```
   if (!PyArg_ParseTuple(args, "ii", &a, &b)) {
      return NULL;
   }
   return Py_BuildValue("ii", a + b, a - b);
}
```

This is what it would look like if implemented in Python:

```
def add_subtract(a, b):
   return (a + b, a - b)
```

The *Py_BuildValue* Function

Here is the standard signature for **Py_BuildValue** function:

```
PyObject* Py_BuildValue(char* format,...)
```

Here *format* is a C string that describes the Python object to build. The following arguments of *Py_BuildValue* are C values from which the result is built. The *PyObject** result is a new reference.

Following table lists the commonly used code strings, of which zero or more are joined into string format.

Code	C type	Meaning
c	char	A C char becomes a Python string of length 1.
d	double	A C double becomes a Python float.
f	float	A C float becomes a Python float.
i	int	A C int becomes a Python int.
l	long	A C long becomes a Python int.
N	PyObject*	Passes a Python object and steals a reference.