A MODULE'S NAME

Byte-compiled .pyc files

Importing a module is a relatively costly affair, so Python does some tricks to make it faster. One way is to create **byte-compiled** files with the extension `.pyc` which is an intermediate form that Python transforms the program into (remember the introduction section on how Python works?). This `.pyc` file is useful when you import the module the next time from a different program - it will be much faster since a portion of the processing required in importing a module is already done. Also, these byte-compiled files are platform-independent.

NOTE	These `.pyc` files are usually created in the same directory as the corresponding `.py` files. If Python does not have permission to write to files in that directory, then the `.pyc` files will *not* be created.

The from … import statement

If you want to directly import the argv variable into your program (to avoid typing the sys. everytime for it), then you can use the from sys import argv statement.

In general, you **should avoid** using this statement and use the import statement instead since your program will avoid name clashes and will be more readable.

Example:

```
from math import sqrt
print "Square root of 16 is", sqrt(16)
```

A module's name

Every module has a name and statements in a module can find out the name of their module. This is handy for the particular purpose of figuring out whether the module is being run standalone or being imported. As mentioned previously, when a module is imported for the first time, the code it contains gets executed. We can use this to make the module behave in different ways depending on whether it is being used by itself or being imported from another module. This can be achieved using the *name* attribute of the module.

Example (save as module_using_name.py):

```
if __name__ == '__main__':
    print 'This program is being run by itself'
else:
    print 'I am being imported from another module'
```

Output:

```
$ python module_using_name.py
This program is being run by itself

$ python
>>> import module_using_name
I am being imported from another module
>>>
```

How It Works

Every Python module has its *name* defined. If this is '*main*', that implies that the module is being run standalone by the user and we can take appropriate actions.

Source: http://www.swaroopch.com/notes/python/