

Forum:[Hack Forums](#)

Thread: [Getting into Python \[The Empire\]](#)

Anxuein - March 29, 2014, 6:31 am

Opening

Python is a great language and some would recommend it to beginners. It has a nice and simple syntax which makes it easier for one to read and write it. Python is an interpreted language, which means you need the Python interpreter installed on the system before you can run any Python standalone scripts. All you need to start programming in Python is the interpreter, a notepad and a shell. Python can also be seen as a powerful language, it has a lot of modules available which are very easy to install which I'll go into later.

Python is multi-platform, you'd need to install it on a Windows system but it would be installed on a linux distribution already.

Here are some examples of programs you may recognize which use Python:BitTorrent

Cinema 4D

Dropbox

Mercurial

You can find more [here](#).

Getting started / Testing installation

Even though Python 3 is available, for this guide I will be using Python 2.7 since not all modules have been ported to 3 yet and I find it more efficient just to use 2. There is a module for converting Python 2 scripts to Python 3 which you can find [here](#).

I'm not going to guide you through installing the interpreter, but it's fairly simple and there are tonnes of tutorials available. If you are on Windows and cannot get a simple script to run (will be below this) then you may likely need to restart your system or edit your system PATH variable.

Before we go, depending if you're on Windows or Linux you may or not need a certain line for it to run. If you are on Linux, always add this line to the top of your scripts:

```
PHP Code
<?php
#!/usr/bin/python
```

To see if Python is simply going to work, make a test.py file somewhere of your preference and move into that directory with a shell using cd. Inside test.py use a simple print

```
PHP Code
<?php
print "Hello"
then run that script with python test.py
```

If that works, then we're good to go (and run any future scripts like you just did).

Basics of functions and why they will benefit you

Functions can make your whole code a lot cleaner and reusing code which you may need throughout the script without repeating it. It's basically an indented block of code which is reusable. To create a function you need to define it with the def() function.

PHP Code

```
<?php
def testFunction(arg1):
    # do whatever you like
```

So the function name would be "testFunction" and you need to parse an argument to it, which in the function would be used as arg1. You can parse multiple arguments by spacing it with a comma, e.g. (arg1, arg2). The colon on the first line where I define the function is the start to the code block, which needs to be indented until you finish the function.

A basic example would be a simple print() with an if statement. Let's say you wanted to print "Your number is higher than 10" if it is higher than 10, and "Your number is not higher than 10" if it is now lower than 10.

PHP Code

```
<?php
def HighOrLow(num):
    if num > 10:
        print "Your number is higher than 10"
    else:
        print "Your number is lower than 10"
```

HighOrLow(1)

HighOrLow(12)

So we are defining the function, then intending everything inside then calling the function by its name.

Lists and dictionaries

Let's go straight to the main huge difference: A dictionary you can use a string as a key, whereas with a list it will need to be an integer.

A basic dictionary would be:

PHP Code

```
<?php
dict = {"key1": "index1", "key2": "index2"}
```

Now to call any value (or index) you would need to call dict with a key. So if we wanted to print the value of key1, then we would use

PHP Code

```
<?php
print dict["key1"]
```

You can add an item to a dictionary simply by assigning a value like how you called it.

PHP Code

```
<?php
dict["key3"] = "index3"
```

Then you will be able to use key3 and print its value.

A dictionary would be great use if you wanted to assign certain values to certain things rather than using all separate variables.

A list would be like this:

PHP Code

```
<?php
list = ["index1", "index2", "index3"]
```

Notice how you're not naming keys? The keys there are going to be integers. Lists in python start as 0, not 1. So to get index2, you would be calling list with the integer 1

PHP Code

```
<?php  
print list[1]
```

You can add an item to a list by using .append() with a string.

PHP Code

```
<?php  
list = ["index1", "index2", "index3"]  
index3 = "index4"  
list.append(index4)
```

Now in the list you will have index1 to index4.

A list would be great use if you just wanted to compile things together to maybe, let's say, loop through them further on in a script or count how many items there are using len.

Loops

Let's start with the basic 1-10 loops.

Using range():

PHP Code

```
<?php  
for x in range(0, 11):  
    print "We're on time %d" % (x)
```

while loop:

PHP Code

```
<?php  
c = 0  
while c <= 10:  
    print c  
    c += 1
```

So here I am defining an integer named "c" then making a loop which is basically "while c is below or equal to ten > do this", then I am printing and adding 1 every time the loop repeats to c.

Let's go through looping through lists and dictionaries.

If you are looping through a list, it will loop through every value until there are no values to loop through then it will continue to further code. You will only be able to use the value (index) within the loop.

PHP Code

```
<?php  
list = ["index1", "index2", "index3"]  
for index in list:  
    print index
```

Looping through a dictionary object is different. If you loop through a dictionary like you did with the list and using "index", rather than getting the index you will get the key. However, you can get both the index and key within the same loop.

PHP Code

```
<?php  
dict = {"key1": "index1", "key2": "index2"}  
for index, key in dict.items():  
    print "%s : %s" % (index, key)
```

Notice how we had to use .items() also? That's because we need to iterate over the actual items

within that dictionary.

Slices

Slices are kind of like substring. The word slice exactly describes what it does, it slices something and returns it to you.

One thing you won't be able to do, is modify a list while iterating through it. You could, create a copy list, loop through that then modify the original, but there's an easier way using a slice.

`item[:]` basically copies the list and loops through the copy list, allowing you to modify the original list.

You can also skip items while looping through a list using slices by using `item[2:]`. This will skip the first two.

You can also do the same, in reverse, so ignoring the last two items but using the rest. For this, you'd use the slice `item[:2]`

The above slices would more likely be used iterating through lists, like so:

PHP Code

```
<?php
item = ["one", "two", "three", "four", "five"]
for x in item[slice]:
    print x
```

Obviously, you'd change the word "slice" to the actual slice..

You can use slices with a string or an array without iterating through them.

`item[-1]` would get the last item of an array or last character of a string

`item[-2:]` would get the last two items of an array and last two characters of a string.

Basically, a slice works like this:

`item[start:end]`

Since I've already gone through some slices, I'm sure you can work out the rest.

This was meant to be a "getting into Python" guide, rather than a beginners to the very basic shit such as getting to know object types, print "Hello World!". I'm sure I've gone a bit off track. I'd like to keep modifying this, but I need content to be able to add.

So if you'd like to see content added, reply and I will try my best to add it.

I was going to add an installing and using module section, but I couldn't really think of a module which a beginner(ish) would find beneficial.

#BORNE - March 29, 2014, 7:31 am

Well explained guide OP, thanks for sharing! ?

MountainLion - March 29, 2014, 9:38 am

Very well written, thank you for sharing.

Anxuein - March 29, 2014, 10:41 am

Thank you both for your comments. I made the subtitles more distinctive.

Coffee is love - June 12, 2014, 9:16 am

I remember when I learned how to code common sense things.. ;) yeah that was an hour ago.

Chalkline - June 12, 2014, 9:31 am

In my opinion I wouldn't have mentioned Cinema 4D in a "getting started" tutorial. You need to understand the fundamentals and basic of Python as well as work with them for awhile before moving into that direction with Python although you didn't really include it in the tutorial. Overall nice guide though.

Anxuein - June 12, 2014, 2:37 pm

(Jun 12, 2014 09:31 AM)Chalkline Wrote: [CITE](#) In my opinion I wouldn't have mentioned Cinema 4D in a "getting started" tutorial. You need to understand the fundamentals and basic of Python as well as work with them for awhile before moving into that direction with Python although you didn't really include it in the tutorial. Overall nice guide though.

They're examples.. It just shows the potential of Python.

Melkor333 - June 13, 2014, 8:09 am

I think it's a pretty cool basic guide for someone who already learned another language and want to learn python now...

I learned python and now I started to learn Javascript with a beginners book. Pretty boring because i do know the basics, since javascript is very similar, but i still need to read it, because i don't know where differences exist..

And I'm pretty sure everyone who already learned a (scripting-) language will understand this tutorial..

I'm not a Robot - June 15, 2014, 11:05 am

Thanks for the great tut, I'm learning python myself and this is very helpful and well laid out.