







```
        return 0;                                // error occurred
    }

// Table of wrapped functions to be exposed by the module
static PyMethodDef methods[] = {
    { ``greet``, greet_wrap, METH_VARARGS, ``return one of 3 parts of a greeting`` }
    , { NULL, NULL, 0, NULL } // segULL,,
}
```

C++ classes and structs are exposed with a similarly-terse interface. Given:

```
struct World
{
    void set(std::string msg) { this->msg = msg; }
    std::string greet() { return msg; }
    std::string msg;
};
```

Since our `World` class is just a plain `struct`, it has an implicit no-argument (nullary) constructor. Boost.Python exposes the nullary constructor by default, which is why we were able to write:

```
>>> planet = hello.World()
```





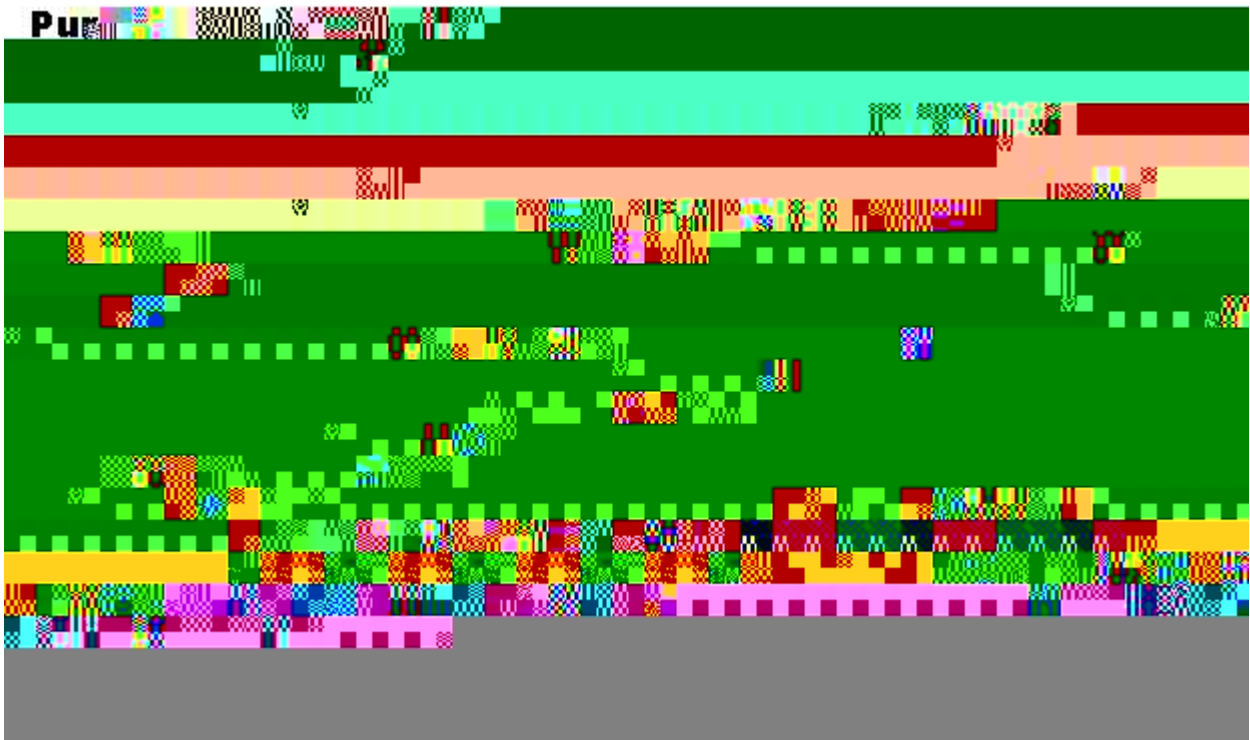






In the example above, 4 and 10 are converted to Python objects before the indexing and multiplication operations are invoked.<sup>11</sup>

The `extract`



This figure shows the estimated ratio of newly added C++ and Python code over time as new algorithms are implemented. We expect this ratio to level out near 70% Python. Being able to solve new problems mostly in Python rather than a more

The emergence of a powerful new type system in Python 2.2 made the choice of whether to maintain compatibility with Python 1.5.2 easy: the opportunity to throw away a great deal of elaborate code for emulating classic Python classes<sup>13</sup>