**Definition:**

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted:** Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive:** You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented:** Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language:** Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

**History of Python**

- Python was developed by **Guido van Rossum** in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.
- Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, Unix shell, and other scripting languages.
- At the time when he began implementing Python, Guido van Rossum was also reading the published scripts from "Monty Python's Flying Circus" (a BBC comedy series from the seventies, in the unlikely case you didn't know). It occurred to him that he needed a name that was short, unique, and slightly mysterious, so he decided to call the language Python.
- Python is now maintained by a core development team at the institute, although **Guido van Rossum** still holds a vital role in directing its progress.
- Python 1.0 was released on **20 February, 1991**.
- Python 2.0 was released on **16 October 2000** and had many major new features, including a cycle detecting garbage collector and support for Unicode. With this release the development process was changed and became more transparent and community-backed.
- Python 3.0 (which early in its development was commonly referred to as Python 3000 or py3k), a major, backwards-incompatible release, was released on **3 December 2008** after a long period of testing. Many of its major features have been back ported to the backwards-compatible Python 2.6.x and 2.7.x version series.
- In January 2017 Google announced work on a Python 2.7 to go transcompiler, which The Register speculated was in response to Python 2.7's planned end-of-life.



**Python Features:**

Python's features include:

- **Easy-to-learn:** Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read:** Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain:** Python's source code is fairly easy-to-maintain.
- **A broad standard library:** Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode:** Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable:** Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable:** You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases:** Python provides interfaces to all major commercial databases.
- **GUI Programming:** Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of UNIX.
- **Scalable:** Python provides a better structure and support for large programs than shell scripting.

**Need of Python Programming**

- **Software quality**

Python code is designed to be *readable*, and hence reusable and maintainable—much more so than traditional scripting languages. The uniformity of Python code makes it easy to understand, even if you did not write it. In addition, Python has deep support for more advanced *software reuse* mechanisms, such as object-oriented (OO) and function programming.

- **Developer productivity**

Python boosts developer productivity many times beyond compiled or statically typed languages such as C, C++, and Java. Python code is typically *one-third to* less to debug, and less to maintain after the fact. Python programs also run immediately, without the lengthy compile and link steps required by some other tools, further boosting programmer speed. *Program portability* Most Python programs run unchanged on *all major computer platforms*. Porting Python code between Linux and Windows, for example, is usually just a matter of copying a script's code between machines.

- **Support libraries**

Python comes with a large collection of prebuilt and portable functionality, known as the *standard library*. This library supports an array of application-level programming tasks, from text pattern matching to network scripting. In addition, Python can be extended with both home grown libraries and a vast collection of third-party application support software. Python's *third-party domain* offers tools for website construction, numeric programming, serial port access, game development, and much more (see ahead for a sampling).

➤ **Component integration**

Python scripts can easily communicate with other parts of an application, using a variety of integration mechanisms. Such integrations allow Python to be used as a product *customization and extension* tool. Today, Python code can invoke C and C++ libraries, can be called from C and C++ programs, can integrate with Java and .NET components, can communicate over frameworks such as COM and Silverlight, can interface with devices over serial ports, and can interact over networks with interfaces like SOAP, XML-RPC, and CORBA. It is not a standalone tool.

➤ **Enjoyment**

Because of Python's ease of use and built-in toolset, it can make the act of programming *more pleasure than chore*. Although this may be an intangible benefit, its effect on productivity is an important asset. Of these factors, the first two (quality and productivity) are probably the most compelling benefits to most Python users, and merit a fuller description.

➤ **It's Object-Oriented**

Python is an object-oriented language, from the ground up. Its class model supports advanced notions such as polymorphism, operator overloading, and multiple inheritance; yet in the context of Python's dynamic typing, object-oriented programming (OOP) is remarkably easy to apply. Python's OOP nature makes it ideal as a scripting tool for object-oriented systems languages such as C++ and Java. For example, Python programs can subclass (specialized) classes implemented in C++ or Java.

➤ **It's Free**

Python is freeware—something which has lately been come to be called *open source software*. As with Tcl and Perl, you can get the entire system for free over the Internet. There are no restrictions on copying it, embedding it in your systems, or shipping it with your products. In fact, you can even sell Python, if you're so inclined. But don't get the wrong idea: "free" doesn't mean "unsupported". On the contrary, the Python online community responds to user queries with a speed that most commercial software vendors would do well to notice.

➤ **It's Portable**

Python is written in portable ANSI C, and compiles and runs on virtually every major platform in use today. For example, it runs on UNIX systems, Linux, MS-DOS, MS-Windows (95, 98, NT), Macintosh, Amiga, Be-OS, OS/2, VMS, QNX, and more. Further, Python programs are automatically compiled to portable *bytecode*, which runs the same on any platform with a compatible version of Python installed (more on this in the section "It's easy to use"). What that means is that Python programs that use the core language run the same on UNIX, MS-Windows, and any other system with a Python interpreter.

➤ **It's Powerful**

From a features perspective, Python is something of a hybrid. Its tool set places it between traditional scripting languages (such as Tcl, Scheme, and Perl), and systems languages (such as C, C++, and Java). Python provides all the simplicity and ease of use of a scripting language, along with more advanced programming tools typically found in systems development languages.

➤ **Automatic memory management**

Python automatically allocates and reclaims ("garbage collects") objects when no longer used, and most grow and shrink on demand; Python, not you, keeps track of low-level memory details.

➤ **Programming-in-the-large support**

Finally, for building larger systems, Python includes tools such as modules, classes, and exceptions; they allow you to organize systems into components, do OOP, and handle events gracefully.

➤ **It's Mixable**

Python programs can be easily "glued" to components written in other languages. In technical terms, by employing the Python/C integration APIs, Python programs can be both extended by (called to) components written in C or C++, and embedded in (called by) C or C++ programs. That means you can add functionality to the Python system as needed and use Python programs within other environments or systems.

➤ **It's Easy to Use**

For many, Python's combination of rapid turnaround and language simplicity make programming more fun than work. To run a Python program, you simply type it and run it. There are no intermediate compile and link steps (as when using languages such as C or C++). As with other interpreted languages, Python executes programs immediately, which makes for both an interactive programming experience and rapid turnaround after program changes. Strictly speaking, Python programs are compiled (translated) to an intermediate form called *bytecode*, which is then run by the interpreter.

➤ **It's Easy to Learn**

This brings us to the topic of this book: compared to other programming languages, the core Python language is amazingly easy to learn. In fact, you can expect to be coding significant Python programs in a matter of days (and perhaps in just hours, if you're already an experienced programmer).

➤ **Internet Scripting**

Python comes with standard Internet utility modules that allow Python programs to communicate over sockets, extract form information sent to a server-side CGI script, parse HTML, transfer files by FTP, process XML files, and much more. There are also a number of peripheral tools for doing Internet programming in Python. For instance, the HTMLGen and pythondoc systems generate HTML files from Python class-based descriptions, and the JPython system mentioned above provides for seamless Python/Java integration.

➤ **Database Programming**

Python's standard pickle module provides a simple object-persistence system: it allows programs to easily save and restore entire Python objects to files. For more traditional database demands, there are Python interfaces to Sybase, Oracle, Informix, ODBC, and more. There is even a portable SQL database API for Python that runs the same on a variety of underlying database systems, and a system named *gadfly* that implements an SQL database for Python programs.

➤ **Image Processing, AI, Distributed Objects, Etc.**

Python is commonly applied in more domains than can be mentioned here. But in general, many are just instances of Python's component integration role in action. By adding Python as a frontend to libraries of components written in a compiled language such as C, Python becomes useful for scripting in a variety of domains. For instance, image processing for Python is implemented as a set of library components implemented in a compiled language such as C, along with a Python frontend layer on top used to configure and launch the compiled components.

### Who Uses Python Today?

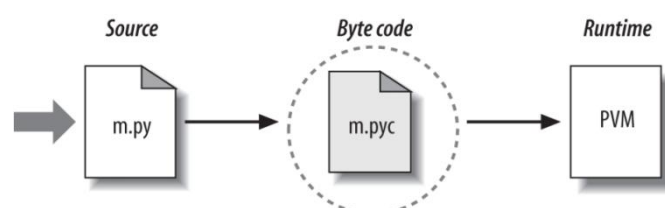
1. *Google* makes extensive use of Python in its web search systems.
2. The popular *YouTube* video sharing service is largely written in Python.
3. The *Dropbox* storage service codes both its server and desktop client software primarily in Python.
4. The *Raspberry Pi* single-board computer promotes Python as its educational language.
5. The widespread *BitTorrent* peer-to-peer file sharing system began its life as a Python program.
6. Google's *App Engine* web development framework uses Python as an application language.
7. *Maya*, a powerful integrated 3D modeling and animation system, provides a Python scripting API.
8. *Intel*, *Cisco*, *Hewlett-Packard*, *Seagate*, *Qualcomm*, and *IBM* use Python for hardware testing.
9. *NASA*, *Los Alamos*, *Fermilab*, *JPL*, and others use Python for scientific programming tasks.

### Byte code Compilation:

Python first compiles your source code (the statements in your file) into a format known as byte code. Compilation is simply a translation step, and byte code is a lower-level, platform independent representation of your source code. Roughly, Python translates each of your source statements into a group of byte code instructions by decomposing them into individual steps. This byte code translation is performed to speed execution —byte code can be run much more quickly than the original source code statements in your text file.

### The Python Virtual Machine:

Once your program has been compiled to byte code (or the byte code has been loaded from existing *.pyc* file), it is shipped off for execution to something generally known as the python virtual machine (PVM).



## Applications of Python:

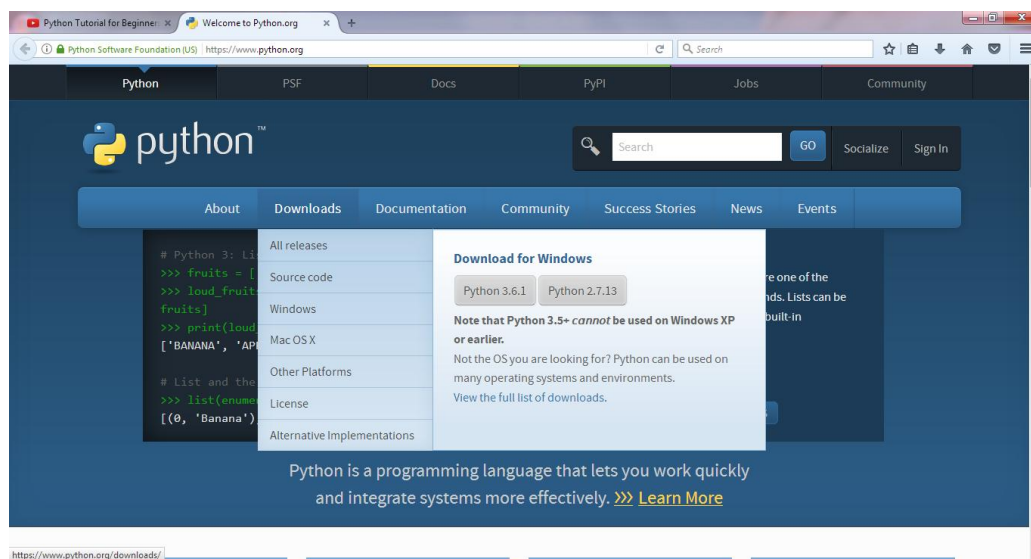
1. Systems Programming
2. GUIs
3. Internet Scripting
4. Component Integration
5. Database Programming
6. Rapid Prototyping
7. Numeric and Scientific Programming

## What Are Python's Technical Strengths?

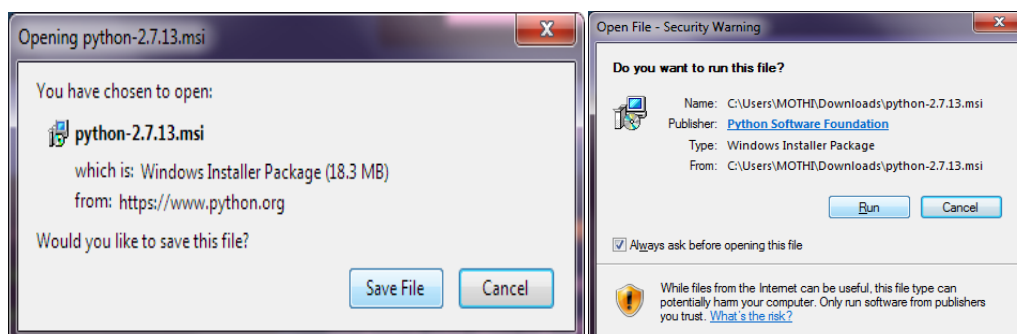
1. It's Object-Oriented and Functional
2. It's Free
3. It's Portable
4. It's Powerful
5. It's Mixable
6. It's Relatively Easy to Use
7. It's Relatively Easy to Learn

## Download and installation Python software:

**Step 1:** Go to website [www.python.org](http://www.python.org) and click downloads select version which you want.

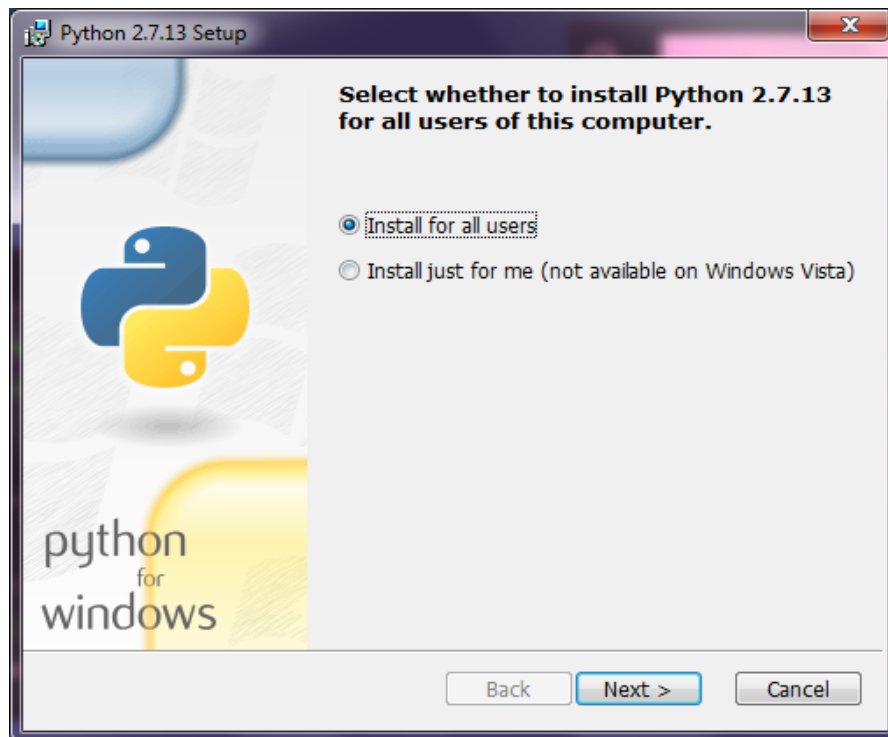


**Step 2:** Click on **Python 2.7.13** and download. After download open the file.

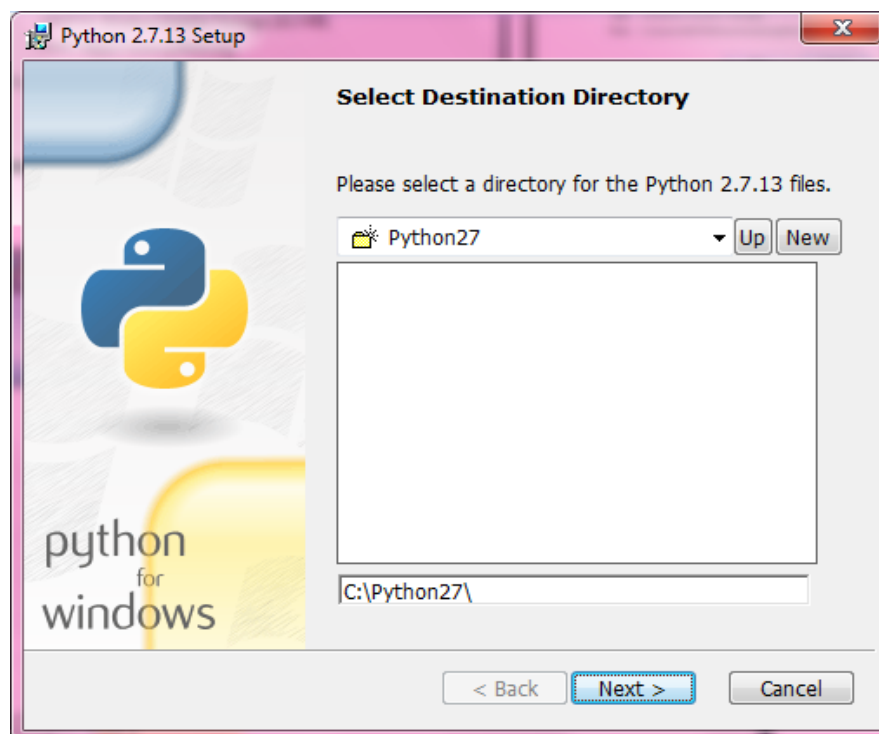




**Step 3:** Click on **Next** to continue.



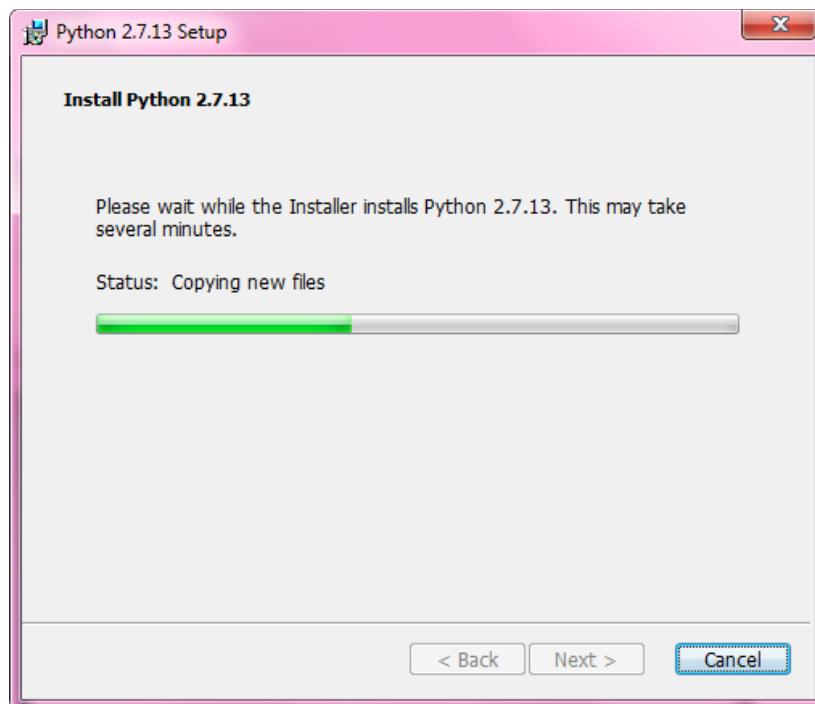
**Step 4:** After installation location will be displayed. The Default location is **C:\Python27**. Click on next to continue.



**Step 5:** After the python interpreter and libraries are displayed for installation. Click on Next to continue.



**Step 6:** The installation has been processed.



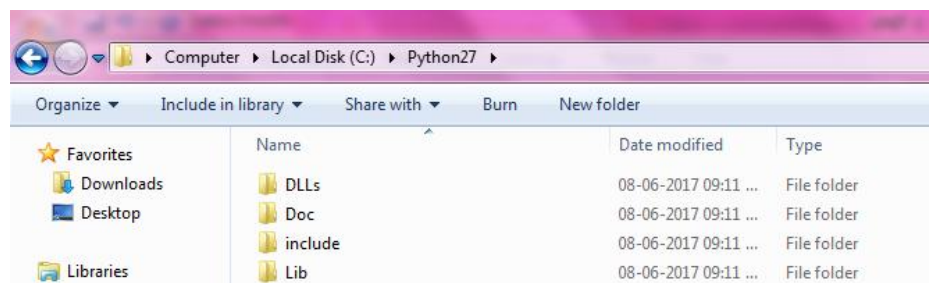


**Step 7:** Click the **Finish** to complete the installation.

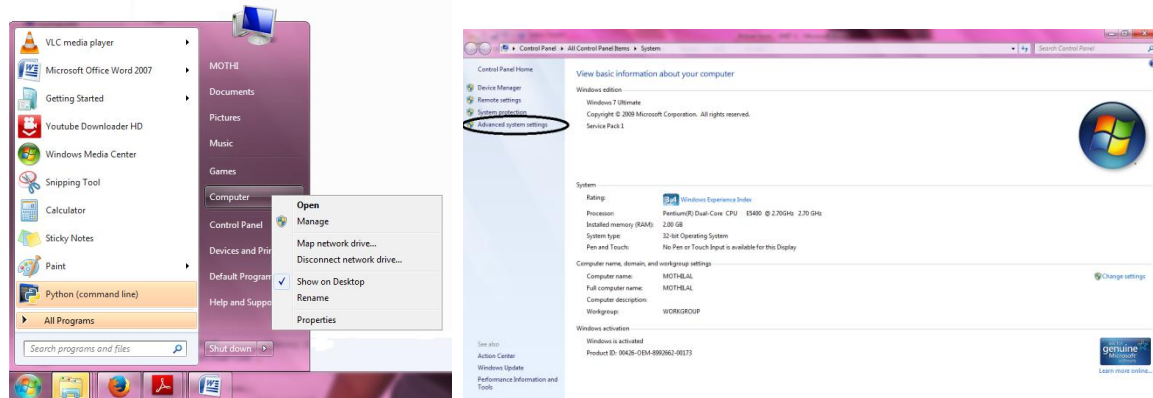


### Setting up PATH to python:

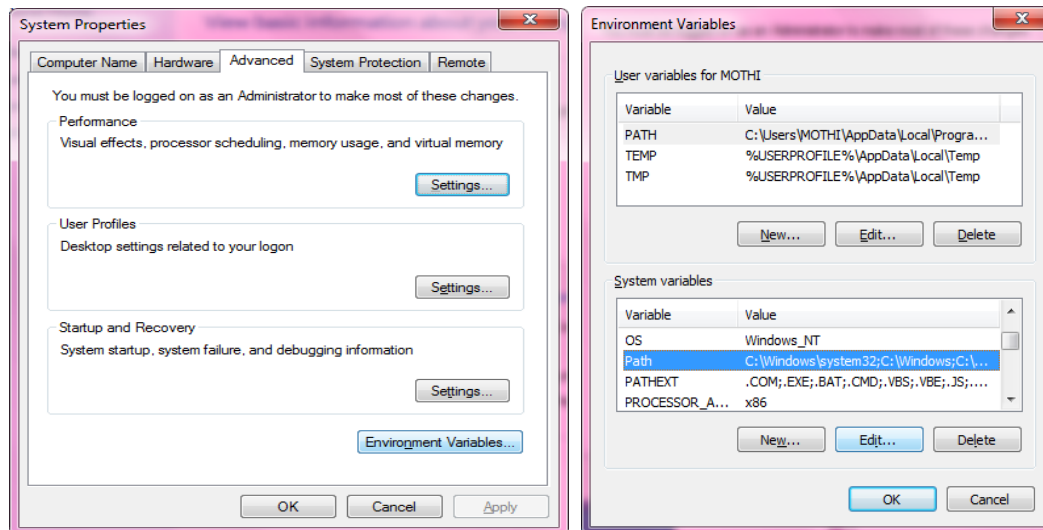
- Programs and other executable files can be in many directories, so operating systems provide a search path that lists the directories that the OS searches for executables.
- The path is stored in an environment variable, which is a named string maintained by the operating system. This variable contains information available to the command shell and other programs.
- Copy the Python installation location C:\Python27



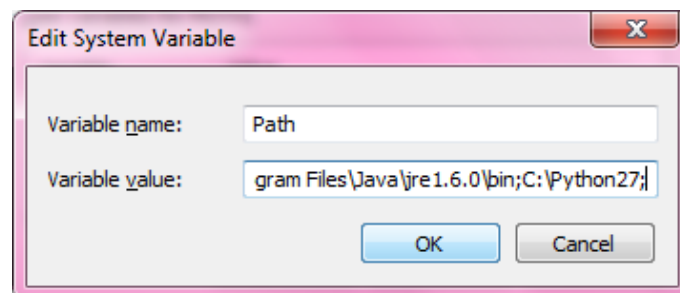
- Right-click the My Computer icon on your desktop and choose **Properties**. And then select **Advanced System properties**.



- Goto **Environment Variables** and go to **System Variables** select **Path** and click on **Edit**.



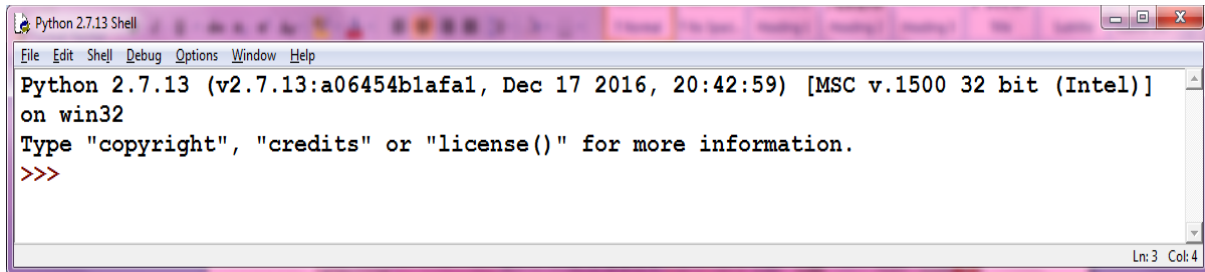
- Add semicolon (;) at end and copy the location **C:\Python27** and give semicolon (;) and click OK.



## Running Python:

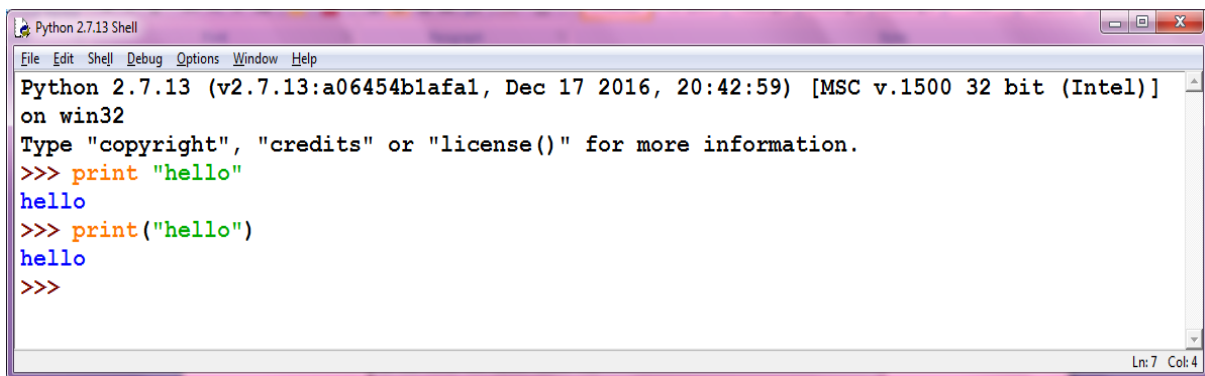
### a. Running Python Interpreter:

Python comes with an interactive interpreter. When you type python in your shell or command prompt, the python interpreter becomes active with a >>> prompt and waits for your commands.



```
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>>
```

Now you can type any valid python expression at the prompt. Python reads the typed expression, evaluates it and prints the result.

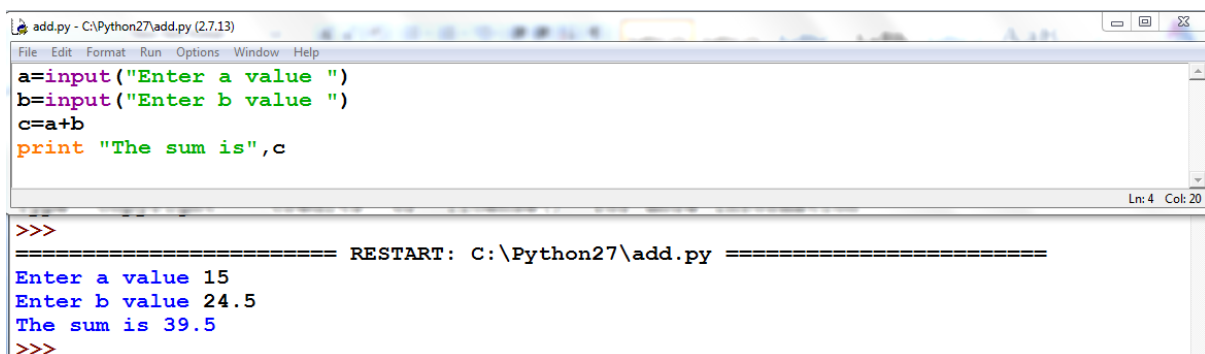


```
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>> print "hello"
hello
>>> print("hello")
hello
>>>
```

### b. Running Python Scripts in IDLE:

- Goto **File** menu click on New File (CTRL+N) and write the code and save add.py

```
a=input("Enter a value ")
b=input("Enter b value ")
c=a+b
print "The sum is",c
```
- And run the program by pressing F5 or Run→Run Module.

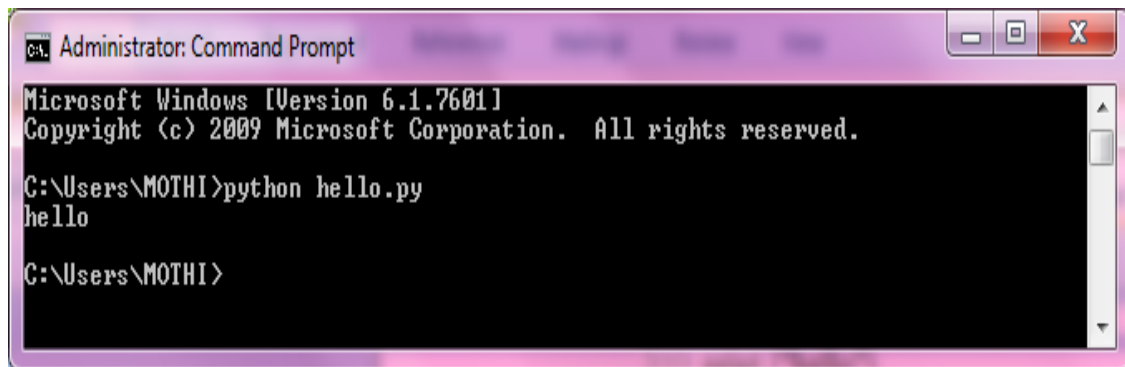


```
add.py - C:\Python27\add.py (2.7.13)
File Edit Format Run Options Window Help
a=input("Enter a value ")
b=input("Enter b value ")
c=a+b
print "The sum is",c
Ln: 4 Col: 20

>>>
===== RESTART: C:\Python27\add.py =====
Enter a value 15
Enter b value 24.5
The sum is 39.5
>>>
```

**c. Running python scripts in Command Prompt:**

- Before going to run we have to check the PATH in environment variables.
- Open your text editor, type the following text and save it as hello.py.  
`print "hello"`
- And run this program by calling `python hello.py`. Make sure you change to the directory where you saved the file before doing it.



```
Administrator: Command Prompt
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\MOTHI>python hello.py
hello

C:\Users\MOTHI>
```

**Variables:**

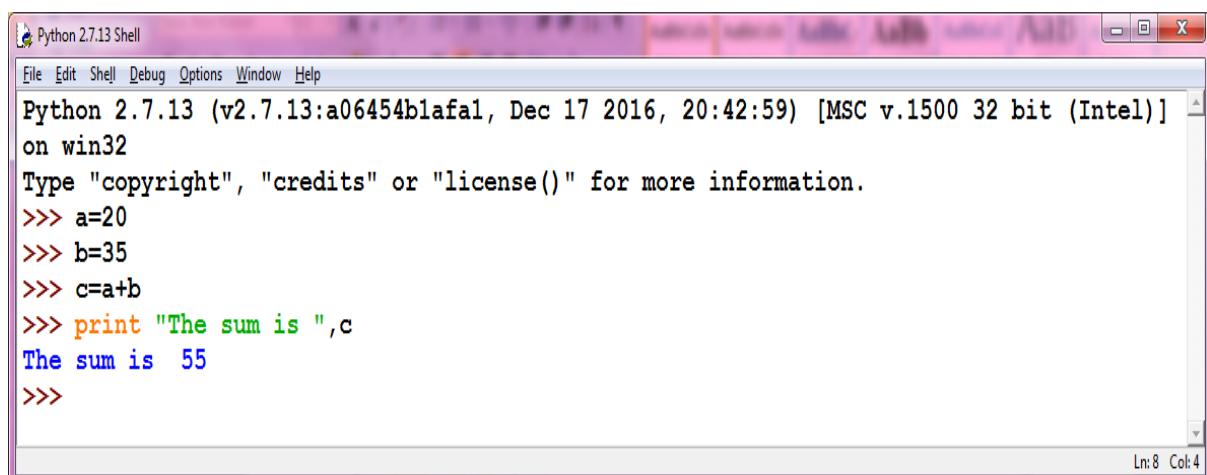
Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

**Assigning Values to Variables**

Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable. The equal sign (=) is used to assign values to variables.

The operand to the left of the = operator is the name of the variable and the operand to the right of the = operator is the value stored in the variable. For example –



```
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454b1afal, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>> a=20
>>> b=35
>>> c=a+b
>>> print "The sum is ",c
The sum is 55
>>>
```

**Multiple Assignments to variables:**

Python allows you to assign a single value to several variables simultaneously.

For example –

**a = b = c = 1**

Here, an integer object is created with the value 1, and all three variables are assigned to the same memory location. You can also assign multiple objects to multiple variables.

For example –

**a, b, c = 1, 2.5, "mothi"**

Here, two integer objects with values 1 and 2 are assigned to variables a and b respectively, and one string object with the value "john" is assigned to the variable c.

**KEYWORDS**

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

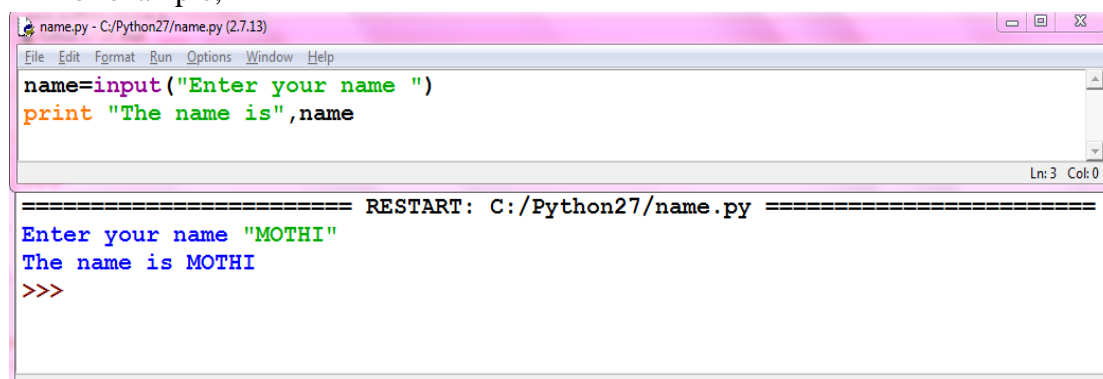
and	elif	if	print
as	else	import	raise
assert	except	in	return
break	exec	is	try
class	finally	lambda	while
continue	for	not	with
def	from	or	yield
del	global	pass	

**INPUT Function:**

To get input from the user you can use the input function. When the input function is called the program stops running the program, prompts the user to enter something at the keyboard by printing a string called the prompt to the screen, and then waits for the user to press the Enter key. The user types a string of characters and presses enter. Then the input function returns that string and Python continues running the program by executing the next statement after the input statement.

Python provides the function input(). input has an optional parameter, which is the prompt string.

For example,



```
name.py - C:/Python27/name.py (2.7.13)
File Edit Format Run Options Window Help
name=input("Enter your name ")
print "The name is",name
Ln: 3 Col: 0

===== RESTART: C:/Python27/name.py =====
Enter your name "MOTHI"
The name is MOTHI
>>>
```

## OUTPUT function:

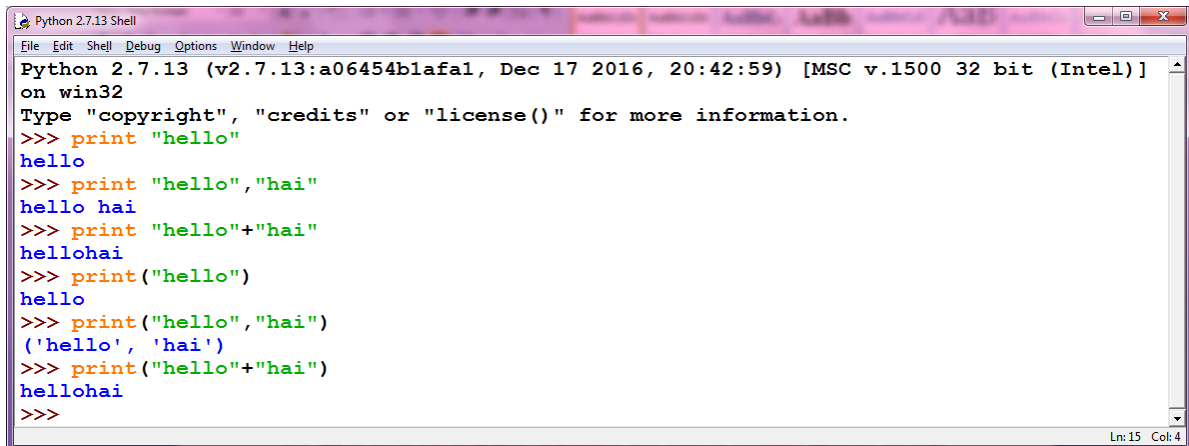
We use the `print()` function or `print` keyword to output data to the standard output device (screen). This function prints the object/string written in function.

The actual syntax of the `print()` function is

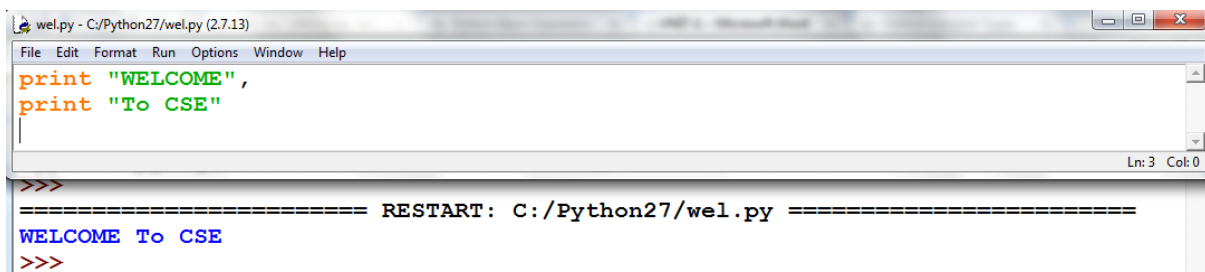
**`print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)`**

Here, objects is the value(s) to be printed.

The `sep` separator is used between the values. It defaults into a space character. After all values are printed, `end` is printed. It defaults into a new line ( `\n` ).



```
Python 2.7.13 Shell
File Edit Shell Debug Options Window Help
Python 2.7.13 (v2.7.13:a06454b1afaf1, Dec 17 2016, 20:42:59) [MSC v.1500 32 bit (Intel)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>> print "hello"
hello
>>> print "hello","hai"
hello hai
>>> print "hello"+"hai"
hellohai
>>> print("hello")
hello
>>> print("hello","hai")
('hello', 'hai')
>>> print("hello"+"hai")
hellohai
>>>
```



```
wel.py - C:/Python27/wel.py (2.7.13)
File Edit Format Run Options Window Help
print "WELCOME",
print "To CSE"

>>>

===== RESTART: C:/Python27/wel.py =====
WELCOME To CSE
>>>
```

## Indentation

Code blocks are identified by indentation rather than using symbols like curly braces. Without extra symbols, programs are easier to read. Also, indentation clearly identifies which block of code a statement belongs to. Of course, code blocks can consist of single statements, too. When one is new to Python, indentation may come as a surprise. Humans generally prefer to avoid change, so perhaps after many years of coding with brace delimitation, the first impression of using pure indentation may not be completely positive. However, recall that two of Python's features are that it is simplistic in nature and easy to read.

Python does not support braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation. All the continuous lines indented with same number of spaces would form a block. Python strictly follow indentation rules to indicate the blocks.



