Start Here:

Python Programming

Made Simple For The Beginner

By

Jody S. Ginther

Author's websites:

www.AlienCatStudios.com

www.toonzcat.com

This page intentionally left blank just to bother people;

You will always wonder...

Table Of Contents

This book is dedicated to my father;
who inspired me to teach myself.

Introduction

Who Should Read This Book?

- Any beginner who wants to learn programming in Python
- Teachers or anyone who wants to teach Python to beginners.
- Parents who want their child to study programming
- People who are curious
- People who are bored and have nothing better to do
- You

This book is meant to help you begin learning the basics of Python programming version 3 or later. It is a brief introduction to Python. At the time of this writing, there are many resources for earlier versions of Python. However, since changes were made in the later versions of Python, using older books and resources can cause some confusion. The author recommends to all new students of programming to begin with Python version 3 or later. If you find source code that you would like to study or use, search the internet for conversion tools that can help you convert the older versions of code to be functional in 3.0 or later.

The author uses the theory that visual learning, humor, and action, (experiential learning), are the best ways for most people to quickly learn something from a book. The author attempts to be as brief as possible to get the new programmer into programming as fast as possible. When you are ready to go deeper into Python, there are many excellent free resources and books on the internet.

Who's The Author?

Who cares? "I just want to get into programming quickly!" The author chose not to include many useless details about himself that you probably don't care about anyway. Suffice it to say that he has taught internationally for many years, and is qualified to teach many subjects. The focus here is on learning to program, not on useless information. If you really want to know, look at the cover of the book or check his websites.

What's A Programming Language?

Every field of study has new words that you must learn to communicate. When you studied biology, chemistry, and other subjects you were faced with learning new words to talk about that subject. Now you are learning programming so you must also learn more new words. Before you know it, you will be able to speak to world renowned geniuses and geeks in words they understand. You will have the added benefit of understanding what you are talking about. A programming language is a language you can use to communicate with a computer. ***Programming*** *is the art and science of making the computer do what you want it to do by creating programs.*

What are programs? ***Programs*** *are algorithms and source code packaged together to achieve your objective(s).* Ahhh! More strange words! What's an algorithm or a source code? ***Algorithms*** *are sets of instructions that tell the computer what to do.* Algorithms tell the computer how to reach a goal or objective. In daily life someone may ask you for directions to the nearest chocolate factory. You may say; "Here's the algorithm for you; first, go straight ahead until you come to a street light. Turn left at the street light and go to the second parking lot on your right. Park your car in the customer parking area. Enter the back entrance to the factory and eat chocolate until you are too fat to fit through the door." This set of instructions is an algorithm. What's an algorithm? If you said, "It's a set of instructions!" you were paying attention. Good boy…or girl; you know what you are.

Ok, an algorithm is a set of instructions; then what is source code? ***Source code*** *is all the algorithms and instructions that we used in a program.* All the words, commands, secret symbols, and other stuff we typed into our program is the source code.

Now that you understand what algorithms and source code are, let's repeat the statement; Programs are algorithms and source code packaged together to achieve an objective(s)

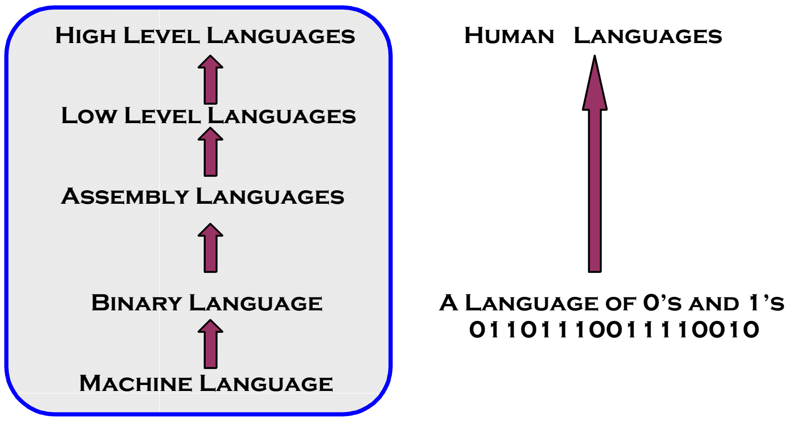
What Is Python?

Python is a computer language. Computers are stupid and don't understand English. So, we have to use computer languages to translate what we say into Computerish or Computerese. Actually, the language computers speak is referred to as a binary language. Binary language is a language based on two words; "on" and "off" represented by the numbers 0 and 1. Humans have trouble communicating in binary. If I say, " 011 001 101," you would say, "Huh?" So, languages that are easier for human brains to grasp were invented.

Why so many computer languages? Different languages were designed for different purposes. Some are better at math, some are better at controlling computer hardware, and some are better for the internet. Python is a general purpose language. It can be used for many different purposes.

Python is known as a scripting language, (uses scripts), and is a high level language. A ***high level language*** *is a computer language that is closer to human language and easier for us to use than low level or machine languages.* High level languages also take care of many tasks like manipulating the memory of the computer for you. ***Low level languages*** *are used when the programmer wants more direct control over the machine he is using.*

This chart gives you a brief over-view of the levels of programming languages.



For the sake of getting you into programming as soon as possible, this book will not expand on Python's history, details on other programming languages, or other details that would delay us from getting started. You can find excellent details on history and other stuff you may want to know at Python's web site, (www.python.org).

How To Use This Book

This book is for the beginner. It was written for the person who knows a little about using a computer; but knows nothing about programming. If you are new to Python and programming, then read and do everything. If you are an experienced programmer and read this book anyway, then jump around at will to the parts you don't know…or just read everything anyway for its entertainment value. If this is the only book you have with you and you run out of toilet paper...use your imagination. Of course, if you using the free PDF version of the book you have a problem.

Getting and installing Python

You can get your free copy of Python; choose the version you want; choose the Python your operating system; and see any special installation instructions you may need at

http://www.python.org/download/.

Chapter 1: The Beginner's Tour Of Python

WHAT YOU WILL LEARN:

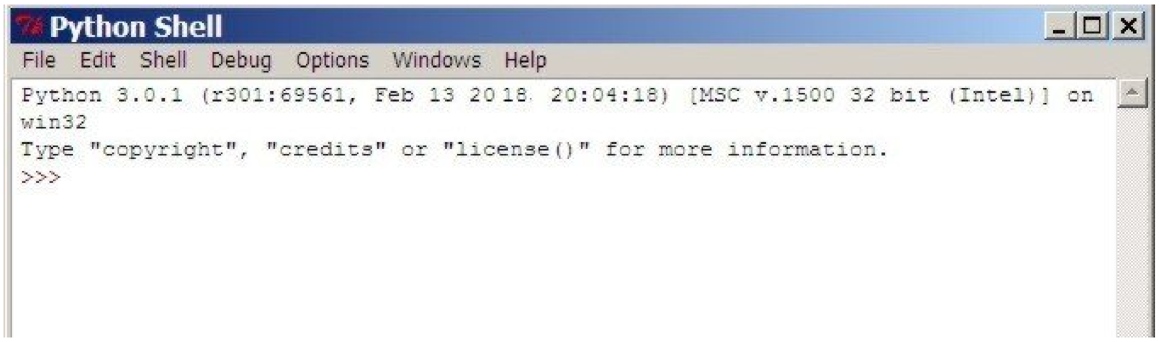
- **HOW TO USE PYTHON'S EDITOR**
- **THE DIFFERENCE BETWEEN IDLE AND THE SHELL**
- **ABOUT THE COLORS USED IN IDLE**
- **WHAT ARE BLOCKS OF CODE?**

IDLE AND THE SHELL

First, you can start Python in Windows by clicking on **Start>Programs>Python 3.x>IDLE**. Go ahead, try it...I'll wait here. ***IDLE** is software that helps you to communicate with the computer.* It is on the outside of the main program, just as a snail's shell is on the outside of the snail. Whack on a snail's shell and he will get the message. IDLE works in the same way. Enter commands into IDLE and it will send the message to your computer. IDLE acts as your *interpreter* and translates what you say into a language that the computer can understand. If you really want to know, IDLE stands for **I**neractive **D**eve**L**opment **E**nvironment. Why did they choose the L in the middle of the word, "development?" I have no idea and it's not important for the purposes of this book, so let's move on and get over it.

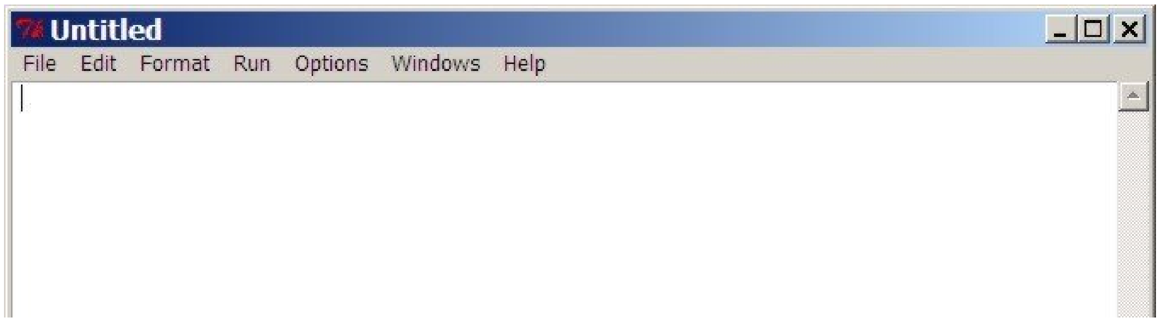
There are two windows to work from in IDLE. There is the **Edit Window** and the **Shell Window**. The Python Shell window will say, "Python Shell" at the top of the window, while the Edit window will say, "Untitled" and have a "run" command listed on the top menu bar. If Python starts in the Shell Window and you want to use the Edit Window, just choose **File<New Window** and it will open an Edit Window. If you want to use the Shell Window if the Edit Window starts go to the "Run" menu at the top of the window and choose "Python Shell." The ***Python Shell** is an interactive interpreter.* This means that when you press the enter key, it checks your source code and may give you some feedback. If you see (>>>), this is the first command prompt. It is letting you know the interpreter is patiently waiting for you to type something. If you see (…), this is the secondary prompt waiting for you to type something more. If you found these two windows they should look something like this:

The Python Shell looks like this:



```
Python 3.0.1 (r301:69561, Feb 13 2018  20:04:18) [MSC v.1500 32 bit (Intel)] on
win32
Type "copyright", "credits" or "license()" for more information.
>>>
```

The editor looks like this:



There are other editors that you can use for programming but to keep things simple in this book we will use the IDLE software that is packaged with Python.

Why are there two windows in IDLE and how do I use them? Python allows you to work in script mode or in interactive mode. What's the difference?
***Script Mode** is great for writing programs you can save and run later. It is generally used for the final product.*
***Interactive mode** is for testing and trying small ideas quickly.*
Most people use both of these together. Script mode for working on their main program and interactive mode for trying new ideas in the same way you would use scratch paper.

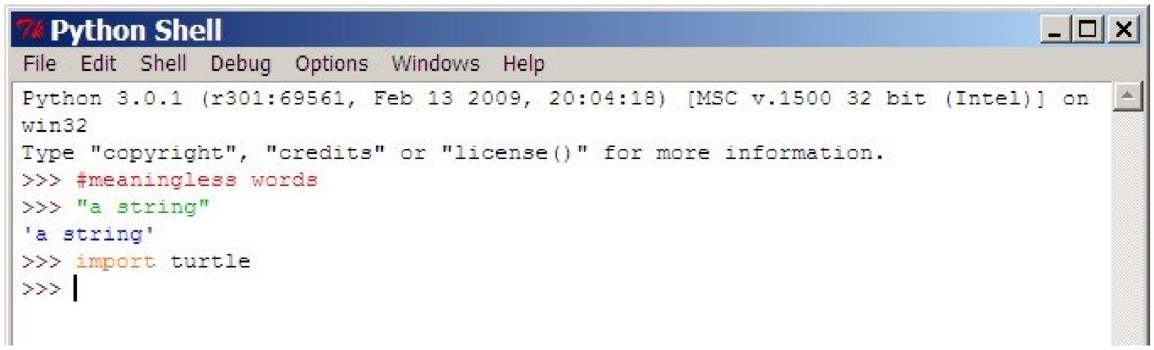
A third function of IDLE is that it is also a debugger. You can find the debugger button at the top of the Shell window. What's a debugger? It's a program to help you kill bugs. No not the one crawling up your leg; but the bugs, or problems, that are in the code we are working on.

IDLE Colors

Did you notice that IDLE changes the colors of your text? What's the meaning? Let's look at a list of some of the most common syntax colors and their meanings in IDLE.

Common Python syntax colors:

Keywords	orange
Strings	green
Comments	red
Definitions	blue
Misc. Words	black



```
Python 3.0.1 (r301:69561, Feb 13 2009, 20:04:18) [MSC v.1500 32 bit (Intel)] on
win32
Type "copyright", "credits" or "license()" for more information.
>>> #meaningless words
>>> "a string"
'a string'
>>> import turtle
>>> |
```

You needn't memorize these at this point, but knowing the meaning of the colors can help you see clearly where you typed something wrong. If you were trying to type a string and its not green, you probably forgot the quotation marks or did something else the computer didn't like. The colors can also be changed according to your personal preferences and may vary in different editors.

Blocks of Code

While we are looking at the windows, how is your text arranged? The text is arranged in lines, groups, and blocks. A ***block*** *is just a group of code that goes together*. Like a city block, it can be divided into smaller groups like houses on the block, cars on the block, dogs, lions, etc. on that block. This concept is important to tell the computer how to read and follow your code. To a computer, arranging your code with the proper grouping of lines and blocks is like a map that says; "first do this, second do that, or repeat this. "

Blocks are one way we dictate the running order in programming. You can have blocks in blocks just as you can have groups in groups. You may have a group of students under

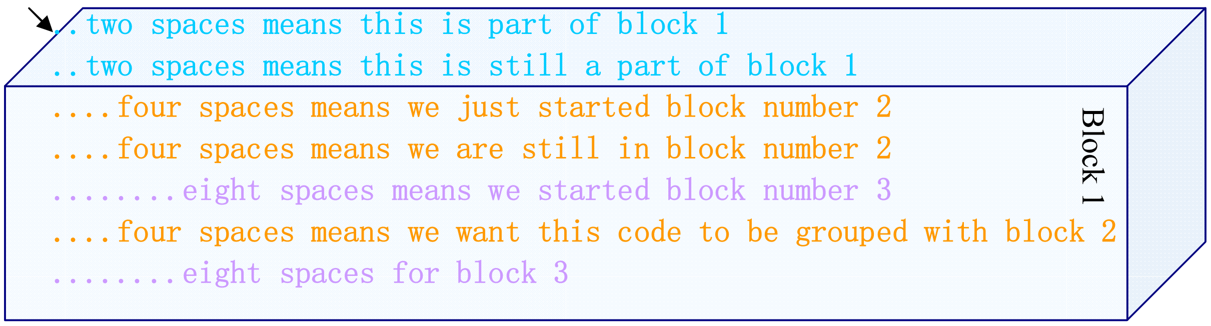
100 years old. But, within that group you may have another group called "girls." Inside of that group you may have another group or block called "girls with green hair."
Blocks make it easy to keep things or instructions in our code together in their correct group. This helps us refer to them and to direct the computer to use that group of instructions, in the order we want the computer to use them. Blocks are defined by the number of spaces used to indent each line of code. In the following examples I will use dots to show you clearly how many spaces would be in the code. This; "**...**" refers to three spaces. Use spaces, not dots, when you type your code.

Note: The color code in the following examples is not a part of IDLE. These colors are used to emphasize what parts of a block belong together.

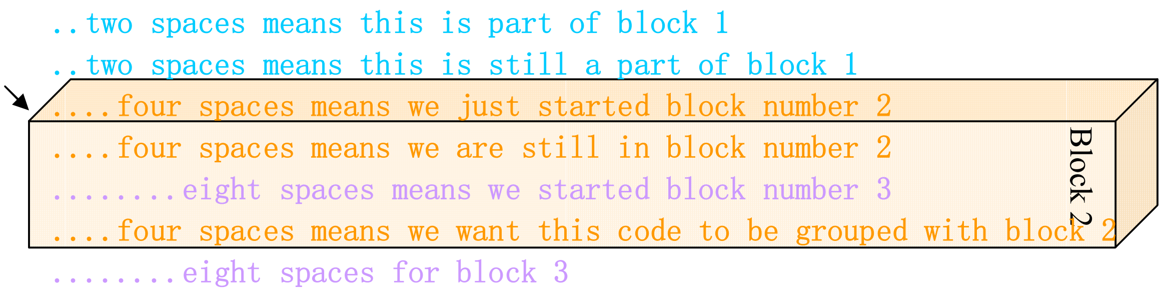
```
2 spaces   ..students under a hundred years old   (Block 1)
4 spaces   ....girls   (Block 2)
8 spaces   ........girls with green hair   (Block 3)
```

The number of times we indent helps the computer know how we are grouping the information. (Hang in there, a few more ideas and you will make your first game).

Here is another example;

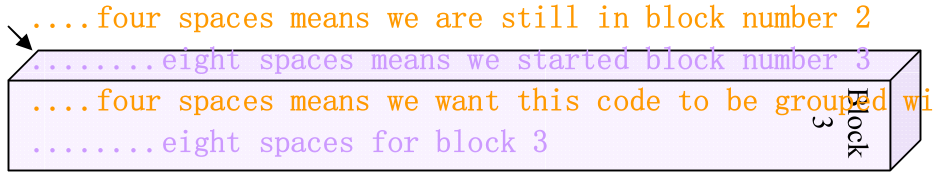


```
..two spaces means this is part of block 1
..two spaces means this is still a part of block 1
....four spaces means we just started block number 2
....four spaces means we are still in block number 2
........eight spaces means we started block number 3
....four spaces means we want this code to be grouped with block 2
........eight spaces for block 3
```



```
..two spaces means this is part of block 1
..two spaces means this is still a part of block 1
....four spaces means we just started block number 2
....four spaces means we are still in block number 2
........eight spaces means we started block number 3
....four spaces means we want this code to be grouped with block 2
........eight spaces for block 3
```

```
..two spaces means this is part of block 1
..two spaces means this is still a part of block 1
....four spaces means we just started block number 2
....four spaces means we are still in block number 2
........eight spaces means we started block number 3
....four spaces means we want this code to be grouped with block 2
........eight spaces for block 3
```



Whatever the number of spaces you choose, keep it consistent so you don't get confused. *Why do we care about blocks and grouping programming statements?* Ah, I'm glad you asked, but I won't tell you until later. (Don't worry you will get to play with these shortly).

VOCABULARY REVIEW: *(Yes, you should read these again)*

Algorithms *are sets of instructions that tell the computer what to do.*
Block *is just a group of code that goes together.*
high level language *is a computer language that is closer to human language and easier for us to use than low level or machine languages.*
IDLE *is software, (an editer), that helps you to communicate with the computer.*
Interactive mode *is for testing and trying small ideas quickly.*
Low level languages *are used when the programmer wants more direct control over the machine he is using.*
Programming *is the art and science of making the computer do what you want it to do by creating programs.*
Programs *are algorithms and source code packaged together to achieve your objective(s).*
Python Shell *is an interactive interpreter.*
Script Mode *is great for writing programs you can save and run later. It is generally used for the final product.*
Source code *is all the algorithms, statements, and instructions that we used in a program.*

Chapter 2: You are now a programmer!

WHAT YOU WILL LEARN:

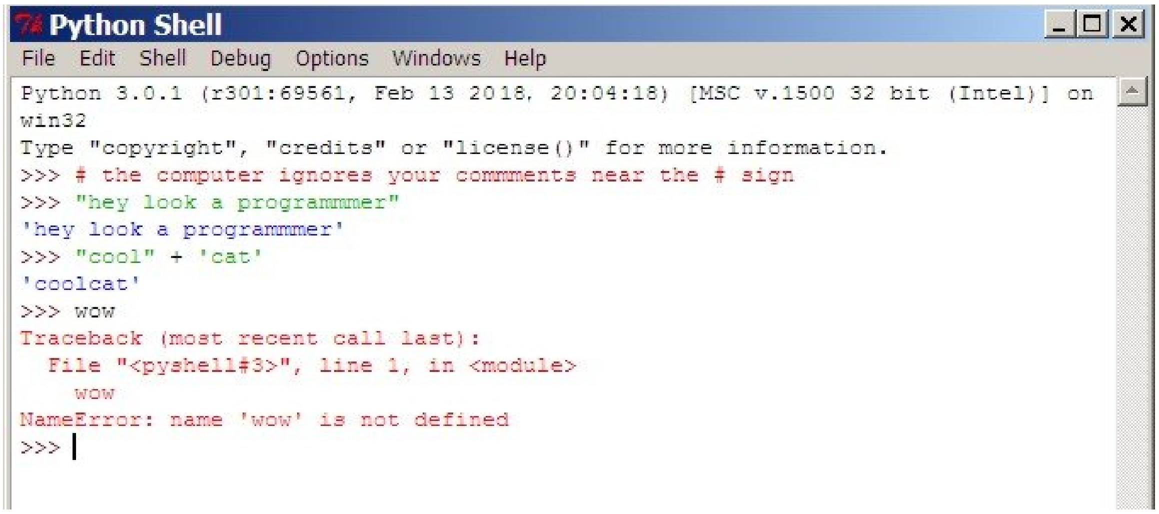
- YOUR FIRST PROGRAM
- STRINGS
- VARIABLES
- OPERATORS
- KEY WORDS (TO USE OR NOT TO USE)
- BOOLEAN DATA TYPES

LET'S START PROGRAMMING

Ok, let's start playing…err… programming. Open your Python Shell window to type some things. Type the following code exactly as you see it. ***Then hit the enter key after each line.***

```
# the computer ignores comments near the # sign
"hey look a programmer"
"cool"  +  'cat'
wow
```

If you entered these correctly, it should look like this:



```
Python 3.0.1 (r301:69561, Feb 13 2018, 20:04:18) [MSC v.1500 32 bit (Intel)] on
win32
Type "copyright", "credits" or "license()" for more information.
>>> # the computer ignores your commments near the # sign
>>> "hey look a programmmer"
'hey look a programmmer'
>>> "cool" + 'cat'
'coolcat'
>>> wow
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    wow
NameError: name 'wow' is not defined
>>> |
```

If you type the # key in front of some text, the computer ignores you. This is useful for adding comments in your programs that will not be misunderstood as instructions by the computer; or if you just feel like being ignored.

Strings

If you noticed I typed the words `“hey look a programmer”` in quotation marks. Later I used; ‘ ‘ and got the same exciting result. When we hit enter, the computer just repeated what we typed back to us. It printed the characters as output on the screen. A sequence of characters, words, or sentences like this one is called a ***string***. When we use single or double quote symbols to tell the computer what is in our string, we call these quote symbols “***delimiters***.” We tell the computer that we are entering or ending a string by using single or double quotation marks. The computer don’t care which kind you use at this point. We can now say we have declared or “delimited” the string. In IDLE strings are green and `the output here is blue`. The `error message is red`.

You can also add strings together using a math operator. We will talk more about that in a moment. Putting the string “cool” with ‘cat’ produced `‘coolcat’` . If I want a space between them when they are added together, I should add one in my string; “cool “ + “cat” or “cool” +” cat” would generate `‘cool cat’` .

What happened when I typed, `“wow?”` The computer said; `“Blah blah blah…is not defined. ”` This is the computer’s way of saying; “huh? I don’t understand.” Python attempts to give you an idea of what went wrong. When I typed the word, I did not include it inside of quotation marks to tell the computer that I was entering a string. So, the computer went, “Huh?”

Have you ever asked; “When am I going to use this kind of math?” In programming you should remember some simple math concepts to make your life easier. Don’t worry, I’ll be brief. To begin with, the world of math has animals called “operators” and “variables.”

Variables

***Variables** are like little like boxes or containers to put different things in.* In math your teacher may have told you that 1 + *x*= some other number. The 1 is an ***integer**, (a complete number as opposed to part of a number like ½),* and the *x is a variable.* In programming, you get to name your variable anything you want. You can create an imaginary box with anything you want to put in it, and define/label that imaginary box, (or variable), by any name you choose.

Let’s try it. Let’s imagine a box of chocolate. We want to tell the computer that the box labeled “chocolate” has happiness and joy inside of it. To do this we use the = sign to define what a variable means for the computer.

In the Python shell window type:

```
Chocolate =  “happiness and joy”
```

Hit the enter key.

Now, let’s type the word without quotes;

```
Chocolate
```

Hit enter.

Your computer should reply, `‘happiness and joy’`

You just *defined* a variable. This is very useful in programming. You will constantly be teaching the computer how to think as you write programs.

Variables *are chunks of data stored in the computers memory.* There are generally three types of data stored in variables. Variables can be in the form of ***integers*** or in a ***string*** as mentioned previously. The second type of data, called a ***float***, refers to the *non-whole numbers like decimals.* Remember to use the = sign to assign a variable. If you want the computer to think the variable *x* means 5 is in the box named *x,* then you type:

```
x=5
```

Now the computer holds a 5 in memory and when you type an x, it tells you ‘5’ if you hit the enter key. You can name a variable almost anything and use symbols like the _ underscore. But there are some rules. You can’t use special key words that Python understands as having special meanings. Don’t use these words:

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