```
#!/usr/bin/python

import helloworld



print helloworld.helloworld()
```

This would produce the following result:

```
Hello, Python extensions!!
```

Passing Function Parameters

As you will most likely want to define functions that accept arguments, you can use one of the other signatures for your C functions. For example, following function, that accepts some number of parameters, would be defined like this:

```
static PyObject *module_func(PyObject *self, PyObject *args) {

   /* Parse args and do something interesting here. */

   Py_RETURN_NONE;

}
```

The method table containing an entry for the new function would look like this:

```
static PyMethodDef module_methods[] = {

   { "func", (PyCFunction)module_func, METH_NOARGS, NULL },

   { "func", module_func, METH_VARARGS, NULL },

   { NULL, NULL, 0, NULL }

};
```

You can use API *PyArg_ParseTuple* function to extract the arguments from the one PyObject pointer passed into your C function.

The first argument to PyArg_ParseTuple is the args argument. This is the object you will be *parsing*. The second argument is a format string describing the arguments as you expect them to appear. Each argument is represented by one or more characters in the format string as follows.

```
static PyObject *module_func(PyObject *self, PyObject *args) {
```