**Модули в Python**

Система модулей позволяет вам логически организовать ваш **код на Python**. Группирование кода в модули значительно облегчает процесс написания и понимания программы. Говоря простым языком, **модуль в Python** это  просто файл, содержащий код на Python. Каждый **модуль в Python** может содержать переменные, объявления [классов](http://pythonicway.com/python-oop) и функций. Кроме того, в модуле может находиться исполняемый код. Получить список всех функций, констант и т.д. нужно выполнить команду >>>hello(«название модуля») или в программе print(hello(«название модуля»)). К примеру: print(hello(«math»)) (http://pythonicway.com/python-modules)

**Команда import в Python:**

Вы можете использовать любой питоновский файл как модуль в другом файле, выполнив в нем команду import. **Команда import в Python** обладает следующим синтаксисом:

|  |  |
| --- | --- |
| 1 | import module\_1[, module\_2[,... module\_N] |

Когда интерпретатор **Python** встречает **команду import**, он импортирует (дает доступ) этот модуль, если он присутствует в пути поиска **Python**. Путь поиска **Python** это список директорий, в которых интерпретатор производит поиск перед попыткой загрузить модуль. Например, чтобы использовать модуль math следует написать:

1. import math
2. print (math.sqrt(9))
3. print (math.pi)

Важно знать, что модуль загружается лишь однажды, независимо от того, сколько раз он был импортирован. Это препятствует цикличному выполнению содержимого модуля.

**Команда from ... import в Python**

Команда **from ... import** позволяет вам импортировать не весь модуль целиком, а только определенное его содержимое. Например:

1. from math import sqrt
2. print (sqrt(144))
3. print (pi)

Выражение **from ... import** не импортирует весь модуль, а только предоставляет доступ к конкретным объектам, которые мы указали.

**Команда from ... import \* в Python:**

В Python так же возможно импортировать всё (переменные, функции, классы) за раз из модуля, для этого используется конструкция **from ... import \***. Например:

1. from math import \*
2. print (sqrt(121))
3. print (pi)
4. print (e)

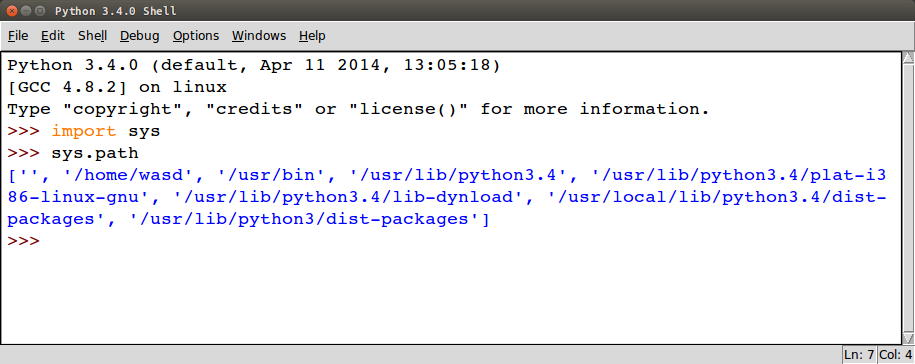
Однако это конструкцию следует использовать с осторожностью, поскольку при импортировании нескольких модулей можно запутаться в своем собственном коде.

**Местонахождение модулей в Python:**

Когда вы импортируете модуль, интерпретатор Python ищет этот модуль в следующих местах:

1. Директория, в которой находится файл, в котором вызывается команда импорта
2. Если модуль не найден, Python ищет в каждой директории, определенной в консольной переменной PYTHONPATH.
3. Если и там модуль не найден, Python проверяет путь заданный по умолчанию

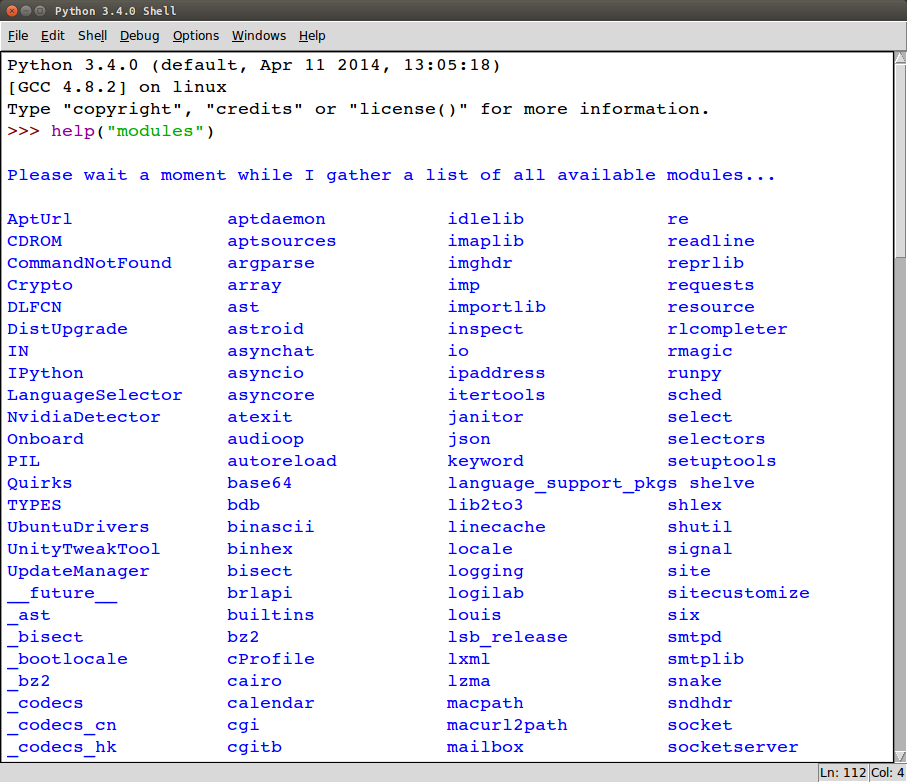
Путь поиска модулей сохранен в системном модуле sys в переменной path. Переменная **sys.path** содержит все три вышеописанных места поиска модулей.



**Получение списка всех модулей Python установленных на компьютере:**

Для того, чтобы получить список всех модулей, установленных на вашем компьютере достаточно выполнить команду:

Через несколько секунд вы получите список всех доступных модулей.



**Создание своего модуля в Python:**

Чтобы создать свой **модуль в Python** достаточно сохранить ваш скрипт с расширением .py Теперь он доступен в любом другом файле. Например, создадим два файла: module\_1.py и module\_2.py и сохраним их в одной директории. В первом запишем:

1. def hello():
2. print ("Hello from module\_1")

А во втором вызовем эту функцию:

1. from module\_1 import hello
2. hello()

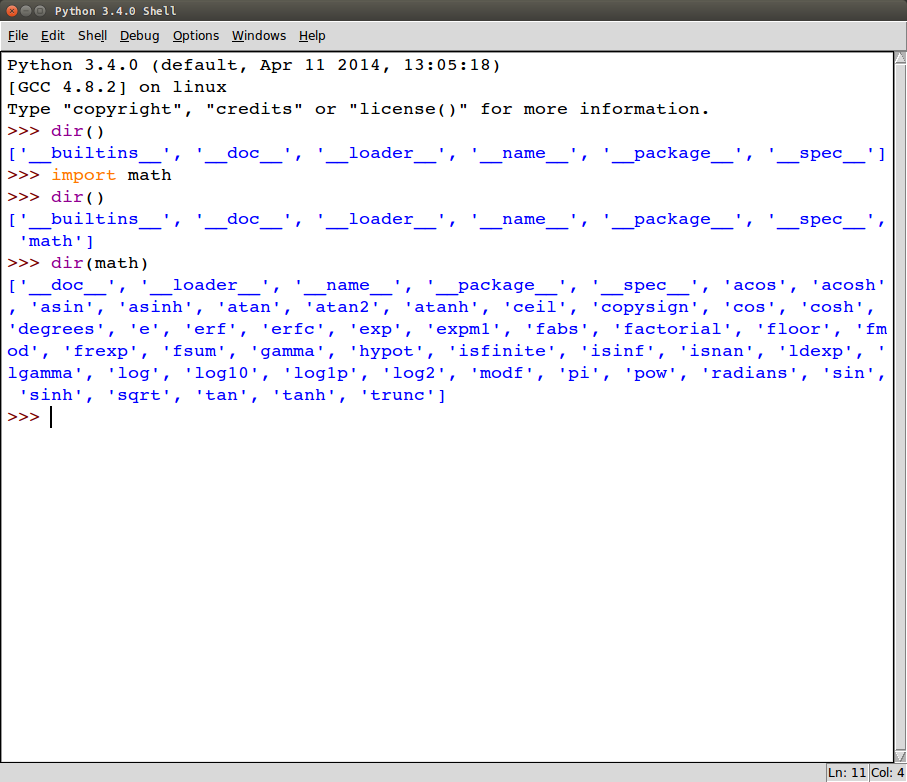
Выполнив код второго файла получим:

Hello from module\_1

**Функция dir():**

Встроенная **функция dir()** возвращает отсортированный список строк, содержащих все имена, определенные в модуле.

1. dir()
2. import math
3. dir()
4. dir(math)



**Архитектура программы на Python:**

Код на Python может быть организован следующим образом:

1. Первый уровень это обычные команды на Python.
2. Команды на Python могут быть собраны в функции.
3. [Функции](http://pythonicway.com/python-functions) могут быть частью класса.
4. Классы могут быть определены внутри модулей.
5. Наконец, модули могут составляться в пакеты модулей.

**Пакеты модулей в Python:**

Отдельные файлы-модули с кодом на Python могут объединяться в **пакеты модулей**. Пакет это директория (папка), содержащая несколько отдельных файлов-скриптов.

Например, имеем следующую структуру:

|\_ my\_file.py

|\_ my\_package

|\_ \_\_init\_\_.py

|\_ inside\_file.py

В файле inside\_file.py определена некая функция foo. Тогда чтобы получить доступ к функции foo, в файле my\_file следует выполнить следующий код:

*from my\_package.inside\_file import foo*

Так же обратите внимание на наличие внутри директории my\_package файла \_\_init\_\_.py. Это может быть пустой файл, который сообщает **Python**, что данная директория является **пакетом модулей**. В Python 3.3 и выше включать файл \_\_init\_\_.py в **пакет модулей** стало необязательно, однако, рекомендуется делать это ради поддержки обратной совместимости.