```
>>> import sys                                                    ①
>>> sys.path                                                      ②
[
    '',
    '/usr/lib/python31.zip',
    '/usr/lib/python3.1',
    '/usr/lib/python3.1/plat-linux2@EXTRAMACHDEPPATH@',
    '/usr/lib/python3.1/lib-dynload',
    '/usr/lib/python3.1/dist-packages',
    '/usr/local/lib/python3.1/dist-packages'
]
>>> sys                                                           ③
<module 'sys' (built-in)>
>>> sys.path.insert(0, '/home/mark/diveintopython3/examples')     ④
>>> sys.path                                                      ⑤
[
    '/home/mark/diveintopython3/examples',
    '',
    '/usr/lib/python31.zip',
    '/usr/lib/python3.1',
    '/usr/lib/python3.1/plat-linux2@EXTRAMACHDEPPATH@',   ⎫
    '/usr/lib/python3.1/lib-dynload',                      ⎬ Divers
    '/usr/lib/python3.1/dist-packages',                   ⎭
    '/usr/local/lib/python3.1/dist-packages'
]
```

① Importing the sys module makes all of its functions and attributes available.

② sys.path is a list of directory names that constitute the current search path. (Yours will look different, depending on your operating system, what version of Python youre running, and where it was originally installed.) Python will look through these directories (in this order) for a .py file whose name matches what youre trying to import.

③ Actually, I lied; the truth is more complicated than that, because not all modules are stored as .py files. Some are built-in modules; they are actually baked right into Python itself. Built-in modules behave just like regular modules, but their Python source code is not available, because they are not written in Python! (Like Python itself, these built-in modules are written in C.)

④ You can add a new directory to Pythons search path at runtime by adding the directory name to sys.path, and then Python will look in that directory as well, whenever you try to import a module. The effect lasts as long as Python is running.

⑤ By using `sys.path.insert(0, new_path)`, you inserted a new directory as the first item of the sys.path list, and therefore at the beginning of Pythons search path. This is almost always what you want. In case of naming conflicts (for example, if Python ships with version 2 of a particular library but you want

to use version 3), this ensures that your modules will be found and used instead of the modules that came with Python.