"Hello World!"

Invent Your Own Computer Games with Python

Taesoo Kwon

Heejin Park

Hanyang University

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.

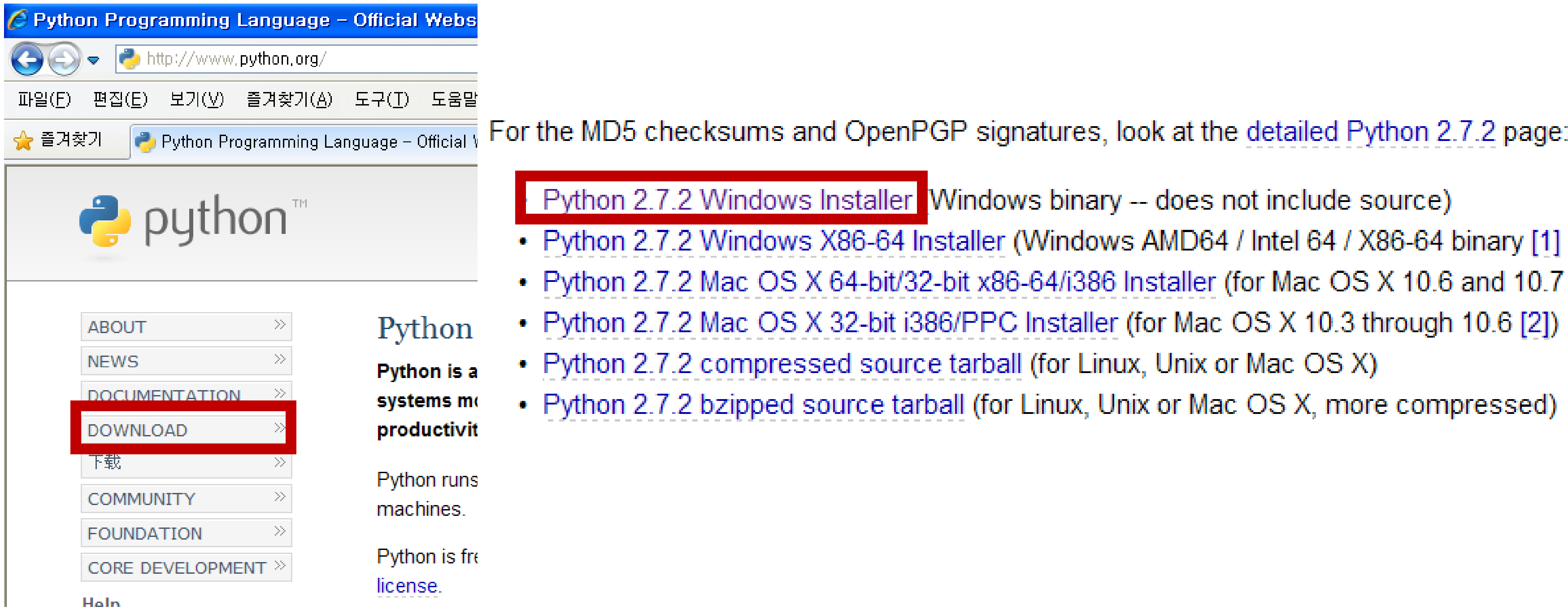
Introduction

- 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



Python Programming Language – Official Webs

http://www.python.org/

파일(F) 편집(E) 보기(V) 즐겨찾기(A) 도구(T) 도움말

즐겨찾기 | Python Programming Language – Official W

python™

ABOUT
NEWS
DOCUMENTATION
DOWNLOAD
下载
COMMUNITY
FOUNDATION
CORE DEVELOPMENT
Help

Python

Python is a
systems m
productivit

Python runs
machines.

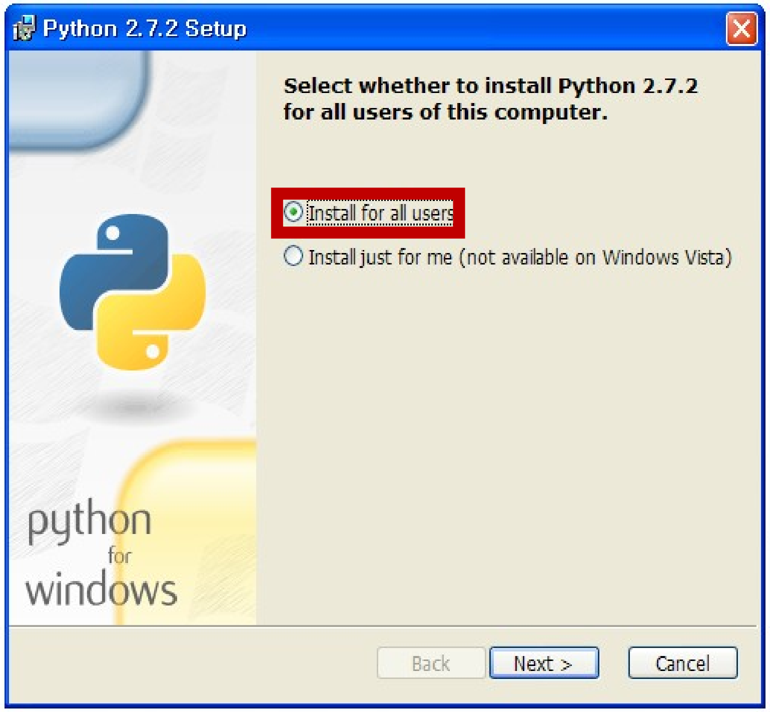
Python is fr
license.

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 bzipped source tarball (for Linux, Unix or Mac OS X, more compressed)

Installing Python

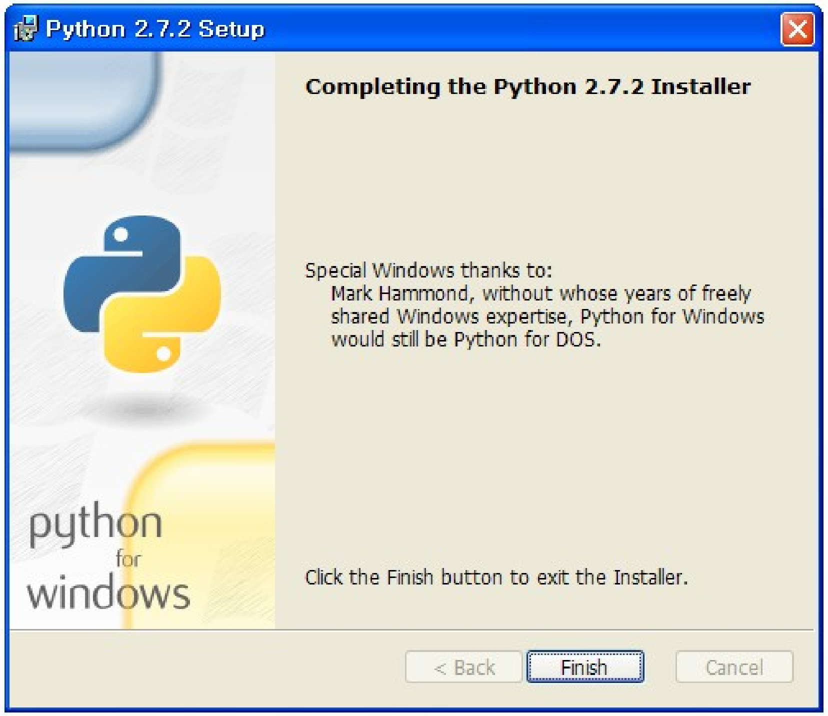
Install Python



Select whether to install Python 2.7.2 for all users of this computer.

[✓ Install for all users]

Install just for me (not available on Windows Vista)



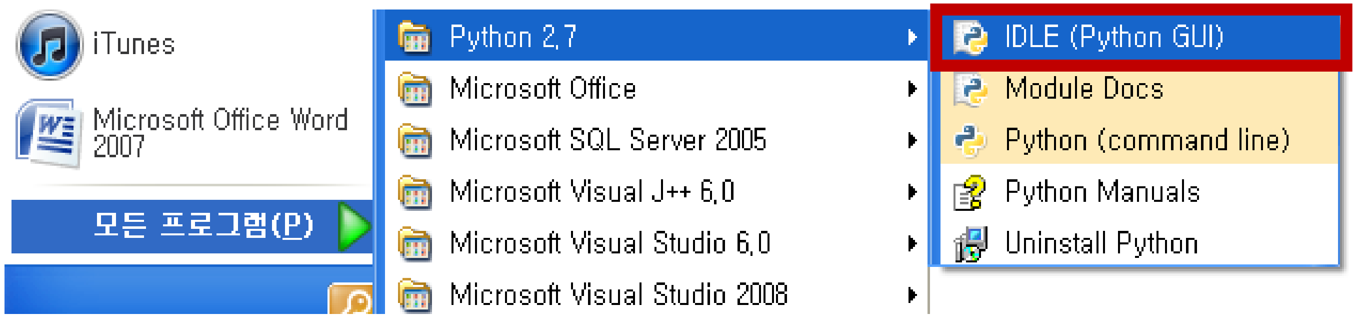
Completing the Python 2.7.2 Installer

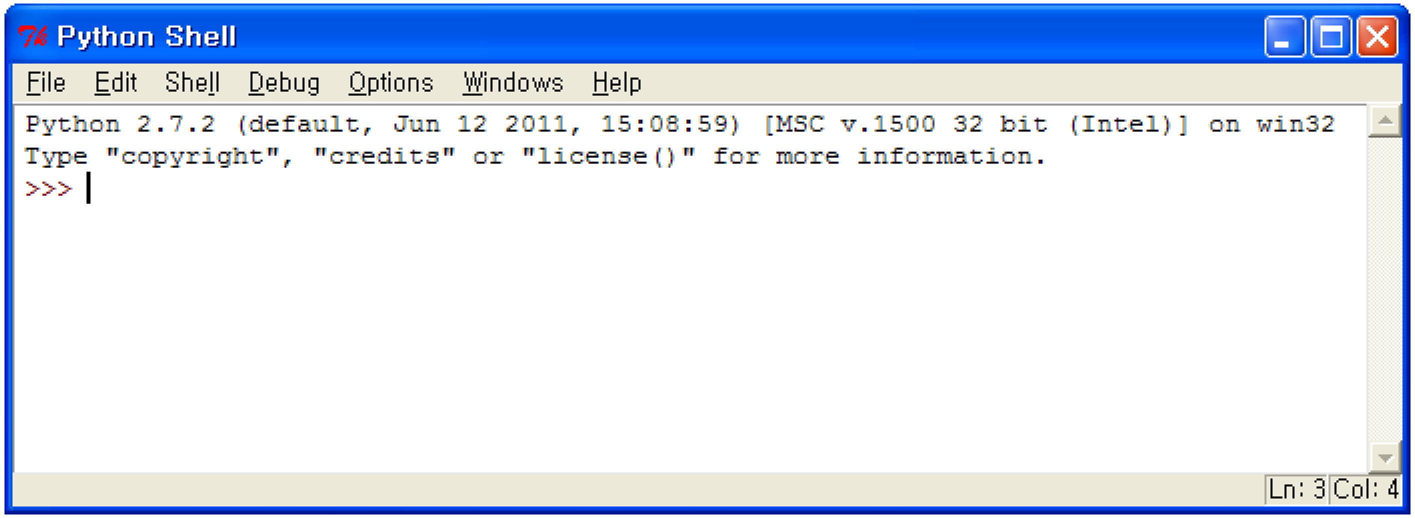
Special Windows thanks to:
Mark Hammond, without whose years of freely shared Windows expertise, Python for Windows would still be Python for DOS.

Click the Finish button to exit the Installer.

Starting the Python Interpreter

IDLE (Python GUI) Program

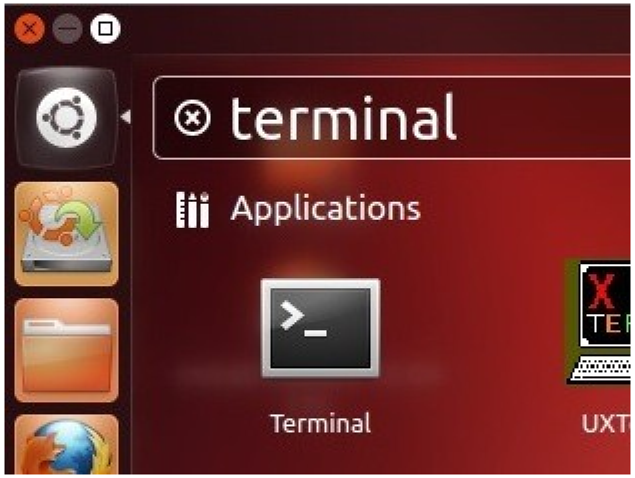




```
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 
```

Starting the Python Interpreter (Linux)

- First, open a “terminal”
- In Ubuntu (Unity), just click the Ubuntu logo and start typing *terminal*.



Starting the Python Interpreter (Linux)

- To open an IDLE session, type this command in a terminal window:
- python
- Type Control-D to terminate the IDLE session.
- If you write a Python script named *filename*.py, you can execute it using the command
- python *filename*.py

Starting the Python Interpreter

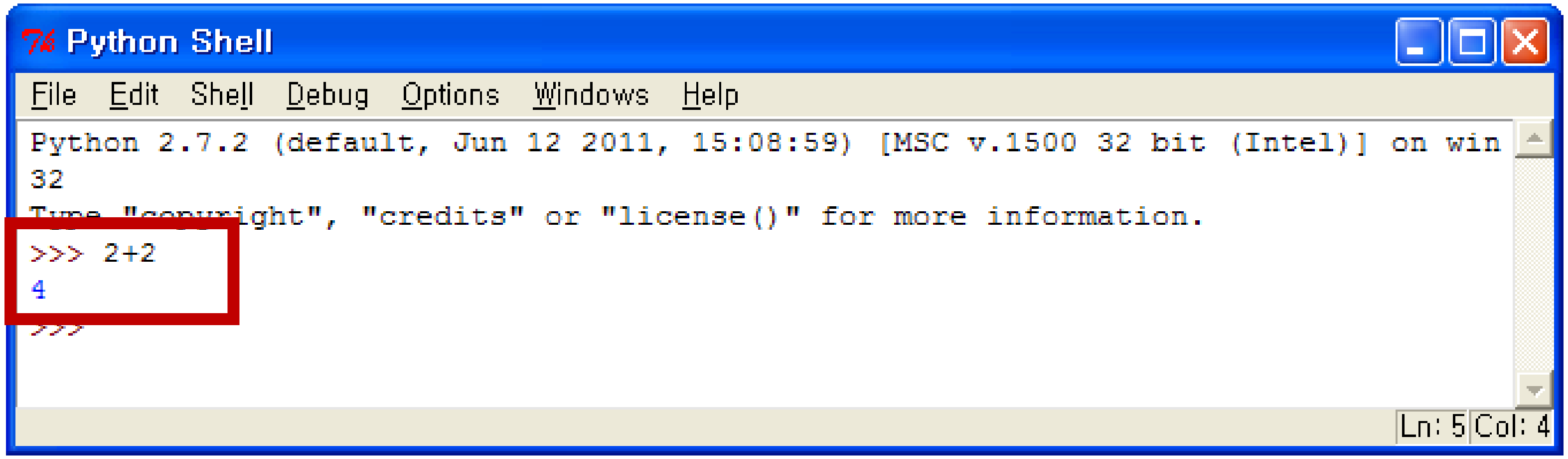
- **IDLE**
  - **I**nteractive **D**eve**L**opment **E**nvironment
  - Program that **helps us** type in our own programs and games.

  - **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**



```
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

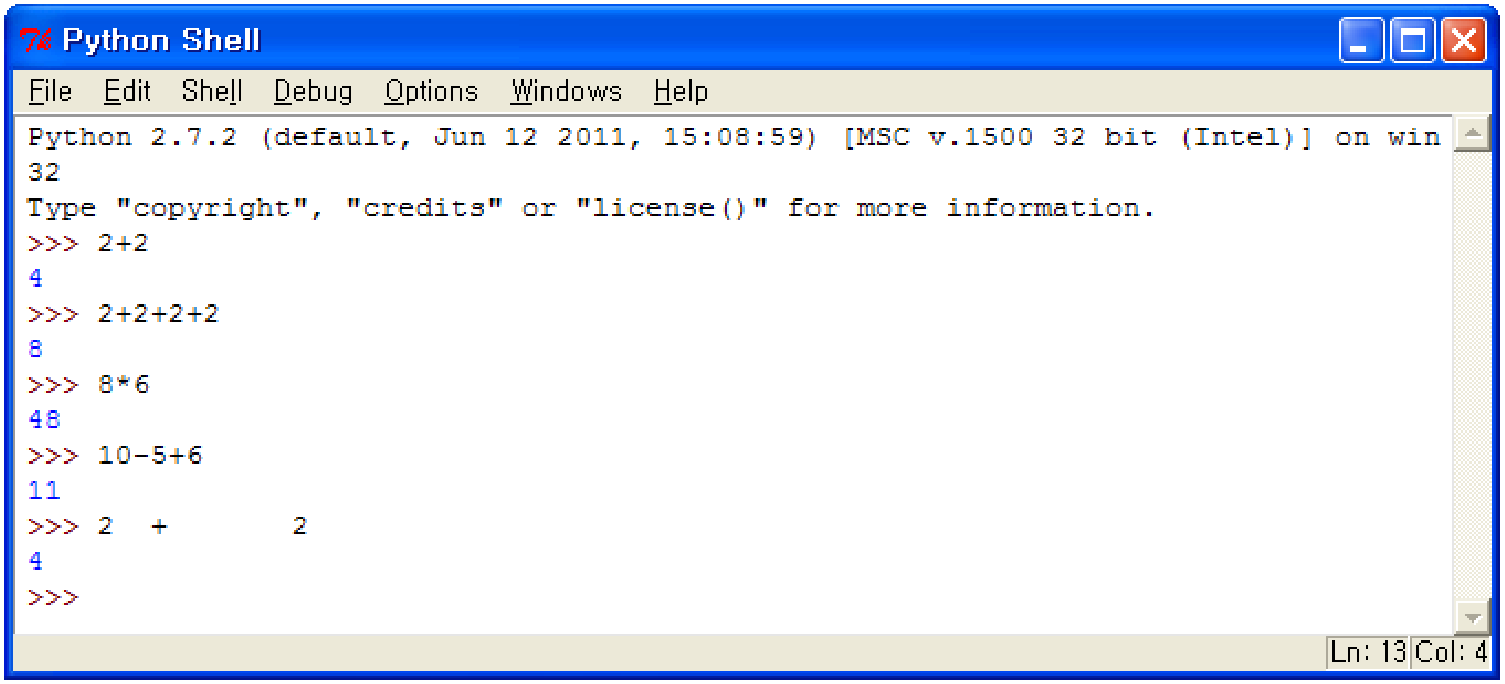
Integers and Floating Point Numbers

- **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 <u>5.0</u> it would <u>**not be an integer**</u>**.**

```
>>> 2/3
0
>>> 2.0/3.0
0.66666
```

Evaluating Expressions

Expressions



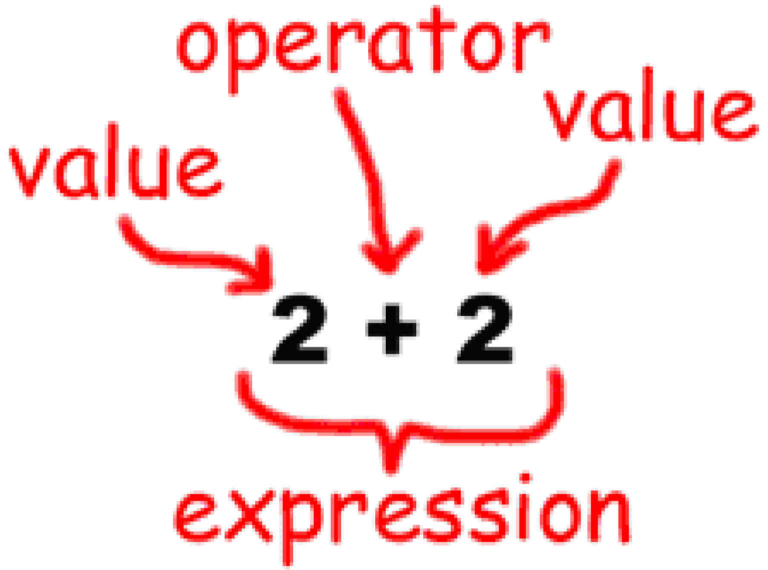
```
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> 2+2
4
>>> 2+2+2+2
8
>>> 8*6
48
>>> 10-5+6
11
>>> 2   +       2
4
>>>
```

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

Evaluating Expressions

Expressions

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



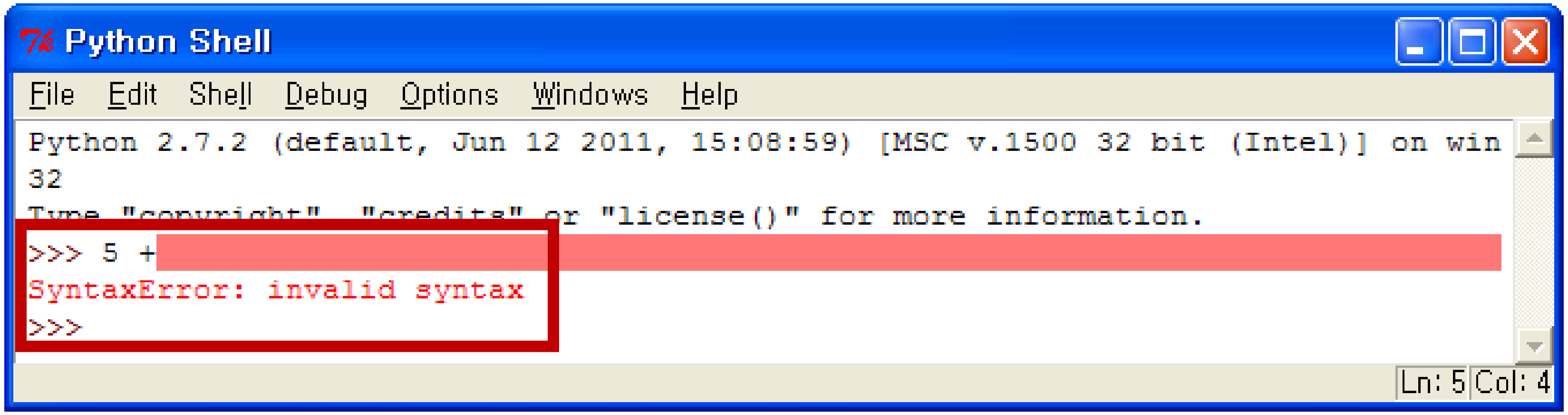
Evaluating Expressions

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.**



```
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
>>>
```

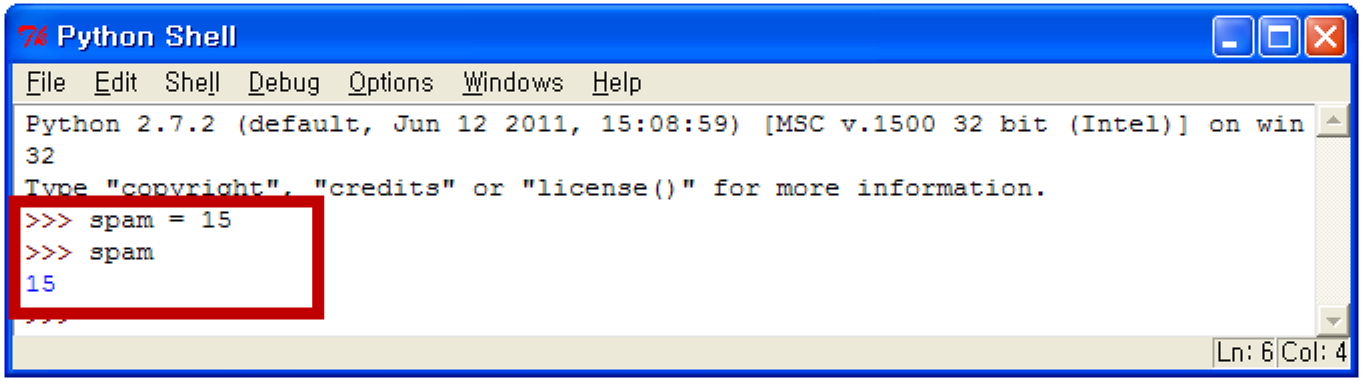
- because **5 +** is **not an expression.**
- expressions have values connected by operators.
- the + **operator** always expects to **connect two things** in Python.

Storing Values in Variables

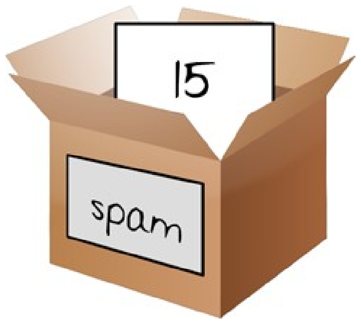
Variables

- 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".**



```
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
15
>>>
```



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

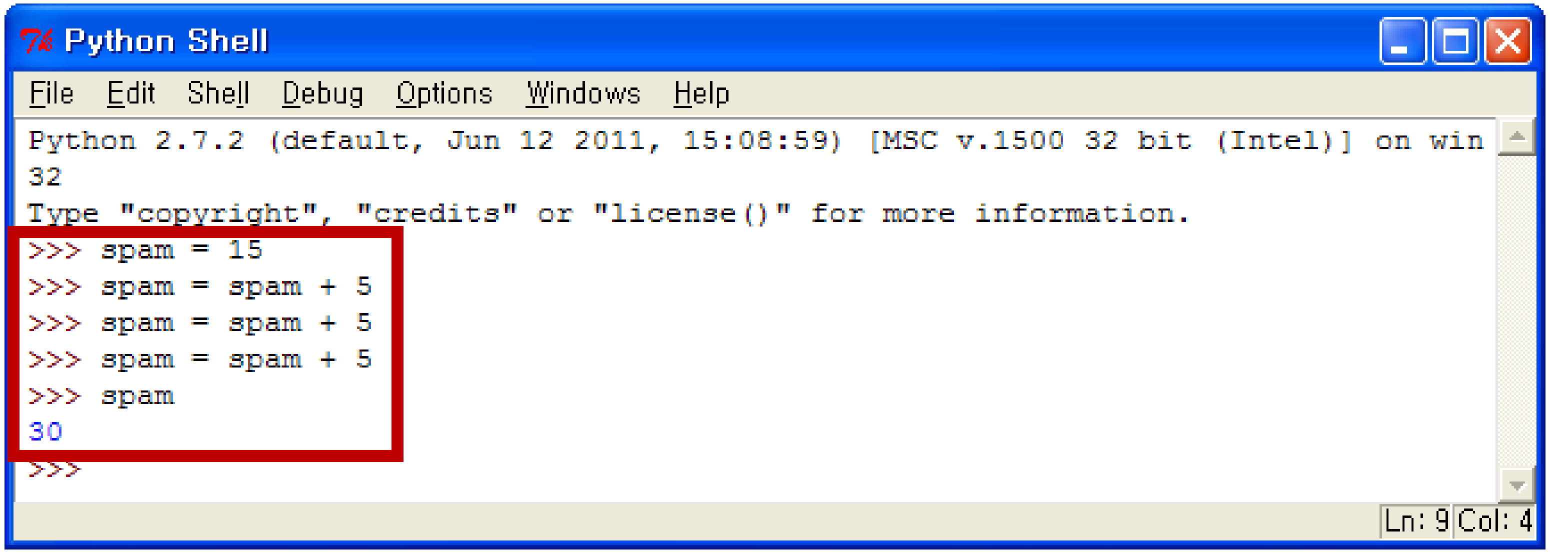
- **Quiz**

```
>>> 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



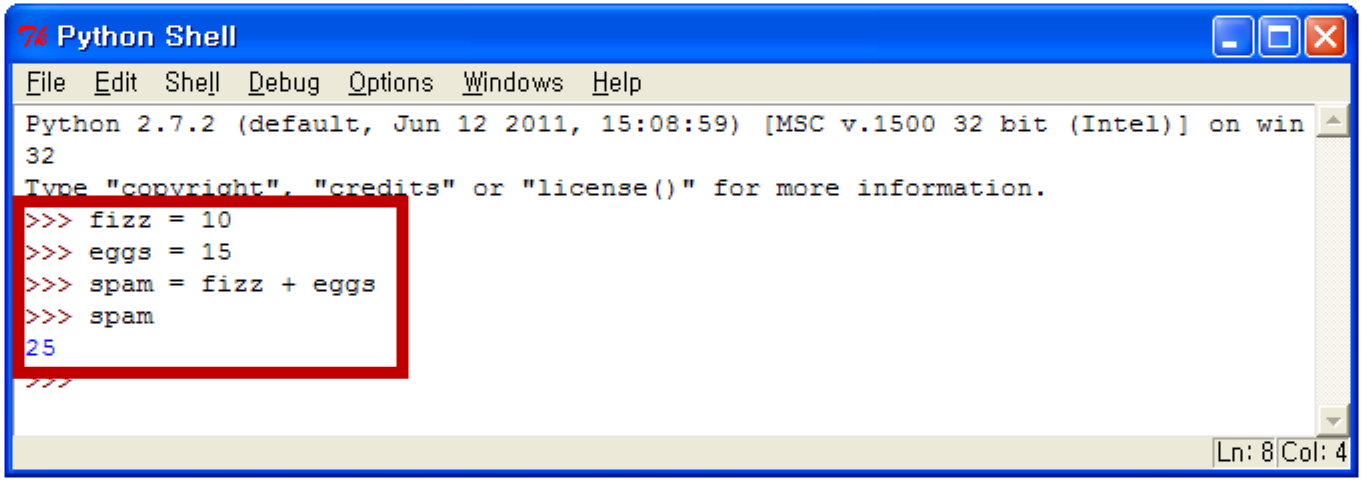
```
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> spam = 15
>>> spam = spam + 5
>>> spam = spam + 5
>>> spam = spam + 5
>>> spam
30
>>>
```

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

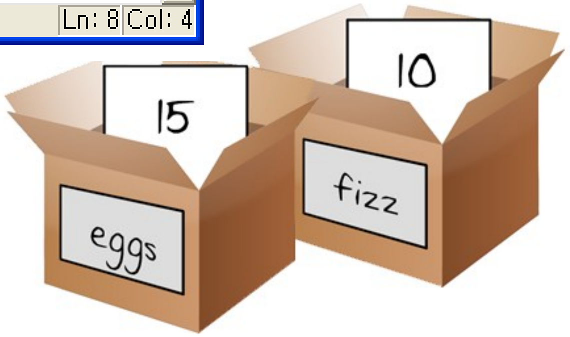
Storing Values in Variables

Using More Than One Variable

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



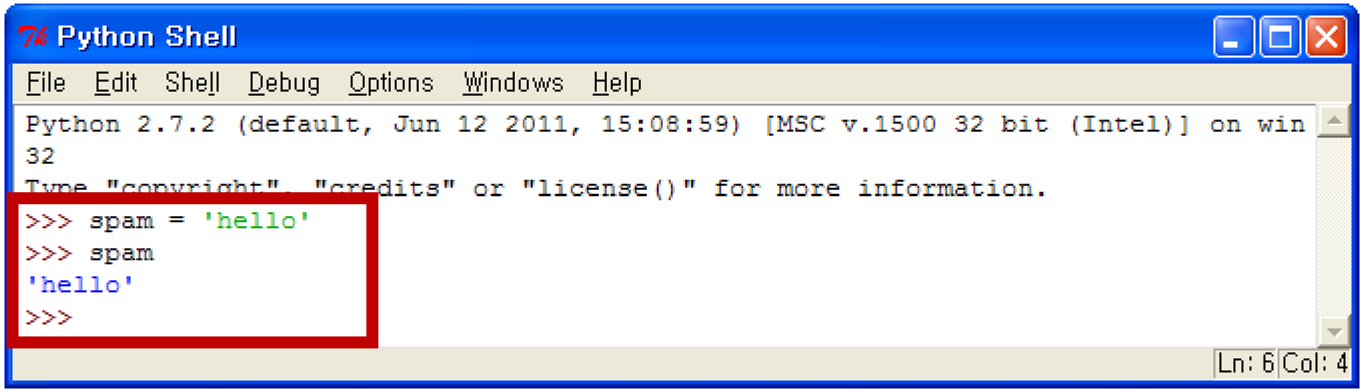
```
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> fizz = 10
>>> eggs = 15
>>> spam = fizz + eggs
>>> spam
25
>>>
```



Strings

Strings

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



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

Strings

Strings

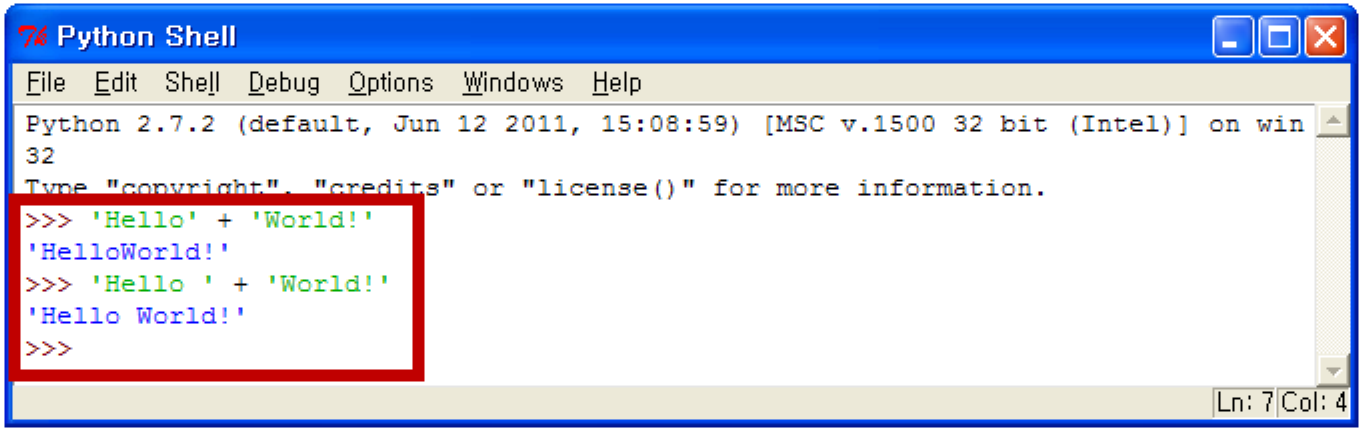
- Strings can have **spaces** and **numbers** as well.
- Examples of strings

```
'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.

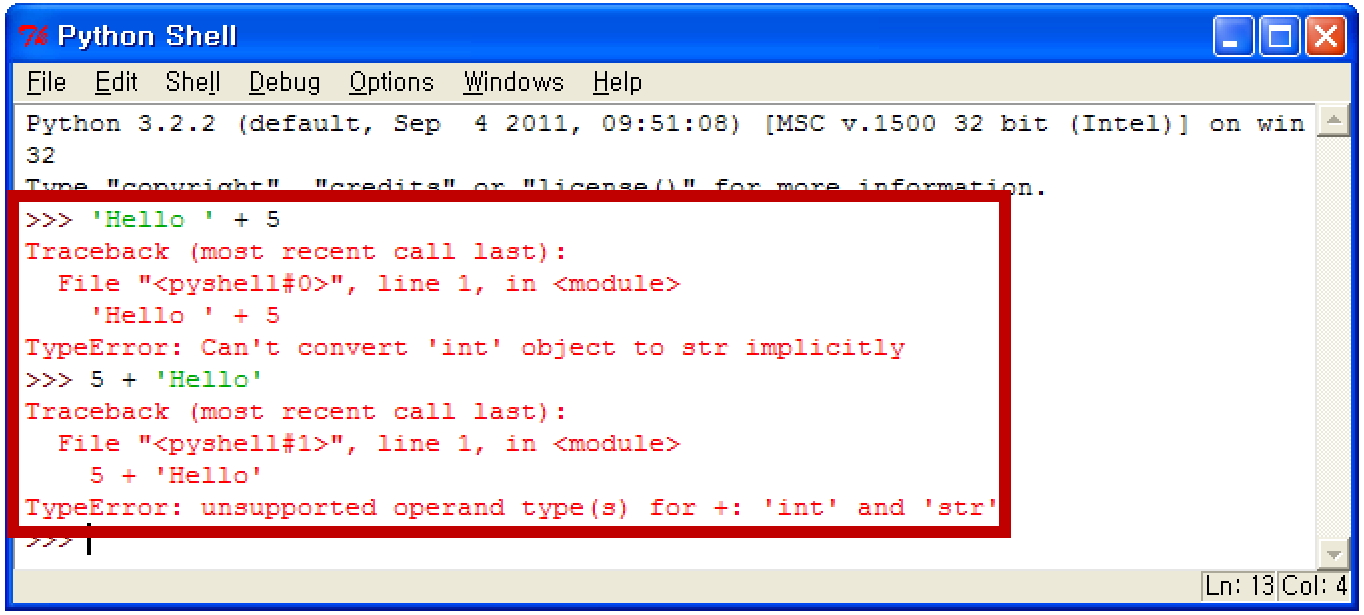


```
Python 2.7.2 (default, Jun 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> 'Hello' + 'World!'
'HelloWorld!'
>>> 'Hello ' + 'World!'
'Hello World!'
>>>
```

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 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'
>>> 
```

Strings

- Quiz

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

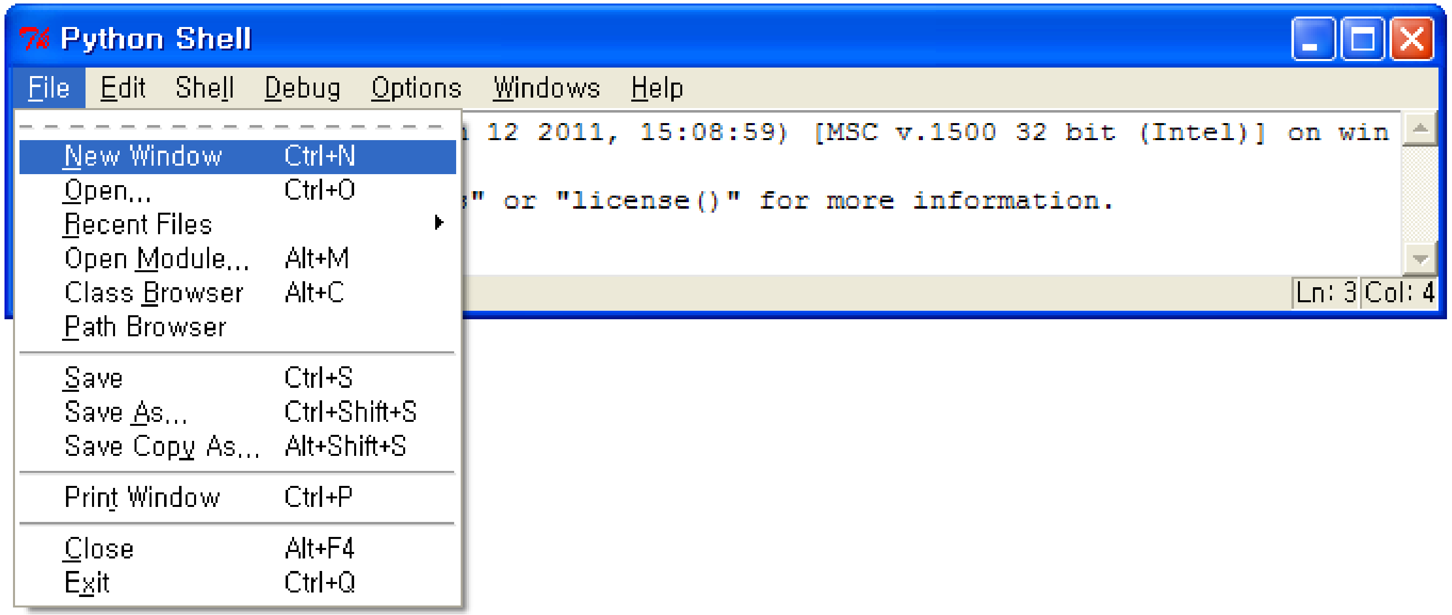
```
>>> spam = 5
>>> 'Hello' + spam
>>>
```

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

Write the first program "Hello World!"

Programs "Hello World!"

- Use "new file editor" window on a windows machine



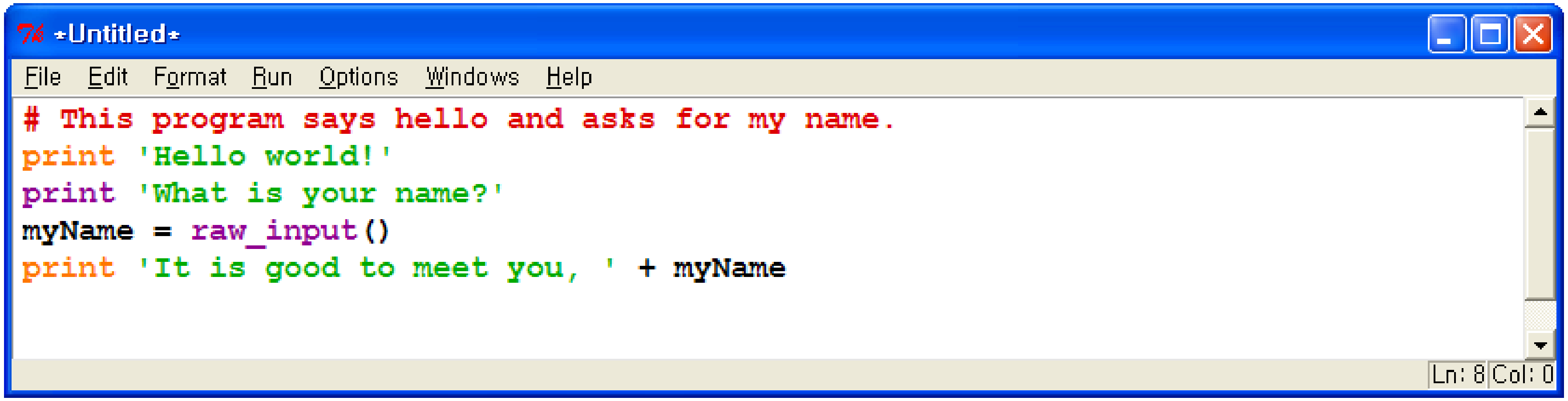
```
1 12 2011, 15:08:59) [MSC v.1500 32 bit (Intel)] on win
;" or "license()" for more information.
```

- 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.



```
# 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
```

Write the first program "Hello World!"

Programs "Hello World!"

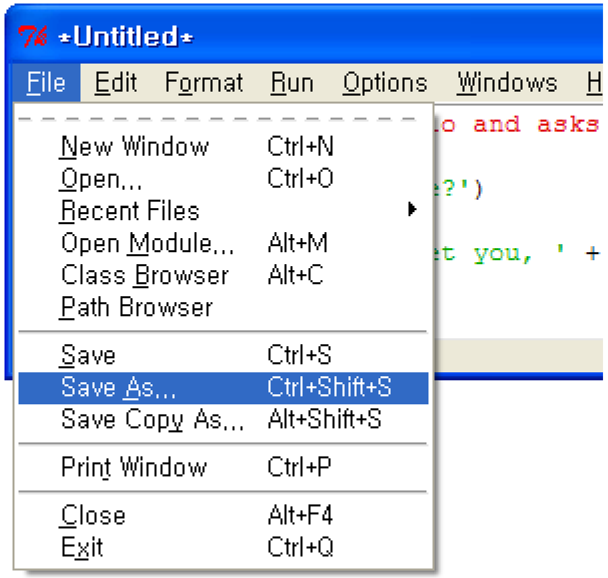
- Type the following text into this new window.

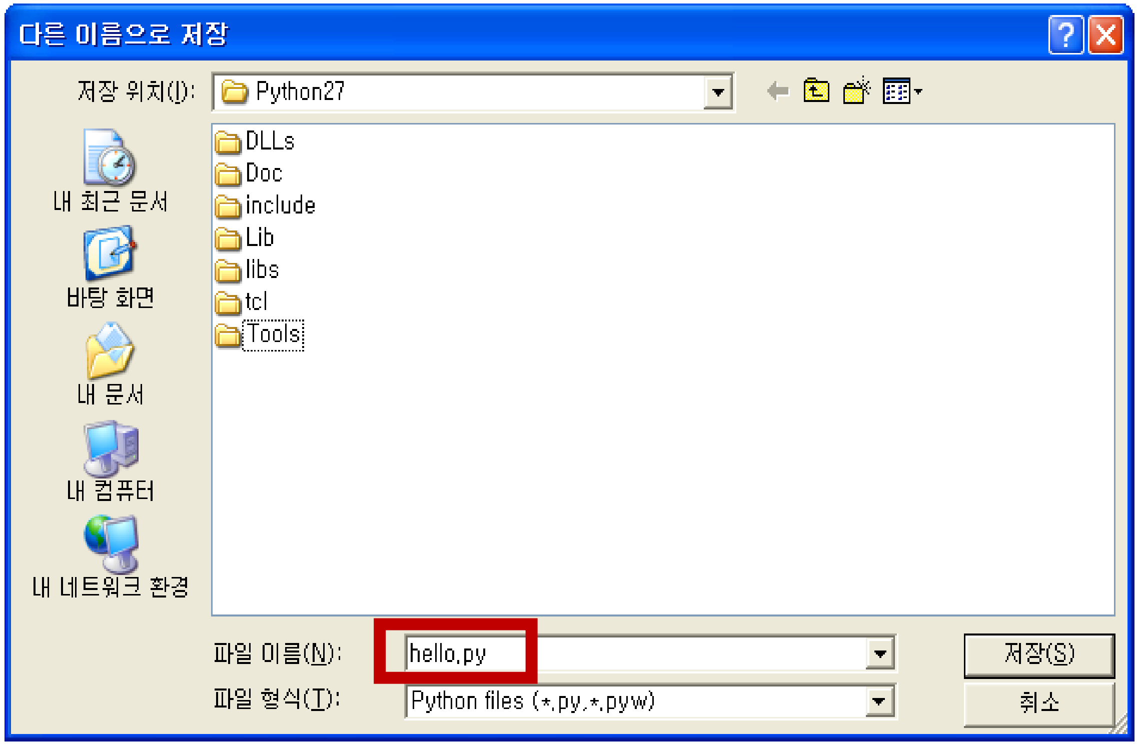
```
# This program says hello and asks for my name.
print 'Hello world!'
print 'What is your name?'
myName = raw_input()
print 'It is good to meet you, ' + myName
```

Write the first program "Hello World!"

Programs "Hello World!"

- Saving Program



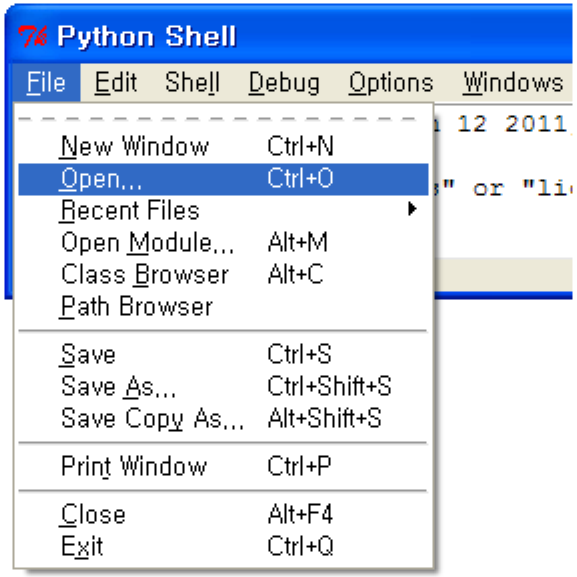


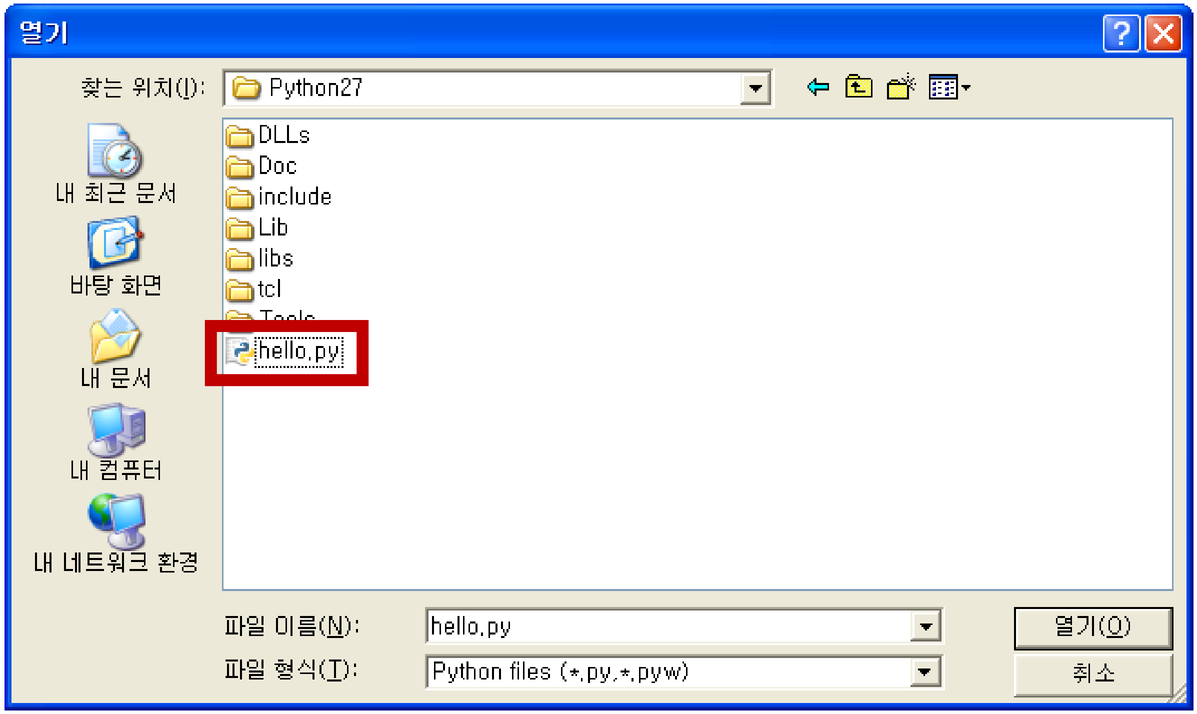
파일 이름(N): hello.py
파일 형식(T): Python files (*.py,*.pyw)

Write the first program "Hello World!"

Programs "Hello World!"

- Opening The Programs You've Saved

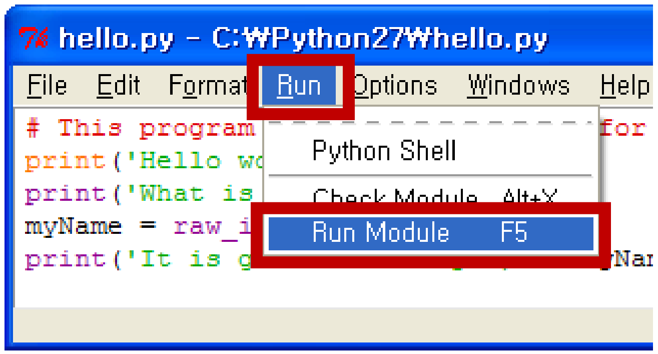




Write the first program "Hello World!"

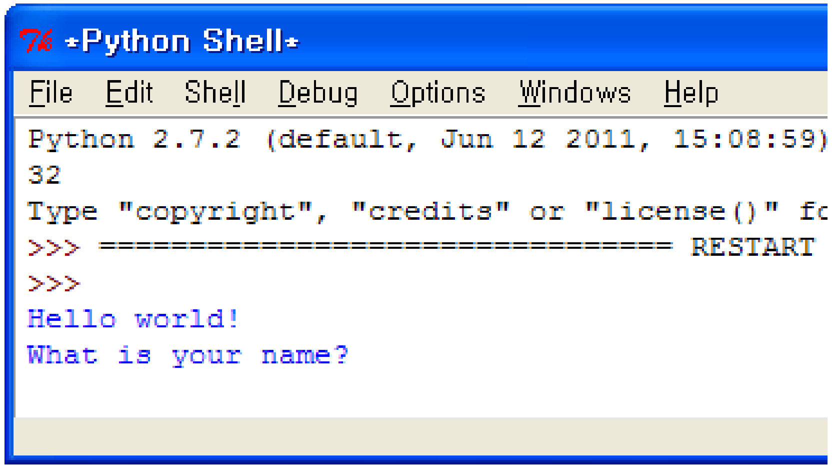
Programs "Hello World!"

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



hello.py - C:₩Python27₩hello.py
File Edit Format Run Options Windows Help
Python Shell
Check Module Alt+X
Run Module F5

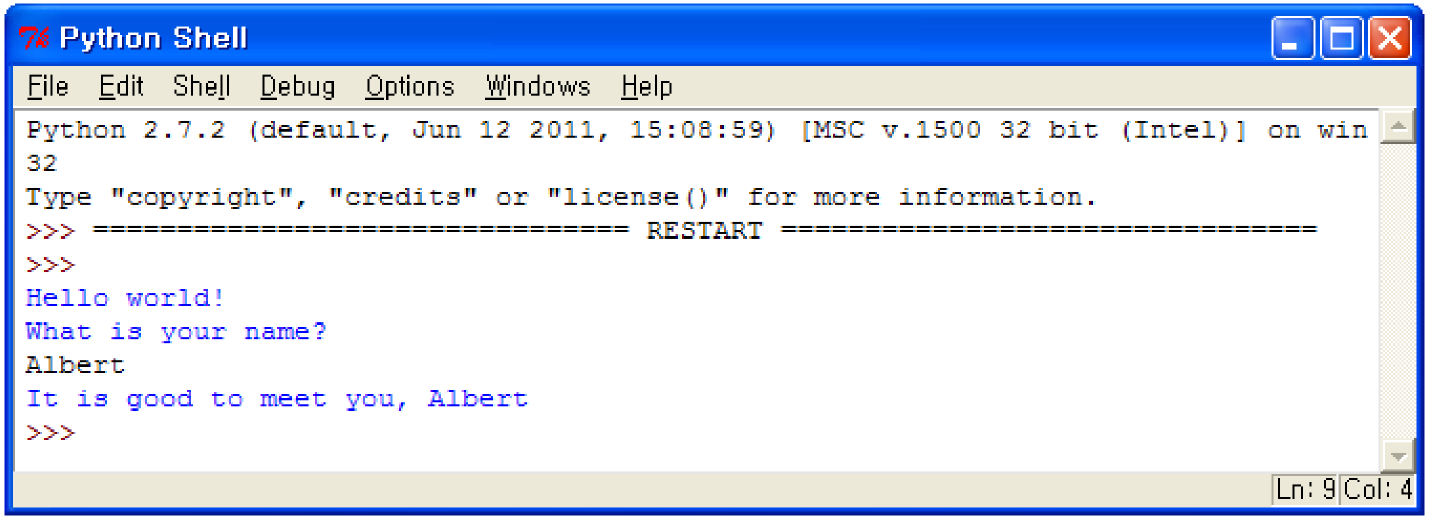
```
# This program                        for
print('Hello wo
print('What is
myName = raw_i
print('It is g                       Nar
```



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

Write the first program "Hello World!"

Programs "Hello World!"



```
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.

```
# 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.

```
print 'Hello world!'
print 'What is your name?'
```

Write the first program "Hello World!"

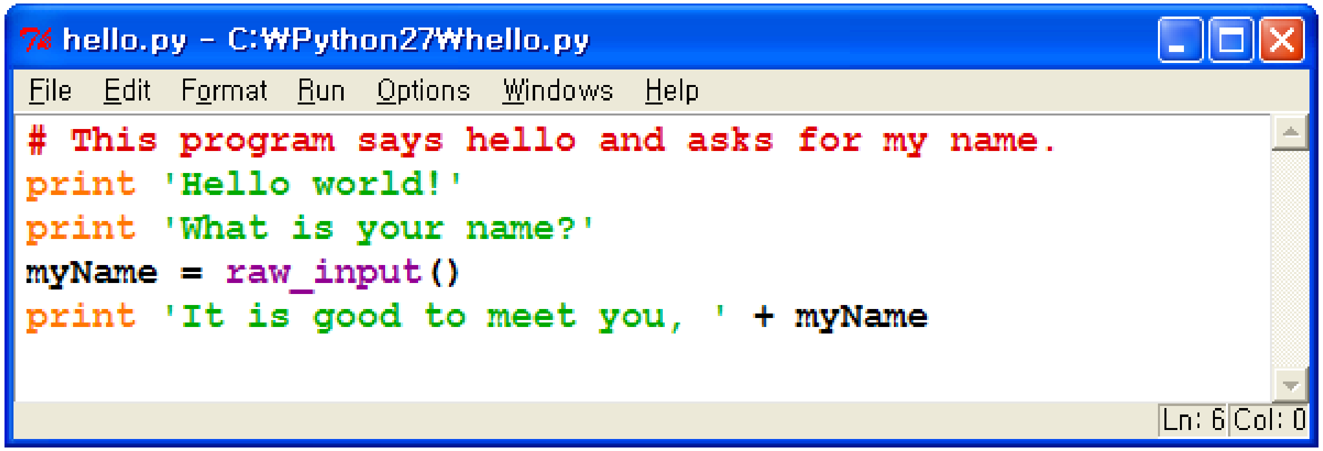
Code Explanation

- **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

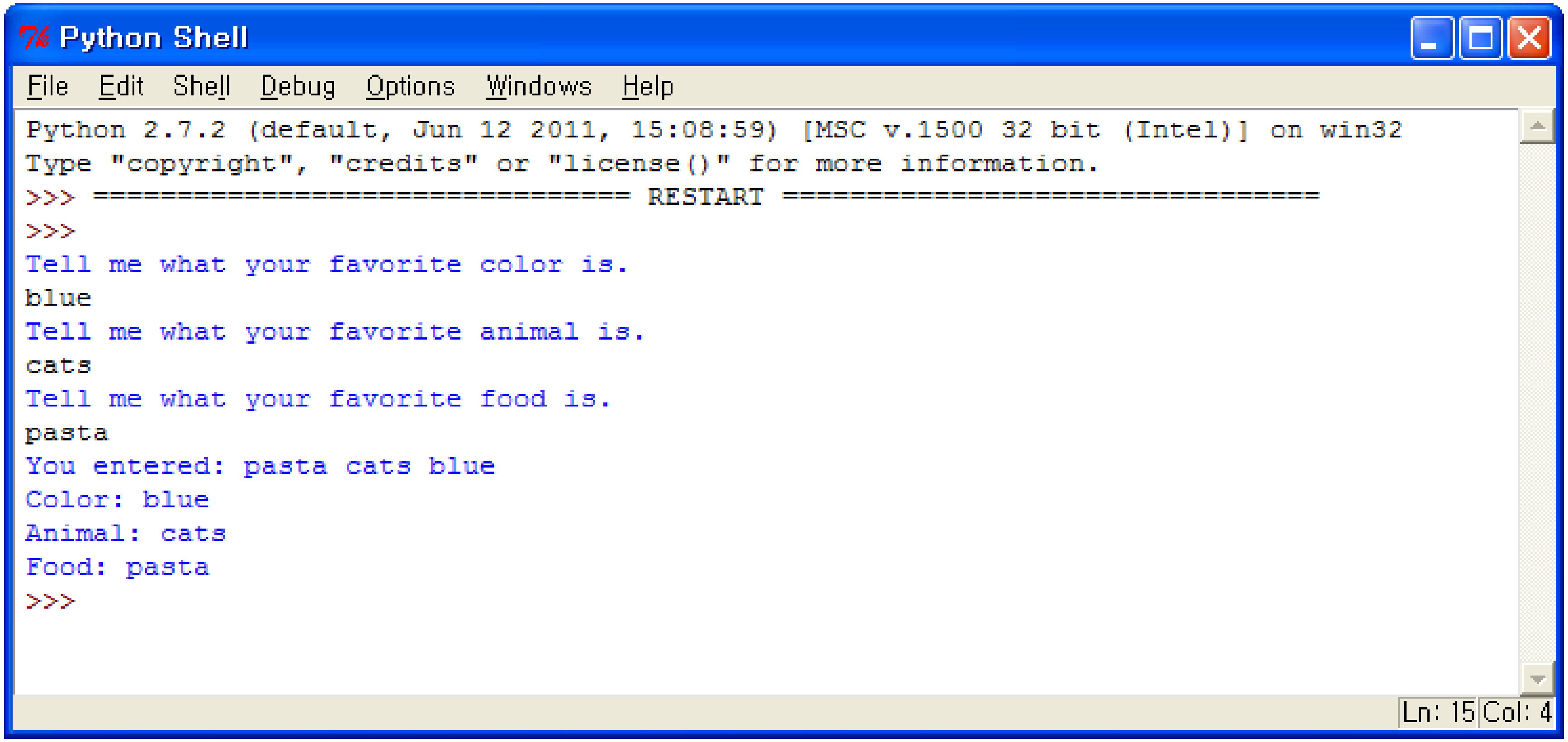

hello.py - C:₩Python27₩hello.py
File Edit Format Run Options Windows Help
Ln: 6 Col: 0

```
# 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
```

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

“My Favorite Stuff”

- Programs “My Favorite Stuff”



```
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
>>> 
```

"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.

```
# Favorite stuff
```

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

```
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.**

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

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

“My Favorite Stuff”

Code Explanation

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

```
# 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.

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

"My Favorite Stuff"

Code Explanation

- `print` statement
  - Another `print` statement

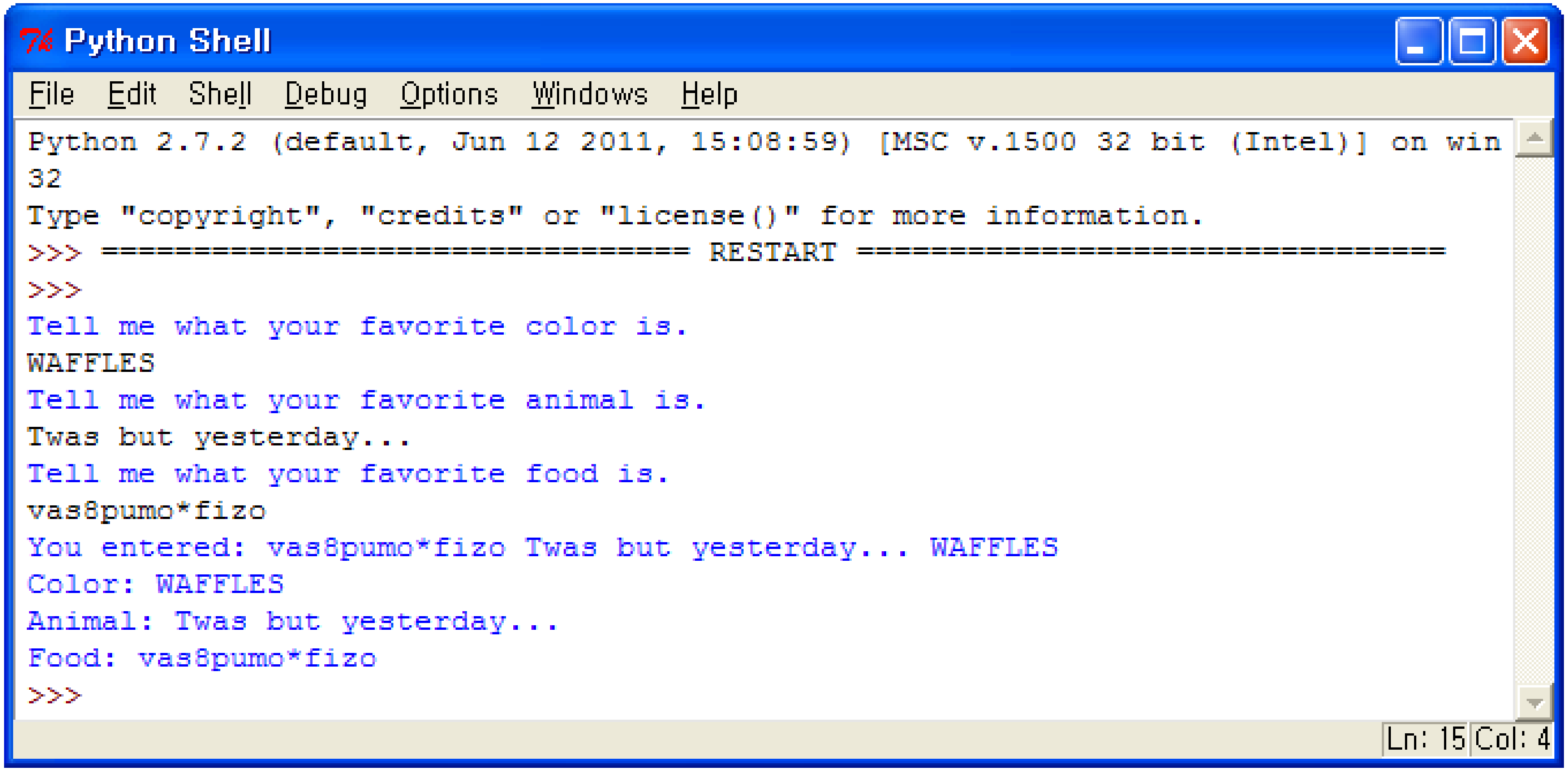
```
# print 'Here is a list of your favorite
  things.'
```

  - These three lines will display our favorite things again.

```
print 'Color: ' + favoriteColor
print 'Animal: ' + favoriteAnimal
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 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
>>>
```

"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.

```
# Favorite stuff 2
print 'Tell me what your favorite color is.'
q = raw_input()

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

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

# display our favorite stuff
print 'You entered: ' + q + ' ' + fizzy + ' ' + AbrahamLincoln
# print 'Here is a list of your favorite things.'
print 'Color: ' + q
print 'Animal: ' + fizzy
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.**

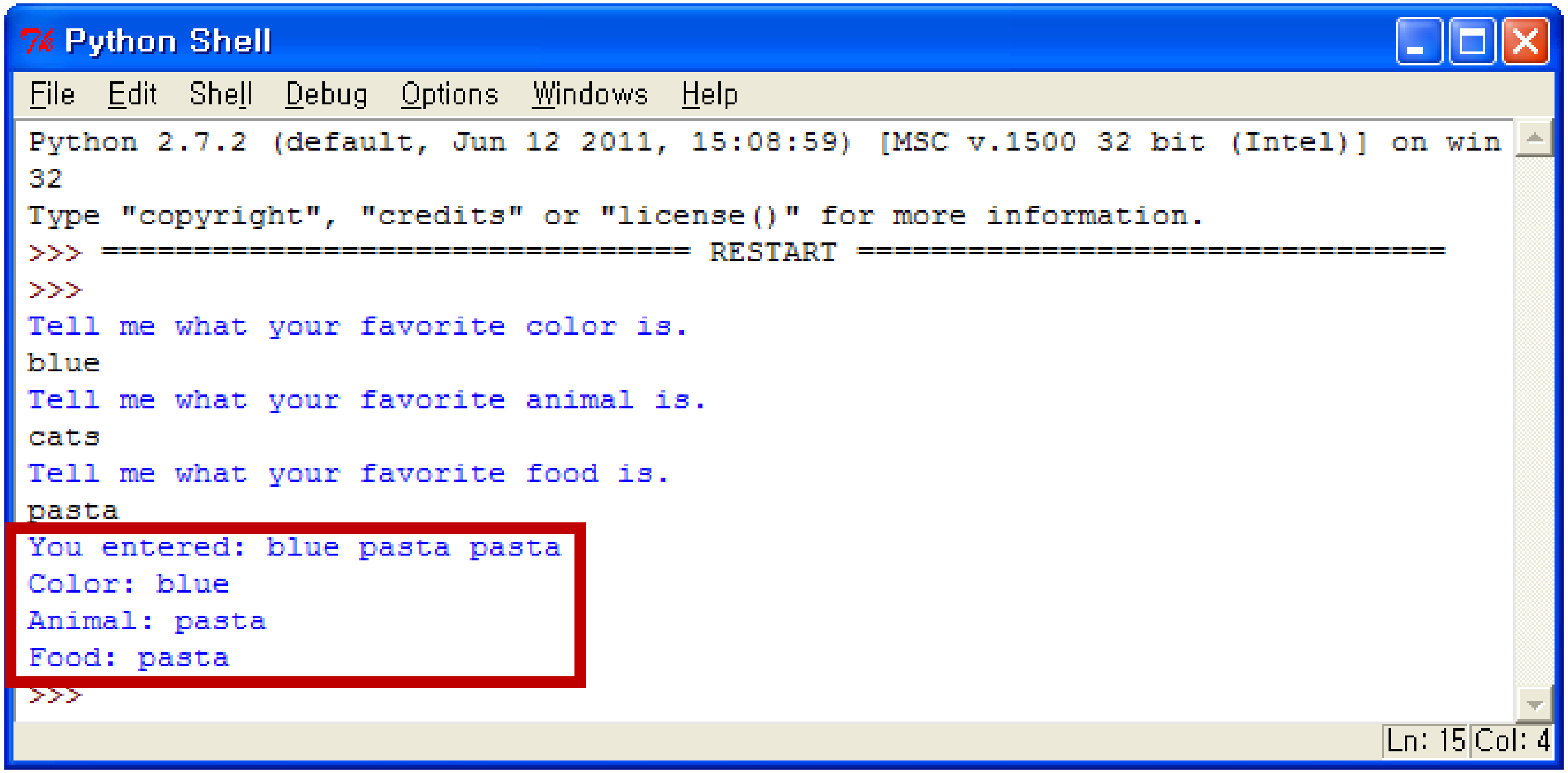
“My Favorite Stuff”

- **Quiz**
  - What happened here?

```
# Favorite stuff 3
print 'Tell me what your favorite color is.'
q = raw_input()

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

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

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

“My Favorite Stuff”

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



```
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.**

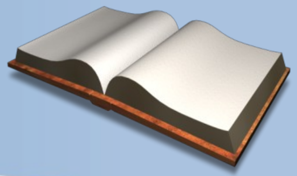
```
fizzy
Fizzy
FIZZY
fIzZy
```

  - – 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