Brief Python

Python Course for Programmers

Learn Python Language for Data Science

CHAPTER 1: DATA

DR. ERIC CHOU

IEEE SENIOR MEMBER

Objectives

- Python Installation
- Python Idle

Python Language

- Python is a general-purpose high-level programming language. It is an open source language, released under a GPL-compatible license.
- These chapters will help you learn Python 3 step by step. Tutorials are broken down into chapters, where each chapter contains several related topics packed with easy-to-understand explanations and real-world examples.
- These chapters are designed for beginners and professionals who want to learn Python programming language.

Installation

LECTURE 1

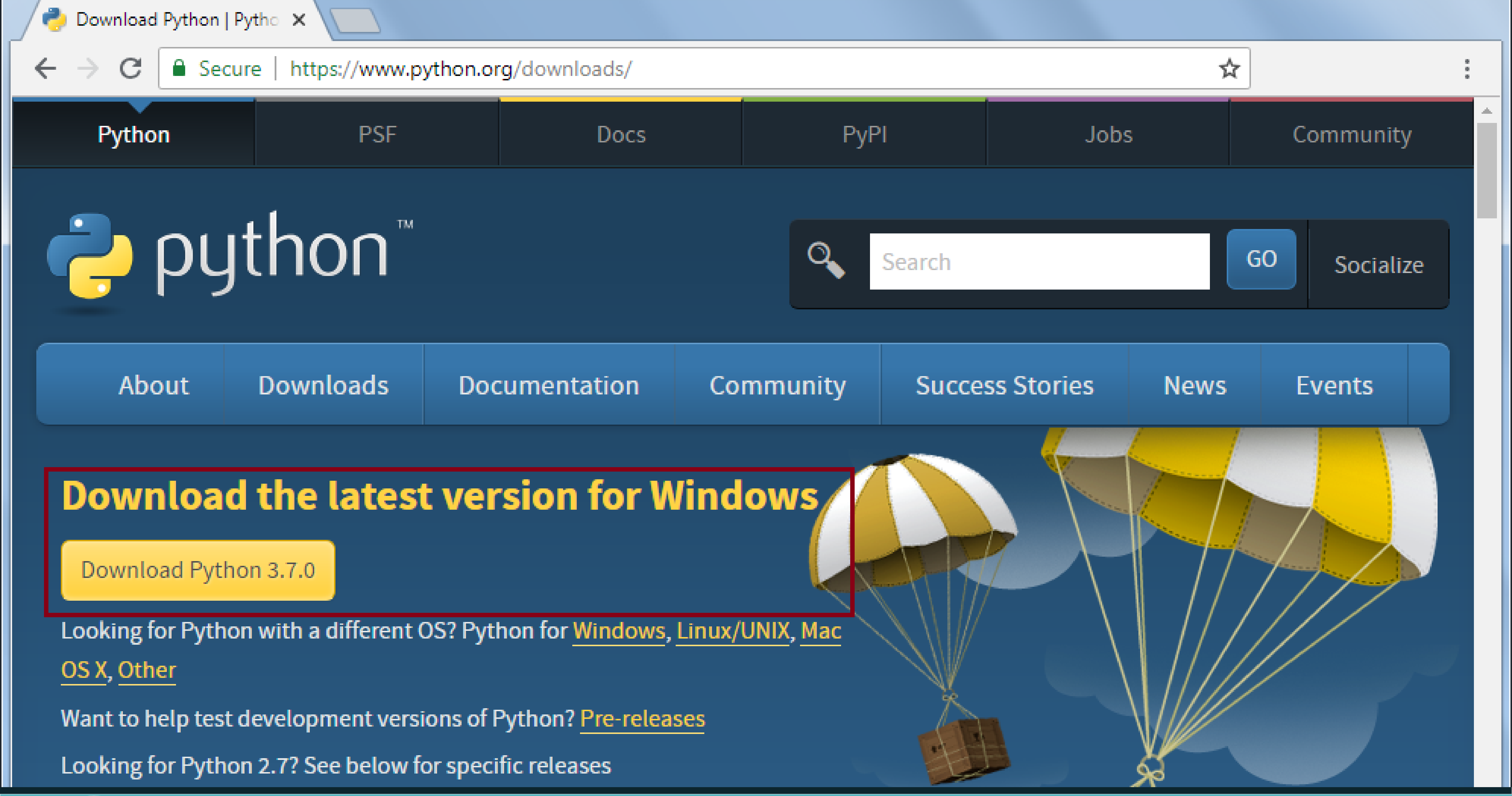
Python

https://www.python.org/

- Python is a dynamic object-oriented programming language that can be compared with Java and .NET languages as a general-purpose substrate for software development. It offers strong support for integrating with other technologies, higher programmer productivity throughout the development life cycle, and is well suited for complex projects.
- Python is the most rapidly growing open source programming language. According to InfoWorld its user base nearly doubled in 2004, and currently includes about 14% of all programmers.
- Python is being used in mission critical applications in the world's largest stock exchange, forms the basis for high end newspaper websites, runs on millions of cell phones, and is used in industries as diverse as ship building, feature length movie animation, and air traffic control.
- Python is available for most operating systems, including Windows, UNIX, Linux, and Mac OS.

Python Installation

1. Python Interpreter
2. Python idle
3. plugins, add-on’s, and site-packages



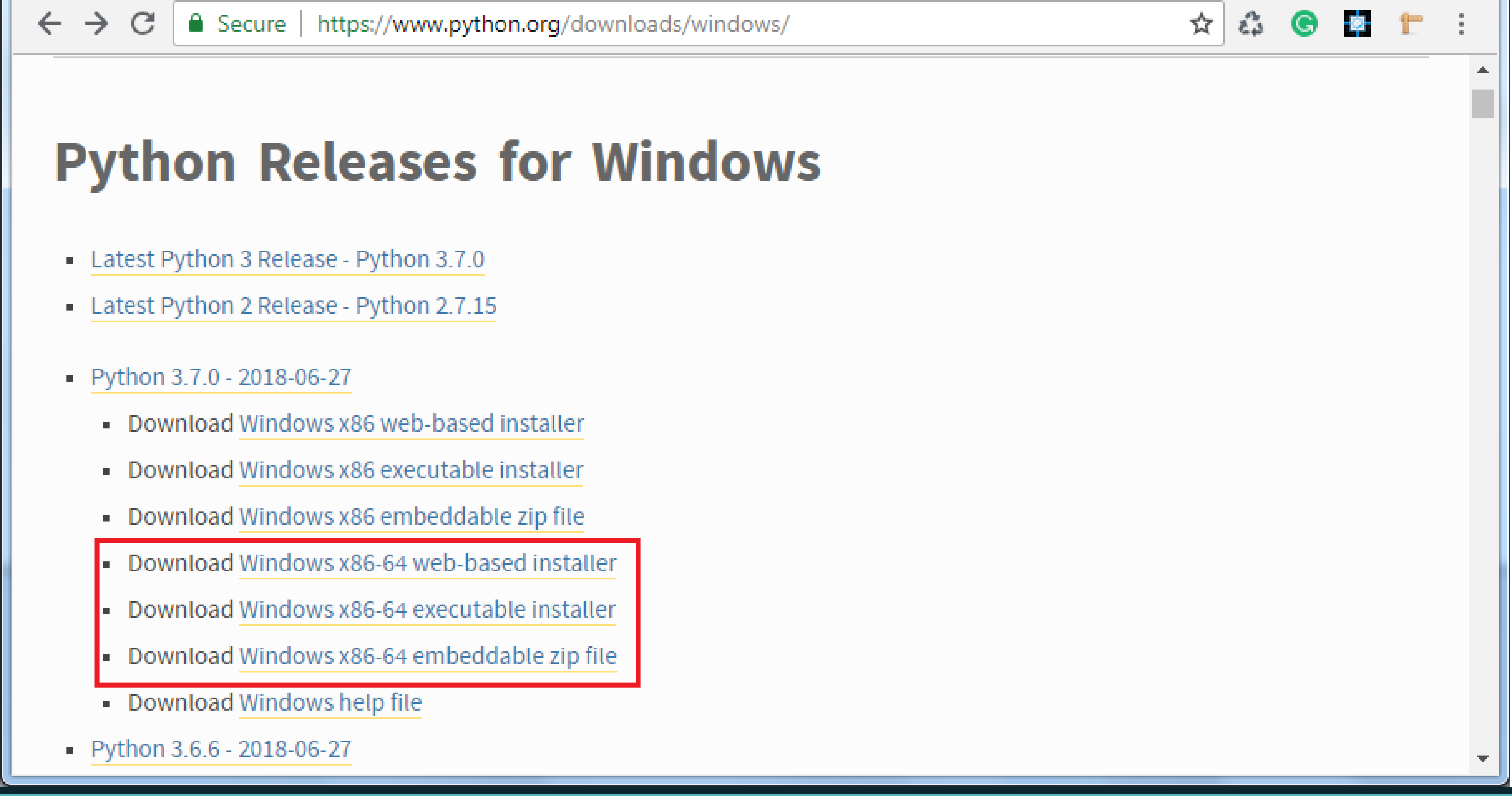
Download the latest version for Windows

Download Python 3.7.0

Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

Want to help test development versions of Python? Pre-releases

Looking for Python 2.7? See below for specific releases

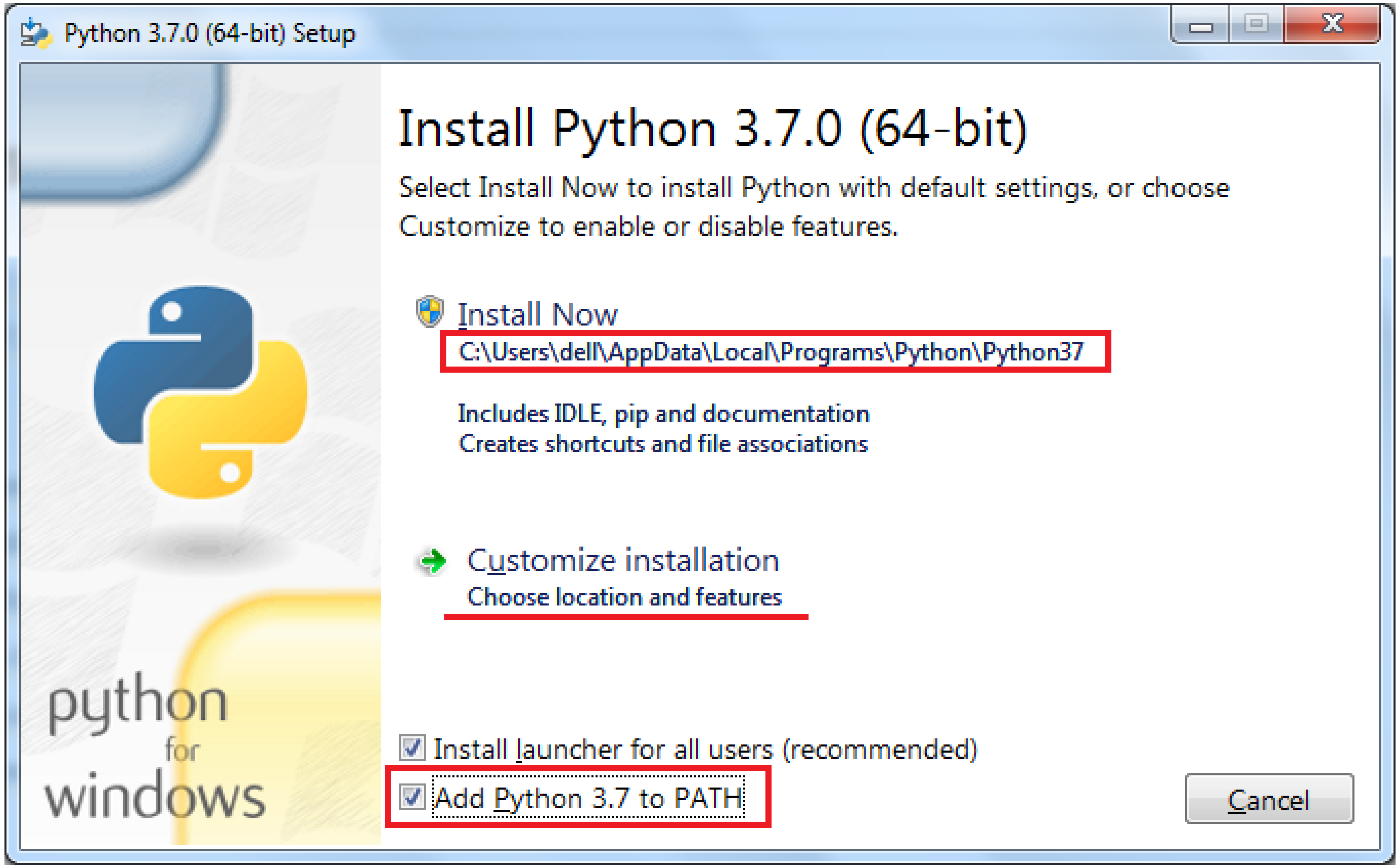


Secure | https://www.python.org/downloads/windows/

Python Releases for Windows

- Latest Python 3 Release - Python 3.7.0
- Latest Python 2 Release - Python 2.7.15

- Python 3.7.0 - 2018-06-27
  - Download Windows x86 web-based installer
  - Download Windows x86 executable installer
  - Download Windows x86 embeddable zip file
  - Download Windows x86-64 web-based installer
  - Download Windows x86-64 executable installer
  - Download Windows x86-64 embeddable zip file
  - Download Windows help file
- Python 3.6.6 - 2018-06-27



Install Python 3.7.0 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

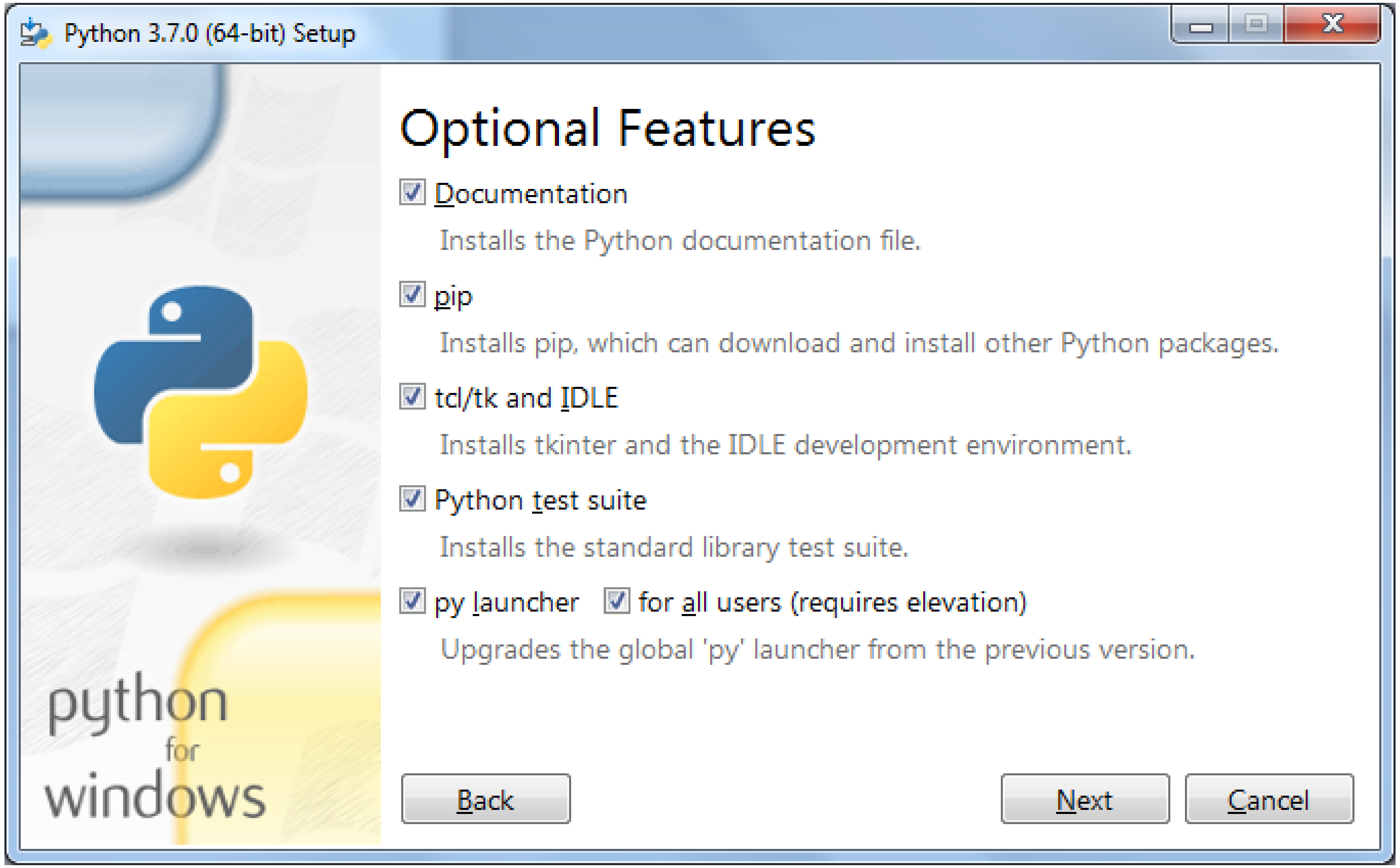
Install Now

C:\Users\dell\AppData\Local\Programs\Python\Python37

Includes IDLE, pip and documentation
Creates shortcuts and file associations

Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☑ Add Python 3.7 to PATH



Python 3.7.0 (64-bit) Setup

Optional Features

[✓] Documentation

Installs the Python documentation file.

[✓] pip

Installs pip, which can download and install other Python packages.

[✓] tcl/tk and IDLE

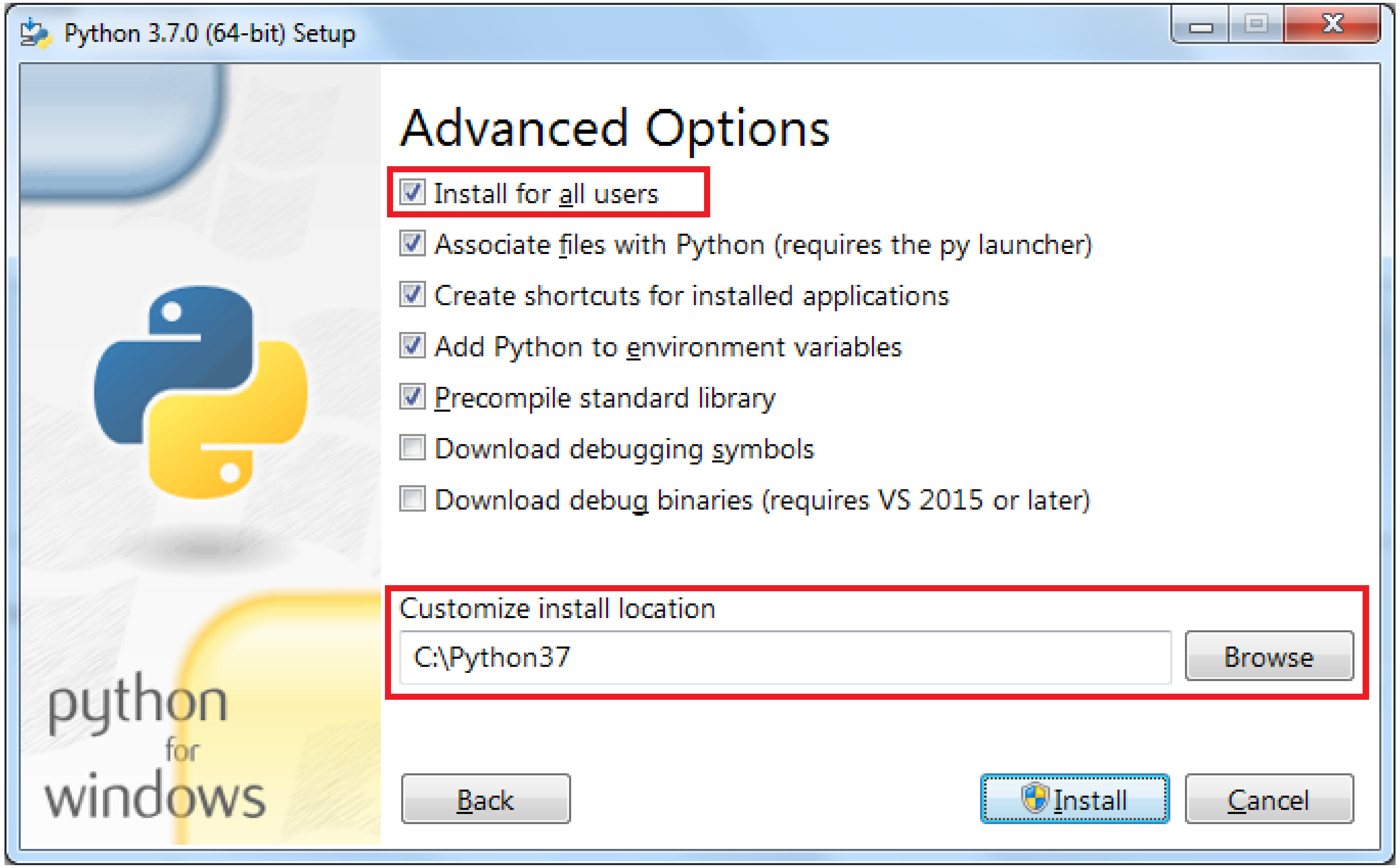
Installs tkinter and the IDLE development environment.

[✓] Python test suite

Installs the standard library test suite.

[✓] py launcher [✓] for all users (requires elevation)

Upgrades the global 'py' launcher from the previous version.



Python 3.7.0 (64-bit) Setup

Advanced Options

- [✓] Install for all users
- [✓] Associate files with Python (requires the py launcher)
- [✓] Create shortcuts for installed applications
- [✓] Add Python to environment variables
- [✓] Precompile standard library
- [ ] Download debugging symbols
- [ ] Download debug binaries (requires VS 2015 or later)

Customize install location

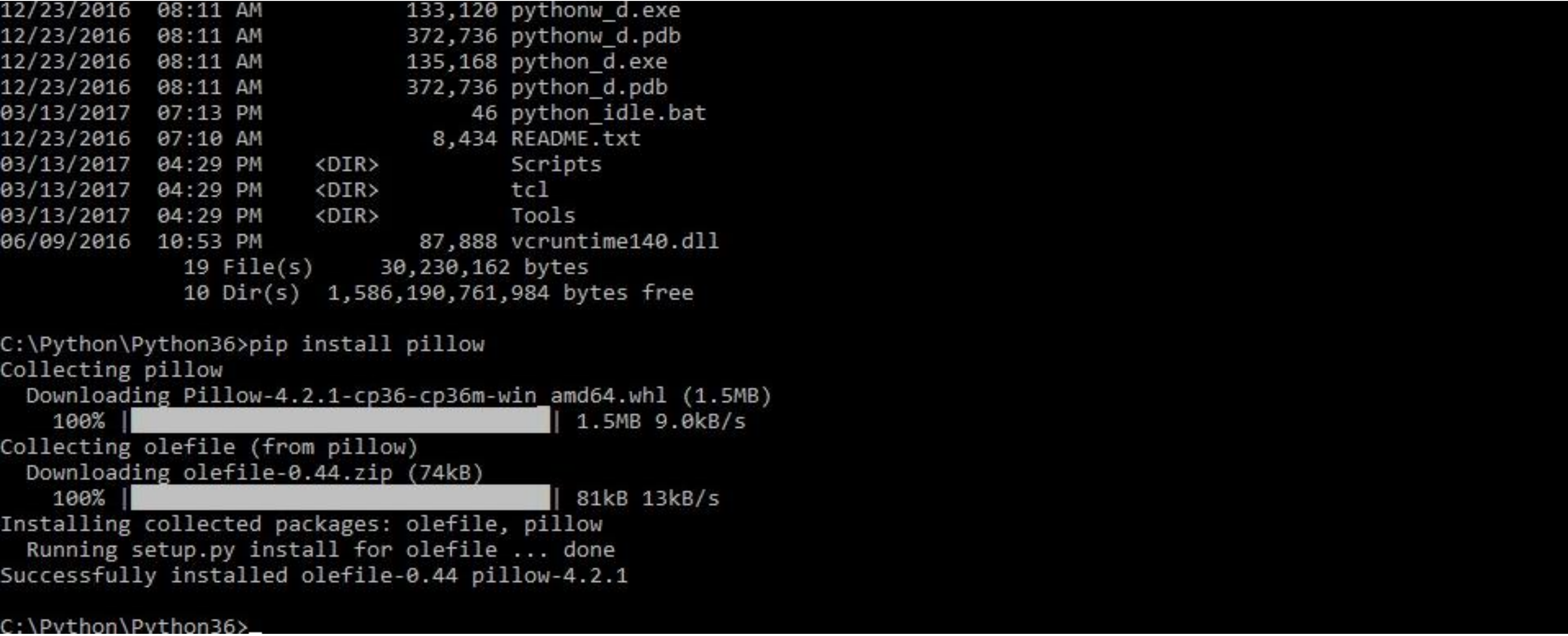
C:\Python37

Basic Graphics

HTTP://MCSP.WARTBURG.EDU/ZELLE/PYTHON/GRAPHICS.PY

Basic GUI package

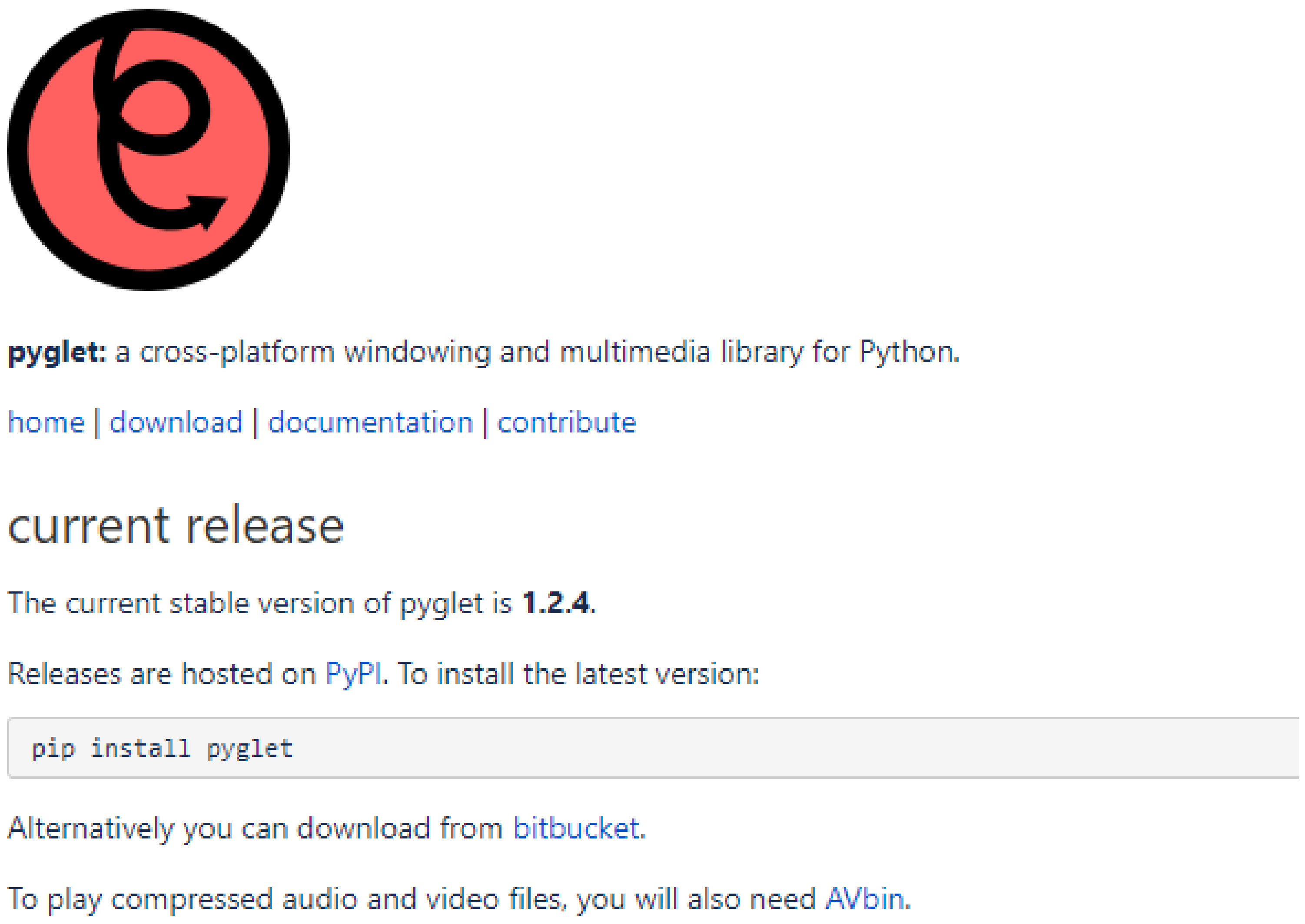
- Tkinter and (TTK) included in Python 3.x
- Pillow (Advanced PIL)
- PyGame
- Pyglet



```
12/23/2016  08:11 AM           133,120 pythonw_d.exe
12/23/2016  08:11 AM           372,736 pythonw_d.pdb
12/23/2016  08:11 AM           135,168 python_d.exe
12/23/2016  08:11 AM           372,736 python_d.pdb
03/13/2017  07:13 PM                46 python_idle.bat
12/23/2016  07:10 AM             8,434 README.txt
03/13/2017  04:29 PM    <DIR>          Scripts
03/13/2017  04:29 PM    <DIR>          tcl
03/13/2017  04:29 PM    <DIR>          Tools
06/09/2016  10:53 PM            87,888 vcruntime140.dll
              19 File(s)     30,230,162 bytes
              10 Dir(s)  1,586,190,761,984 bytes free

C:\Python\Python36>pip install pillow
Collecting pillow
  Downloading Pillow-4.2.1-cp36-cp36m-win_amd64.whl (1.5MB)
    100% |████████████████████████████████| 1.5MB 9.0kB/s
Collecting olefile (from pillow)
  Downloading olefile-0.44.zip (74kB)
    100% |████████████████████████████████| 81kB 13kB/s
Installing collected packages: olefile, pillow
  Running setup.py install for olefile ... done
Successfully installed olefile-0.44 pillow-4.2.1

C:\Python\Python36>_
```



pyglet: a cross-platform windowing and multimedia library for Python.

home | download | documentation | contribute

current release

The current stable version of pyglet is **1.2.4**.

Releases are hosted on PyPI. To install the latest version:

```
pip install pyglet
```

Alternatively you can download from bitbucket.

To play compressed audio and video files, you will also need AVbin.

Pygame Installation

Pygame requires Python; if you don't already have it, you can download it from python.org. **Use python 3.6.1** or greater, because it is much friendlier to newbies, and additionally runs faster.

The best way to install pygame is with the pip tool (which is what python uses to install packages). Note, this comes with python in recent versions. We use the --user flag to tell it to install into the home directory, rather than globally.

```
python3 -m pip install pygame --user
```

To see if it works, run one of the included examples:

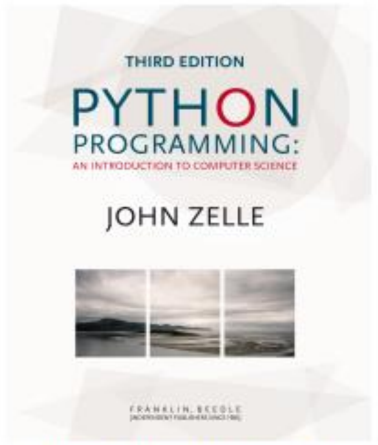
```
python3 -m pygame.examples.aliens
```

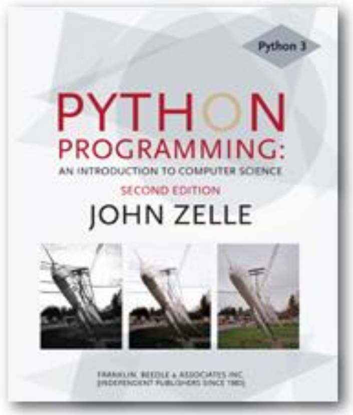
If it works, you are ready to go! Continue on to the tutorials.

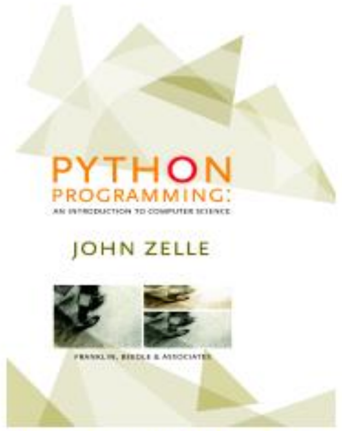
Python Programming: An Introduction to Computer Science

by John M. Zelle, Ph.D.

I have written an introductory CS textbook using Python. It is published by Franklin, Beedle & Associates. If you are interested in reviewing or adopting this textbook, contact Tom Sumner at FBA. Instructor resources including problem solutions and exam questions are available from the publisher. Click an edition below to access edition-specific public resources.


Third Edition (Python 3.x)


Second Edition (Python 3.x)


First Edition (Python 2.x)

Simple Graphics Library: graphics.py

I have developed an easy-to-use graphics library to introduce object-oriented concepts. The library is released under the GPL, so it is freely available for use and modification.

graphics.py	The simple graphics module used in textbook examples. This is version 5.0, and will work with both Python 2.x and 3.x. This is the latest version of the graphics library can be used with any edition of the book. It is known to work under Linux, Windows, and Mac OSX.
Graphics Reference (HTML)	Browseable reference documentation for the graphics package
Graphics Reference (PDF)	Downloadable/printable documentation for the graphics package

Numerical Packages NumPy/SciPy/Matplotlib

HTTPS://WWW.SCIPY.ORG/INSTALL.HTML#

NumPy

- **NumPy** is the fundamental package for scientific computing with **Python**. It contains among other things:
  - a powerful N-dimensional array object
  - sophisticated (broadcasting) functions
  - tools for integrating C/C++ and Fortran code
  - useful linear algebra, Fourier transform, and random number capabilities

- Besides its obvious scientific uses, **NumPy** can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows **NumPy** to seamlessly and speedily integrate with a wide variety of databases.

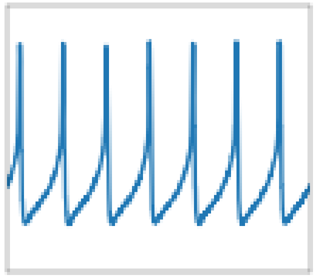
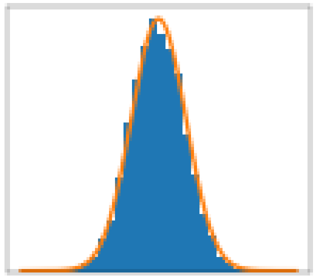
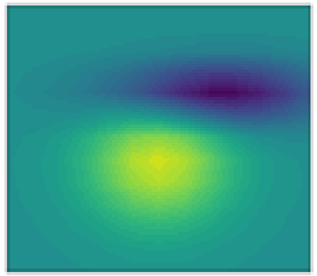
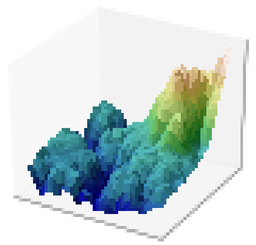
SciPy

CURRENT PACKAGES

- Special Functions (scipy.special)
- Signal Processing (scipy.signal)
- Image Processing (scipy.ndimage)
- Fourier Transforms (scipy.fftpack)
- Optimization (scipy.optimize)
- Numerical Integration (scipy.integrate)
- Linear Algebra (scipy.linalg)
- Input/Output (scipy.io)
- Statistics (scipy.stats)
- Fast Execution (scipy.weave)
- Clustering Algorithms (scipy.cluster)
- Sparse Matrices (scipy.sparse)
- Interpolation (scipy.interpolate)
- More (e.g. scipy.odr, scipy.maxentropy)

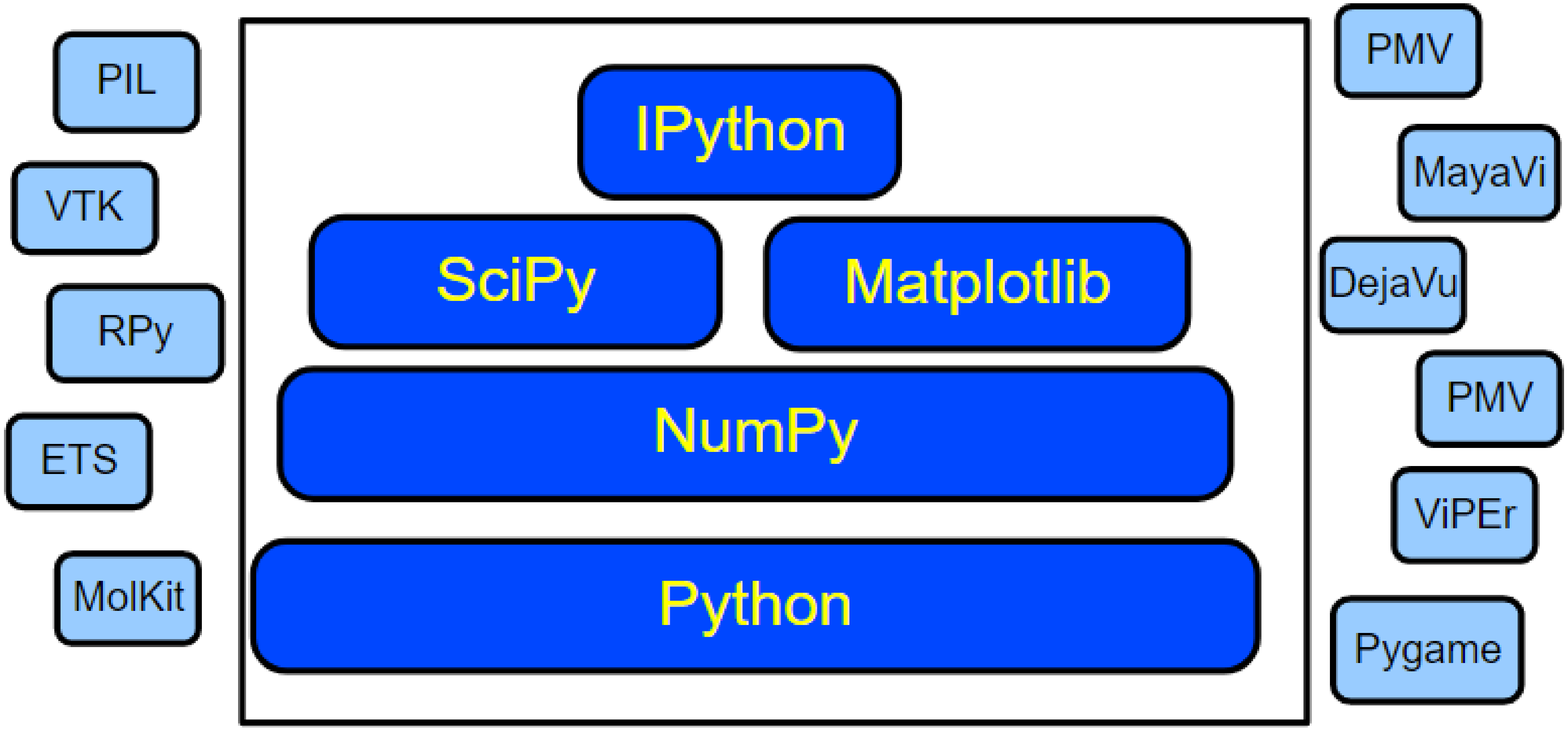
Matplotlib

- Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.
- Matplotlib can be used in Python scripts, the Python and IPython shell, the jupyter notebook, web application servers, and four graphical user interface toolkits.

Matplotlib

- Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc., with just a few lines of code. For a sampling, see the screenshots, thumbnail gallery, and examples directory
- For simple plotting the **pyplot** module provides a MATLAB-like interface, particularly when combined with **IPython**.
- For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.


PIL
VTK
RPy
ETS
MolKit
IPython
SciPy
Matplotlib
NumPy
Python
PMV
MayaVi
DejaVu
PMV
ViPEr
Pygame

Python Interpreter and Idle

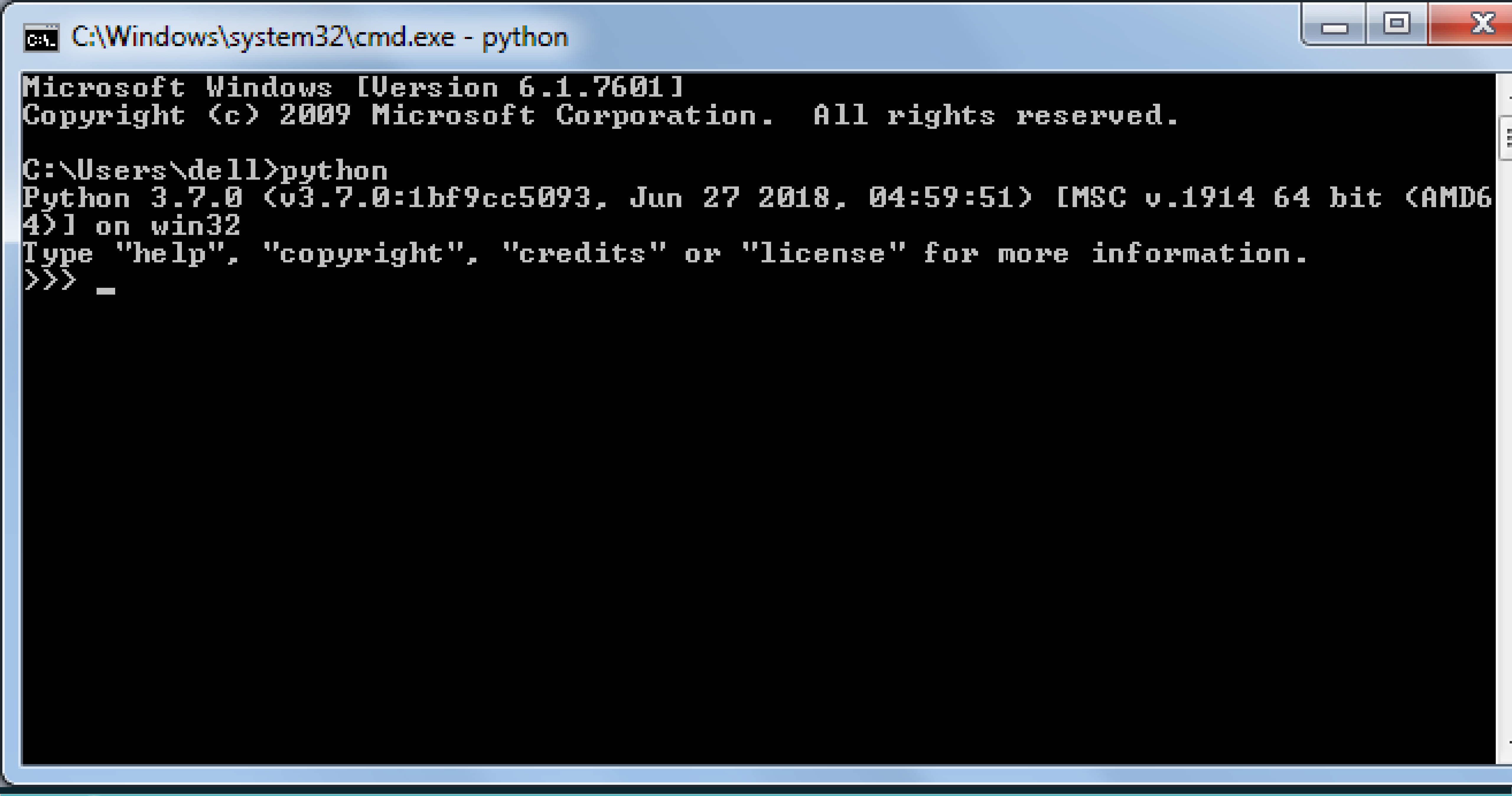
LECTURE 1

Python Interpreter

ACTIVITY

Python - Shell (Interpreter)

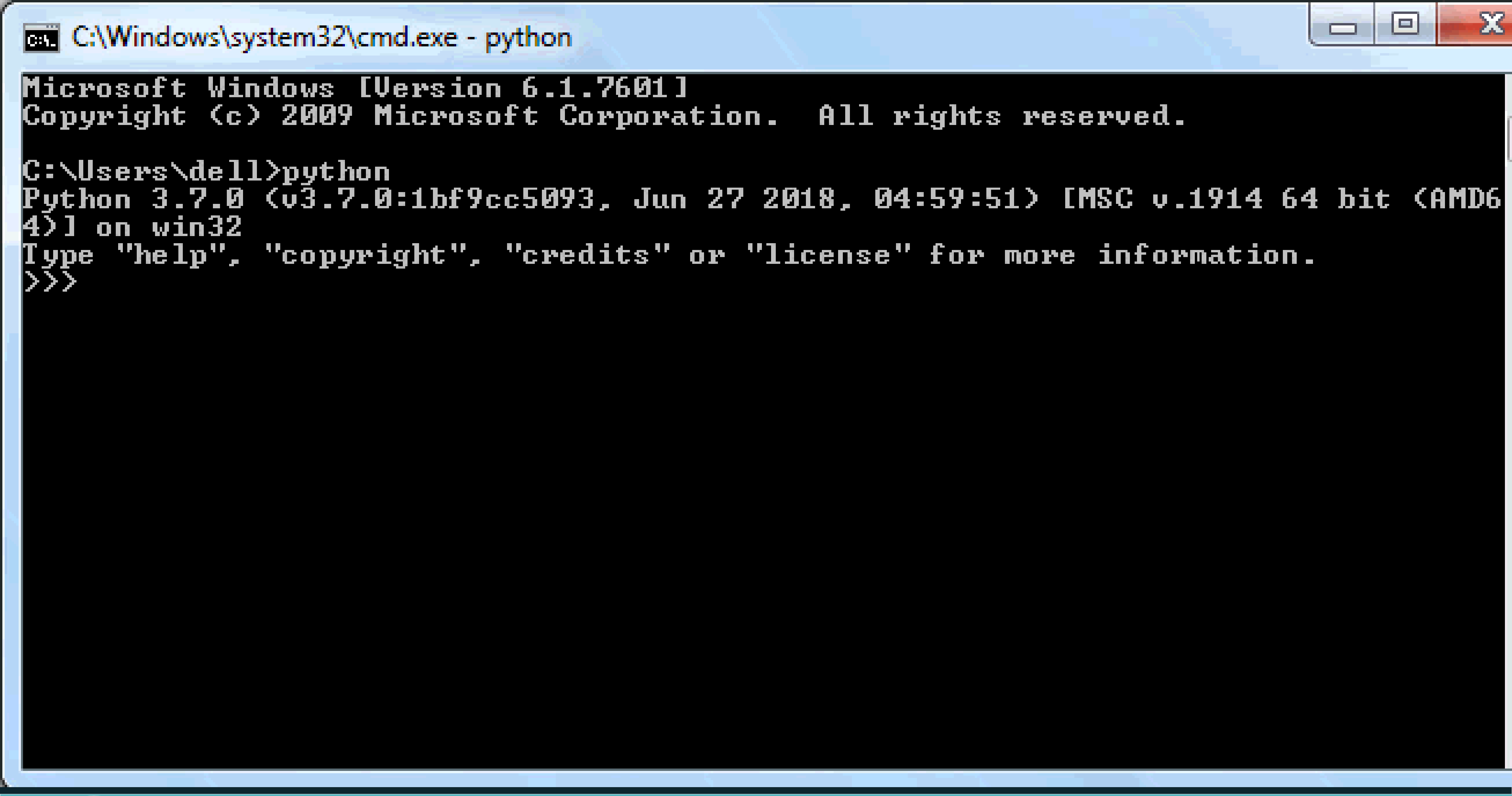
- Python is an interpreter language. It means it executes the code line by line. Python provides a Python Shell (also known as Python Interactive Shell) which is used to execute a single Python command and get the result.
- Python Shell waits for the input command from the user. As soon as the user enters the command, it executes it and displays the result.
- To open the Python Shell on Windows, open the command prompt, write **python** and press **enter**.



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

C:\Users\dell>python
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD6
4)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> _
```

eC Learning Channel



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

C:\Users\dell>python
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD6
4)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Execute Python Script

As you have seen above, Python Shell executes a single statement. To execute multiple statements, create a Python file with extension .py, and write Python scripts (multiple statements).

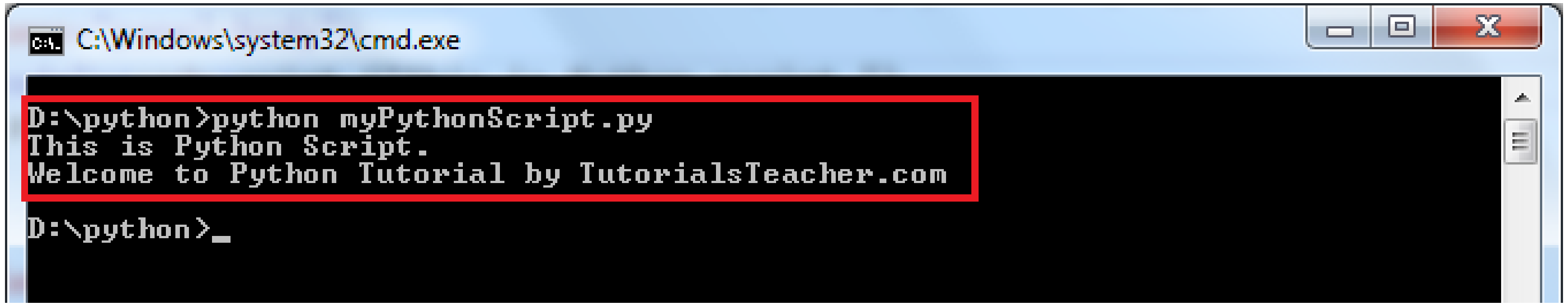
For example, enter the following statement in a text editor such as Notepad++.

Example: myPythonScript.py

```
print ("This is Python Script.")
print ("Welcome to Python Tutorial by eCode24.com")
```

Execute Python Script

Save it as myPythonScript.py, navigate command prompt to the folder where you have saved this file and execute the python myPythonScript.py command, as shown below.



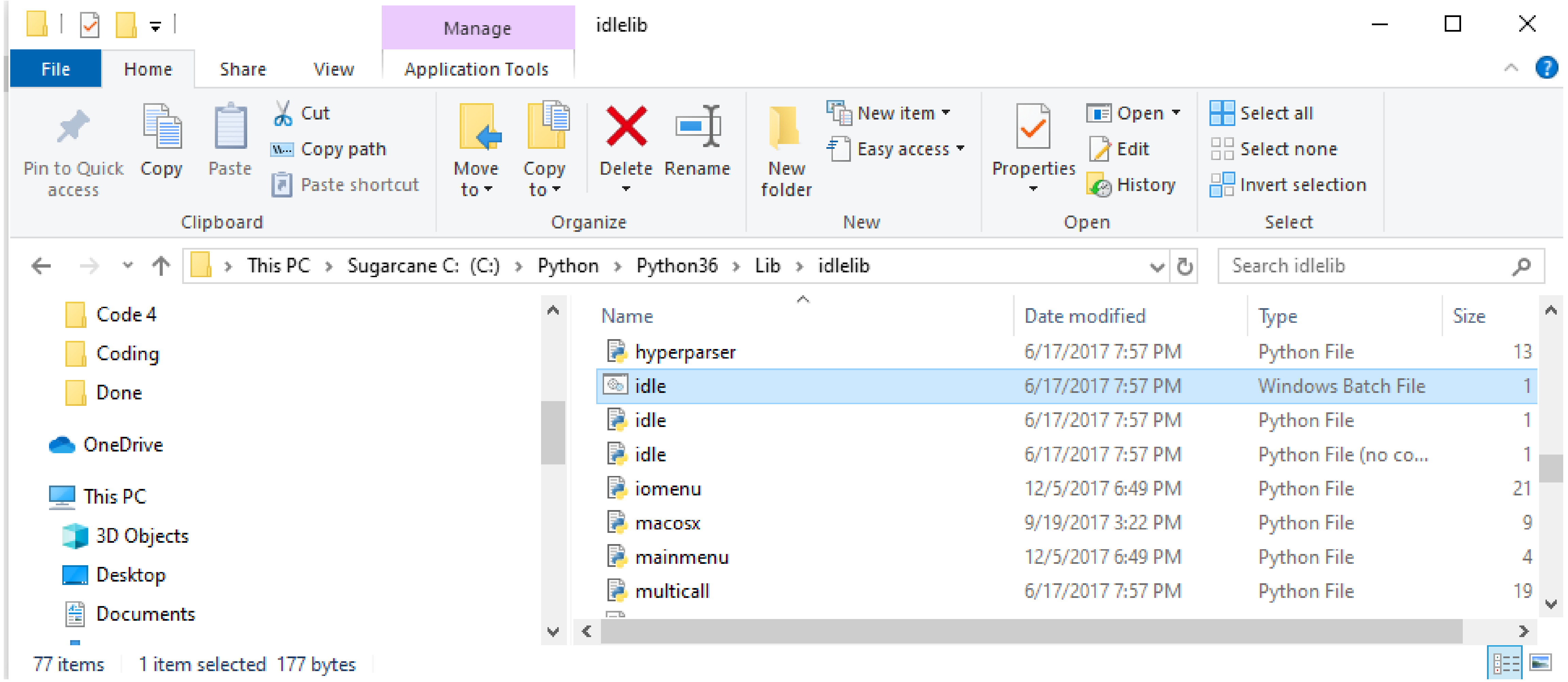
```
D:\python>python myPythonScript.py
This is Python Script.
Welcome to Python Tutorial by TutorialsTeacher.com

D:\python>_
```

Thus, you can execute Python expressions and commands using Python Shell. We will be using Python Shell to demo Python features through out these tutorials.

Idle

ACTIVITY

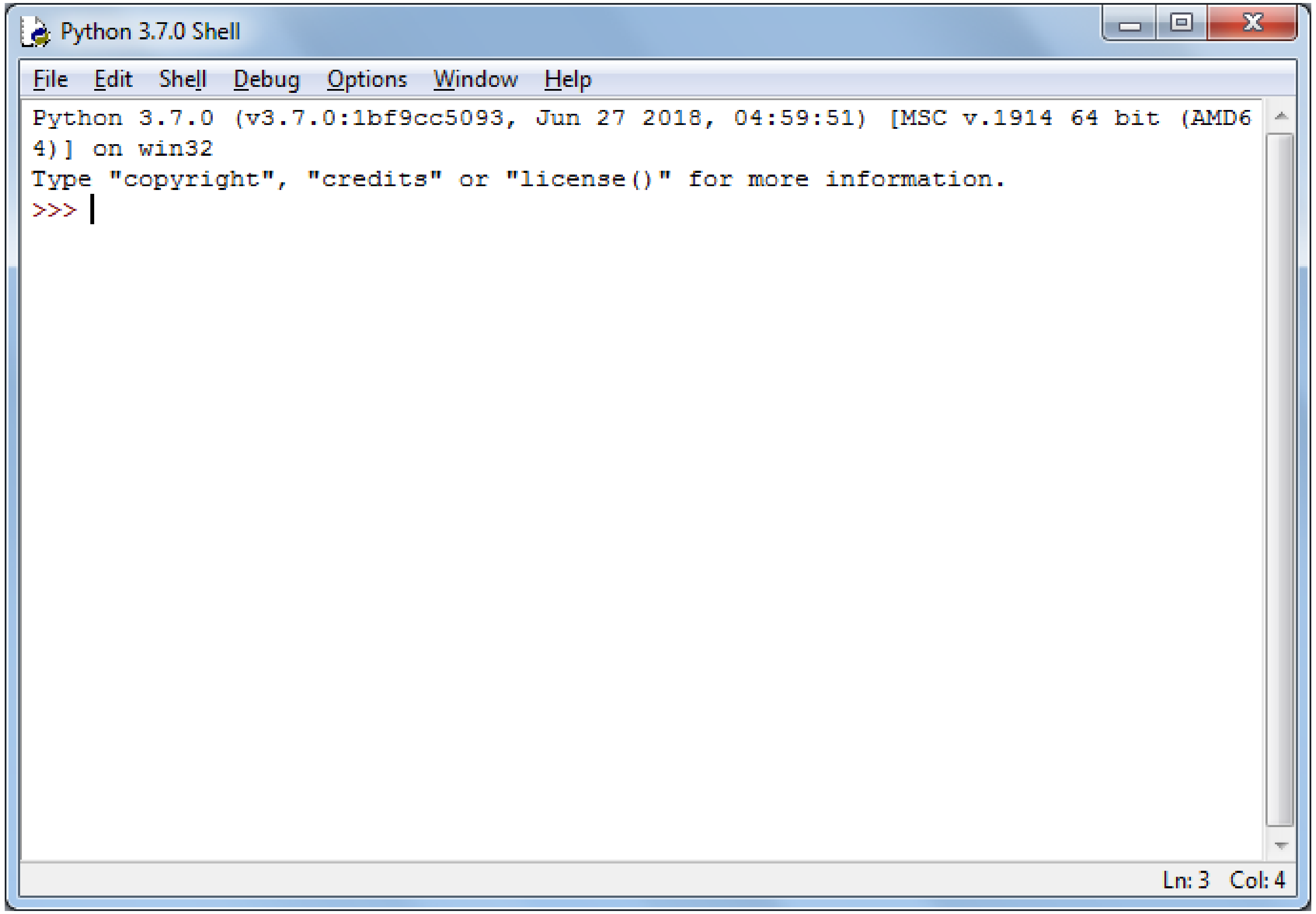


Name	Date modified	Type	Size
hyperparser	6/17/2017 7:57 PM	Python File	13
idle	6/17/2017 7:57 PM	Windows Batch File	1
idle	6/17/2017 7:57 PM	Python File	1
idle	6/17/2017 7:57 PM	Python File (no co...	1
iomenu	12/5/2017 6:49 PM	Python File	21
macosx	9/19/2017 3:22 PM	Python File	9
mainmenu	12/5/2017 6:49 PM	Python File	4
multicall	6/17/2017 7:57 PM	Python File	19

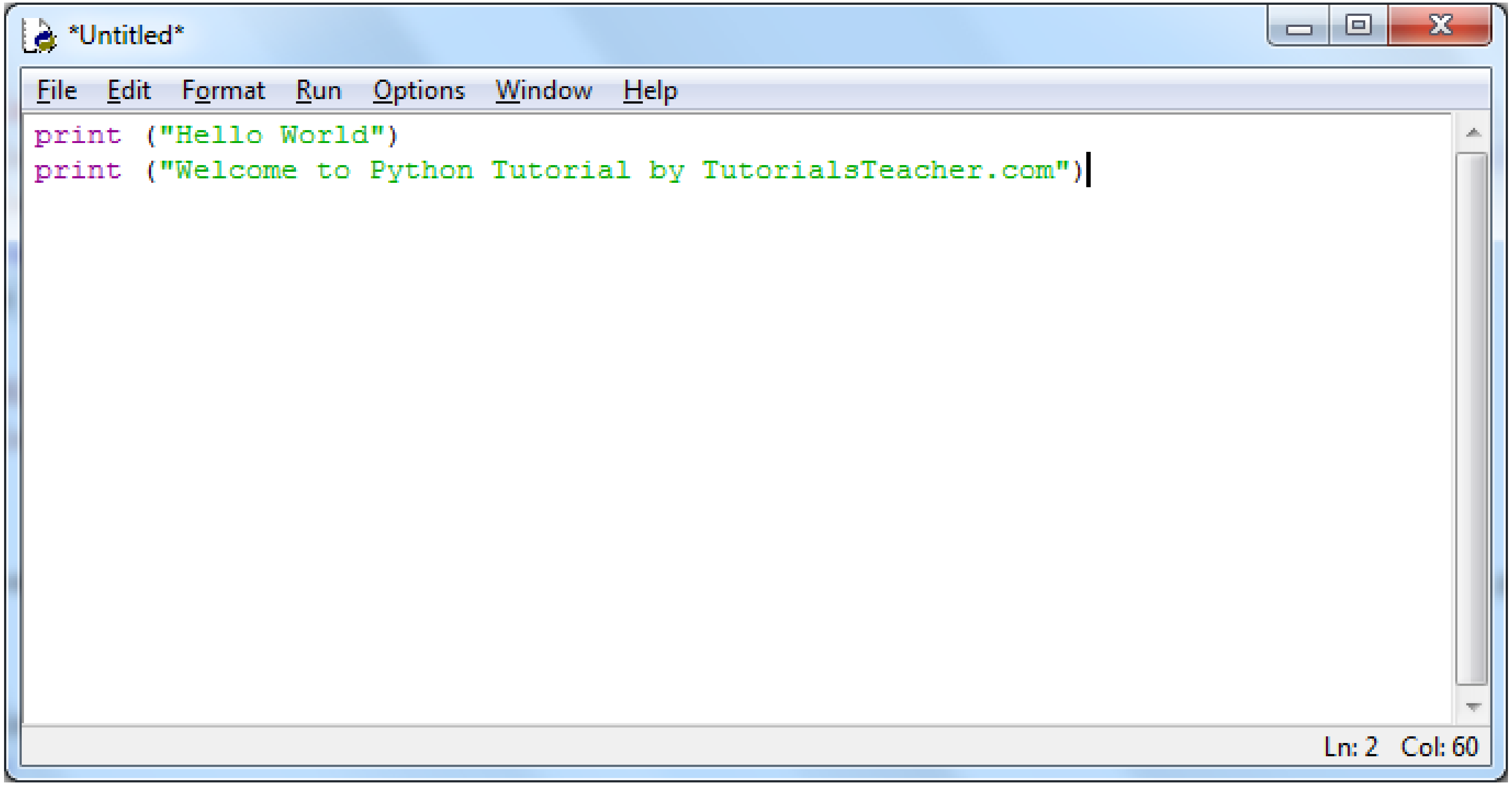
77 items 1 item selected 177 bytes

Python - IDLE

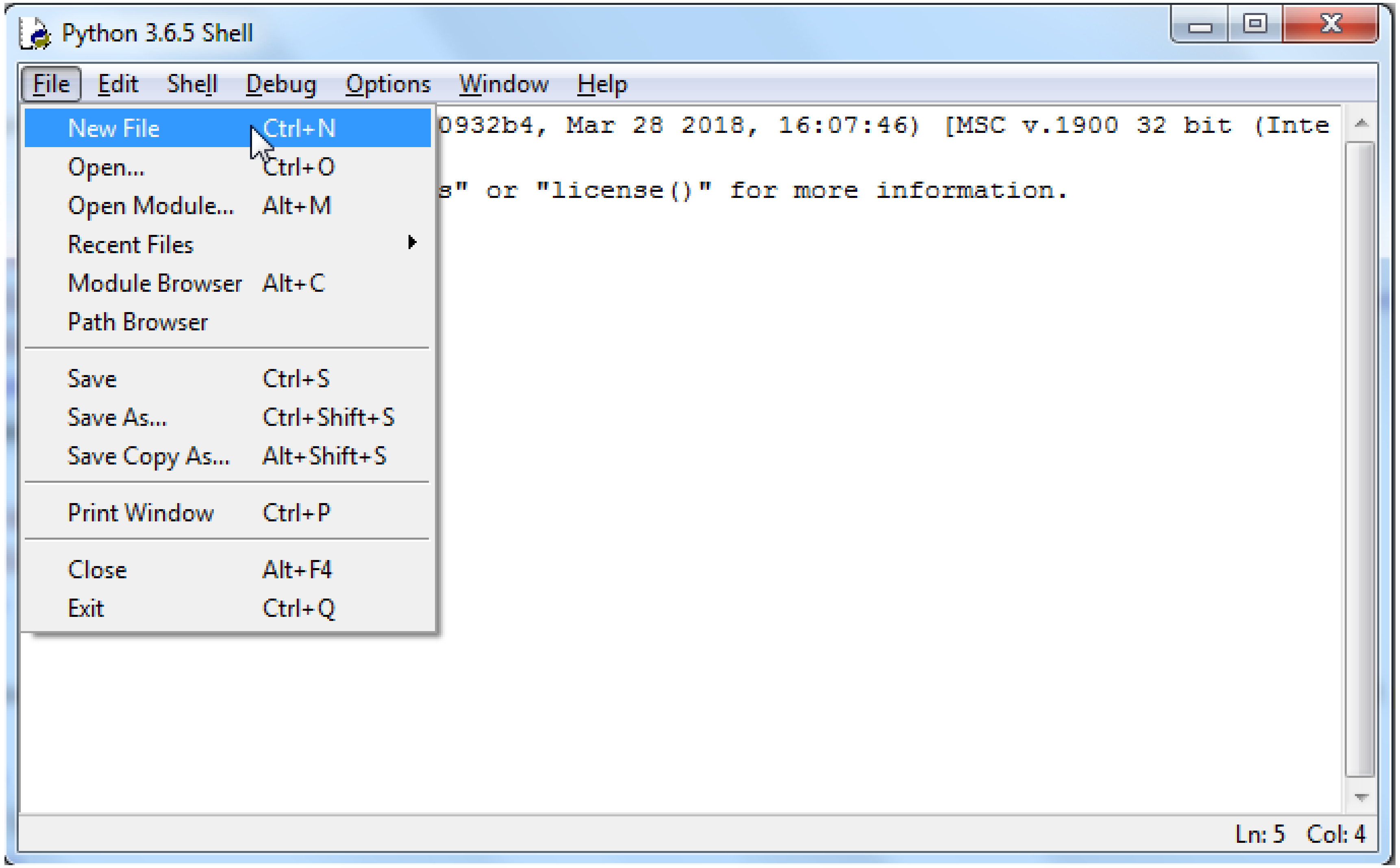
- IDLE (Integrated Development and Learning Environment) is an integrated development environment (IDE) for Python. The Python installer for Windows contains the IDLE module by default.
- IDLE can be used to execute a single statement just like Python Shell and also to create, modify and execute Python scripts. IDLE provides a fully-featured text editor to create Python scripts that includes features like syntax highlighting, autocompletion and smart indent. It also has a debugger with stepping and breakpoints features.



```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD6
4)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 
```

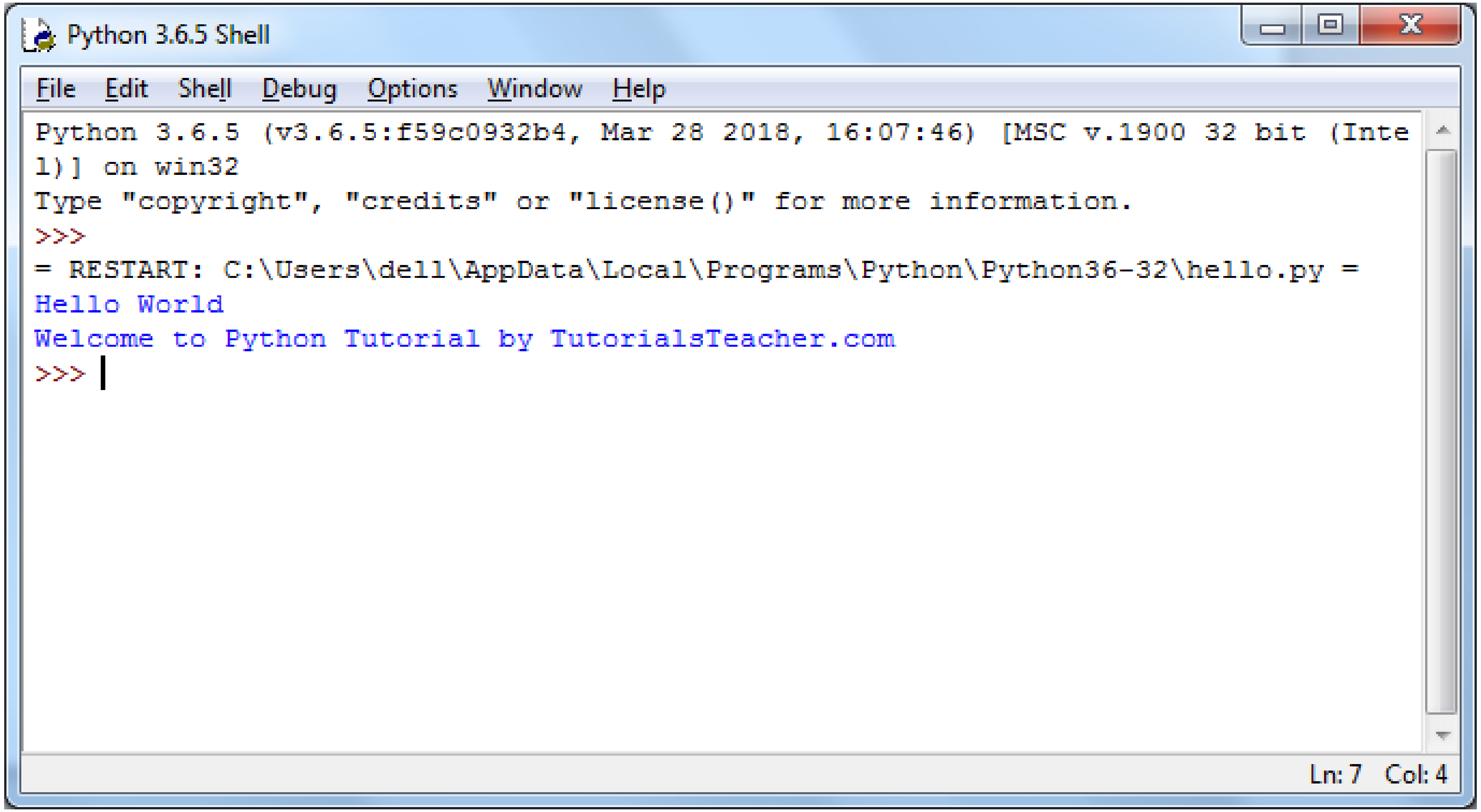


```
print ("Hello World")
print ("Welcome to Python Tutorial by TutorialsTeacher.com")
```



```
0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Inte

s" or "license()" for more information.
```



```
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 16:07:46) [MSC v.1900 32 bit (Inte
l)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\dell\AppData\Local\Programs\Python\Python36-32\hello.py =
Hello World
Welcome to Python Tutorial by TutorialsTeacher.com
>>> 
```

Basic Syntax

LECTURE 1

Python - Basic Syntax

- Vocabulary / Words - Variables and Reserved words
- Sentence structure - valid syntax patterns
- Program structure - constructing a program for a purpose

Reserved Words

You cannot use **reserved words** as variable names / identifiers

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

Python Identifiers

- A Python program can have variables, functions, classes, modules, packages etc.
- **Identifier** is the name given to these programming elements. An identifier should start with either an alphabet letter (lower or upper case) or an underscore (_). After that, more than one alphabet letters (a-z or A-Z), digits (0-9) or underscores may be used to form an identifier.
- No other characters are allowed.

Python Identifiers

- Conventionally, the name of the class begins with an uppercase alphabet letter. Others start with lowercase alphabet letters.
- Use of one or two underscore characters has a special significance when naming the instance attributes of a class. More about this will follow in the discussion about inheritance.
- Two leading and trailing underscores are used in the language itself for a special purpose. For example (e.g. __add__, __init__)

Python Statement

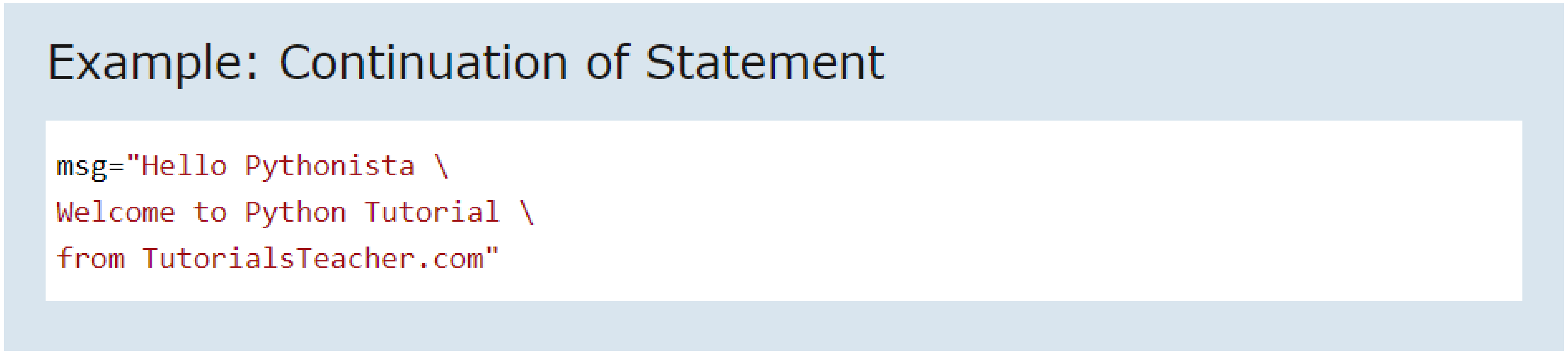
By default, the Python interpreter treats a piece of text terminated by hard carriage return (new line character) as one statement. It means each line in a Python script is a statement. (Just as in C/C++/C#, a semicolon ; denotes the end of a statement).

Example: Python Statements

```
msg="Hello World"
code=123
name="Steve"
```

Python Statement

However, you can show the text spread over more than one lines to be a single statement by using the backslash (\) as a continuation character. Look at the following examples:



Example: Continuation of Statement

```
msg="Hello Pythonista \
Welcome to Python Tutorial \
from TutorialsTeacher.com"
```

Python Statement

Similarly, use the semicolon ; to write multiple statements in a single line.

Example: Multiple Statements in Single Line

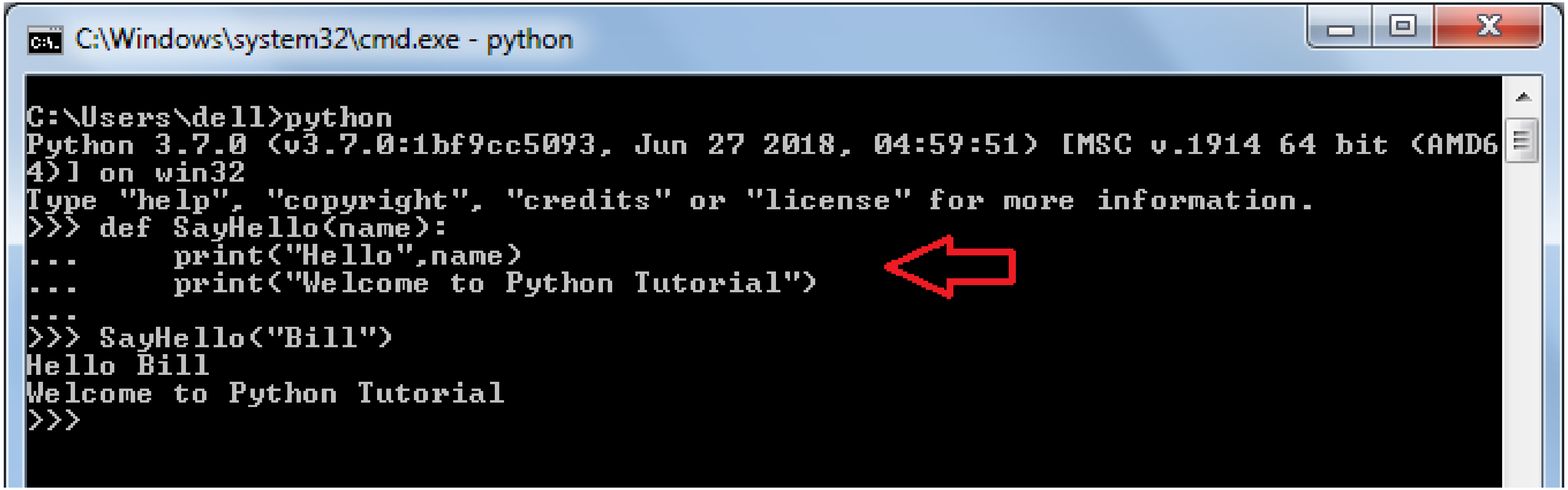
```
msg="Hello World";code=123;name="Steve"
```

Indents in Python

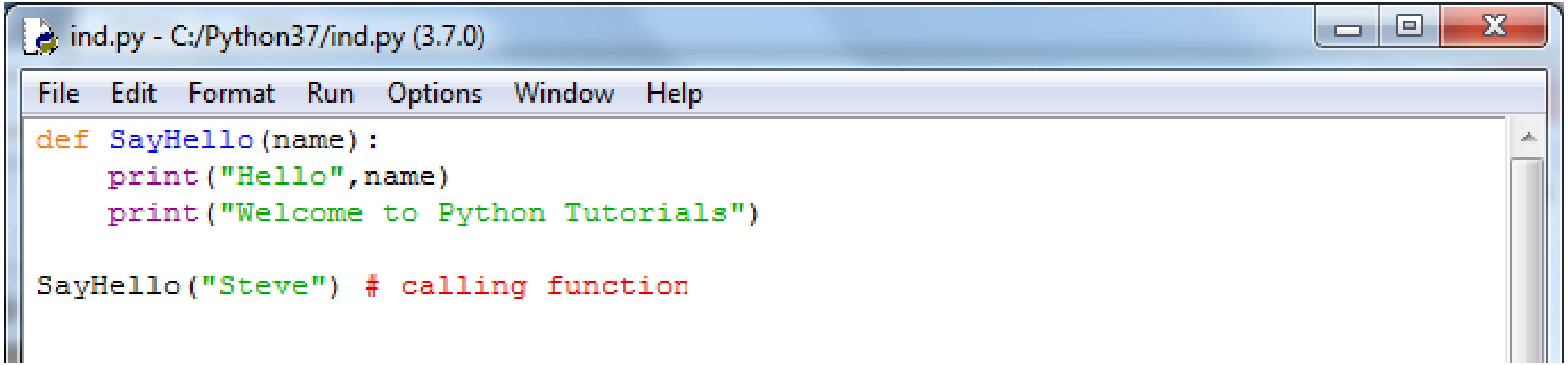
- Many times it is required to construct a block of more than one statements. For example there are usually multiple statements that are part of the definition of a function. There can be one or more statements in a looping construct.
- Different programming languages use different techniques to define the scope and extent of a block of statements in constructs like class, function, conditional and loop. In C, C++, C# or Java, statements inside curly brackets { and } are treated as a block.

Indents in Python

- Python uses uniform indentation to denote a block of statements. When a block is to be started, type the exclamation symbol (:) and press Enter. Any Python-aware editor (like IDLE) goes to the next line leaving an additional whitespace (called indent). Subsequent statements in the block follow the same level of indent.
- In order to signal the end of a block, the whitespace is de-dented by pressing the backspace key. If your editor is not configured for Python, you may have to ensure that the statements in a block have the same indentation level by pressing the spacebar or Tab key. The Python interpreter will throw an error if the indentation level in the block is not same.


C:\Windows\system32\cmd.exe - python

```
C:\Users\dell>python
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD6
4)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> def SayHello(name):
...     print("Hello",name)
...     print("Welcome to Python Tutorial")
...
>>> SayHello("Bill")
Hello Bill
Welcome to Python Tutorial
>>>
```


ind.py - C:/Python37/ind.py (3.7.0)
File Edit Format Run Options Window Help

```
def SayHello(name):
    print("Hello",name)
    print("Welcome to Python Tutorials")

SayHello("Steve") # calling function
```

Comments in Python

In a Python script, the symbol # indicates the start of a comment line. It is effective till the end of the line in the editor. If # is the first character of the line, then the entire line is a comment. It can be used also in the middle of a line. The text before it is a valid Python expression, while the text following is treated as a comment.

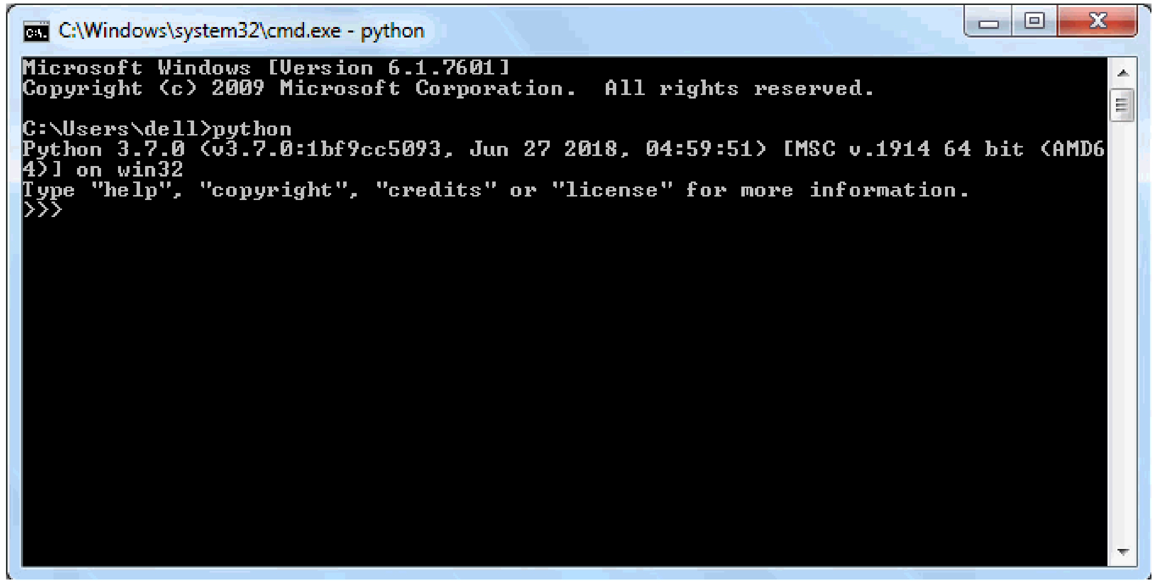
Example: Comments

```
# this is a comment
print ("Hello World")
print ("Welcome to Python Tutorial") #this is also a comment but after a statement.
```

Example: Multi-line Comments

```
'''
comment1
comment2
comment3
'''
print ("Hello World")
```

Getting the User's Input

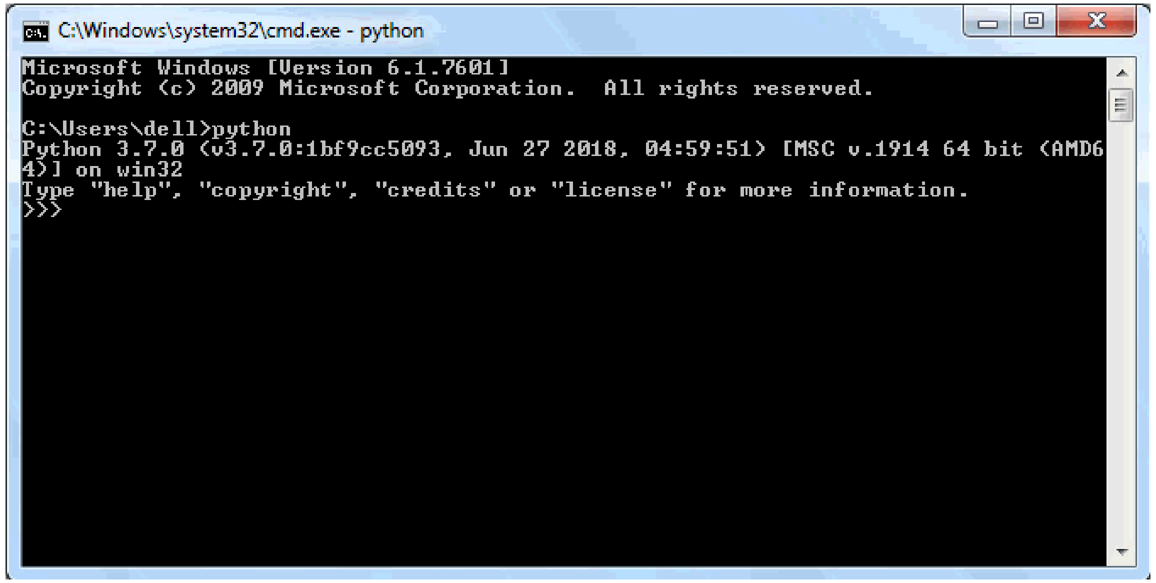


```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\dell>python
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD6
4)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

The **input()** function is a part of the core library of standard Python distribution. It reads the key strokes as a string object which can be referred to by a variable having a suitable name.

Getting the User's Input


C:\Windows\system32\cmd.exe - python

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

C:\Users\dell>python
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD6
4)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

In the above example, the **input()** function takes the user's input from the next line, e.g. 'Steve' in this case. **input()** will capture it and assign it to a name variable. The **name** variable will display whatever the user has provided as the input.

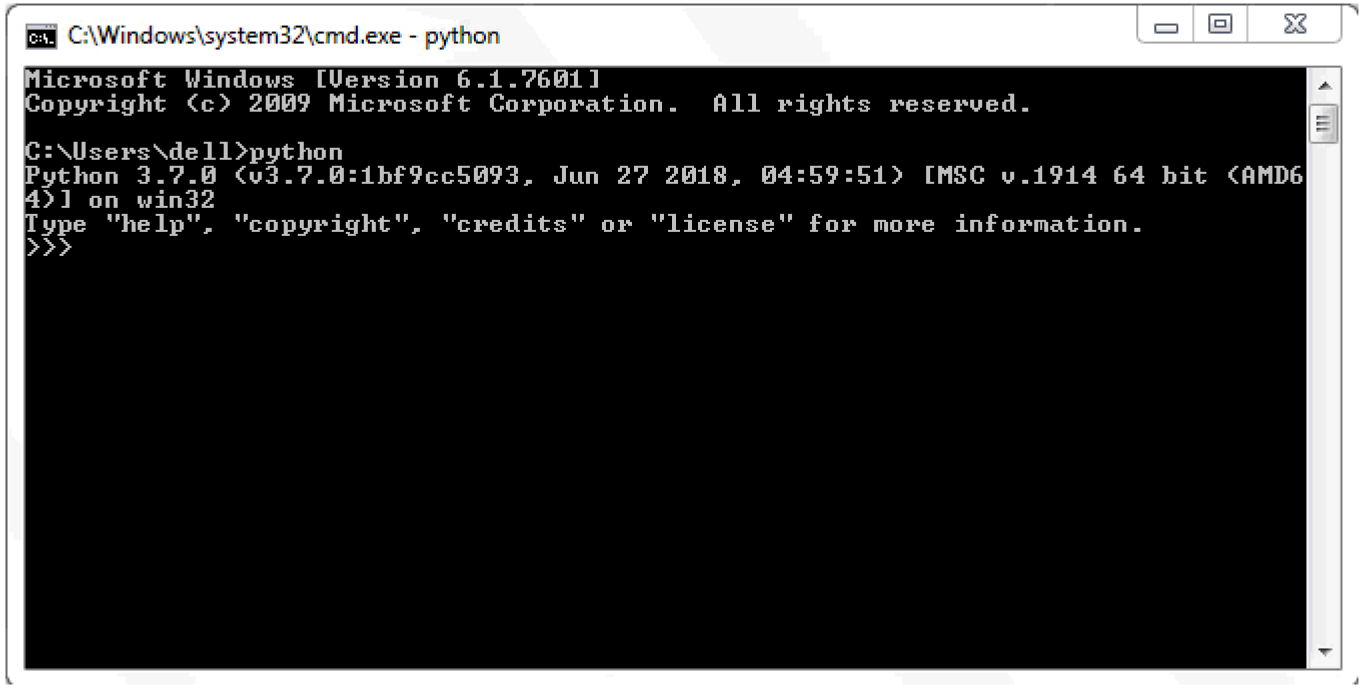
Getting the User's Input

The input() function always reads the input as a string, even if comprises of digits. The type() function used earlier confirms this behavior.

```
>>> name=input("Enter your name: ")
Enter your name: Steve
>>> type(name)
<class 'str'>
>>> age=input("Enter your age: ")
Enter your age: 21
>>> type(age)
<class 'str'>
```

Display the Output

Another built-in function **print()** serves as an output statement in Python. It echoes the value of any Python expression on the Python shell.



C:\Windows\system32\cmd.exe - python

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

C:\Users\dell>python
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:51) [MSC v.1914 64 bit (AMD6
4)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Display the Output

Multiple values can be displayed by the single print() function separated by comma. The following example displays values of name and age variables using the single **print()** function.

```
>>> name="Ram"
>>> age=21
>>> print("Name:", name, "Age:",age)
Name: Ram Age: 21
```

Display the Output

By default, a single space (' ') acts as a separator between values. However, any other character can be used by providing a sep parameter. In the following example, "=" is used as a separator character.

```
>>> name="Ram"
>>> age=21
>>> print(name,age)
Ram 21
>>> print(name,age,sep=",")
Ram,21
```

Display the Output

The output of the print() function always ends by a newline character. The print() function has another optional parameter end, whose default value is "\n". This value can be substituted by any other character such as a single space (' ') to display the output of the subsequent print() statement in the same line. This is especially useful in a Python script like the one shown below:

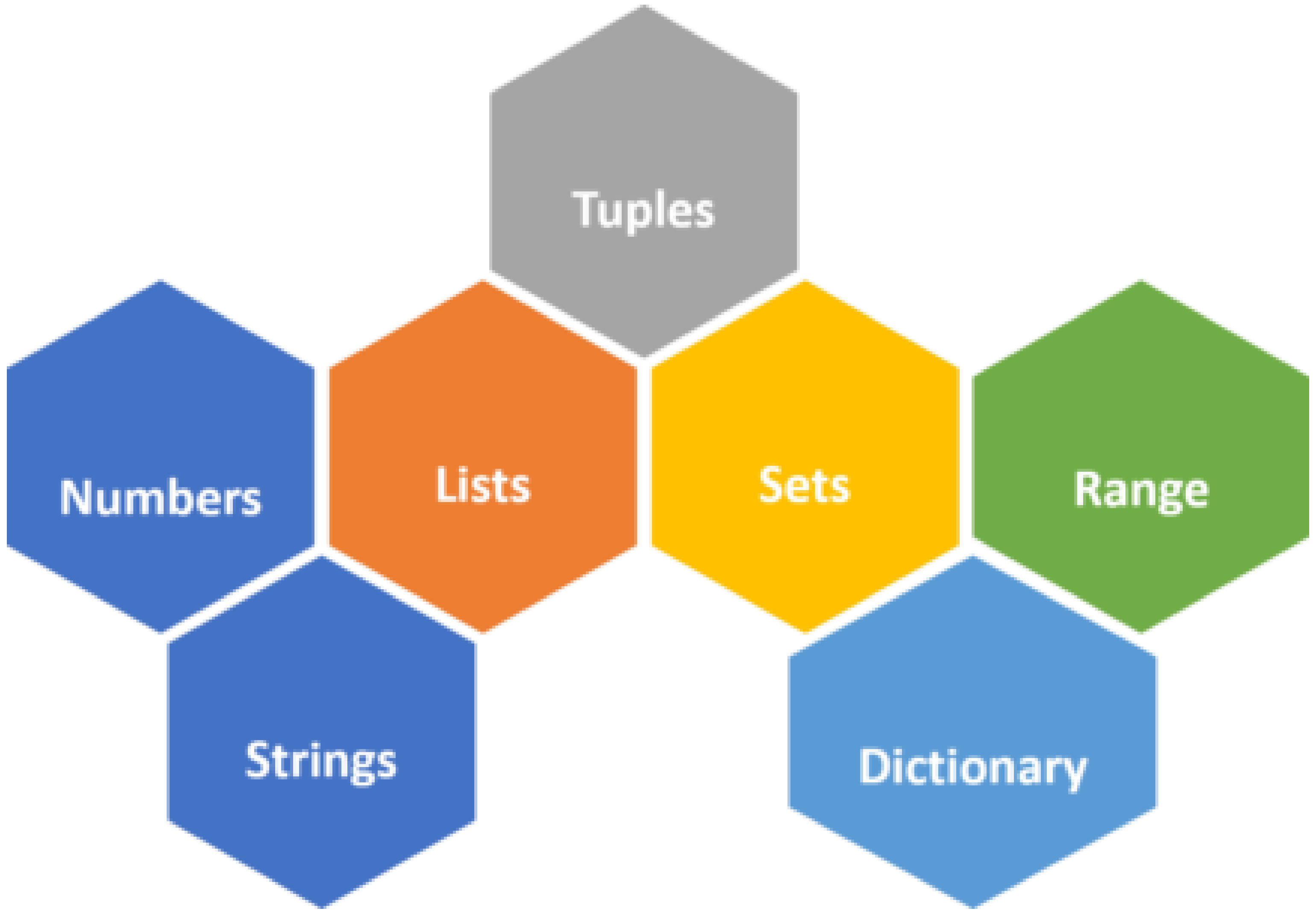
Example: display.py

```
name="Amar"
age=21
print("Name:", name, end=" ")
print("Age:", age)
```

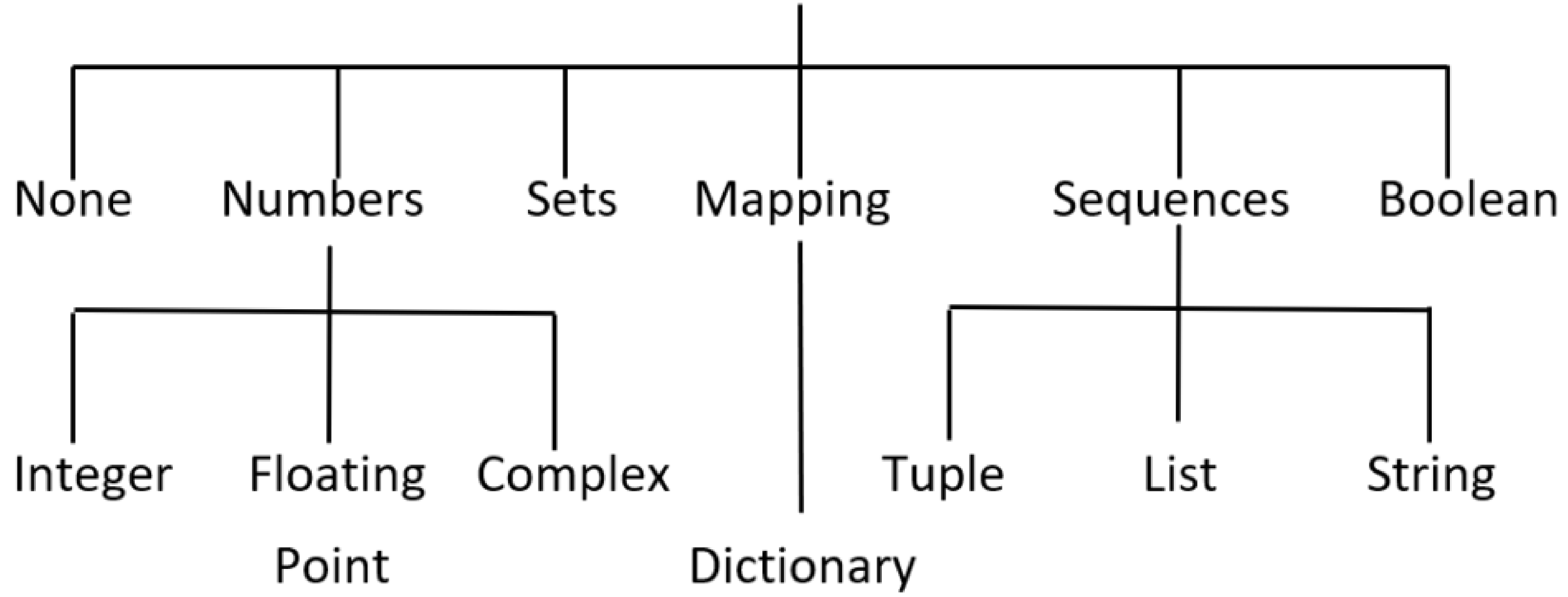
```
Name: Amar Age: 21
```

Data Types

LECTURE 1


Tuples
Numbers
Lists
Sets
Range
Strings
Dictionary

DATA TYPES


None
Numbers
Sets
Mapping
Sequences
Boolean
Integer
Floating Point
Complex
Dictionary
Tuple
List
String

Basic Datatypes

- Integers (default for numbers)
    - `z = 5 / 2 # Answer 2, integer division`
- Floats
    - `x = 3.456`
- Strings
    - Can use “” or ‘’ to specify with “abc” == ‘abc’
    - Unmatched can occur within the string: “matt’s”
    - Use triple double-quotes for multi-line strings or strings that contain both ‘ and “ inside of them: “““a‘b“c”””

Numeric

A numeric value is any representation of data which has a numeric value. Python identifies three types of numbers:

- **Integer:** Positive or negative whole numbers (without a fractional part)
- **Float:** Any real number with a floating point representation in which a fractional component is denoted by a decimal symbol or scientific notation
- **Complex number:** A number with a real and imaginary component represented as x+yj. x and y are floats and j is $\sqrt{-1}$

Boolean

- Data with one of two built-in values **True** or **False**. Notice that 'T' and 'F' are capital. true and false are not valid booleans and Python will throw an error for them.

Sequence Type

A sequence is an ordered collection of similar or different data types. Python has the following built-in sequence data types:

- **String**: A string value is a collection of one or more characters put in single, double or triple quotes.
- **List** : A list object is an ordered collection of one or more data items, not necessarily of the same type, put in square brackets.
- **Tuple**: A Tuple object is an ordered collection of one or more data items, not necessarily of the same type, put in parentheses.

Dictionary

A dictionary object is an unordered collection of data in a key:value pair form. A collection of such pairs is enclosed in curly brackets. For example:

```
{1:"Steve", 2:"Bill", 3:"Ram", 4: "Farha"}
```

type() function

Python has an in-built function type() to ascertain the data type of a certain value. For example, enter type(1234) in Python shell and it will return <class 'int'>, which means 1234 is an integer value. Try and verify the data type of different values in Python shell, as shown below.

```
>>> type(1234)
<class 'int'>
>>> type(55.50)
<class 'float'>
>>> type(6+4j)
<class 'complex'>
>>> type("hello")
<class 'str'>
>>> type([1,2,3,4])
<class 'list'>
>>> type((1,2,3,4))
<class 'tuple'>
>>> type({1:"one", 2:"two", 3:"three"})
<class 'dict'>
```

Variables

LECTURE 1

Constants

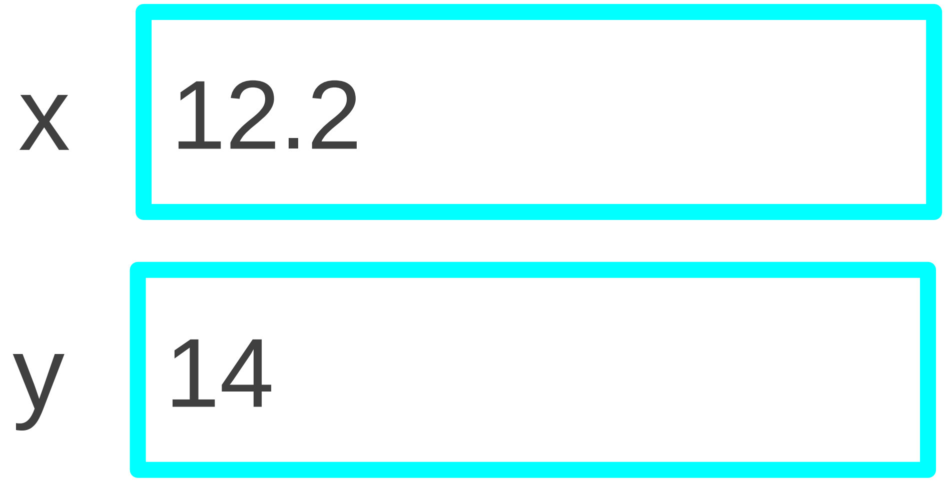
- Fixed values such as numbers, letters, and strings, are called “**constants**” because their value does not change
- Numeric constants are as you expect
- String constants use single quotes (') or double quotes (")

```
>>> print(123)
123
>>> print(98.6)
98.6
>>> print('Hello world')
Hello world
```

Variables

- A variable is a named place in the memory where a programmer can store data and later retrieve the data using the variable "name"
- Programmers get to choose the names of the variables
- You can change the contents of a variable in a later statement

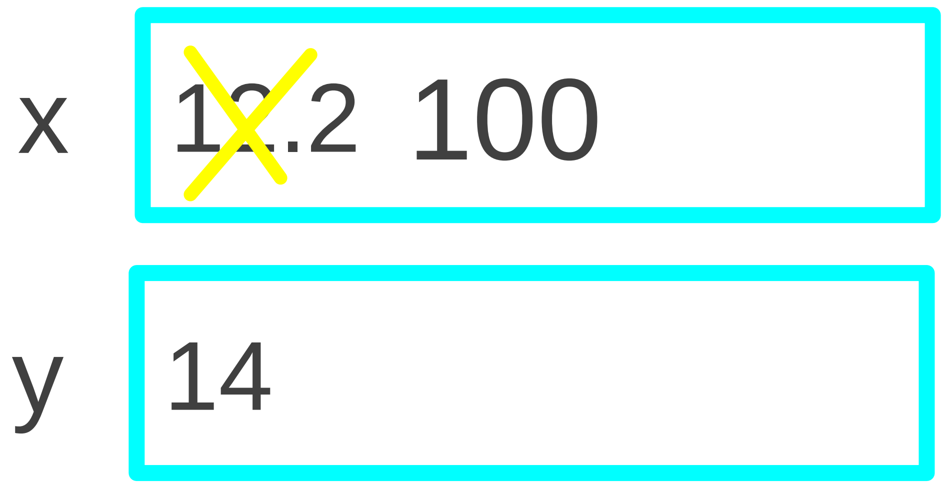
```
x = 12.2
y = 14
```



Variables

- A variable is a named place in the memory where a programmer can store data and later retrieve the data using the variable "name"
- Programmers get to choose the names of the variables
- You can change the contents of a variable in a later statement

```
x = 12.2
y = 14
x = 100
```



Python Variable Name Rules

- Must start with a letter or underscore _
- Must consist of letters, numbers, and underscores
- Case Sensitive

```
Good:        spam      eggs     spam23      _speed
Bad:         23spam    #sign    var.12
Different:   spam      Spam      SPAM
```

Mnemonic Variable Names

- Since we programmers are given a choice in how we choose our variable names, there is a bit of “best practice”
- We name variables to help us remember what we intend to store in them (“mnemonic” = “memory aid”)
- This can confuse beginning students because well-named variables often “sound” so good that they must be keywords

What is this bit of code doing?

```
x1q3z9ocd = 35.0
x1q3z9afd = 12.50
x1q3p9afd = x1q3z9ocd * x1q3z9afd
print(x1q3p9afd)
```

What are these bits of code doing?

```
x1q3z9ocd = 35.0
x1q3z9afd = 12.50
x1q3p9afd = x1q3z9ocd * x1q3z9afd
print(x1q3p9afd)
```

```
a = 35.0
b = 12.50
c = a * b
print(c)
```

What are these bits of code doing?

```
x1q3z9ocd = 35.0
x1q3z9afd = 12.50
x1q3p9afd = x1q3z9ocd * x1q3z9afd
print(x1q3p9afd)
```

```
a = 35.0
b = 12.50
c = a * b
print(c)
```

```
hours = 35.0
rate = 12.50
pay = hours * rate
print(pay)
```

Expression

LECTURE 1

Numeric Expressions

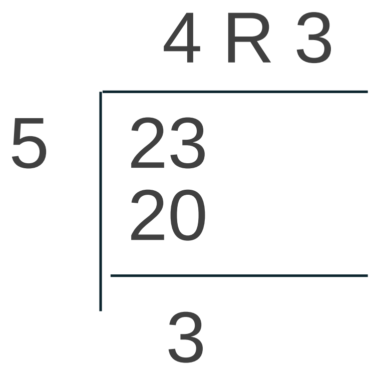
- Because of the lack of mathematical symbols on computer keyboards - we use “computer-speak” to express the classic math operations
- Asterisk is multiplication
- Exponentiation (raise to a power) looks different than in math

Operator	Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division
//	Integer Division
**	Power
%	Remainder

Numeric Expressions

```
>>> xx = 2
>>> xx = xx + 2
>>> print(xx)
4
>>> yy = 440 * 12
>>> print(yy)
5280
>>> zz = yy / 1000
>>> print(zz)
5.28
```

```
>>> jj = 23
>>> kk = jj % 5
>>> print(kk)
3
>>> print(4 ** 3)
64
```



Operator	Operation
+	Addition
-	Subtraction
*	Multiplication
/	Division
//	Integer Division
**	Power
%	Remainder

Order of Evaluation

- When we string operators together - Python must know which one to do first
- This is called “operator precedence”
- Which operator “takes precedence” over the others?

```
x = 1 + 2 * 3 - 4 / 5 ** 6
```

Operator Precedence Rules

Highest precedence rule to lowest precedence rule:

- Parentheses are always respected
- Exponentiation (raise to a power)
- Multiplication, Division, and Remainder
- Addition and Subtraction
- Left to right

Parenthesis
Power
Multiplication
Addition
Left to Right

```
>>> x = 1 + 2 ** 3 / 4 * 5
>>> print(x)
11.0
>>>
```

Parenthesis
Power
Multiplication
Addition
Left to Right

$$\begin{tikzcd}
1 + 2 ** 3 / 4 * 5 \arrow[d] \\
1 + 8 / 4 * 5 \arrow[d] \\
1 + 2 * 5 \arrow[d] \\
1 + 10 \arrow[d] \\
11
\end{tikzcd}$$

Operator Precedence

- Remember the rules top to bottom
- When writing code - use parentheses
- When writing code - keep mathematical expressions simple enough that they are easy to understand
- Break long series of mathematical operations up to make them more clear

Parenthesis
Power
Multiplication
Addition
Left to Right

What Does “Type” Mean?

- In Python variables, literals, and constants have a “type”
- Python knows the difference between an integer number and a string
- For example “+” means “addition” if something is a number and “concatenate” if something is a string

```
>>> ddd = 1 + 4
>>> print(ddd)
5
>>> eee = 'hello ' + 'there'
>>> print(eee)
hello there
```

concatenate = put together

Type Matters

- Python knows what “type” everything is
- Some operations are prohibited
- You cannot “add 1” to a string
- We can ask Python what type something is by using the type() function

```
>>> eee = 'hello ' + 'there'
>>> eee = eee + 1
Traceback (most recent call last):
File "<stdin>", line 1, in
<module>TypeError: Can't convert
'int' object to str implicitly
>>> type(eee)
<class'str'>
>>> type('hello')
<class'str'>
>>> type(1)
<class'int'>
>>>
```

Several Types of Numbers

Numbers have two main types

- Integers are whole numbers: -14, -2, 0, 1, 100, 401233
- Floating Point Numbers have decimal parts: -2.5 , 0.0, 98.6, 14.0

There are other number types - they are variations on float and integer

```
>>> xx = 1
>>> type (xx)
<class 'int'>
>>> temp = 98.6
>>> type(temp)
<class'float'>
>>> type(1)
<class 'int'>
>>> type(1.0)
<class'float'>
>>>
```

Type Conversions

- When you put an integer and floating point in an expression, the integer is implicitly converted to a float
- You can control this with the built-in functions int() and float()

```
>>> print(float(99) + 100)
199.0
>>> i = 42
>>> type(i)
<class'int'>
>>> f = float(i)
>>> print(f)
42.0
>>> type(f)
<class'float'>
>>>
```

Integer Division

- Division produces a floating point result.
- Integer Division produces integer results

```
>>> print(10 / 2)
5.0
>>> print(9 / 2)
4.5
>>> print(99 / 100)
0.99
>>> print(10.0 / 2.0)
5.0
>>> print(99.0 / 100.0)
0.99
```

Statements

LECTURE 1

Sentences or Lines

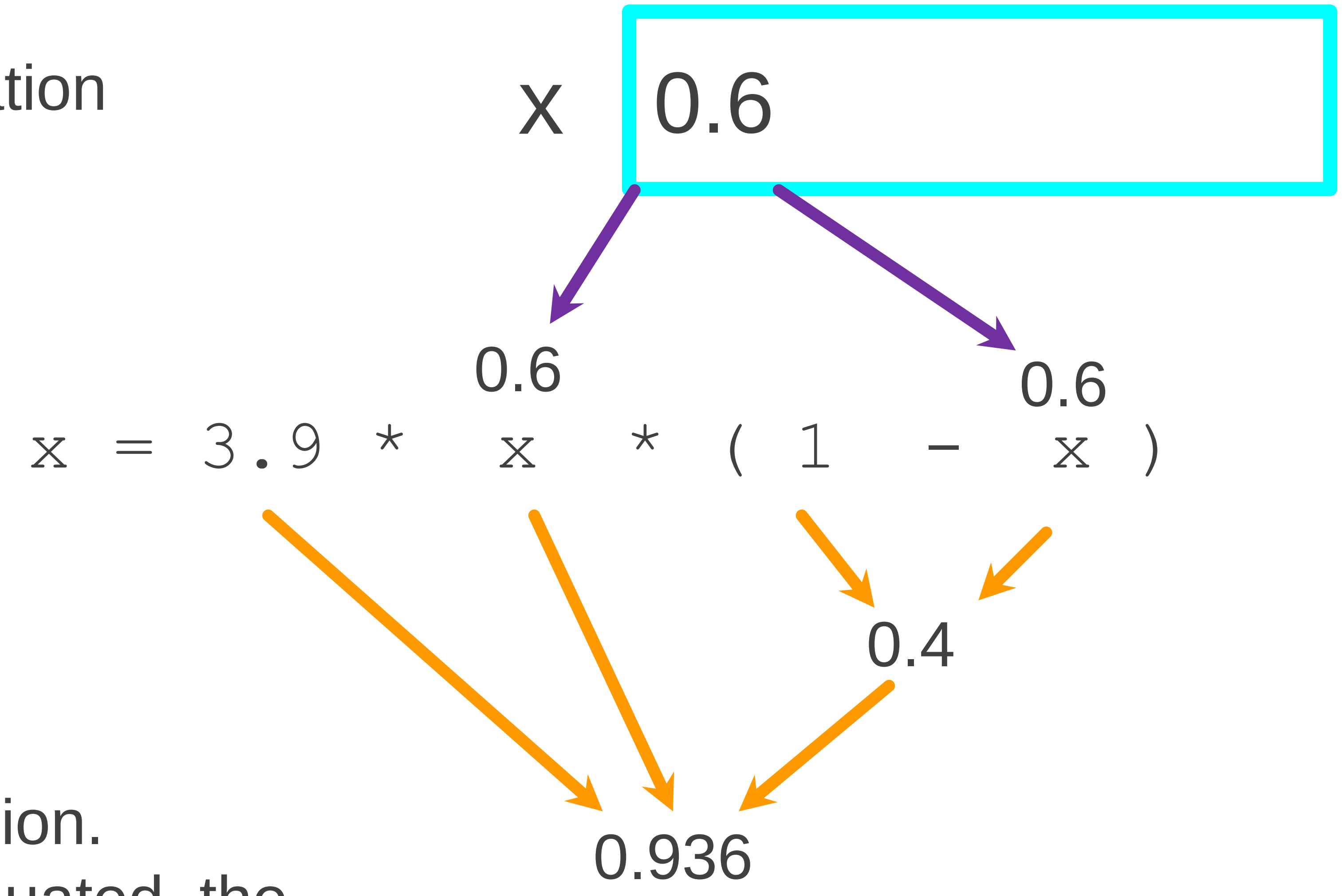
`x = 2`	Assignment statement
`x = x + 2`	Assignment with expression
`print(x)`	Print statement

Assignment Statements

- We assign a value to a variable using the assignment statement (=)
- An assignment statement consists of an expression on the right-hand side and a variable to store the result

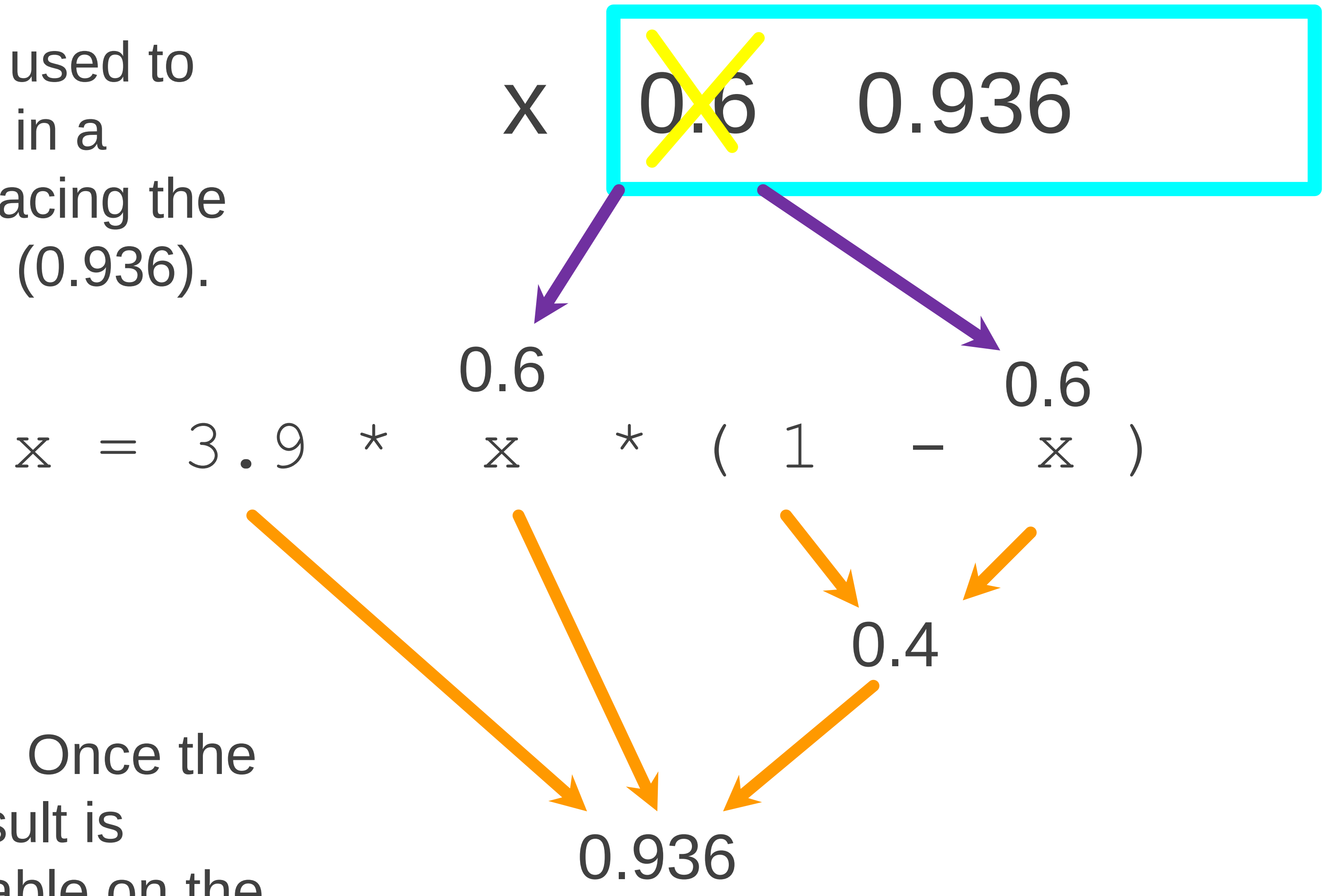
```
x = 3.9 * x * ( 1 - x )
```

A variable is a memory location used to store a value (0.6)



The right side is an expression.
Once the expression is evaluated, the result is placed in (assigned to) x.

A variable is a memory location used to store a value. The value stored in a variable can be updated by replacing the old value (0.6) with a new value (0.936).



The right side is an expression. Once the expression is evaluated, the result is placed in (assigned to) the variable on the left side (i.e., x).

Type Conversion

LECTURE 1

String Conversions

- You can also use int() and float() to convert between strings and integers
- You will get an error if the string does not contain numeric characters

```
>>> sval = '123'
>>> type(sval)
<class 'str'>
>>> print(sval + 1)
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: Can't convert 'int' object
to str implicitly
>>> ival = int(sval)
>>> type(ival)
<class 'int'>
>>> print(ival + 1)
124
>>> nsv = 'hello bob'
>>> niv = int(nsv)
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
ValueError: invalid literal for int()
with base 10: 'x'
```

User Input

We can instruct Python to pause and read data from the user using the input() function

The input() function returns a string

```
nam = input('Who are you? ')
print('Welcome', nam)
```

```
Who are you? Chuck
Welcome Chuck
```

Converting User Input

- If we want to read a number from the user, we must convert it from a string to a number using a type conversion function

- Later we will deal with bad input data

```
inp = input('Europe floor?')
usf = int(inp) + 1
print('US floor', usf)
```

```
Europe floor? 0
US floor 1
```