



소프트웨어입문설계

Hosung Jo
Hanyang University

소프트웨어입문설계 소개

■ 과목개요

구분	내용
교과목명	소프트웨어입문설계
과목구분	SCSC
학점	3학점
강의시간	목요일, 월요일, 수요일 17:00-19:00
강의장소	목 510, 수 510, 월 609
강의형태	이론 - 실습

소프트웨어입문설계 소개

■ 평가기준

구분	내용
평가구분	절대평가
출석	15%
중간고사	30%
기말고사	40%
과제	15%

소프트웨어입문설계 소개

■ 강사

- 조호성

- 한양대학교 공학박사 [컴퓨터이론, 알고리즘, 정보보호]
- ustog@hanyang.ac.kr, [소프트웨어입문설계]를 말머리로 메일 발송

Introduction to Python

■ Python

- Easier to learn than C.
- Serious programming language.
- Many expert programmers use Python in their work.

■ Python programming

- Need to install software called the **Python interpreter**.
 - **Interpreter**
: a program that understands the instructions that you'll write in the Python language.

Introduction

- **Installing Python on a Windows Machine**
- **Starting the Python Interpreter**
- **Evaluating Expressions**
- **Storing values in variables**
- **Strings**
- **Write the first program**
 - "Hello world"
 - "My Favorite Stuff"

Installing Python

■ Download Python

- Official website: <http://www.python.org>
- Download Python: [Python 2.7.2 Windows Installer](#)

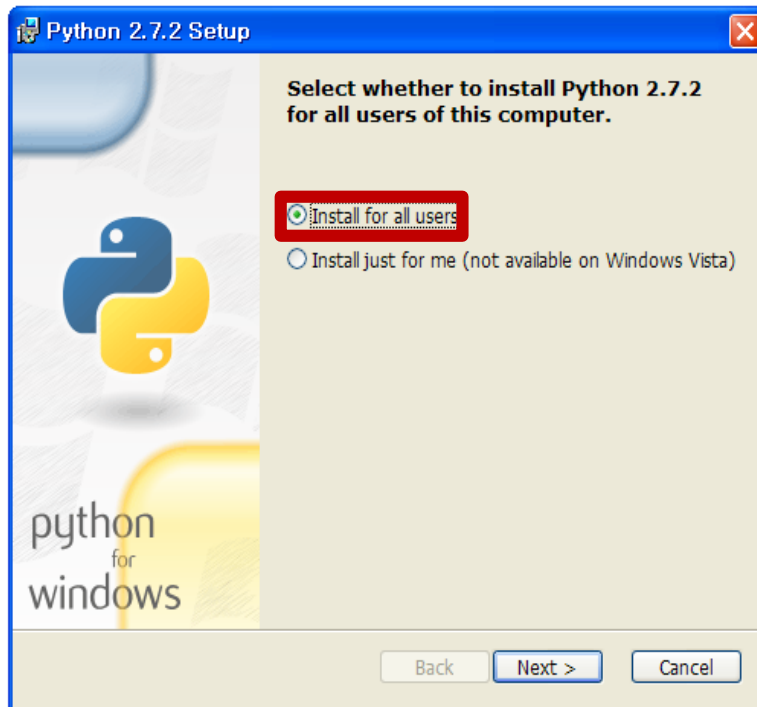


For the MD5 checksums and OpenPGP signatures, look at the [detailed Python 2.7.2 page](#):

- [Python 2.7.2 Windows Installer](#) (Windows binary -- does not include source)
- [Python 2.7.2 Windows X86-64 Installer](#) (Windows AMD64 / Intel 64 / X86-64 binary [1])
- [Python 2.7.2 Mac OS X 64-bit/32-bit x86-64/i386 Installer](#) (for Mac OS X 10.6 and 10.7)
- [Python 2.7.2 Mac OS X 32-bit i386/PPC Installer](#) (for Mac OS X 10.3 through 10.6 [2])
- [Python 2.7.2 compressed source tarball](#) (for Linux, Unix or Mac OS X)
- [Python 2.7.2 bzipipped source tarball](#) (for Linux, Unix or Mac OS X, more compressed)

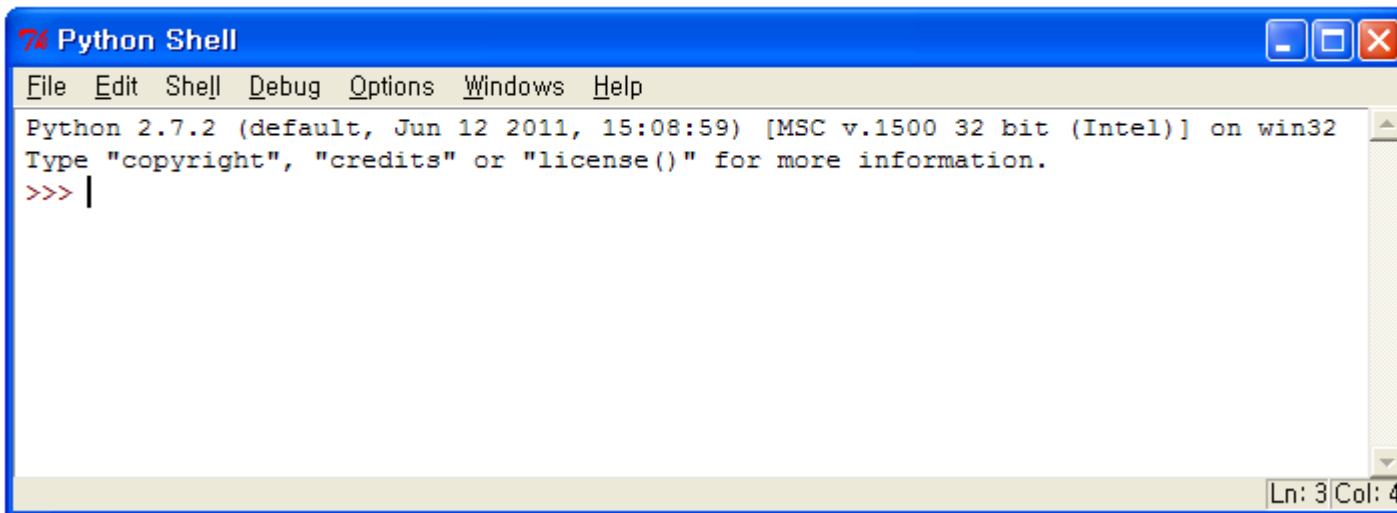
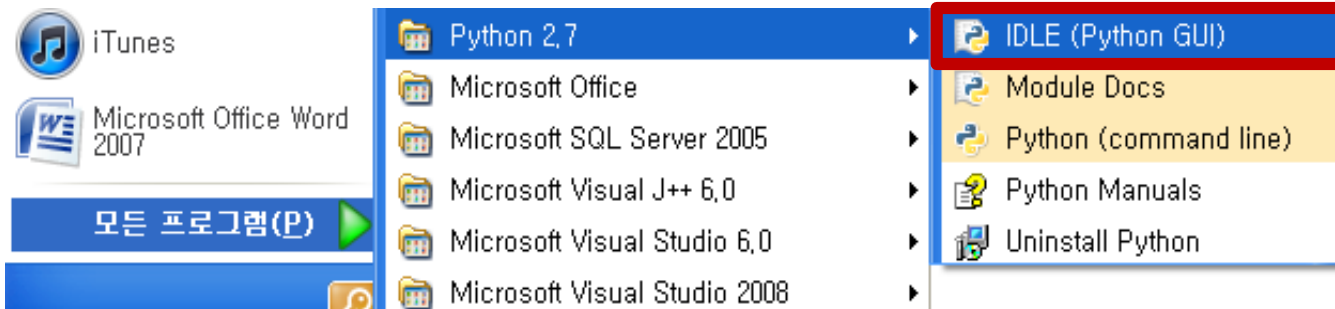
Installing Python

■ Install Python



Starting the Python Interpreter

■ IDLE (Python GUI) Program



Starting the Python Interpreter

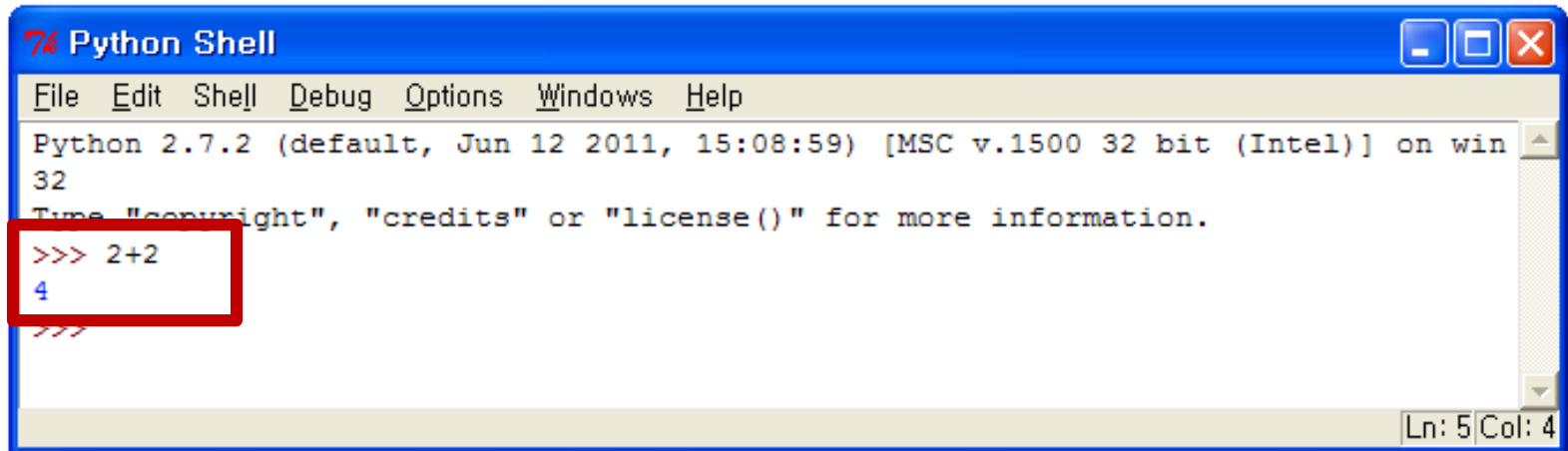
■ IDLE

- Interactive DeveLopment Environment
- Program that **helps us** type in our own programs and games.
- **Interactive shell**
 - First run IDLE window.
 - Can work just like a calculator.

Starting the Python Interpreter

■ Some Simple Math Stuff

- Type **2+2** into the shell and press the **Enter** key.
- Computer should respond with the number **4**.
: the sum of **2+2**



A screenshot of a Windows-style application window titled "Python Shell". The window has a blue title bar with standard minimize, maximize, and close buttons. Below the title bar is a menu bar with options: File, Edit, Shell, Debug, Options, Windows, and Help. The main text area shows the following text: "Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win32", "Type \"copyright\", \"credits\" or \"license()\" for more information.", and a prompt ">>>". The user has entered "2+2" on the line following the prompt, and the shell has responded with "4" on the next line. A red rectangular box highlights the input "2+2" and the output "4". At the bottom right of the window, a status bar shows "Ln: 5 Col: 4".

```
Python Shell
File Edit Shell Debug Options Windows Help
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> 2+2
4
>>>
```

Starting the Python Interpreter

■ Some Simple Math Stuff

- The various math operators in Python.

2+2	Addition
2-2	Subtraction
2*2	Multiplication
2/2	Division

- +, -, *, and / are called **operators**.
- * sign is called an **asterisk**.

Starting the Python Interpreter

■ Integers and Floating Point Numbers

- **Integers**

- whole numbers (like 4, 0, and 99)

- **Floating point numbers**

- numbers with a decimal point (like 5.0)

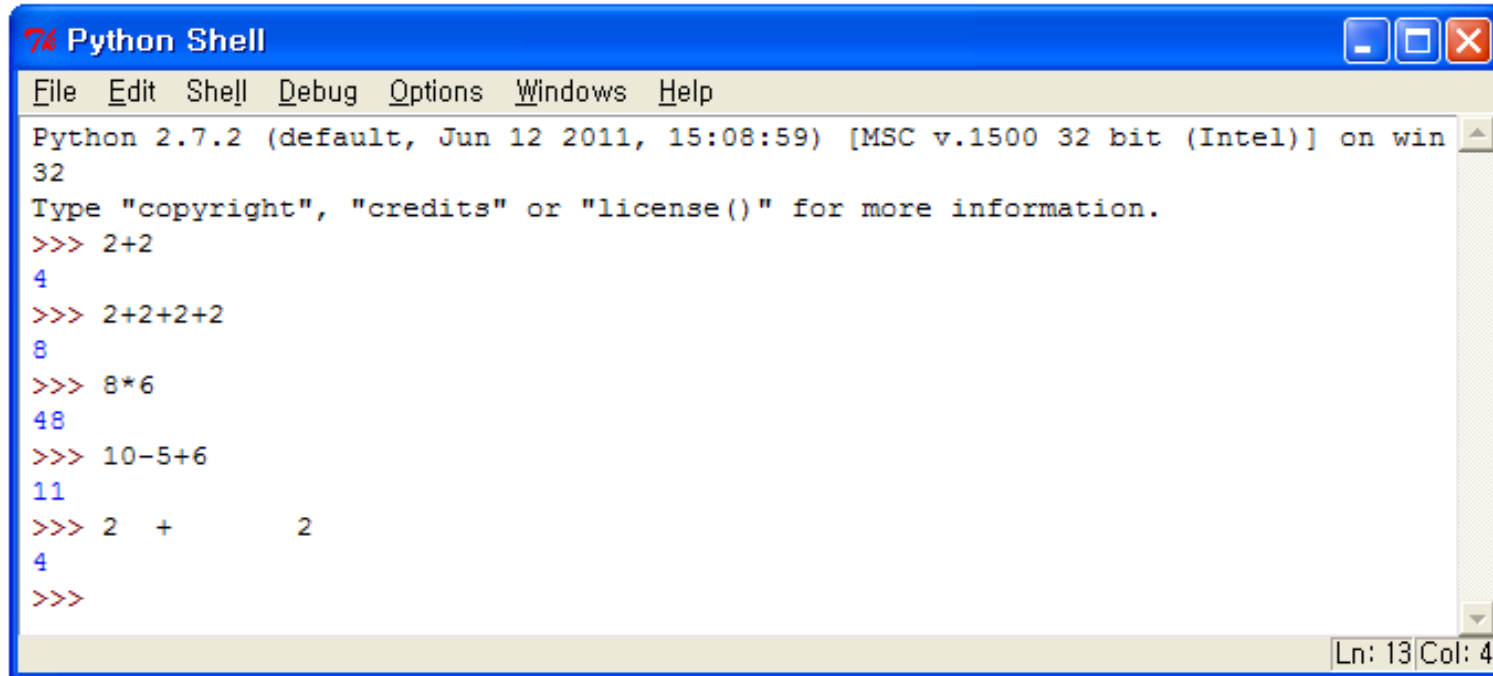
- **In Python**

- the number 5 is an **integer**
- but if we wrote it as 5.0 it would **not be an integer.**



Evaluating Expressions

■ Expressions



```
Python Shell
File Edit Shell Debug Options Windows Help
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> 2+2
4
>>> 2+2+2+2
8
>>> 8*6
48
>>> 10-5+6
11
>>> 2 +      2
4
>>>
```

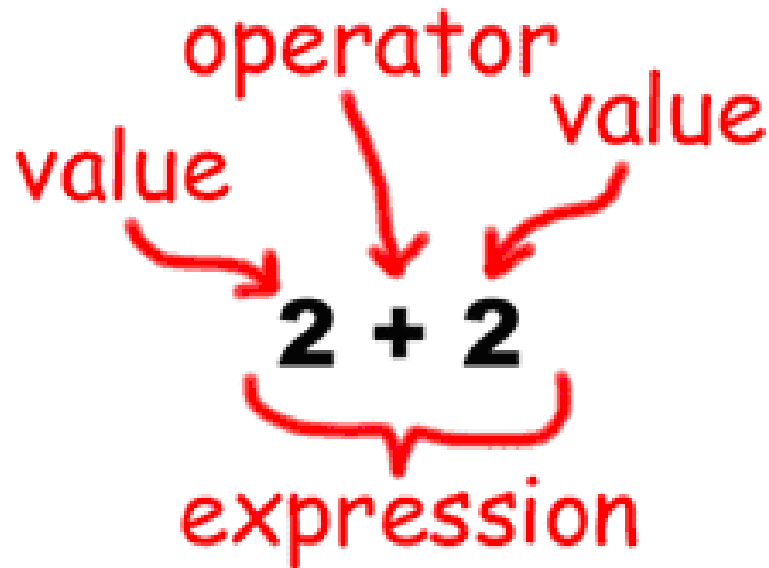
The screenshot shows a Python Shell window with a blue title bar and standard Windows window controls. The menu bar includes File, Edit, Shell, Debug, Options, Windows, and Help. The main text area displays the Python version (2.7.2) and the prompt 'Type "copyright", "credits" or "license()" for more information.' followed by several arithmetic expressions and their results: 2+2 results in 4, 2+2+2+2 results in 8, 8*6 results in 48, 10-5+6 results in 11, and 2 + 2 results in 4. The status bar at the bottom right shows 'Ln: 13 Col: 4'.

- These math problems are called **expressions**.
- These integers are also called **values**.

Evaluating Expressions

■ Expressions

- An expression is made up of **values** and **operators**.



Evaluating Expressions

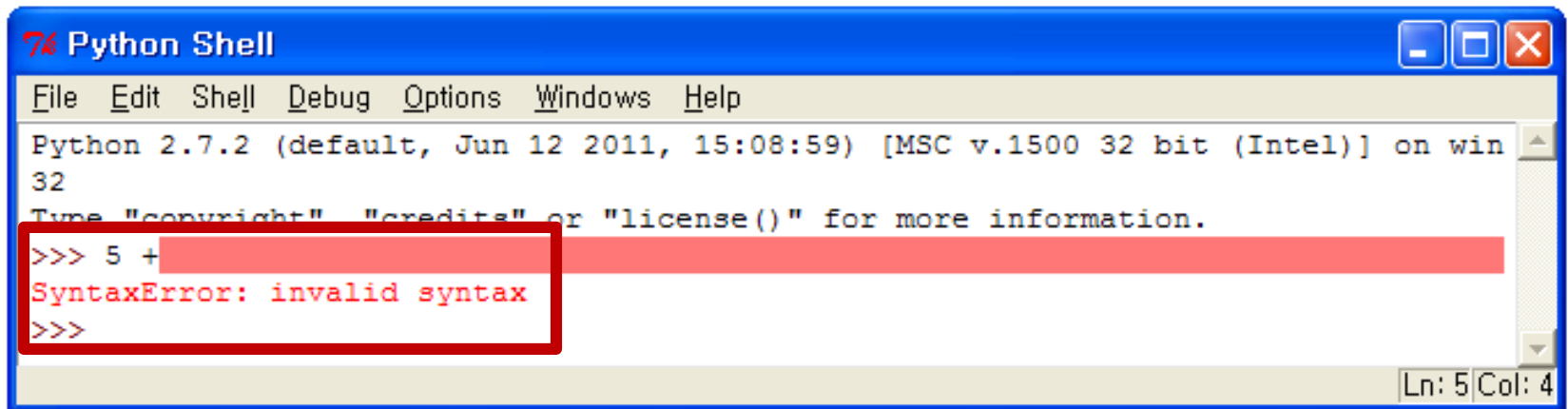
■ Evaluated the expression

- computer solves the **expression** $10 + 5$ and gets the **value** 15.
- The expressions $10 + 5$ and $10 + 3 + 2$
 - have the **same value**.
 - they both **evaluate to 15**.
- Single values are considered expressions.
 - The expression 15 evaluates to the value 15.

Evaluating Expressions

■ However,

- If you **just type 5 +**, you will get an **error message**.



The screenshot shows a Python Shell window with a blue title bar and standard Windows window controls. The menu bar includes File, Edit, Shell, Debug, Options, Windows, and Help. The main text area displays the Python version and environment information: "Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win32". Below this, it says "Type 'copyright', 'credits' or 'license()' for more information." The prompt ">>>" is followed by "5 +", which is highlighted with a red rectangular box. Below the input, the error message "SyntaxError: invalid syntax" is displayed in red text. The prompt ">>>" appears again on the next line. The status bar at the bottom right shows "Ln: 5 Col: 4".

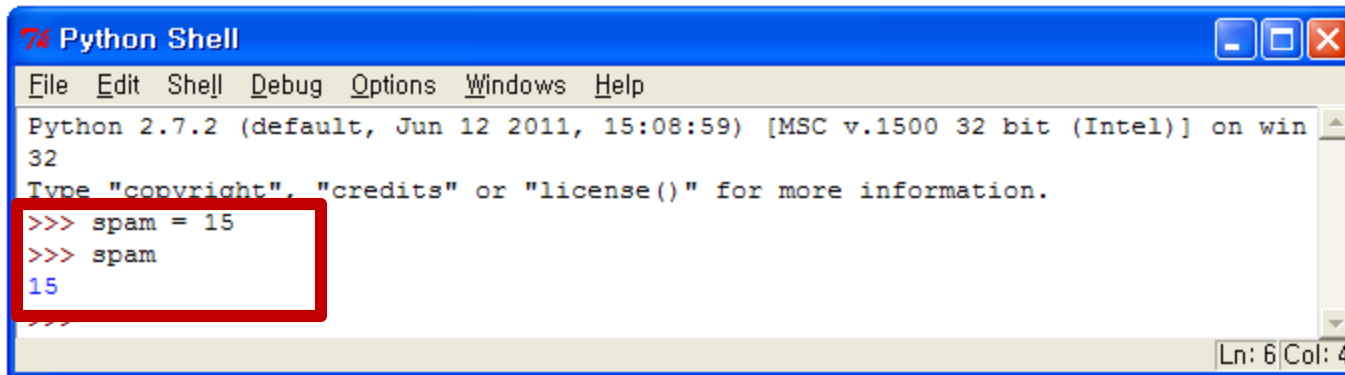
```
Python Shell
File Edit Shell Debug Options Windows Help
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> 5 +
SyntaxError: invalid syntax
>>>
```

- because **5 +** is **not an expression**.
- expressions have values connected by operators.
- the **+** **operator** always expects to **connect two things** in Python.

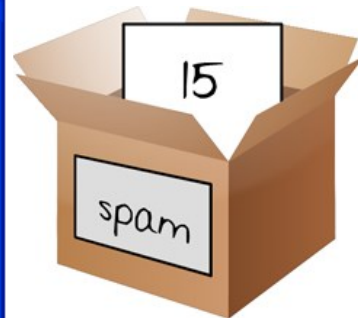
Storing Values in Variables

■ Variables

- Variables like a box that can hold values.
- Can **store values** in variables.
 - with the **= sign** (called the **assignment operator**)
- For example, **to store the value 15** in a variable named **“spam”**.



A screenshot of a Python Shell window titled "Python Shell". The window has a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Windows", and "Help". The main text area shows the following content: "Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win 32", "Type 'copyright', 'credits' or 'license()' for more information.", and a red rectangular box highlighting the code: ">>> spam = 15", ">>> spam", and "15". The status bar at the bottom right shows "Ln: 6 | Col: 4".



Storing Values in Variables



■ Quiz

```
>>> spam = 15
>>> spam + 5
```

```
>>> spam = 15
>>> spam + 5
```

```
>>> spam = 3
>>> spam + 5
```

```
>>> spam = 5 + 7
>>> spam
```

```
>>> spam = 15
>>> spam + spam
```

```
>>> spam - spam
```

Storing Values in Variables



■ Quiz

```
>>> spam = 15
>>> spam + 5
20
```

```
>>> spam = 15
>>> spam + 5
20
>>> spam = 3
>>> spam + 5
8
```

```
>>> spam = 5 + 7
>>> spam
12
>>> spam = 15
>>> spam + spam
30
>>> spam - spam
0
```

Storing Values in Variables



■ Quiz

```
>>> spam = 15
>>> spam = spam + 5
>>> spam
```

```
>>> spam = 15
>>> spam = spam + 5
>>> spam = spam + 5
>>> spam = spam + 5
>>> spam
```

Storing Values in Variables



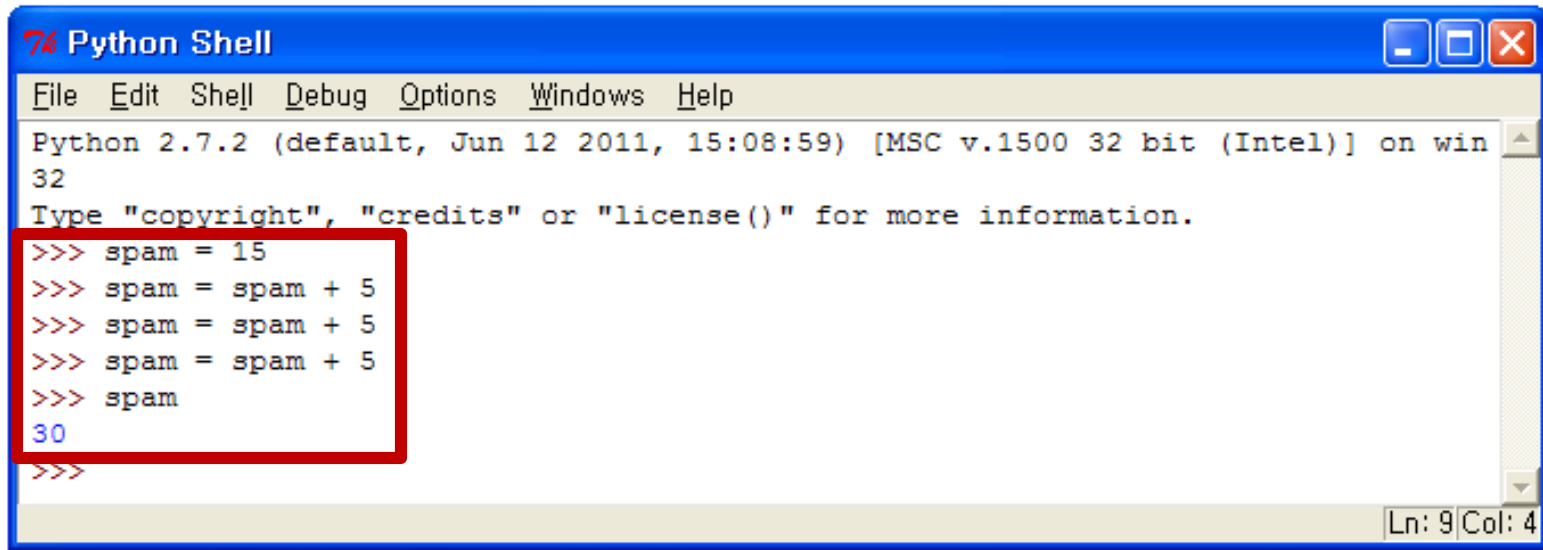
■ Quiz

```
>>> spam = 15
>>> spam = spam + 5
>>> spam
20
```

```
>>> spam = 15
>>> spam = spam + 5
>>> spam = spam + 5
>>> spam = spam + 5
>>> spam
30
```

Storing Values in Variables

■ Write Expressions with Variables



The screenshot shows a Python Shell window with a blue title bar and standard Windows window controls. The menu bar includes File, Edit, Shell, Debug, Options, Windows, and Help. The text area displays the following content:

```
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> spam = 15
>>> spam = spam + 5
>>> spam = spam + 5
>>> spam = spam + 5
>>> spam
30
>>>
```

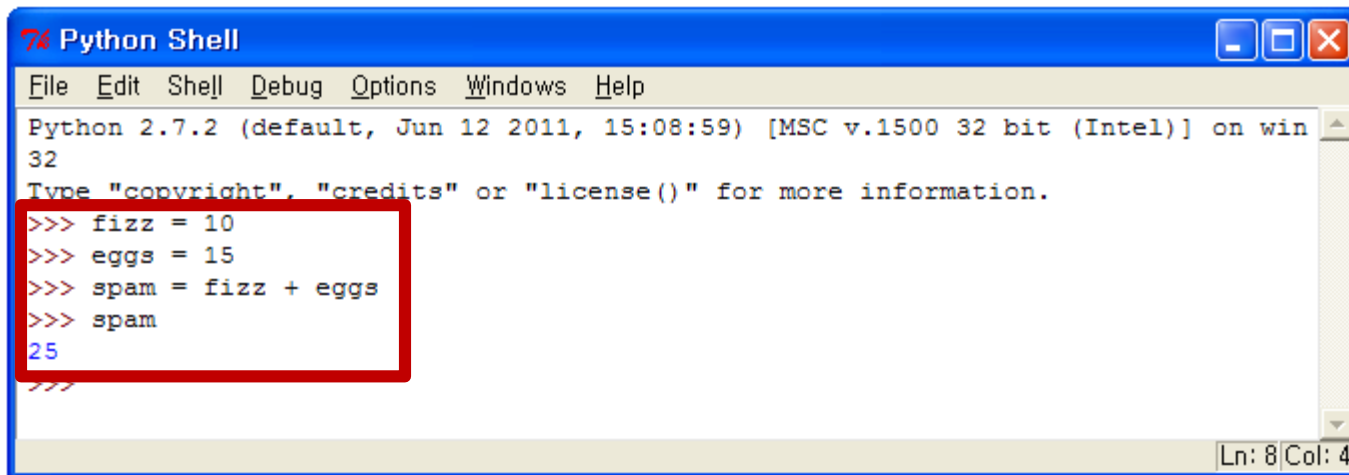
A red rectangular box highlights the five lines of code starting from `>>> spam = 15` down to the final `>>>` prompt. The output `30` is displayed in blue text. The status bar at the bottom right shows `Ln: 9|Col: 4`.

- The value of **15** was **overwritten**.

Storing Values in Variables

■ Using More Than One Variable

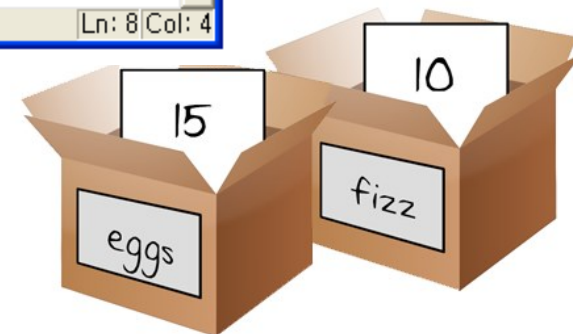
- Often we'll need to use **multiple variables**.
 - The "**fizz**" and "**eggs**" variables have values stored in them.



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

```
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> fizz = 10
>>> eggs = 15
>>> spam = fizz + eggs
>>> spam
25
>>>
```

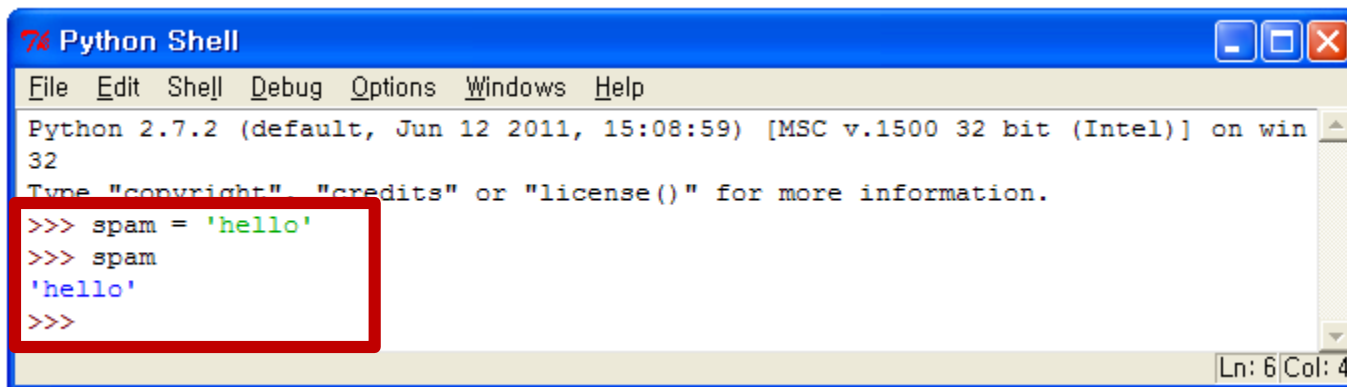
The code is enclosed in a red rectangular box. The status bar at the bottom right indicates "Ln: 8 Col: 4".



Strings

■ Strings

- **Little chunks of text.**
- Can store string values inside variables.
- Put them in between **two single quotes (')**.



A screenshot of a Python Shell window titled "Python Shell". The window has a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Windows", and "Help". The main text area shows the following text: "Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win32" and "Type 'copyright', 'credits' or 'license()' for more information." Below this, a red rectangle highlights the following code and its output:

```
>>> spam = 'hello'
>>> spam
'hello'
>>>
```

 The status bar at the bottom right shows "Ln: 6 | Col: 4".

Strings

■ Strings

- Strings can have **spaces** and **numbers** as well.
- Examples of strings

```
'hello'
```

```
'Hi there!'
```

```
'KITTENS'
```

```
'7 apples, 14 oranges, 3 lemons'
```

```
'Anything not pertaining to elephants is irrelephant.'
```

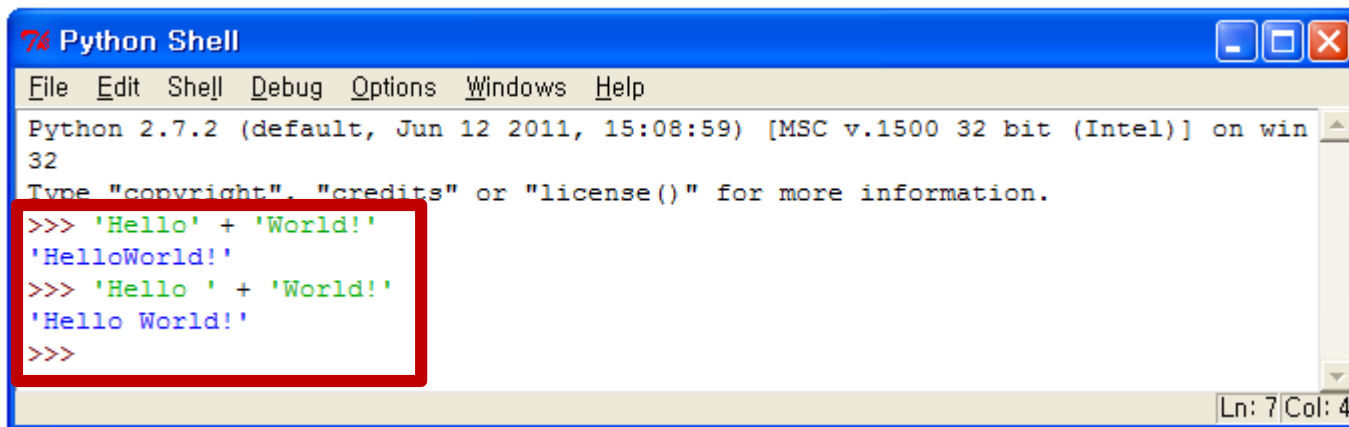
```
'A long time ago in a galaxy far, far away...'
```

```
'O*&#wY%*&OCfsdYO*&gfC%YO*&%3yc8r2'
```

Strings

■ Strings Concatenation

- Can **add one string** to the end of another by **using the + operator**.
- **Put a space** at the end of the 'Hello' string.

A screenshot of a Python Shell window titled "Python Shell". The window has a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Windows", and "Help". Below the menu bar, it says "Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win32". The main text area shows the following code and output:

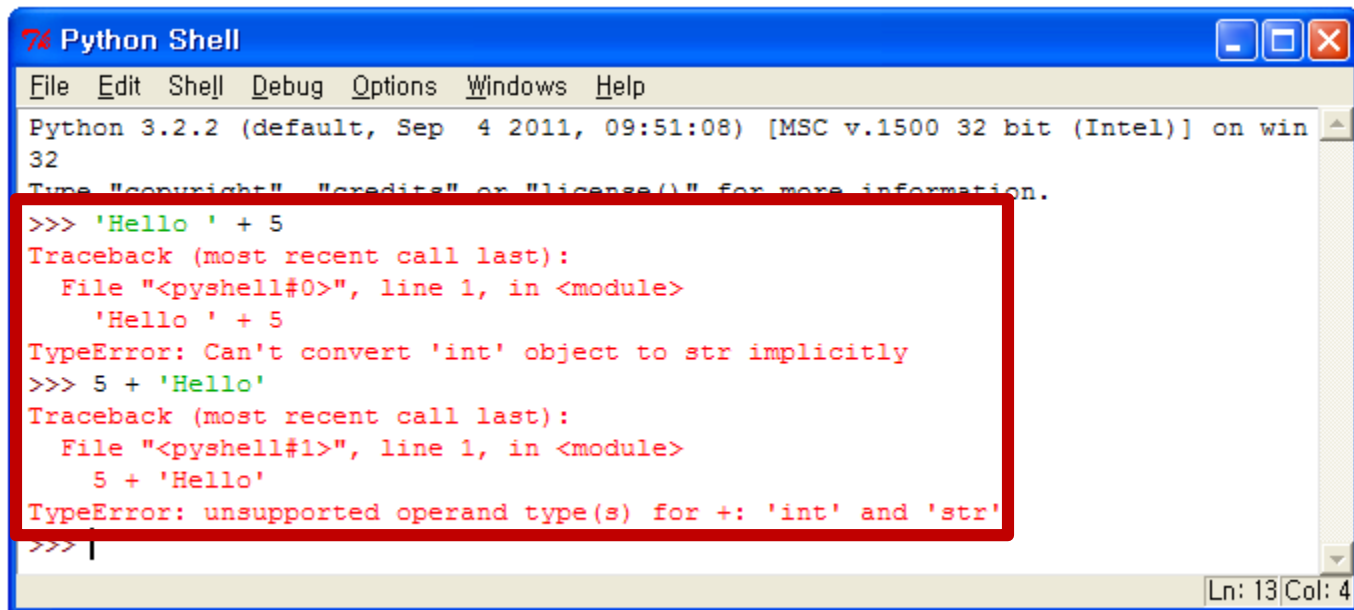
```
>>> 'Hello' + 'World!'
'HelloWorld!'
>>> 'Hello ' + 'World!'
'Hello World!'
>>>
```

The code and output are highlighted with a red rectangular box. The status bar at the bottom right shows "Ln: 7|Col: 4".

Strings

■ Data Types

- Can't add a **string** to an **integer**, or an **integer** number to a **string**.
 - Because a **string** and an **integer** are **different data types**.



```
Python Shell
File Edit Shell Debug Options Windows Help
Python 3.2.2 (default, Sep  4 2011, 09:51:08) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.

>>> 'Hello ' + 5
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    'Hello ' + 5
TypeError: Can't convert 'int' object to str implicitly
>>> 5 + 'Hello'
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    5 + 'Hello'
TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>> |
```

Ln: 13|Col: 4

Strings



■ Quiz

```
>>> 'Hello' + '5'  
>>>
```

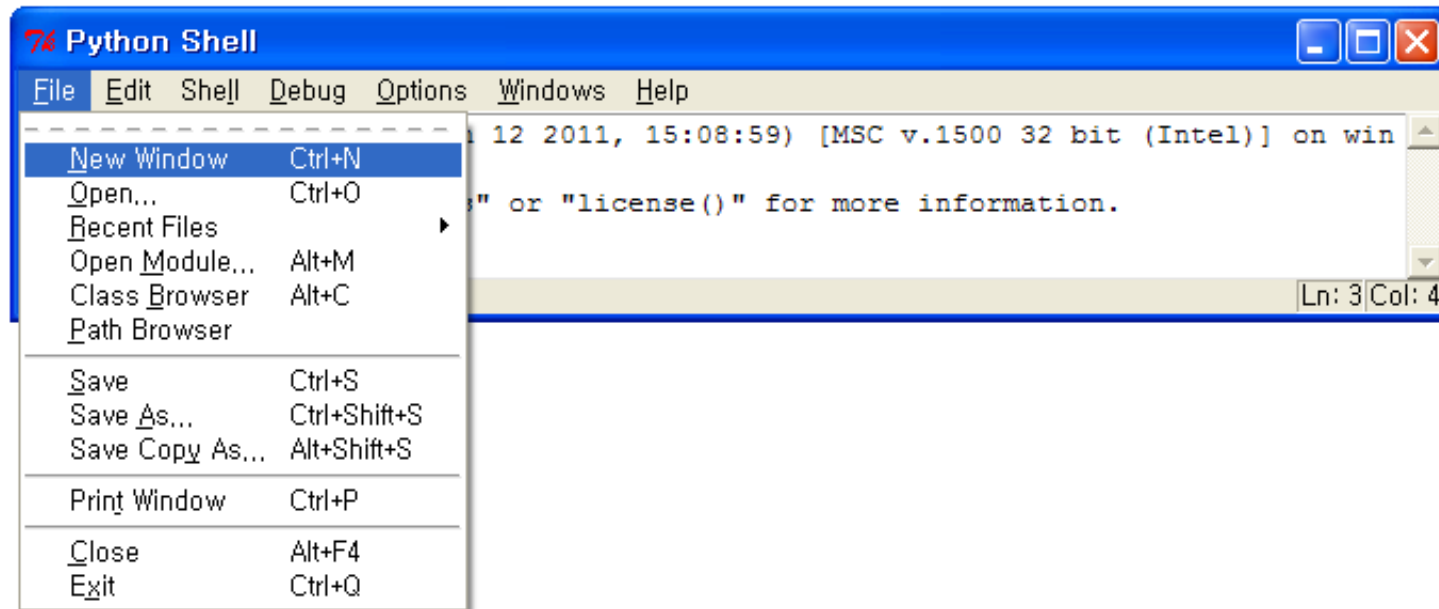
```
>>> spam = 5  
>>> 'Hello' + spam  
>>>
```

```
>>> spam = 'Hello'  
>>> spam = 'World!'  
>>> spam + spam  
>>>
```

Write the first program “Hello World!”

■ Programs “Hello World!”

- Use “new file editor” window on a windows machine

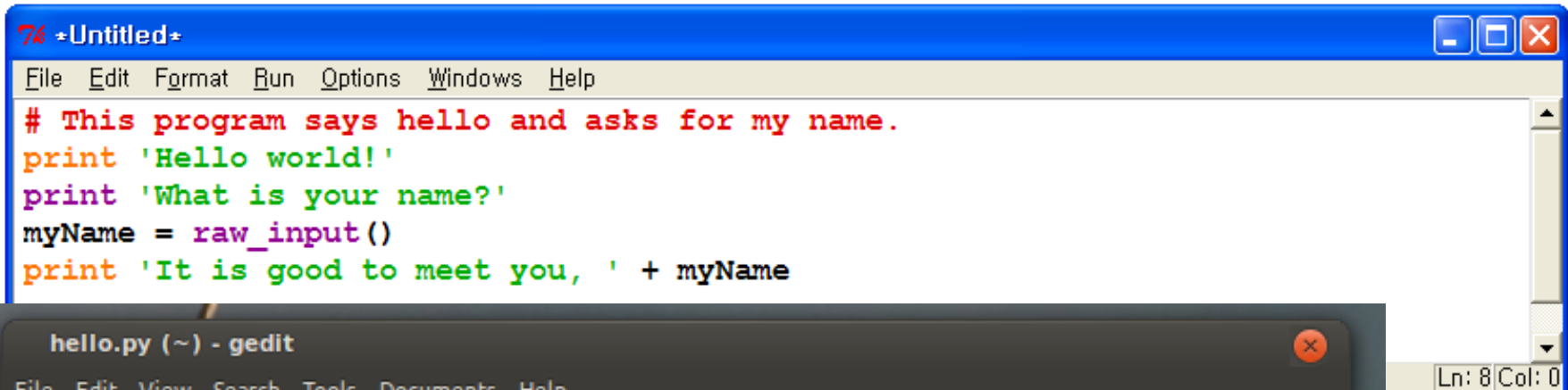


- Use a “gedit” window on a linux machine
 - click the Ubuntu logo and start typing *gedit*.

Write the first program “Hello World!”

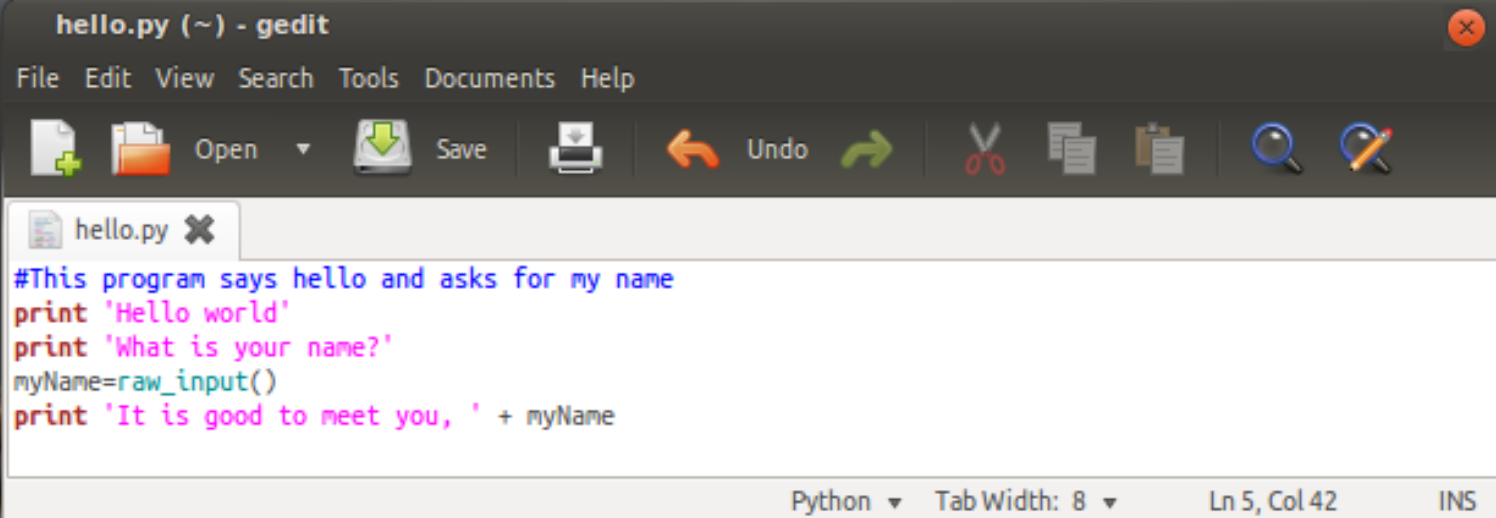
■ Programs “Hello World!”

- We call this text the **source code** of the program.



```
*Untitled*
File Edit Format Run Options Windows Help
# This program says hello and asks for my name.
print 'Hello world!'
print 'What is your name?'
myName = raw_input()
print 'It is good to meet you, ' + myName
```

Ln: 8 Col: 0



```
hello.py (~) - gedit
File Edit View Search Tools Documents Help
#This program says hello and asks for my name
print 'Hello world'
print 'What is your name?'
myName=raw_input()
print 'It is good to meet you, ' + myName

Python Tab Width: 8 Ln 5, Col 42 INS
```

Write the first program “Hello World!”

■ Programs “Hello World!”

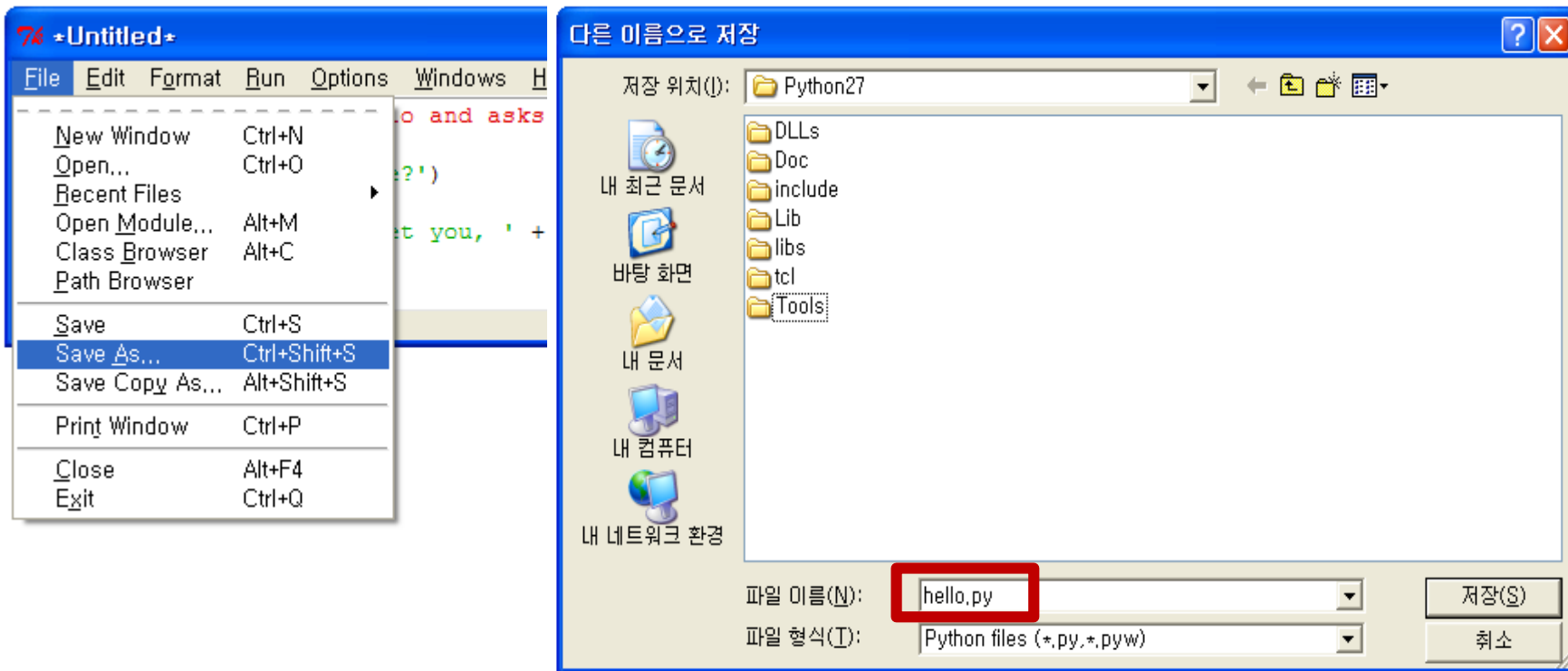
- Type the following text into this new window.

```
1. # This program says hello and asks for my name.  
2. print 'Hello world!'  
3. print 'What is your name?'  
4. myName = raw_input()  
5. print 'It is good to meet you, ' + myName
```


Write the first program “Hello World!”

■ Programs “Hello World!”

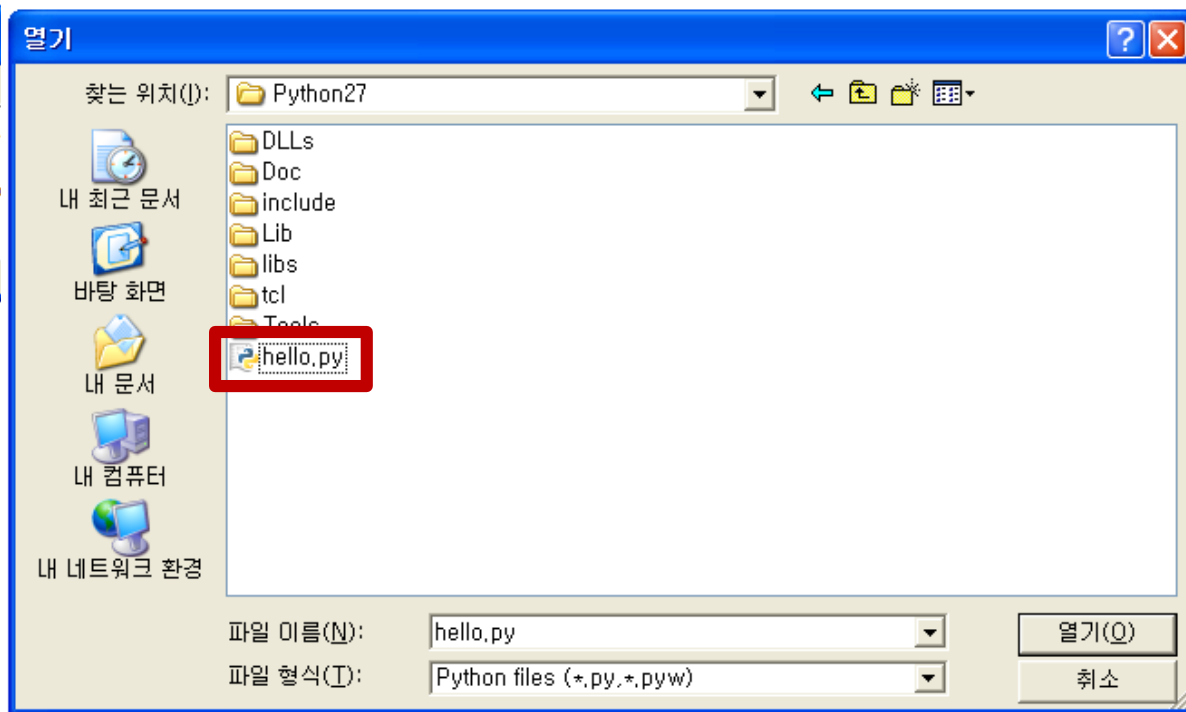
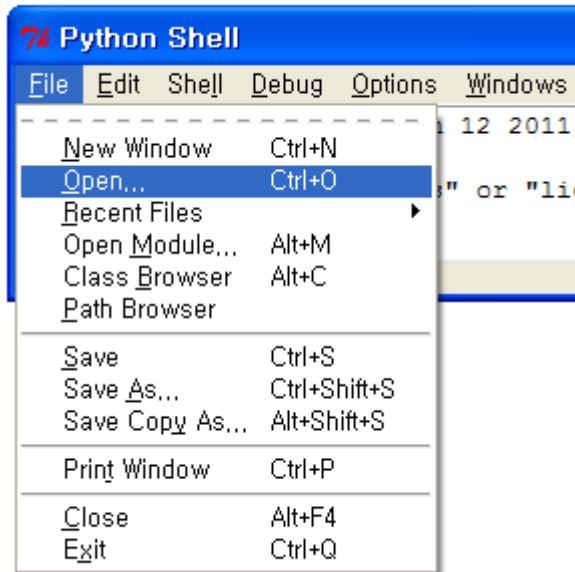
- Saving Program



Write the first program “Hello World!”

■ Programs “Hello World!”

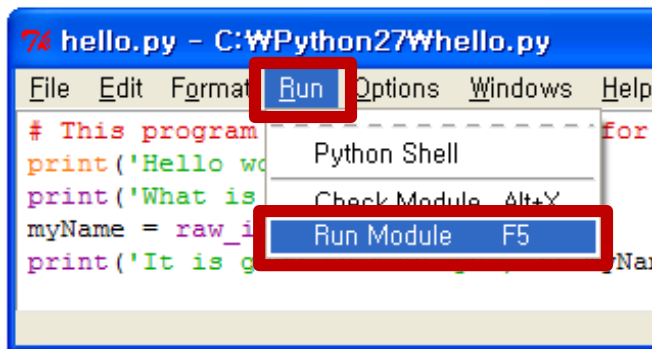
- Opening The Programs You've Saved



Write the first program “Hello World!”

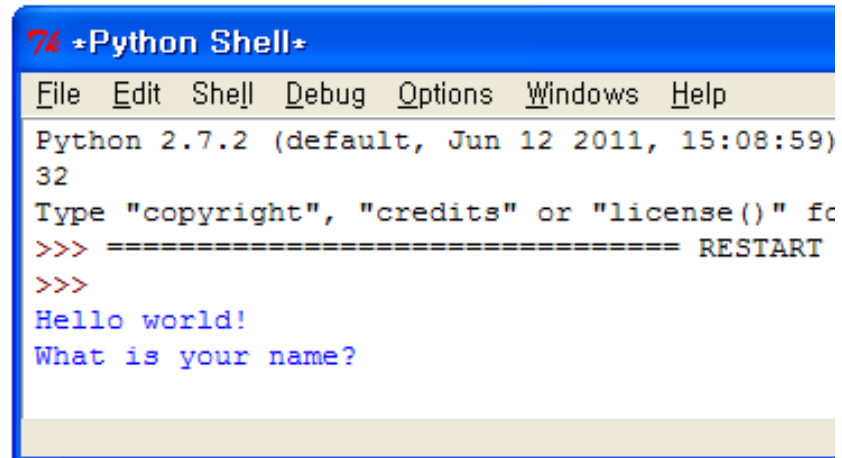
■ Programs “Hello World!”

- Run “Hello World!” program.
- choose **Run > Run Module** or just press the **F5** key.



The screenshot shows a Python IDE window titled 'hello.py - C:\WPython27\hello.py'. The menu bar includes File, Edit, Format, Run, Options, Windows, and Help. The 'Run' menu is open, showing options: Python Shell, Check Module (Alt+X), and Run Module (F5). The 'Run Module' option is highlighted with a red box.

```
# This program
print('Hello world!')
print('What is your name?')
myName = raw_input()
print('It is good to meet you, ' + myName + '!')
```

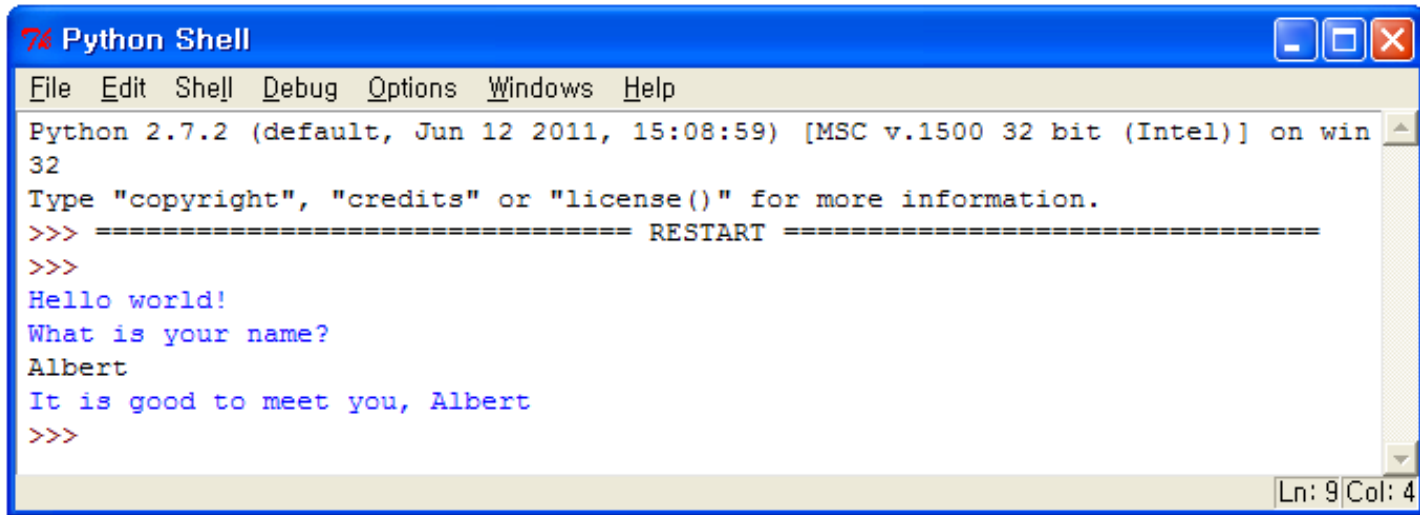


The screenshot shows a Python Shell window titled '+Python Shell+'. The menu bar includes File, Edit, Shell, Debug, Options, Windows, and Help. The shell displays the output of the program: 'Hello world!' and 'What is your name?'.

```
Python 2.7.2 (default, Jun 12 2011, 15:08:59)
32
Type "copyright", "credits" or "license()" for more
>>> ===== RESTART
>>>
Hello world!
What is your name?
```

Write the first program “Hello World!”

■ Programs “Hello World!”



The screenshot shows a Windows-style application window titled "Python Shell". The menu bar includes "File", "Edit", "Shell", "Debug", "Options", "Windows", and "Help". The status bar at the bottom right indicates "Ln: 9 Col: 4". The main text area contains the following text:

```
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
Hello world!
What is your name?
Albert
It is good to meet you, Albert
>>>
```

Users	run and use the program
Programmers	wrote the program
Executes	program starts at the very top and then executes each line
Flow of execution, or execution	program's following of instructions step-by-step

Write the first program “Hello World!”

■ Code Explanation

- **Comment**

- Any text following a **# sign** (called the **pound sign**) is a comment.
- Not for the computer, but for the programmer.

```
1. # This program says hello and asks for my name.
```

- **Print statement**

- The **print** keyword followed by an expression.
- Will **display** the text on the screen.

```
2. print 'Hello world! '  
3. print 'What is your name?'
```

Write the first program “Hello World!”

■ Code Explanation

- **Function**

- a bit of code that does a particular action.

- **Function call**

- a piece of code that tells our program to run the code inside a function.

- **Return value**

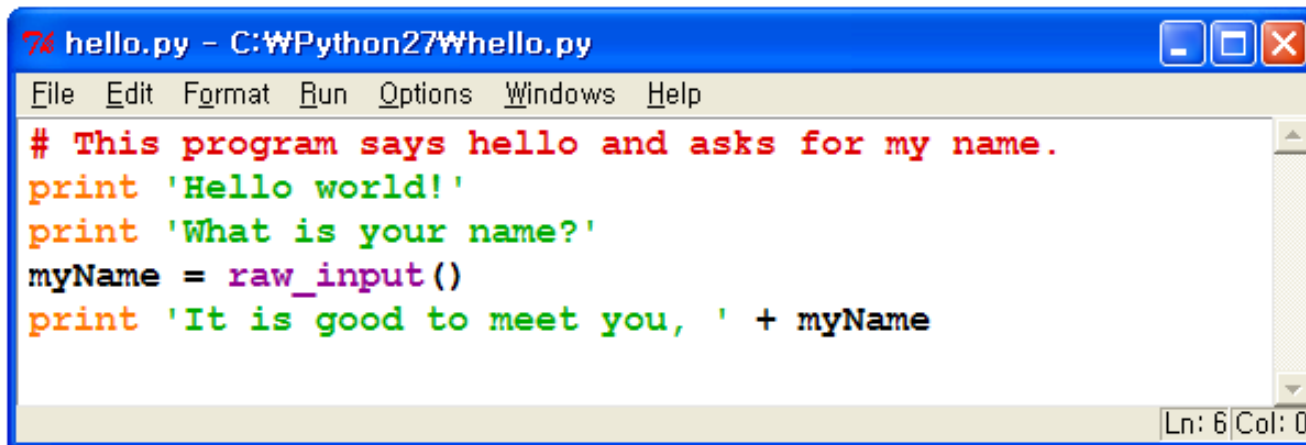
- The **value that the function call** evaluates to is called the return value.

- **Ending the Program**

- Once the program executes the **last line, it stops.**
At this point it has **terminated** or **exited**.

Write the first program “Hello World!”

■ Code Explanation



The screenshot shows a window titled 'hello.py - C:\WPython27\Whello.py'. The menu bar includes File, Edit, Format, Run, Options, Windows, and Help. The code is as follows:

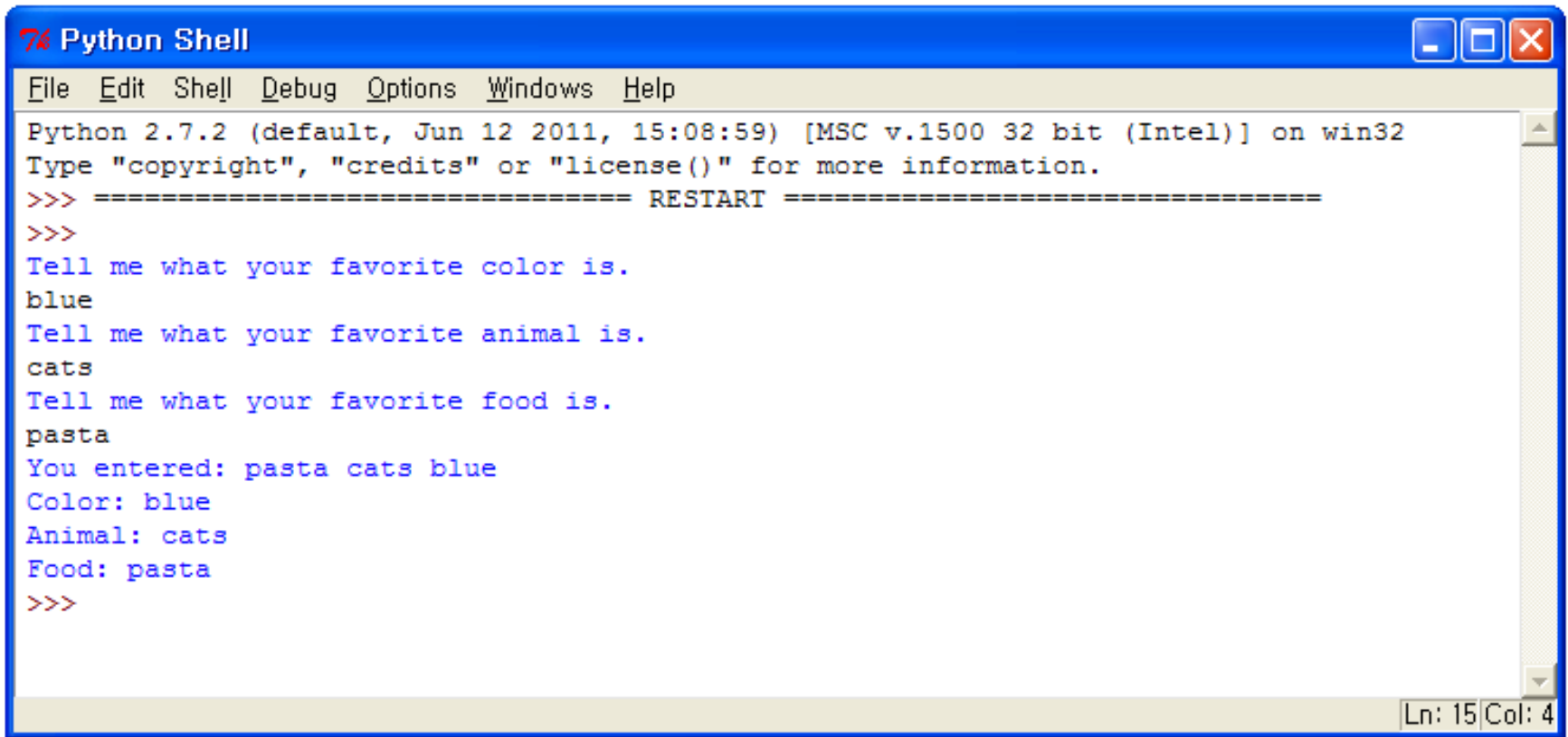
```
# This program says hello and asks for my name.  
print 'Hello world!'  
print 'What is your name?'  
myName = raw_input()  
print 'It is good to meet you, ' + myName
```

The status bar at the bottom right indicates 'Ln: 6 Col: 0'.

- **Variable**
 - myName
- **Function**
 - print(), raw_input()

“My Favorite Stuff”

■ Programs “My Favorite Stuff”



```
Python Shell
File Edit Shell Debug Options Windows Help
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
Tell me what your favorite color is.
blue
Tell me what your favorite animal is.
cats
Tell me what your favorite food is.
pasta
You entered: pasta cats blue
Color: blue
Animal: cats
Food: pasta
>>>
```

Ln: 15 Col: 4

“My Favorite Stuff”

■ Programs “My Favorite Stuff”

```
# Favorite stuff
print 'Tell me what your favorite color is.'
favoriteColor = raw_input()

print 'Tell me what your favorite animal is.'
favoriteAnimal = raw_input()

print 'Tell me what your favorite food is.'
favoriteFood = raw_input()

# display our favorite stuff
print 'You entered: ' + favoriteFood + ' ' + favoriteAnimal + ' ' + favoriteColor
# print 'Here is a list of your favorite things.'
print 'Color: ' + favoriteColor
print 'Animal: ' + favoriteAnimal
print 'Food: ' + favoriteFood
```

“My Favorite Stuff”

■ Code Explanation

- **Comment**

- The program will **ignore it**.
- All the text after the **pound sign(#)** will be ignored by the program.

```
1. # Favorite stuff
```

- Display a bit of text asking the user to type in their **favorite color**.

```
2. print 'Tell me what your favorite color is.'
```

“My Favorite Stuff”

■ Code Explanation

- **raw_input()** function
 - Let the user type in their favorite color.
 - string the user entered is stored in the **favoriteColor** variable.

```
3. favoriteColor = raw_input()
```

“My Favorite Stuff”

■ Code Explanation

- `raw_input()` function
 - These lines are similar to the ones before.
 - There is a **blank line** in between them.
 - Python language, blank lines are just **ignored**.

```
5. print 'Tell me what your favorite animal is.'  
6. favoriteAnimal = raw_input()
```

```
8. print 'Tell me what your favorite food is.'  
9. favoriteFood = raw_input()
```

“My Favorite Stuff”

■ Code Explanation

- **Another comment.**
 - Don’t always have to go at the top (can show up **anywhere**).

```
11. # display our favorite stuff
```

- **print statement**
 - Show us the favorite food, animal, and other we entered.
 - The **plus sign** is used to combine the string.

```
12. print 'You entered: ' + favoriteFood + ' '  
    + favoriteAnimal + ' ' + favoriteColor
```

“My Favorite Stuff”

■ Code Explanation

- **print** statement
 - Another `print` statement

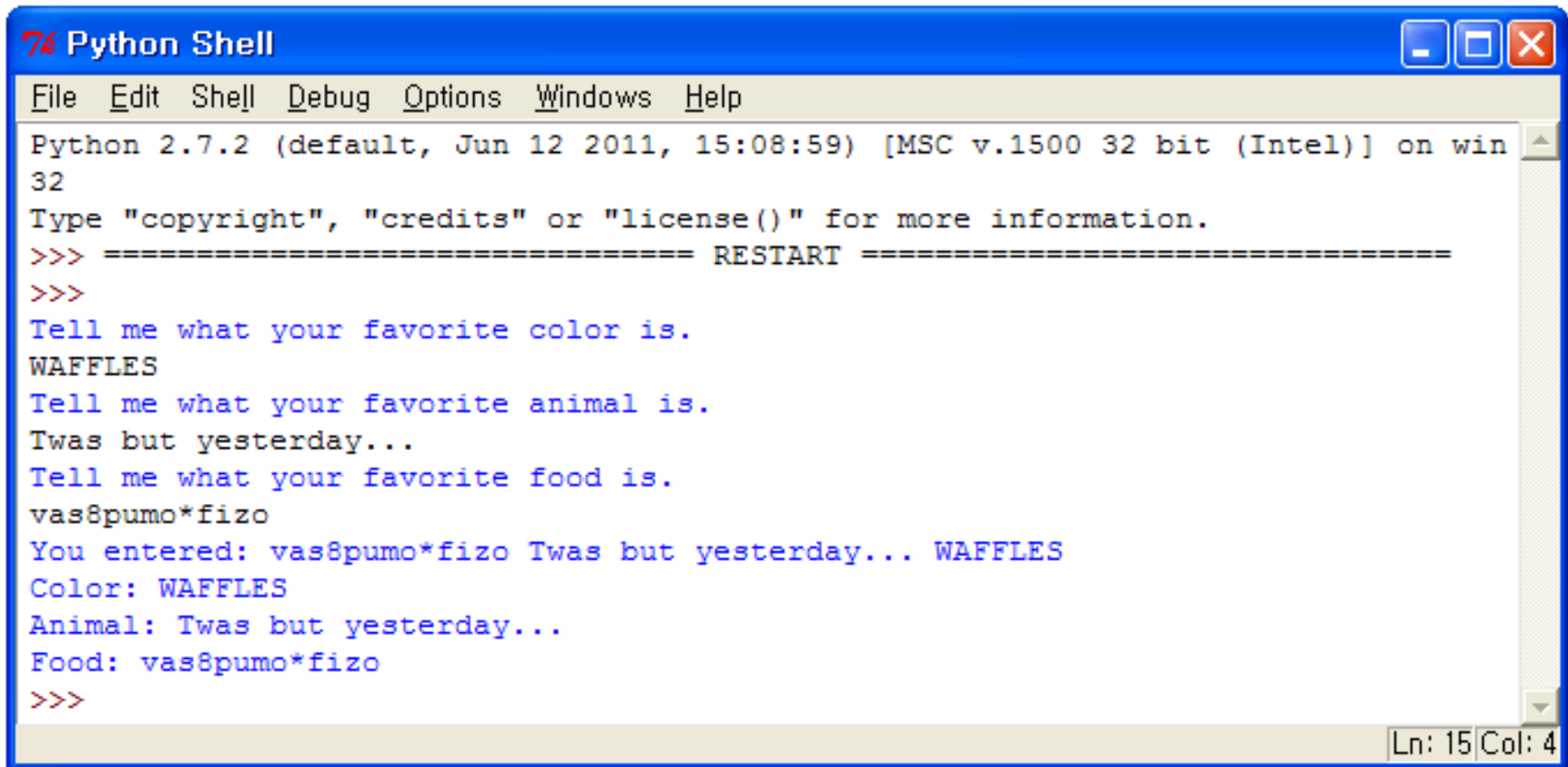
```
13. # print 'Here is a list of your favorite  
    things.'
```

- These three lines will display our favorite things again.

```
14. print 'Color: ' + favoriteColor  
15. print 'Animal: ' + favoriteAnimal  
16. print 'Food: ' + favoriteFood
```

“My Favorite Stuff”

- Crazy Answers and Crazy Names for our Favorite Stuff
 - The computer doesn't really care what you type in.



```
Python Shell
File Edit Shell Debug Options Windows Help
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
Tell me what your favorite color is.
WAFFLES
Tell me what your favorite animal is.
Twas but yesterday...
Tell me what your favorite food is.
vas8pumo*fizo
You entered: vas8pumo*fizo Twas but yesterday... WAFFLES
Color: WAFFLES
Animal: Twas but yesterday...
Food: vas8pumo*fizo
>>>
Ln: 15 Col: 4
```

“My Favorite Stuff”

■ Crazy Answers and Crazy Names for our Favorite Stuff

- The program also **does not care** what name we give to our variables.

```
1. # Favorite stuff 2
2. print 'Tell me what your favorite color is.'
3. q = raw_input()
4.
5. print 'Tell me what your favorite animal is.'
6. fizzy = raw_input()
7.
8. print 'Tell me what your favorite food is.'
9. AbrahamLincoln = raw_input()
10.
11. # display our favorite stuff
12. print 'You entered: ' + q + ' ' + fizzy + ' ' + AbrahamLincoln
13. # print 'Here is a list of your favorite things.'
14. print 'Color: ' + q
15. print 'Animal: ' + fizzy
16. print 'Food: ' + AbrahamLincoln
```


“My Favorite Stuff”

■ Capitalizing our Variables

- This is to make the variable names **easier to read**.
 - Because variable names **can’t have spaces** in them.

```
thisnameiskindofhardtoread  
thisNameIsEasierToRead
```

- Leave the first word in **lowercase**.
 - » But start the other words in **uppercase**.
 - We call something in a certain way like this a **convention**.

“My Favorite Stuff”



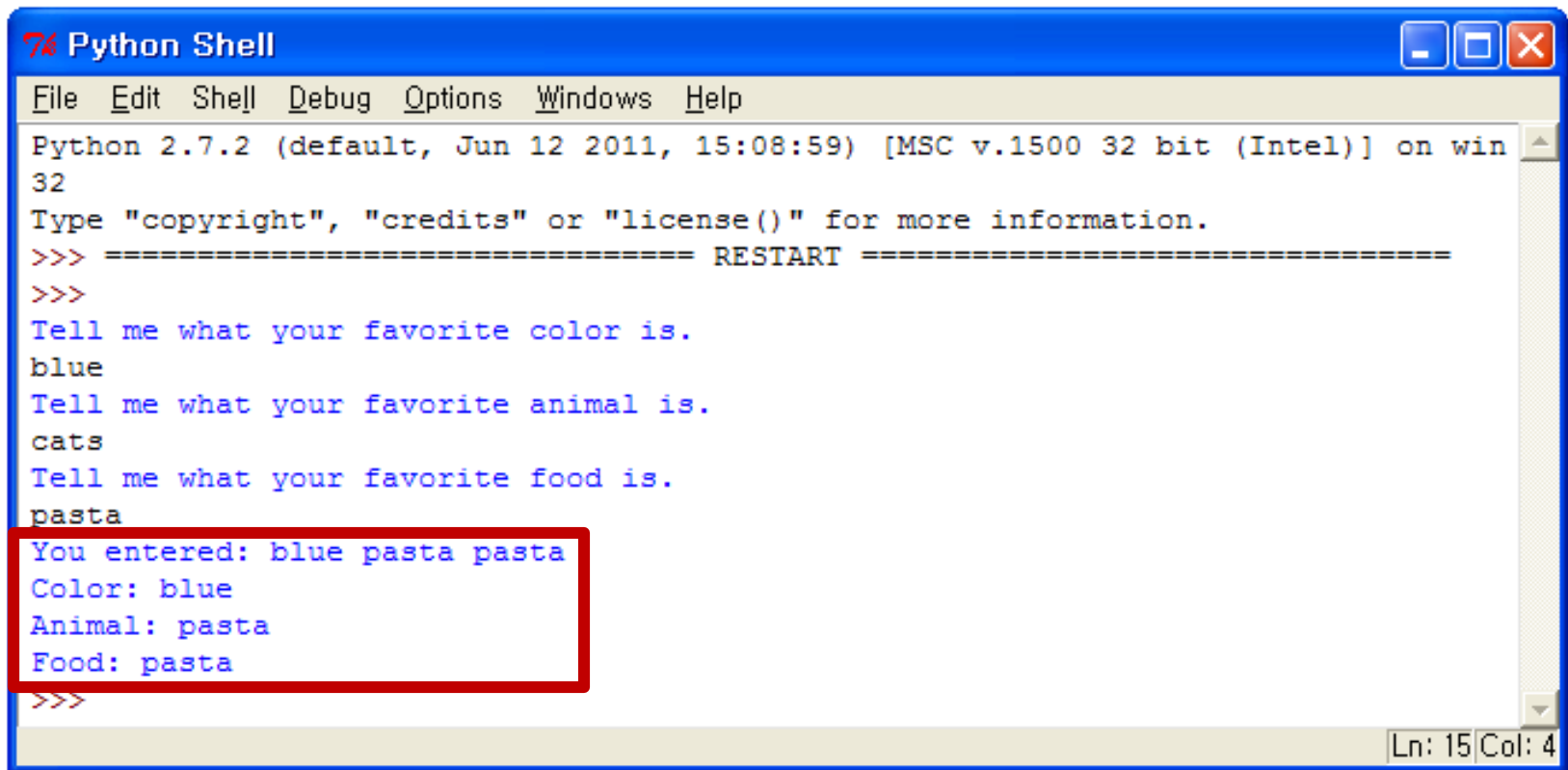
■ Quiz

- What happened here?

```
1. # Favorite stuff 3
2. print 'Tell me what your favorite color is.'
3. q = raw_input()
4.
5. print 'Tell me what your favorite animal is.'
6. AbrahamLincoln = raw_input()
7.
8. print 'Tell me what your favorite food is.'
9. AbrahamLincoln = raw_input()
10.
11. # display our favorite stuff
12. print 'You entered: ' + q + ' ' + AbrahamLincoln + ' ' + AbrahamLincoln
13. # print 'Here is a list of your favorite things.'
14. print 'Color: ' + q
15. print 'Animal: ' + AbrahamLincoln
16. print 'Food: ' + AbrahamLincoln
```

“My Favorite Stuff”

- The favorite food value was **overwritten**.
 - A variable can only store **one value at a time**.



```
Python Shell
File Edit Shell Debug Options Windows Help
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
Tell me what your favorite color is.
blue
Tell me what your favorite animal is.
cats
Tell me what your favorite food is.
pasta
You entered: blue pasta pasta
Color: blue
Animal: pasta
Food: pasta
>>>
```

“My Favorite Stuff”

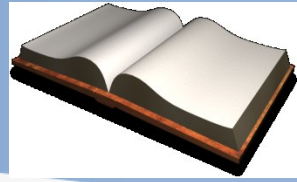
■ Case-sensitivity

- The computer considers these names to be **four separate variables**.

```
fizzy  
Fizzy  
FIZZY  
fIzZy
```

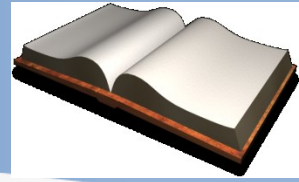
- The computer doesn't know of a function named **RAW_INPUT()**.
- It only knows a function named **raw_input()**.

Things Covered In This Chapter(1/2)



- Downloading and installing the Python interpreter.
- Using IDLE's interactive shell to run instructions.
- Flow of execution
- Expressions, and evaluation expressions
- Integer
- Operators(such as + - *)
- Variables
- Assignment statements
- Overwriting values in variables.

Things Covered In This Chapter(2/2)



- Strings, String concatenation
- Data types (such as strings or integers)
- Using IDLE to write source code.
- Saving and running programs in IDLE.
- The `print` statement.
- The `raw_input()` function.
- Comments
- Conventions
- Case-sensitivity