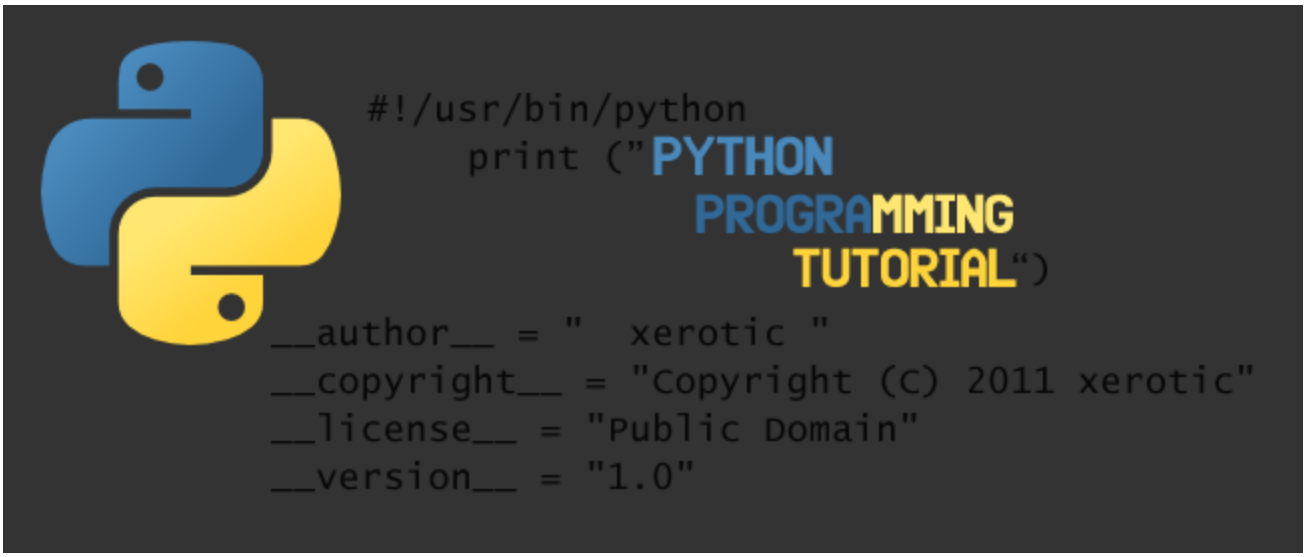
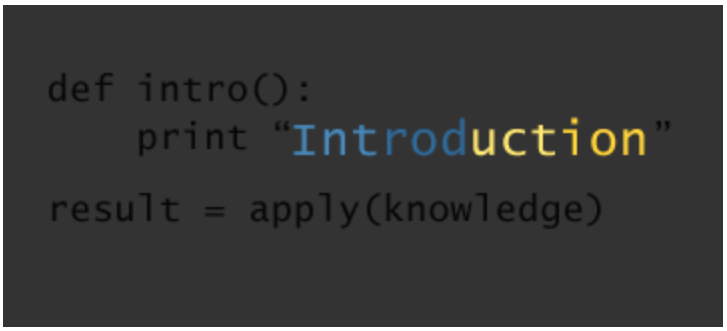


- -



What is Python?

Python is an interpreted, object-oriented, high-level programming language. This means the Python requires an interpreter to run. It cannot be compiled in the way C++ can. It must have the interpreter.

Some may think this is a disadvantage, but I think otherwise. With Python, you can debug at the source level, increasing productivity. You can write and test your programs instantly, and find any possible errors in seconds!

- - - - - - - - - -

Where is it used?

Python can be ran on almost any OS. While it is *not* native to Windows machines, it *is* native to most Linux distributions. The great thing with Python is its flexibility and usefulness in many environments.

- - - - - - - - - -

When can I use it?

Python is not limited to your local machine. You can write Python programs/scripts to interact with other computers, either on your Local Area Network (LAN) or across the Internet. Python can be used to make interactive webpages and some of the best programs. You may know some programs made in Python. A few examples include:

- BitTorrent
- Blender 3D
- Battlefield 2
- And many more! You can find a huge list of programs made in Python HERE

As you can see, many popular programs are created using Python. You can make these programs

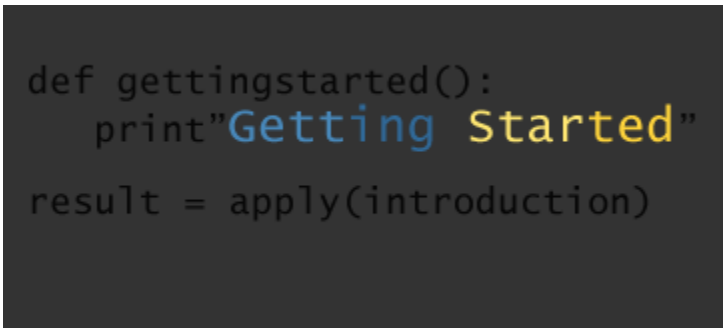
too!

- - - - - - - - - -

What else do I need to know before I start?

Python can become addicting! It is easy to learn and easy to code with. Before you continue with this tutorial, please have an urge or push to really start your Python Programming career!

- -



Choosing the right Python version

Many people will have many different opinions on which Python version to choose. Personally, I use either Python 2.6 or 2.7. The reason I use 2.6 is because it is *supported*, meaning that it has the most support and additional modules *as of this time*. In the future, Python may offer support for versions 3.x.
I also use Python 2.7 because it is what some people call *future-proof*. This means that 2.7 was written *after* 3.x, so it offers some compatibility between the two.

Currently, Python 2.x would be your best bet. Name companies, such as Google, still use the 2.x versions of Python. So can you. Do NOT feel pressured to need the latest versions, they are not always the best.
In this case, functions are handled differently and some things have a different syntax. It can get confusing for a new user.

- - - - - - - - - -

Downloading and Installing Python

To download Python, you will need to visit their website:

http://python.org/

And their download page:

http://python.org/download/

You should have decided in the last section which version to download. If you are still questioning which version to choose, just get python 2.7

For Windows, here are two of the MSI installer links:

Python 2.7.1 Windows Installer
Python 2.7.1 Windows X86-64 Installer

Once you have downloaded the appropriate file (if you are not sure, use the first download link) run the MSI file and leave all options at default. It should install at "C:\Python27" with the numbers as your Python version. If it did not install that, no worries, anywhere is fine.

If you are using Linux, you may find it easier to get Python using the following video. (Note** Many Linux Distributions already include Python.)

Linux-Python Installing Video (Click to View)

Once you have Python installed on your machine, move on to the next section.

- - - - - - - - - -

Figuring out how it works

For the rest of this Tutorial and Guide, I will be explaining Python from a Windows point of view. I will include how to run Python files from Linux, but I will not delve into Linux Text Editors and the like.

To make sure Python is working, press the Windows Start Button (bottom left) and type in "python" (assuming you have Windows Vista/7, if not, use 'Run' to run "python") and press enter. You should see a DOS type of box appear on the screen with the insertion mark ("_") after three arrows '>>>'.

If you can see this, Python is installed correctly! Good job, now let's move on to coding!

- -

The Python Syntax

Python's syntax is extremely easy to learn. Unlike many programming languages, it does NOT require a semi-colon (";") at the end of every line. That is a change for most people.

Another thing is that Python uses 'white-space' instead of curly brackets ("{}") to separate functions and classes. This keeps the source neat and very easy to understand. Due to the simple syntax and readability of code, it is easy to pick up another coders work from nearly any point in their project. This gives Python loads of flexibilty with collaborative projects.

Another good thing to remember with Python is that it IS case sensitive. This means that "apple" is different that "Apple". This is important.

I also want to point out now, you can add a comment to a Python program using the pound ('#') symbol. Use it like so:

Code:
```
#This is a comment!
```

Comments are ignored by the interpreter and usually are only used to include version information or to keep notes on the source code.

- - - - - - - - - -

Hello World!

It is time for what many of you have been waiting for. Programming. Scripting. Whatever you would like to call it. It is starting now, so get ready!

Okay, so as with many programming languages, I will start out with a simple program, which is called 'Hello World'. What this program will do is display the text "Hello World!" on the screen and await user input to close.

Alright, so you will need to open up IDLE (for Windows users). This can be done by typing 'IDLE' in the Windows Vista/7 search box in the Start Menu. Once that opens, go to 'File' and click 'New Window'. This gives us a blank page for us to write out our programs.

Alright! All set up, time for some code!

Here is the first bit of code we will use:

Code:
```
print "Hello World!"
```

Do NOT copy that straight into IDLE. Instead, type it out yourself, you will benefit more in the end.
What this code does is tell Python to *Display* (print to screen) the following to the screen. Since we enclosed the text ("Hello World!") in quotes, that means it is a string. A string just means text, no numbers. Simple.

So that one lines of code displays "Hello World!" to the screen. The only problem is that if you try running this, it will close instantly, so let us add some code to stop it.

Code:
```
print "Hello World!"
raw_input("Press <enter> to close.")
```

Type that out into your text editor.

"raw_input()" is a simple way to grab data from the user. It is called *input*. You may have noticed the quotes in the "raw_input()". Let me explain. That function by itself without the quotes ("raw_input()") would work 100% to keep your program from instantly shutting down, but the quotes with text gives the user *instruction* of what to do. This is important.

So now that you have those two lines of code, either go to 'File' > 'Save' ('Save As') or press 'CTRL' + 'S'. This comes up with the save window. Due to simplicity, you can save the file at your desktop. By default, it should have the option to either save as a "*.txt" file or a "*.py" file. In our case, we want a "*.py" file.

So type in the file name bar: "example.py"
Then hit 'Save'.

From here, you have a few different options. You can double-click or run the file as you would any program. Doing this will result in the following on your screen:

Spoiler (Click to View)

When you press enter, the program will end.

Another way you can run the program is to just press 'F5' after you save it (while the IDLE text editor is open). This will open it in Python's native debugger. This is very useful when writing large projects, as it can pinpoint exact errors in your coding.

In Linux, simply type the following into the terminal to run the file:

Code:
```
python fileName.py
```

If you have successfully ran the file, congratulations! If not, go back up and read again. Time to get into some interesting concepts.

- - - - - - - - - -

Math Functions and Variables

In any programming language, Mathematics and Variables are of KEY importance. You cannot possibly write a leading program without them. They are they key to programming, so without further ado, lets explain how they work in Python.

Firstly, a variable can hold assorted types of data.

These include:

- Integers (int)
- Strings (str)
- Boolean Values (True or False)

We will cover those three in this section.

In Python, it is beyond easy to declare a variable. All you have to do is type the variable name and its value. A variable can be named nearly anything (do not use spaces though!) so it is very flexible. In my examples, I will use an assortment of words and single letters as variables.

This is how you declare a string variable:

Code:
```
x = "This is a string"
```

What happened? First, you have the variable name (in this case 'x'). Then, you put an '=' sign. Finally, you put its value, enclosed in quotes. With python, you can use single quotes or double quotes, the choice is up to you.

Now let us use the integer variable type.

Code:
```
y = 23
```

Could that get any easier? Again, all you have to do is type the variable name, an equal sign, and its value. With integers, you do not use quotes. If you do use quotes with a number, it will be treated as a string and NOT a number. So with integers, leave out the quotes.

Finally, we have the Boolean variable type. This holds only two different options. True or False. You declare Boolean variables like this:

Code:
```
z = True
a = False
```

Simple. You either write 'True' or 'False' (remember, case matters!).

Now that Variables are covered, it is time for basic math functions.

The first one I will cover it addition and subtraction.
If you can program, you should already know how this works, mathematically, but let me show you how it works in Python.

You can use IDLE for this. Use this snippet of code:

Code:
```
print 4+5
print 7-3
```

If you run that program, you will get the following output:

Code:
```
9
4
```

Basic math. With addition and subtraction, you can only use integer values.

Now, multiplication and division.

Multiplication ("*") can be handled with String variables and Integer variables. For example, use the following snippet of code:

Code:
```
x = 5
y = "5"
print x * 5
print y * 5
```

Your output will be:

Code:

```
25
55
```

For the integer variable, it is self-explanatory, basic math. However, the string variable acted differently! When you multiply a string variable, it just repeats itself. Simple enough, right?

Now time for division (and modulus).

To divide with Python, use the division sign ("/")

Code:
```
x = 20
print x / 4
```

Your output will be '5'.
With Python, if you try dividing numbers that will result in a decimal value, you have to include a decimal place in the operation. Let me show you in action.

Code:
```
print 20 / 3
print 20 / 3.0
```

If you run that, you will get the following output:

Code:
```
6
6.667
```

The simple ".0" makes a big difference. Always remember to include a decimal place somewhere when using division!

Earlier, I mentioned modulus ('%') with division, because they are very similar. When you were younger, you were probably taught how to divide and then how to 'leave the remainder'. This is exactly what modulus does. It does the division and gives you the remainder. Lets see how it works.

Code:
```
print 20 % 5
print 20 % 3
```

Your output will be:

Code:
```
0
2
```

Simple. All modulus does if give you the remainder.

The last math function I will show you is exponents ('**'). An exponent tells you how many

times to multiply a number. For example, '6**3' is the same as '6*6*6'. Let us see Python work this out.

Code:
```
print 8**4
```

Your output will be:

Code:
```
4096
```

Simple. You should have already been familiar with all of these math operations, but now you know how to utilize it in Python!

One more thing with variables..

They can either be local, or global. This will be important later in the tutorial, when function definitions are included, but this is how you make a variable global (used everywhere in the program).
Code:
```
global x
```

With 'x' being the variable.

- -

```
def onemorestep():
   print"One more step.."

result = apply(introduction, getting started,
lets start programming!)
```

If, elif, and else

In Python, you will want to check against user input at some point, so the *if* statement will come in handy. Before I show you the code for it, let me tell you about it.
The 'if' statement pretty much just checks against something, and returns a result based on the answer. Let me give you an example of where you have to enter the correct phrase in order to see the "Success!" text.

Code:

```
x = raw_input("Password:  ")
if x == 'lemon':
    print 'I love LEMONS!'
elif x == 'apple':
    print 'I hate APPLES!'
else:
    print 'I have no idea what that is!'
```

Let us dissect this line by line.

In the first line, we get input from the user with the statement, "Password:". The user input is assigned to variable 'x'.

Next, we open our 'if statement' with the first condition. If 'x' (what the user typed) is equal ('==') to (exactly) 'lemons', then do the following (':') and display ('print') the text.

Firstly, 'if' must be lowercase. Secondly, you must use double equal signs ('==') when using 'if/elif'. Thirdly, if you are checking for a string, make sure the what you are checking (in this case 'lemon') is enclosed in quotes. Finish it off with a colon then go to the next line.
The next line MUST be indented (4 spaces) in an if statement.

The 'elif' works exactly the same, EXCEPT that it handles every other "check/question" after the initial "if". You can have an infinite number of "elif's".

Finally, 'else'. This is only raised if the "if" and none of the "elif's" were met. So a way of thinking of all this is..

"If the user types in lemon, tell them you love lemons. But if they type apple, tell them you hate apples. If they do not type one of those two, tell them you don't know what they are saying."

That is how you can think of it, in English.

I suggest that you use the techniques you have learned so far to create a simple program, such as a calculator program (I will post the source below of a simple calculator). Until then.. lets move on!

- - - - - - - - - - -

While loop

The 'while loop' is the most basic of Python loops. What a loop does, is perform a series of events as long as something is in a certain condition. That condition can be anything. For example, your condition can be that if 'x = 5' then to perform something 5 times. It is simple.

I will show you an example of a basic while loop.

Code:

```
x = 5
while x > 0:
    print x
    x = x - 1
print 'We are out of the loop!'
```

With this, your output will be:

Code:
```
5
4
3
2
1
We are out of the loop!
```

Now let me explain each part.
First, we declare a variable, but you already should know about that.
Next, is the while loop. What this does is check to see if 'x' is *greater than* 0. As long as x is larger than 0, it will execute the content in the loop. Also notice the colon (':') at the end of the 'while loop' line.

Next, I have the program display what 'x' is currently set to.
Finally, I use basic math functions to reassign "x's" value. What I do is set its value equal to one less than its current value. Without this, the loop would go on forever.

This is a basis loop. You can check for other conditions as well, such as

Code:
```
while x == 3:
while x < 10:
```

Those are some other examples. You can probably think of many more.

To get out of a loop early, use the following code:

Code:
```
break
```

This simply takes (breaks) you from the loop and continues with the program.

- - - - - - - - - - -

For loop and Lists

In Python, lists are very important, as are for loops. They are used in order to create complex programs and to create ease with variables.

A list, in Python, is a variable that holds multiple values. They are similar to arrays in other

languages. It is declared the same as any other variable, except that it needs to be enclosed with brackets ("[]") and separated with commas (",").

Code:
```
x = ["Item One", "Item Two", "Item Three"]
print x
print x[0]
print x[1]
print x[2]
```

When you run that in IDLE, you will get:

Code:
```
['Item One', 'Item Two', 'Item Three']
Item One
Item Two
Item Three
```

Now, let me explain it.

In the first line, we declare that variable 'x' will hold the list. We start the list with an opening bracket ('[') and start with our first item (notice that since it is in quotes, it is a string). You then separate each item with a comma. Simple.

Next, you print (display) the list contents. That is also simple. Now the next line is new! We told Python to print (display) 'x[0]'. Let me explain this. When you work with list items, you will use brackets ('[]') rather than parenthesis. You are probably wondering though, why did I say '0' instead of '1'? Well, with Python, as with most programming languages, lists start with '0'. So when I say 'print x[0]', I am telling Python to give me the first list items.

This is just the basics of lists. Later in this guide, I will mention a link to where you can learn further with Python lists.

Now, the 'for loop'. The for loops syntax is different that the while loop. Let me show you an example.

Code:
```
for i in xrange(5):
    print i
```

Your output:

Code:
```
0
1
2
3
4
```

Python 'for loops' iterate over a list. This means it goes through ever item in the list to perform

the loop.

You initialize the 'for loop' with "for" and then a variable. In this case, I chose 'i'. Next, you have to include the word "in" and then the function. In this case, "xrange()". This means that it will perform the loop 5 times.

When I print "i" on the next line, it is for each iteration of the loop (remember, Python starts counting at '0').
This is a simple explanation of Python 'for loops'. There are many more things you can do with them, all of which can be found in the Python Documentation.

Time to move on to Function Definitions.

- - - - - - - - - -

Function Definitions

Functions in Python are very important. They separate areas of the code to not only organize it, but to utilize it to its fullest potential. The first thing you have to do is define it, with its name, and then write the contents of the function on the following line(s).

Code:
```
def Xero():
    print 'Here is some content!'
    print 'And some more...!'
    for i in xrange(5):
  print 'Hello!'
Xero()
```

And your output in IDLE:

Code:
```
Here is some content!
And some more...!
Hello!
Hello!
Hello!
Hello!
Hello!
```

Alright, now let me explain it. You start it off with defining it ('def') and then whatever name you want followed by parenthesis ('()') and a colon (':'). Then you indent (4 spaces) and begin writing you code for that function!

To call the function, simply type the function name with parameters ('()'). Once the function is done, it will return back to the point to where it was called. This is just a simple example of Python functions. As a beginner, you will not need to fill the parameters.

- - - - - - - - - -

Modules

Modules, simply stated, are Python programs that you call to use. They are Python scripts already written by another person in order for you to have an easier time doing certain functions. I will provide an example of one function, and you can find any other on Google.

To call a function, you need to *import* it. To do this, simply type 'import' and the Python file name.

Code:
```
import time
```

'Time' is a basic Python module that comes with the Python package that you downloaded earlier. Now, it is time for us to use this module.

Type in the following code, and run it.

Code:
```
import time
print 'This box will close in 5 seconds!'
time.sleep(5)
```

Alright, so the first thing it does is import the time module. Simple enough, right? Then I added a 'print' function, and finally the "time.sleep(5)". This will stop (sleep) the program for five seconds. Be noted, using 'time.sleep' will freeze the entire program for five seconds. You will not be allowed input. You may also wonder.. how will you know everything in a given module? Well, you can always look up the documentation for it. Another thing you can do is this:

Code:
```
import time
dir(time)
```

Replace 'time' with any module of your choosing. What this does is list everything that the module defines.

- -

```
def programs():
   print"Simple Programs"

result = apply(introduction, getting started,
lets start programming!, one more step..)
```

Calculator

In this section, I will show you some source codes of simple programs, using the techniques I taught you in this tutorial. I highly suggest that you try making the programs before looking at my source.

The first program will be a simple calculator to handle addition, subtraction, multiplication, and division.

Here is my source code:

Code:

```
global x
x = 1
def main():
    global x
    print "\n\n\n1.) Addition\n2.) Subtraction\n3.) Multiplication\n4.)
Division\n5.) Free Hand\n6.) Exit"
    choice = int(raw_input("Pick an option (1-6):  "))
    if choice == 1:
  Addition()
    elif choice == 2:
  Subtraction()
    elif choice == 3:
  Multiplication()
    elif choice == 4:
  Division()
    elif choice == 5:
  Free()
    elif choice == 6:
  x = 0
    else:
  print 'Invalid Choice!'

def Addition():
    x = int(raw_input("First number:  "))
    y = int(raw_input("Second number: "))
    print 'Your result is:  ' + str(x+y)

def Subtraction():
    x = int(raw_input("First number:  "))
    y = int(raw_input("Second number: "))
    print 'Your result is:  ' + str(x-y)

def Multiplication():
    x = int(raw_input("First number:  "))
    y = int(raw_input("Second number: "))
    print 'Your result is:  ' + str(x*y)

def Division():
    x = int(raw_input("First number:  "))
    y = int(raw_input("Second number: "))
    print 'Your result is:  ' + str(x/y)
```

```
def Free():
    x = raw_input("First number:  ")
    print eval(x)

while x == 1:
    main()
```

Learn from that source code. If you do not know what something does, look it up on Google. Reading another's source, and learning from it is a great experience for a coder, in my opinion.

Now, I have an example of a Logbook type of program. What this does, is retrieve input from the user and stores it in a text document in a human-readable form. Again, please learn from this source code:

Code:

```
global x
x = 1
def main():
    global x
    choice = int(raw_input("1.) Log a Person\n2.) Exit\n\nChoice (1-2):  "))
    if choice == 1:
  Log()
    elif choice == 2:
  x =0
    else:
  print 'Invalid Choice!'

def Log():
    name = raw_input("Name:  ")
    age = raw_input("Age:  ")
    gender = raw_input("Gender:  ")
    eye = raw_input("Eye Color:  ")

    f = open("datafile.txt", "a")
    f.write("\n\n"+"-"*25+"\nName: "+name+"\nAge: "+age+"\nGender:
"+gender+"\nEye Color: "+eye)
    f.close()
    print 'Log Successful!\n\n'

while x == 1:
    main()
```

These two example should give you ideas and teach you the basics of Python. Utilize them to learn and expand your knowledge, and don't hesitate to ask question!

- -

```
def whatnext():
   print"What next?"

result = apply(introduction, getting started,
lets start programming!, one more step..,
simple programs)
```

Deciding what to do

No doubt that once you finish this tutorial, you will want to make a cool program. Every new coder wants to make something really cool and advanced, but face it, knowledge is key! Don't take me wrong, I want you to test your limits, but you should also be reasonable while doing so. If this is your first introduction into the foray into programming, then of course you will not have all the knowledge necessary to make that game or that crypter you had thoughts on. As with everything, programming takes patience, practice, and time.

Here is my advice to you, find something locally (at your computer) that could be easier done with a small program. Look at the logbook. If you sell items online, you can make something like that to log your sales. You could even take it a step farther so it can retrieve data as well as store data. Your imagination is the only limitation, so let your mind be free when programming.

Find something that needs to be done, and start doing it!

- - - - - - - - - -

Finding out how to do it

A big block for many coders is that they simply do not know how to do something. This can easily be fixed by reading the appropriate documentation on what you are trying to accomplish. For example, read the Python documentation to learn about loads of different modules and functions.

Use other resources. Go to Google and simply search for what you need to do. Simple. Search for "python lists" and you will get TONS of results. It is all there!

Tutorials, videos, and ebooks. There are loads of great ebooks and video tutorials for those who want to further expand their knowledge. Personally, I suggest TheNewBoston for his video tutorials, and for ebooks, I suggest A Byte of Python. Those two are some of the leading resources for new Python programmers. I know they helped me!

- - - - - - - - - -

Programming and beyond!

Programming can not only be a job, but a great hobby. If you can enjoy programming, and have a great imagination, then there are no limits to what you can do with a computer! Use your knowledge and expand it with tutorials and your own imagination. There are no limits. Use your imagination! :)

- -

Credits

I would like to give a big thanks to Slankey™, who made the images of this thread.

- -

Happy Programming!