6.3 Shell window. The window has a title bar with the Python logo and text 'Python 3.6.3 Shell'. Below the title bar is a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main area contains a Python REPL session. The user enters '>>>' followed by 'mins', which returns ''4''. Then they enter '>>> type(mins)', which returns '<class 'str'>'. Next, they enter '>>> m = int(mins)', which returns '>>> m'. Finally, they enter '>>> type(m)', which returns '<class 'int'>'. The session ends with three more '>>>' prompts. The status bar at the bottom right shows 'Ln: 19 Col: 4'.

```
>>>
>>> mins
'4'
>>> type(mins)
<class 'str'>
>>> m = int(mins)
>>> m
4
>>> type(m)
<class 'int'>
>>>
>>>
```



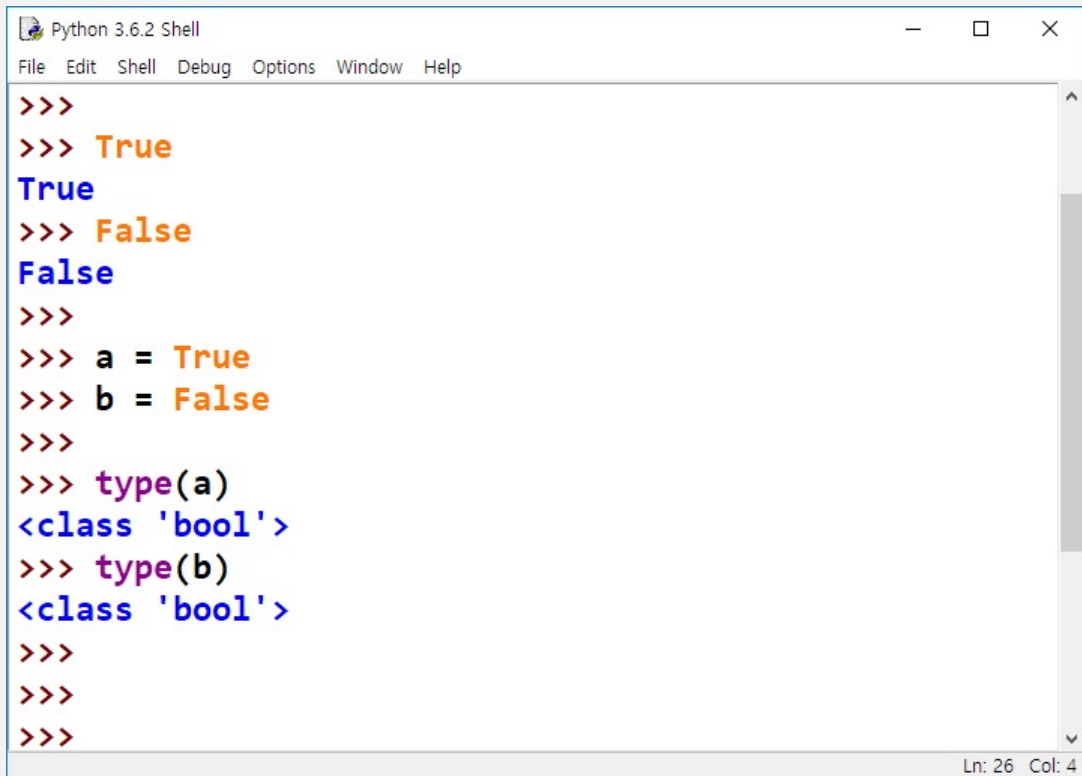
# 자료형 변환

---

```
>>> str(0)
'0'
>>> str(-3.14)
'-3.14'
>>> int('42')
42
>>> int(1.25)
1
>>> float('3.14')
3.14
>>> float(10)
10.0
```

# 자료형: bool

- 참(True), 또는 거짓(False)을 나타내는데 사용되는 자료형



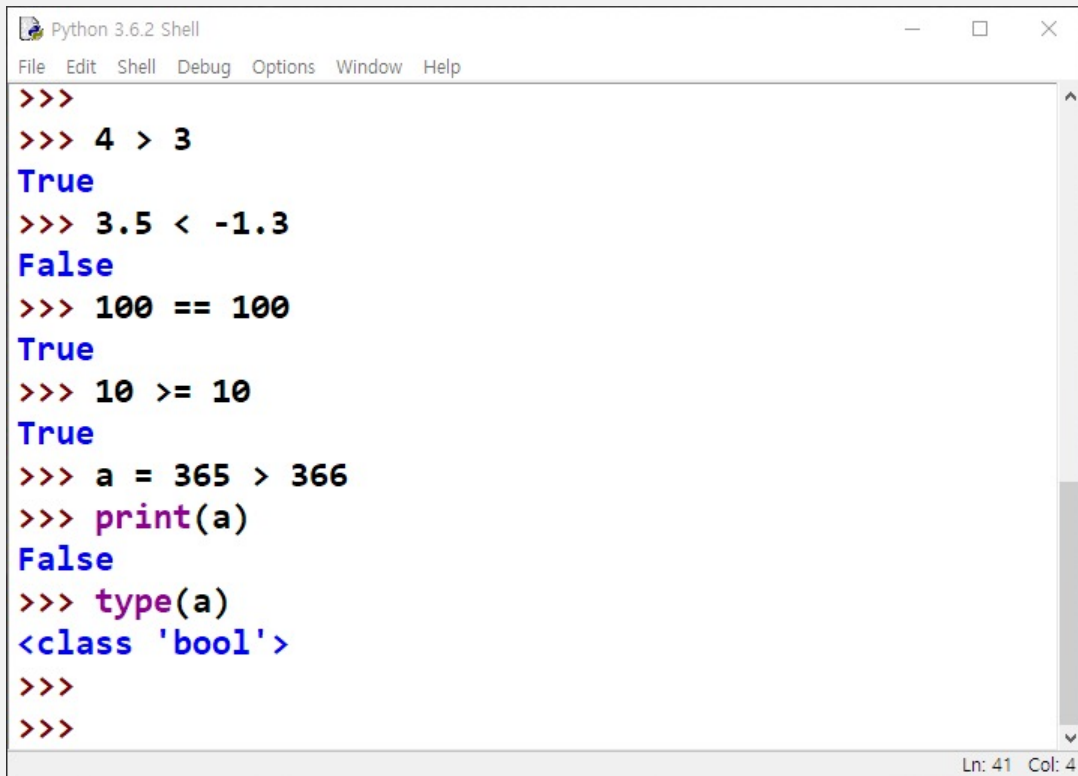
```
Python 3.6.2 Shell
File Edit Shell Debug Options Window Help

>>>
>>> True
True
>>> False
False
>>>
>>> a = True
>>> b = False
>>>
>>> type(a)
<class 'bool'>
>>> type(b)
<class 'bool'>
>>>
>>>
>>>
```

Ln: 26 Col: 4

# 비교 연산(Comparison Operation)

- 두개의 값의 대소, 동일 등을 확인하는 계산.
- 결과는 참(True) 또는 거짓(False)임.

A screenshot of a Python 3.6.2 Shell window. The window has a title bar with the text 'Python 3.6.2 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: '>>>', '>>> 4 > 3', '>>> 3.5 < -1.3', '>>> 100 == 100', '>>> 10 >= 10', '>>> a = 365 > 366', '>>> print(a)', and '>>> type(a)'. The outputs are: 'True', 'False', 'True', 'True', 'False', and '<class \'bool\'>'. The window also has a status bar at the bottom right showing 'Ln: 41 Col: 4'.

```
Python 3.6.2 Shell
File Edit Shell Debug Options Window Help
>>>
>>> 4 > 3
True
>>> 3.5 < -1.3
False
>>> 100 == 100
True
>>> 10 >= 10
True
>>> a = 365 > 366
>>> print(a)
False
>>> type(a)
<class 'bool'>
>>>
>>>
Ln: 41 Col: 4
```

# 비교 연산 기호

---

기호	뜻
<	작다
<=	작거나 같다
==	같다
>=	크거나 같다
>	크다
!=	다르다

# 문자열의 비교



```
Python 3.6.3 Shell
File Edit Shell Debug Options Window Help

>>>
>>>
>>> "KOREA" == "korea"
False
>>>
>>>
>>> 'abcdefg' == "abcdefg"
True
>>>
>>> |
```

Ln: 327 Col: 4

# 문자열 str 의 다양한 연산

```
Python 3.7.0 Shell - C:/Users/dustinlee/Desktop/test.py (3.7.0)
File Edit Shell Debug Options Window Help
>>>
>>> first = "Daehyun"
>>> last = "Lee"
>>> name = first + " " + last
>>> name
'Daehyun Lee'
>>> print(name)
Daehyun Lee
>>>
>>> name * 2
'Daehyun LeeDaehyun Lee'
>>> name * 3
'Daehyun LeeDaehyun LeeDaehyun Lee'
>>>
>>> name[0]
'D'
>>> name[2]
'e'
>>> name[-1]
'e'
>>> name[-2]
'e'
```

Ln: 157 Col: 4

# Slice(슬라이스)

## ■ 문자열의 일부분을 잘라내는 기법

□ `name[ start : stop : step ]`

```
>>> title = "Python 2D Game Programming"
>>> title[0:6]
'Python'
>>> title[7:9]
'2D'
>>> title[10:14]
'Game'
>>> title[:6]
'Python'
>>> title[-11:]
'Programming'
>>> title[::2]
'Pto DGm rgamn'
>>> title[::-1]
'gnimmargorP emaG D2 nohtyP'
```

# List

---

```
>>> twice = ['momo', 'sana', 'zwi', 'nayun', 'dahyun']
>>> black_pink = ['jisu', 'jeni', 'rose', 'risa']
>>> twice
['momo', 'sana', 'zwi', 'nayun', 'dahyun']
>>> twice.append('jihyo')
>>> twice
['momo', 'sana', 'zwi', 'nayun', 'dahyun', 'jihyo']
>>> twice.sort()
>>> twice
['dahyun', 'jihyo', 'momo', 'nayun', 'sana', 'zwi']
>>> len(twice)
6
>>> unite = twice + black_pink
>>> unite
['dahyun', 'jihyo', 'momo', 'nayun', 'sana', 'zwi', 'jisu', 'jeni', 'rose', 'risa']
>>> unite.remove('momo')
>>> unite
['dahyun', 'jihyo', 'nayun', 'sana', 'zwi', 'jisu', 'jeni', 'rose', 'risa']
```



# List 에서 Slice 가 적용됨.

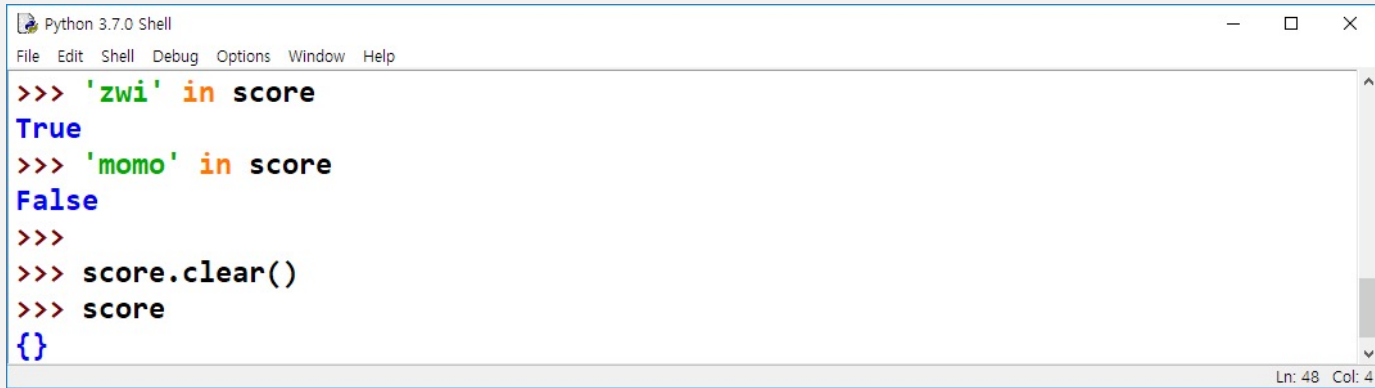
---

```
>>> unite[0]
'dahyun'
>>> unite[-1]
'risa'
>>> unite[:3]
['dahyun', 'jihyo', 'nayun']
>>> unite[-3:]
['jeni', 'rose', 'risa']
```

# Dictionary

```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
>>> score = { 'momo' : 80, 'zwi' : 85, 'sana' : 98 }
>>> type(score)
<class 'dict'>
>>> score['momo']
80
>>> score['nayun']
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    score['nayun']
KeyError: 'nayun'
>>> score['nayun'] = 100
>>> score
{'momo': 80, 'zwi': 85, 'sana': 98, 'nayun': 100}
>>> del score['momo']
>>> score
{'zwi': 85, 'sana': 98, 'nayun': 100}
>>> score.keys()
dict_keys(['zwi', 'sana', 'nayun'])
>>> score.values()
dict_values([85, 98, 100])
>>>
```

Ln: 12 Col: 0



A screenshot of a Python 3.7.0 Shell window. The window has a title bar with the text "Python 3.7.0 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 Python REPL session with the following code and output:

```
>>> 'zwi' in score
True
>>> 'momo' in score
False
>>>
>>> score.clear()
>>> score
{}
```

The status bar at the bottom right of the window displays "Ln: 48 Col: 4".

# Tuple

- 여러 개의 값을 동시에 관리. 리스트와 유사.
- 하지만, 기본적으로 값을 바꿀 수는 없음. ==> 프로그램 중 변경이 되지 않는 값들의 모음이 필요할 때 사용하면 됨.

```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
>>>
>>> t1 = (1,2,3)
>>> t2 = (1, )
>>> t3 = ()
>>> t4 = 1,2,3,4
>>> t4
(1, 2, 3, 4)
>>> type(t4)
<class 'tuple'>
>>> t5 = (1, 'a', "park", (1, 2))
>>> t1[1:]
(2, 3)
>>> t1 + t5
(1, 2, 3, 1, 'a', 'park', (1, 2))
>>> t4 * t4
Traceback (most recent call last):
  File "<pyshell#15>", line 1, in <module>
    t4 * t4
TypeError: can't multiply sequence by non-int of type 'tuple'
>>> t4 * 2
(1, 2, 3, 4, 1, 2, 3, 4)
>>> |
```

Ln: 37 Col: 4

# set

- 집합 자료형, 리스트와 달리, 중복을 허용하지 않고, 순서가 없음.

```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
>>> s1 = {1,2,3}
>>> type(s1)
<class 'set'>
>>> s1 = {1,2,2,4}
>>> s1
{1, 2, 4}
>>> l1 = [1,2,2,2,2,3,3,3,3,5,5,5,5,5]
>>> s1 = set(l1)
>>> s1
{1, 2, 3, 5}
>>> s2 = {3,5,6,7}
>>> s1 + s2
Traceback (most recent call last):
  File "<pyshell#36>", line 1, in <module>
    s1 + s2
TypeError: unsupported operand type(s) for +: 'set' and 'set'
>>> s1 | s2
{1, 2, 3, 5, 6, 7}
>>> s1 & s2
{3, 5}
>>> s2 - s1
{6, 7}
>>> s1 - s2
{1, 2}
>>> s1.add(8)
>>> s1
{1, 2, 3, 5, 8}
>>> s2.remove(6)
>>> s2
{3, 5, 7}
```

Ln: 86 Col: 4

# Complex Data Type

---

## ■ List – list

- 순서가 있는, 중복을 허용하는 데이터들의 집합.
- 원하는 데이터를 찾기 위해, 순서 index 를 이용.

[ val1, val2, ... ]

## ■ Dictionary – dict

- 검색을 위한 키를 갖는 데이터들의 집합
- key – value 쌍 들의 집합

{ key1: val1, key2: val2, ... }

## ■ Tuple – tuple

- 순서가 있는, 중복을 허용하는 데이터들의 집합
- 다만, 데이터값을 변경하는 것은 불가

( val1, val2, ... )

## ■ Set – set

- 중복을 허용하지 않는, 순서에 상관없는 데이터들의 집합

{ val1, val2, ... }