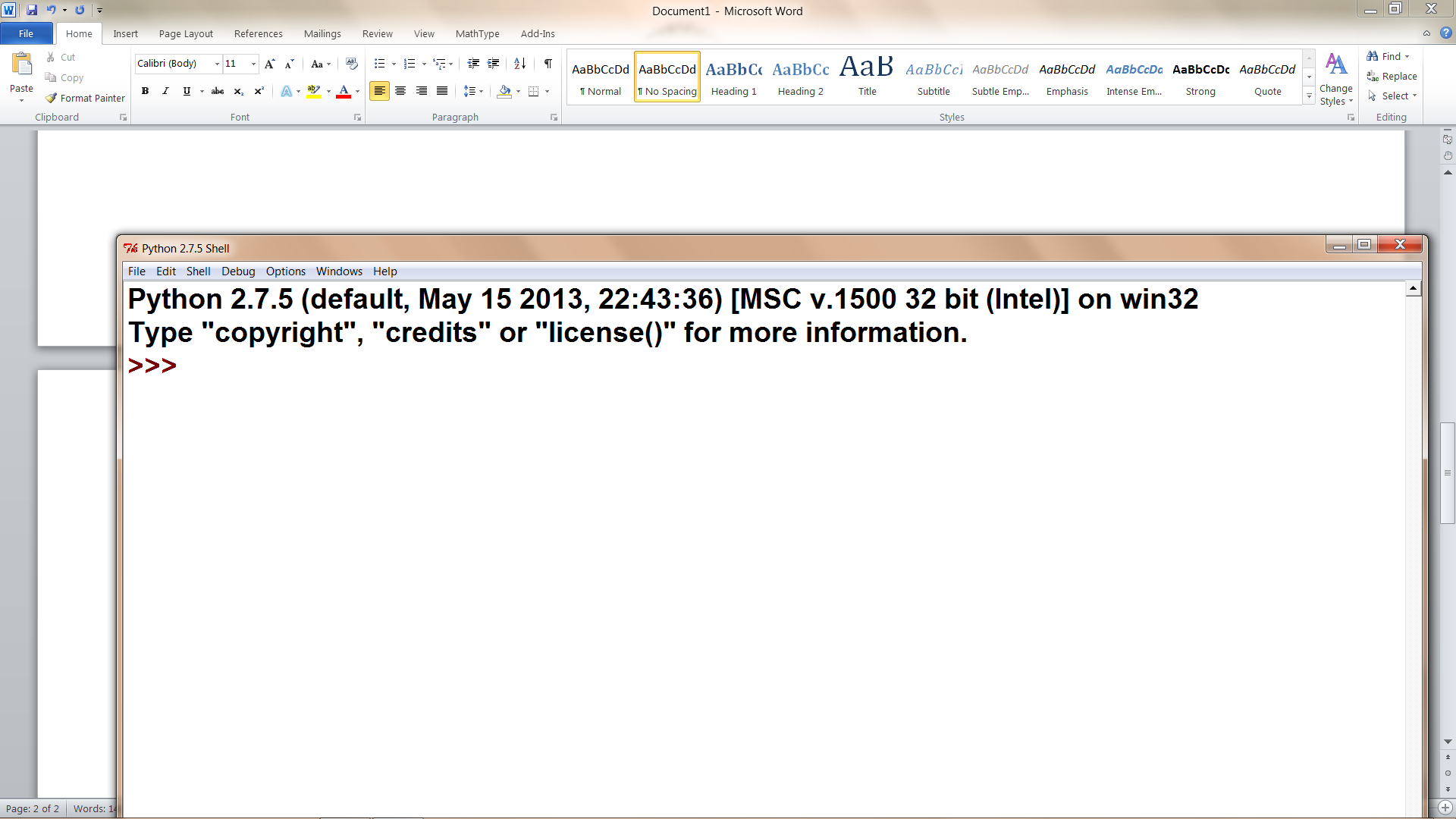
**Getting Started with Python**

The following list contains the basic knowledge required to create basic programs in Python. Once you have a basic understanding of these topics, you are ready to build some projects.

1. Python Basics
   1. IDLE
   2. Python Shell
   3. Editor
   4. Numbers
   5. Variables
   6. For Loops
   7. Relational Operators
   8. Logical Operators
   9. if …. Else
   10. while loops
2. Strings, Lists, Functions, and Dictionaries
   1. Strings
   2. Lists
   3. Functions
   4. Dictionaries
   5. Tuples
   6. Multiple Assignment
   7. Multiple Returns
   8. Number Functions
   9. String Functions
   10. List Functions
   11. Dictionary Functions
   12. Type Conversions
3. Modules, Classes, and Methods
   1. Modules
   2. Using Modules
   3. Python Libraries
   4. Object Orientation
   5. Defining Classes
   6. Inheritance

**IDLE**

IDLE is the development environment we will use to create Python scripts (programs) in Python.

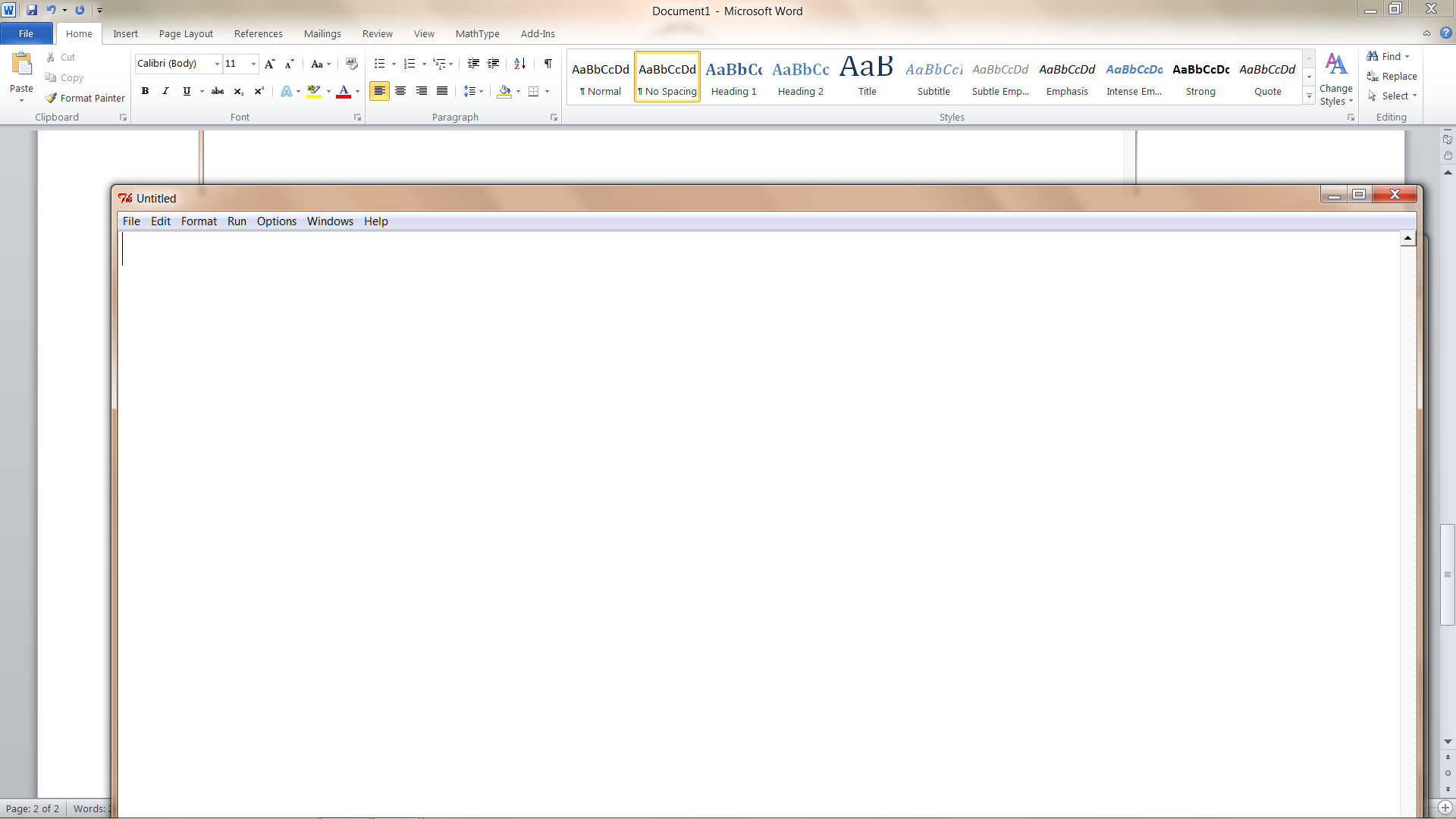


**Python Shell**

In interactive mode you can type commands here. It also provides a console output for program output.

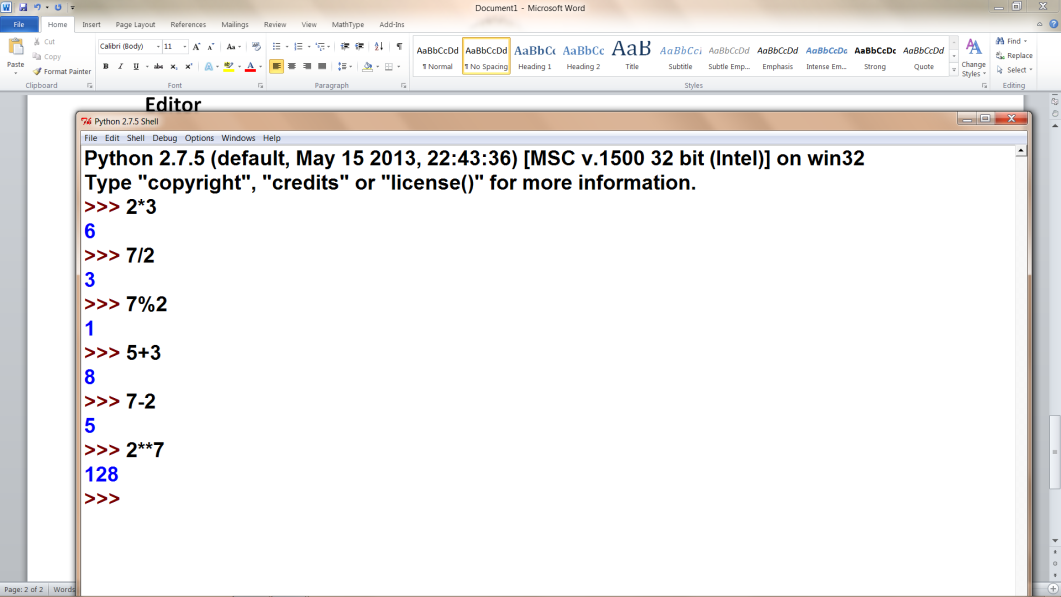
**Editor**

This is opened when you use File|New Window. In this new window you will write your scripts. The editor provides color coding for Python reserved words and syntax characteristics. In order to maintain this, you must save your file using the filename extension .py.



**Numbers**

These are manipulated and combined using the arithmetic operators found in Python. These include \*, /, +, -, \*\*, and %.



**Variables**

Memory references where data is stored. Data may be numbers, strings, or objects. Variables are typeless; for example a variable may be assigned an integer and then be assigned a string.

number = 123 #assign an integer

number = “123” #assign a string

number = 123.0 #assign a floating point

**For Loops**

Will iterate of a list

for number in range(10): #number will take on integer values [0,1,2,3,4,5,6,7,8,9]

for number in range(1, 11): #number will take on integer values [1,2,3,4,5,6,7,8,9,10]

for number in range(0, 11, 5): #number will take on values of [0, 5, 10]

for number in range(10, 0, -1) #number will take on values [10,9,8,7,6,5,,4,3,2,1]

Example: Print the cubes of the first 100 integers

for i in range(1, 101):

print i\*\*3

Example: Count down for rocket ship

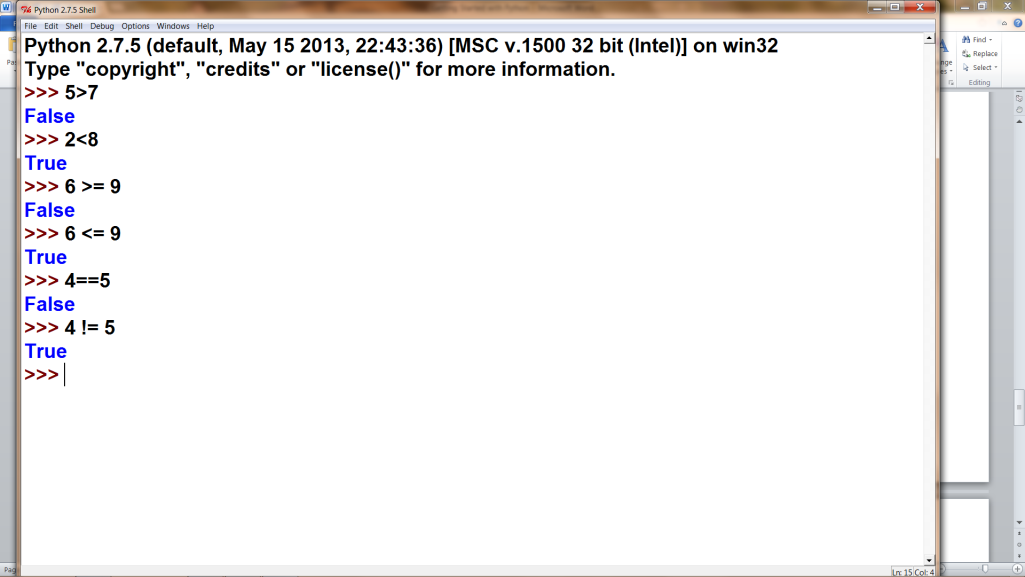
for i in range(10, 0, -1):

print i

print “Blast off!!!”

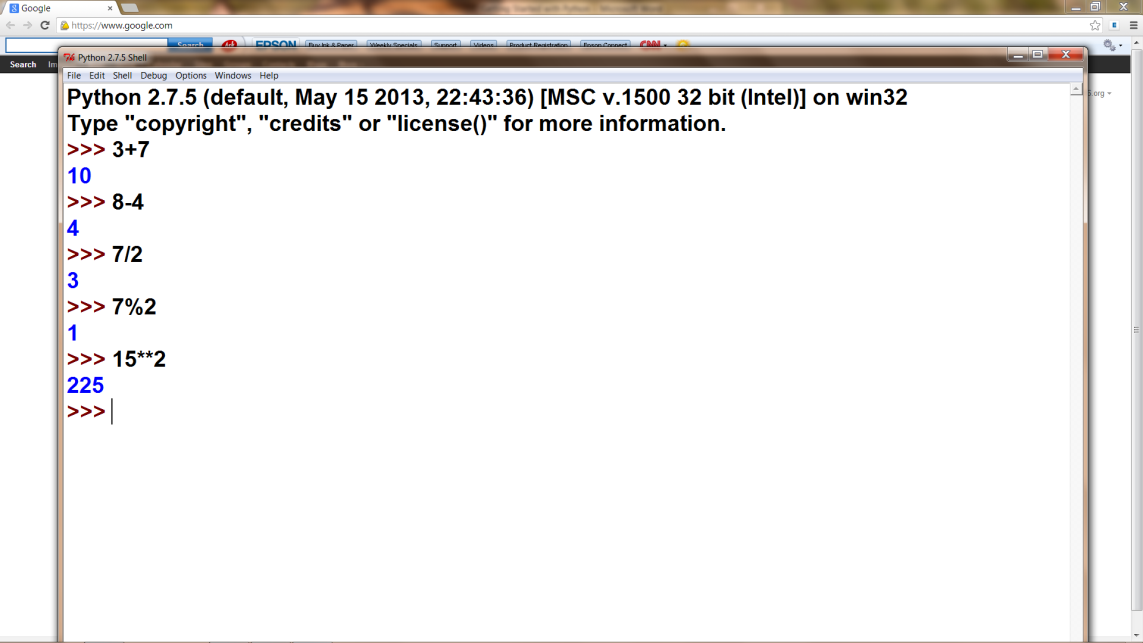
**Relational Operators**

> < >= <= == !=



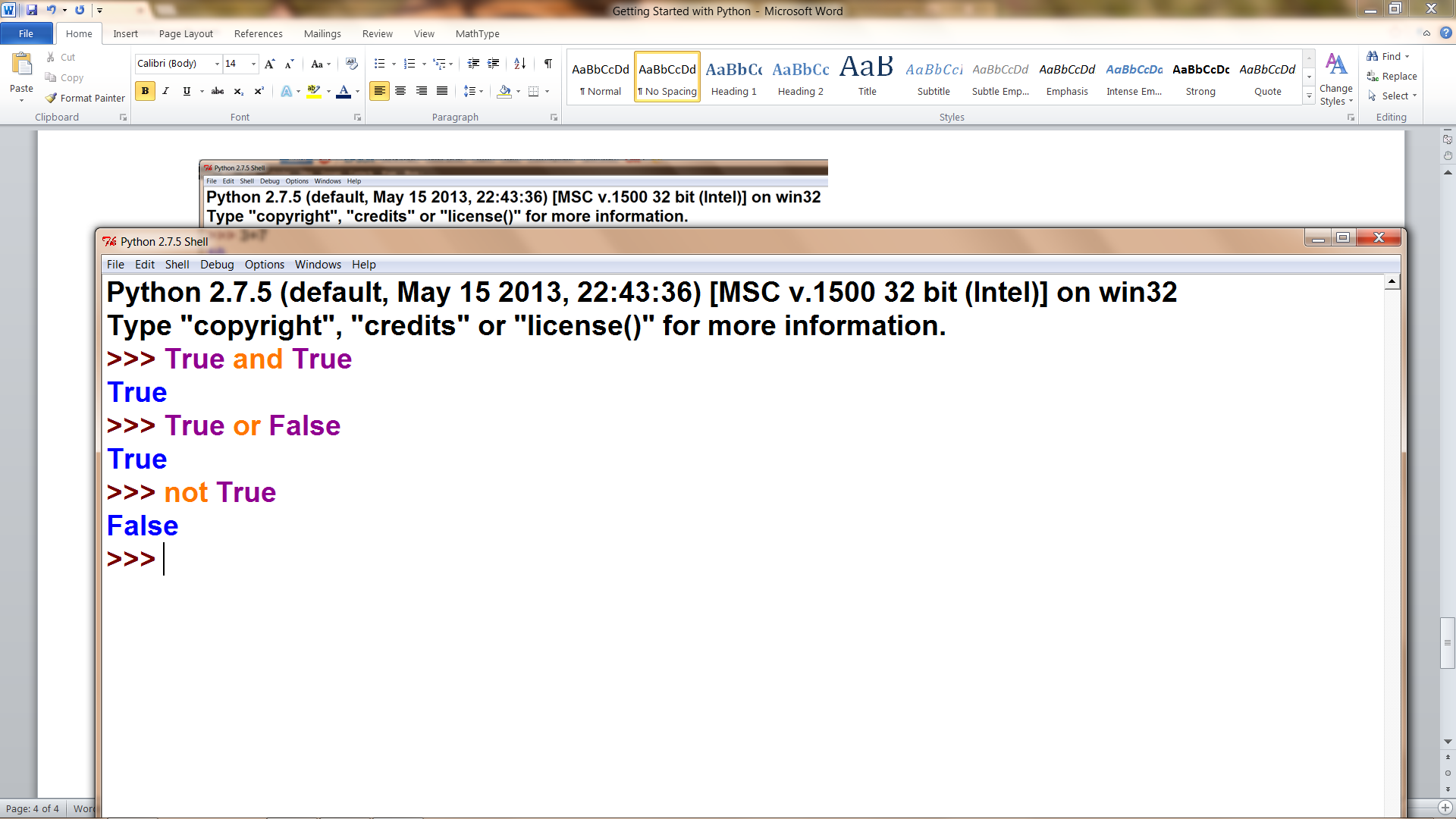
**Arithmetic Operators**

+ - / \* % \*\*



**Logical Operators**

and or not



**if / else**

if score >= 90:

print ‘A’

elif score >= 80:

print ‘B’

elif score >= 70:

print ‘C’

elif score >= 60:

print ‘D’

else:

print ‘F’

**while Loops**

while (condition is true):

#execute code

Example: Rocket ship countdown

number = 10

while number > 0:

print number

number = number – 1

print “Blast off!!!”

**Strings**

A sequence of characters delineated by single of double quotes

Example:

title = 'Programming is fun'

name = "John Smith"

There are a variety of built in string functions such as len()

Each character in the string has an index starting with 0

len(name) is 11

title[0] is 'P'

title[3] is 'g'

**Lists**

A mutable collection of objects.

Example:

numbers = [123, 34, 55, 321, 9]

There a variety of built-in list functions.

Each element in a list has an index position starting with 0

len(numbers) is 5

numbers[0] is 123

numbers[3] is 321

**Functions**

A function is a small program that can be called.

The function may or may not return something.

A function may or may not require parameters.

Example:

def addEm(a, b):

result = a + b

return result

**Dictionaries**

A collection of key:value pairs

Example:

months = {1:'January', 2:'February', 3:'March' ..... ,11:'November', 12:'December'}

months[3] would return 'March'

**Tuples**

A non-mutable collection of objects.

Example:

rgb\_value = (123, 34, 171)

**Multiple Assignment**

A method to assign multiple values to multiple variables

Example:

red, green, blue = 123, 34, 171

**Multiple Returns**

A method of getting multiple items returned from a function

Example:

def tableEntry(number):

square = number\*\*2

cube = number\*\*3

squareRoot = math.sqrt(number)

return square, cube, squareRoot

#Function call

sqr, kube, root = tableEntry(number)

**Number Functions**

May be built in to language such as:

abs(x), bin(x), and round(x, n)

May be part of math module such as:

math.sqrt(x), math.sin(x), math.exp(x)

**String Functions**

There are a variety of functions that can be applied to strings such as:

s. lower(), s.find(str), s.split()

**List Functions**

There are a variety of functions that can be applied to lists such as:

a.append(x), a.index(x), a.pop()

**Dictionary Functions**

There are a variety of functions that can be applied to dictionaries such as:

len(d), d.clear(), key in d, del(d[key])

**Type Conversions**

Used to convert one data type to another data type

Example:

int('123') will result in 123

float(123) will result in 123.0

str(7) will result in '7'

int('hello') will give an error

**Modules**

Allows you to create a group of related functions that can be used at some future time.

Any Python file can be thought of as a module

**Using Modules**

Modules are use with import

Four methods of importing:

import random

diceRoll = random.randint(1, 6)

import random as r

diceRoll = r.randint(1, 6)

from random import randint

diceRoll = randint(1, 6)

from random import \*

diceRoll = randint(1, 6)

**Python Libraries**

Some of the most useful include:

string

datetime

math

random

tkinter

**Object Orientation**

An object is an instance of a class

An object contains data as well as methods that can operate on the data

Properties of Object Orientation:

Abstraction

Encapsulation

Polymorphism

Inheritance

**Defining Classes**

class Fraction:

def \_\_init\_\_ (self, num, den):

self.numerator = num

self.denominator = den

def \_\_str\_\_ (self):

message = str(self.numerator) + '/' + str(self.denominator)

return message

#Test of Fraction class

myFraction = Fraction(3, 4) #Creates a fraction with the value of 3/4

print myFraction #prints out 3/4

**Inheritance**

class DecimalFraction(Fraction):

def \_\_init\_\_(self, num, den):

Fraction.\_\_init\_\_(self, num, den)

self.decimal = float(num)/den

def \_\_str\_\_(self):

message = str(self.numerator) + '/' + str(self.denominator) + ' = ' + str(self.decimal)

return message

myDecimalFraction = DecimalFraction(7, 8)

print myDecimalFraction