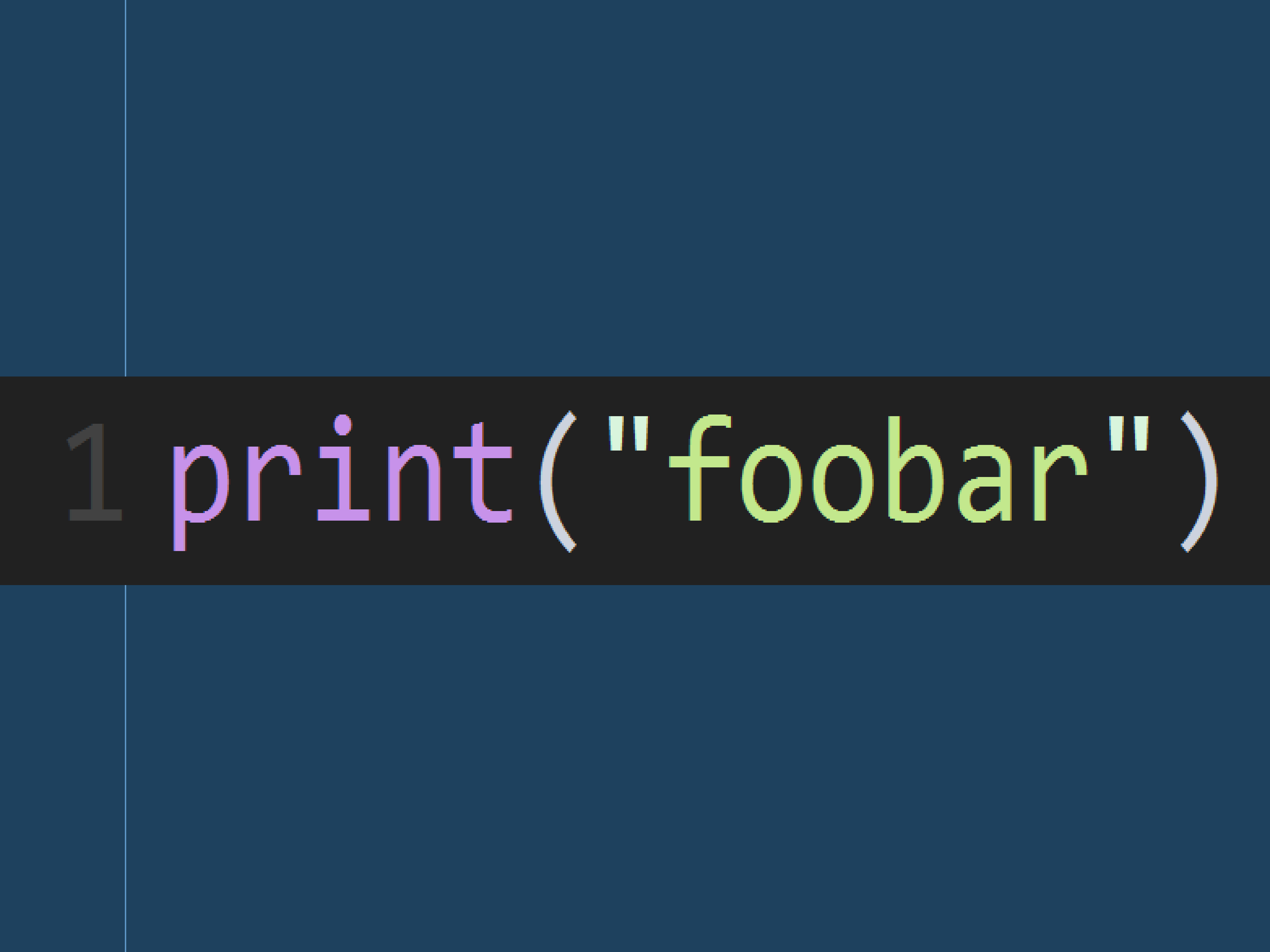
Python for .NET

About Me

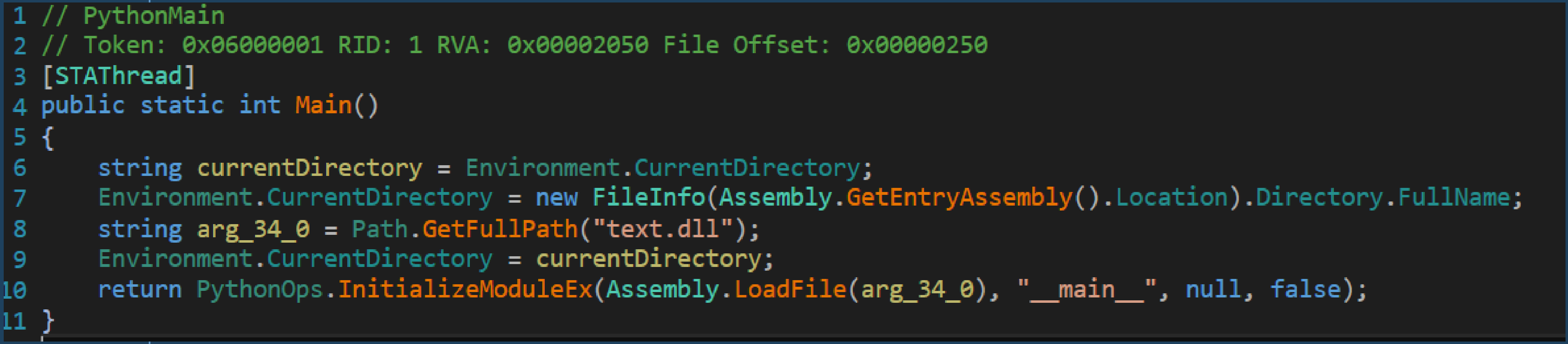
- Software Developer at United Shore
- About 2 years of professional experience
- email: hmfarran@gmail.com
- github: ijwu

IronPython

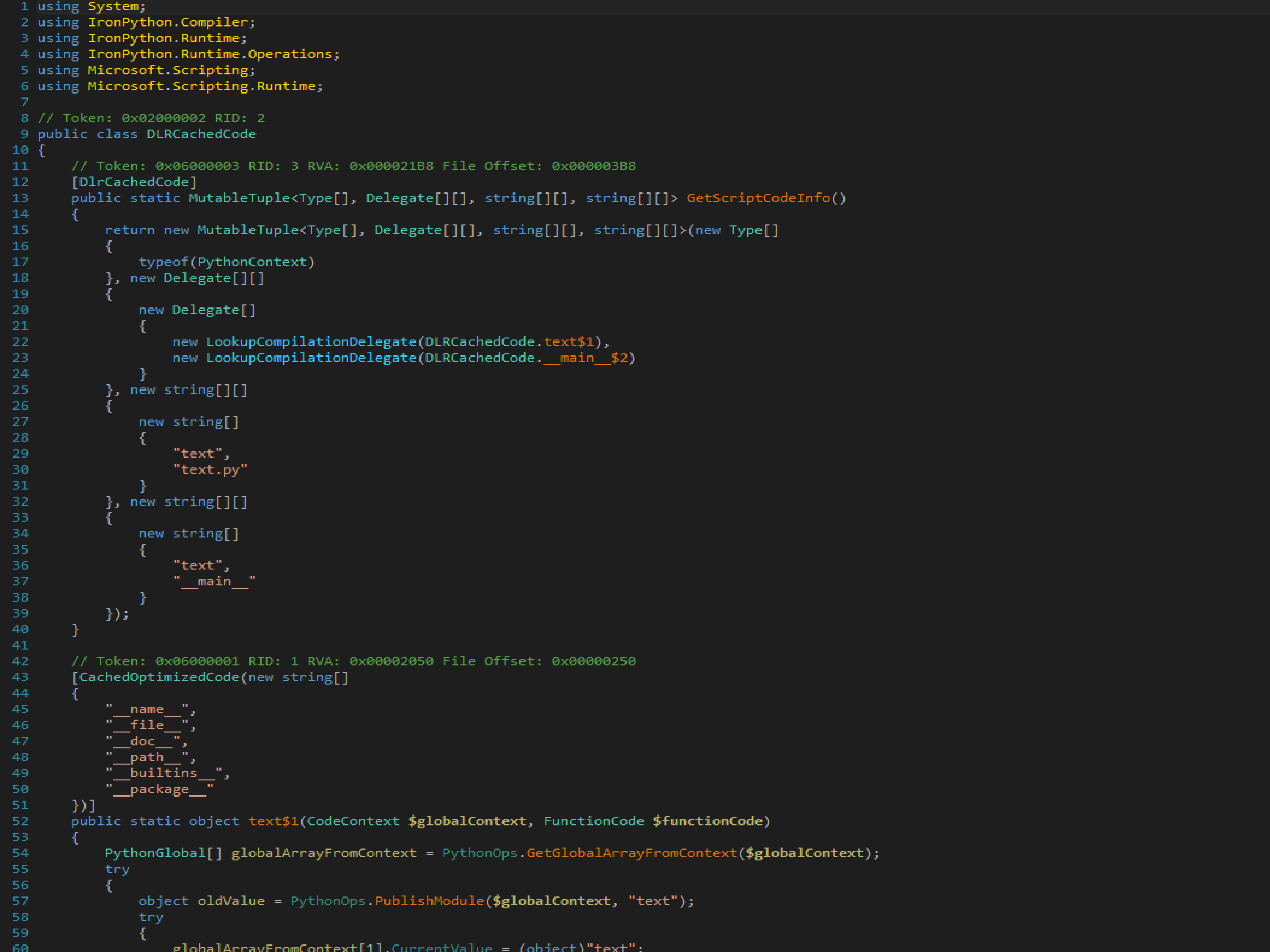
- IronPython implements the Python API on the CLR
- Includes the full power of the CLR including GIL-less threading
- (Eventually) compiles down to CIL



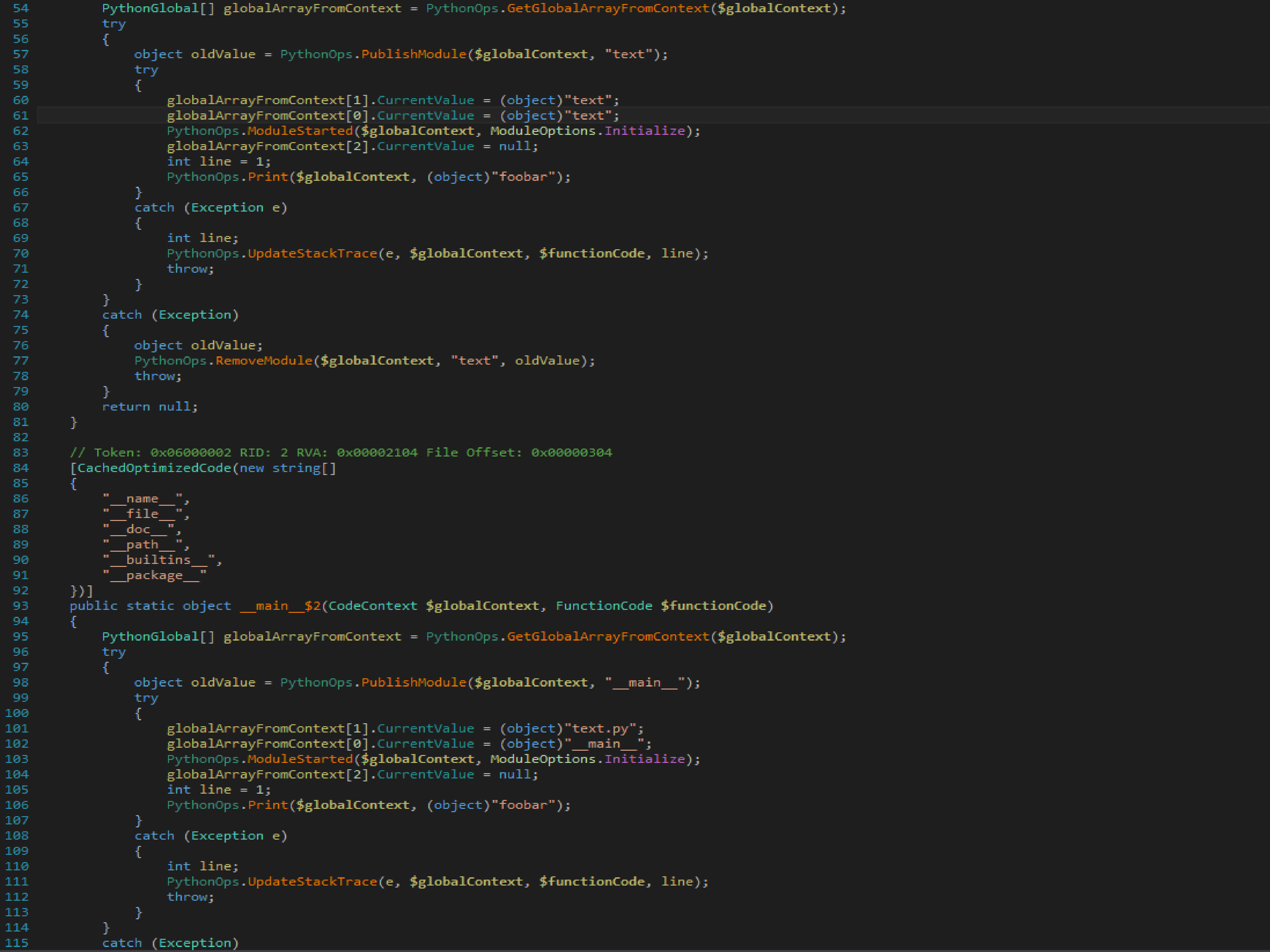
```
print("foobar")
```



```csharp
// PythonMain
// Token: 0x06000001 RID: 1 RVA: 0x00002050 File Offset: 0x00000250
[STAThread]
public static int Main()
{
    string currentDirectory = Environment.CurrentDirectory;
    Environment.CurrentDirectory = new FileInfo(Assembly.GetEntryAssembly().Location).Directory.FullName;
    string arg_34_0 = Path.GetFullPath("text.dll");
    Environment.CurrentDirectory = currentDirectory;
    return PythonOps.InitializeModuleEx(Assembly.LoadFile(arg_34_0), "__main__", null, false);
}
```



```
using System;
using IronPython.Compiler;
using IronPython.Runtime;
using IronPython.Runtime.Operations;
using Microsoft.Scripting;
using Microsoft.Scripting.Runtime;

// Token: 0x02000002 RID: 2
public class DLRCachedCode
{
    // Token: 0x06000003 RID: 3 RVA: 0x000021B8 File Offset: 0x000003B8
    [DlrCachedCode]
    public static MutableTuple<Type[], Delegate[][], string[][], string[][]> GetScriptCodeInfo()
    {
        return new MutableTuple<Type[], Delegate[][], string[][], string[][]>(new Type[]
        {
            typeof(PythonContext)
        }, new Delegate[][]
        {
            new Delegate[]
            {
                new LookupCompilationDelegate(DLRCachedCode.text$1),
                new LookupCompilationDelegate(DLRCachedCode.__main__$2)
            }
        }, new string[][]
        {
            new string[]
            {
                "text",
                "text.py"
            }
        }, new string[][]
        {
            new string[]
            {
                "text",
                "__main__"
            }
        });
    }

    // Token: 0x06000001 RID: 1 RVA: 0x00002050 File Offset: 0x00000250
    [CachedOptimizedCode(new string[]
    {
        "__name__",
        "__file__",
        "__doc__",
        "__path__",
        "__builtins__",
        "__package__"
    })]
    public static object text$1(CodeContext $globalContext, FunctionCode $functionCode)
    {
        PythonGlobal[] globalArrayFromContext = PythonOps.GetGlobalArrayFromContext($globalContext);
        try
        {
            object oldValue = PythonOps.PublishModule($globalContext, "text");
            try
            {
                globalArrayFromContext[1].CurrentValue = (object)"text";
```



```
        PythonGlobal[] globalArrayFromContext = PythonOps.GetGlobalArrayFromContext($globalContext);
        try
        {
            object oldValue = PythonOps.PublishModule($globalContext, "text");
            try
            {
                globalArrayFromContext[1].CurrentValue = (object)"text";
                globalArrayFromContext[0].CurrentValue = (object)"text";
                PythonOps.ModuleStarted($globalContext, ModuleOptions.Initialize);
                globalArrayFromContext[2].CurrentValue = null;
                int line = 1;
                PythonOps.Print($globalContext, (object)"foobar");
            }
            catch (Exception e)
            {
                int line;
                PythonOps.UpdateStackTrace(e, $globalContext, $functionCode, line);
                throw;
            }
        }
        catch (Exception)
        {
            object oldValue;
            PythonOps.RemoveModule($globalContext, "text", oldValue);
            throw;
        }
        return null;
    }

    // Token: 0x06000002 RID: 2 RVA: 0x00002104 File Offset: 0x00000304
    [CachedOptimizedCode(new string[]
    {
        "__name__",
        "__file__",
        "__doc__",
        "__path__",
        "__builtins__",
        "__package__"
    })]
    public static object __main__$2(CodeContext $globalContext, FunctionCode $functionCode)
    {
        PythonGlobal[] globalArrayFromContext = PythonOps.GetGlobalArrayFromContext($globalContext);
        try
        {
            object oldValue = PythonOps.PublishModule($globalContext, "__main__");
            try
            {
                globalArrayFromContext[1].CurrentValue = (object)"text.py";
                globalArrayFromContext[0].CurrentValue = (object)"__main__";
                PythonOps.ModuleStarted($globalContext, ModuleOptions.Initialize);
                globalArrayFromContext[2].CurrentValue = null;
                int line = 1;
                PythonOps.Print($globalContext, (object)"foobar");
            }
            catch (Exception e)
            {
                int line;
                PythonOps.UpdateStackTrace(e, $globalContext, $functionCode, line);
                throw;
            }
        }
        catch (Exception)
```

Python for .NET

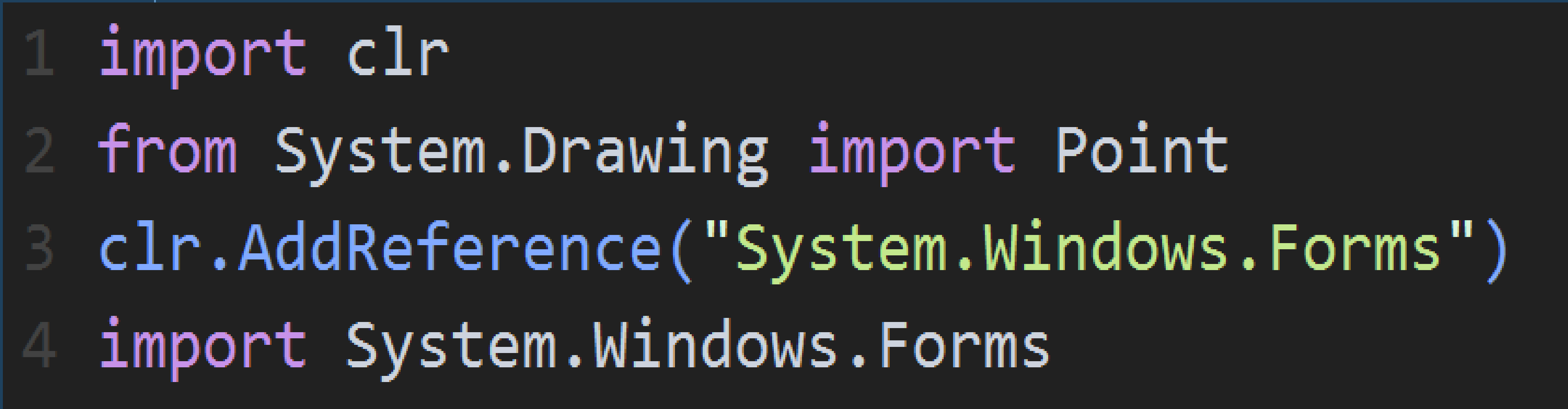
- Python for .NET accesses the CPython C API from the CLR
- Runs the CPython runtime directly.
- Allows access to CPython C extensions.
- No way around the GIL

Installing Python for .NET

- `pip install pythonnet`
- `import clr`

Imports

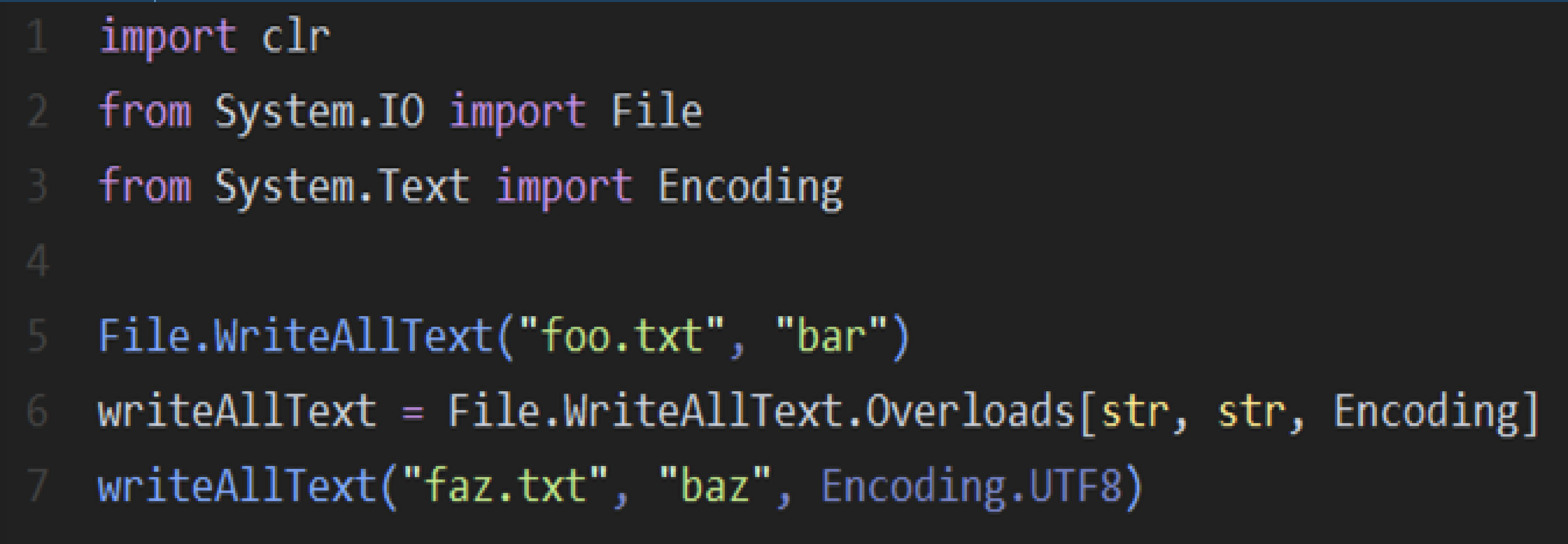
- `clr` module
- .NET namespaces may be imported like typical Python modules



```python
import clr
from System.Drawing import Point
clr.AddReference("System.Windows.Forms")
import System.Windows.Forms
```

Classes

- Same class instantiation syntax as Python
- `Overloads` collection for overload resolution



```python
import clr
from System.IO import File
from System.Text import Encoding

File.WriteAllText("foo.txt", "bar")
writeAllText = File.WriteAllText.Overloads[str, str, Encoding]
writeAllText("faz.txt", "baz", Encoding.UTF8)
```

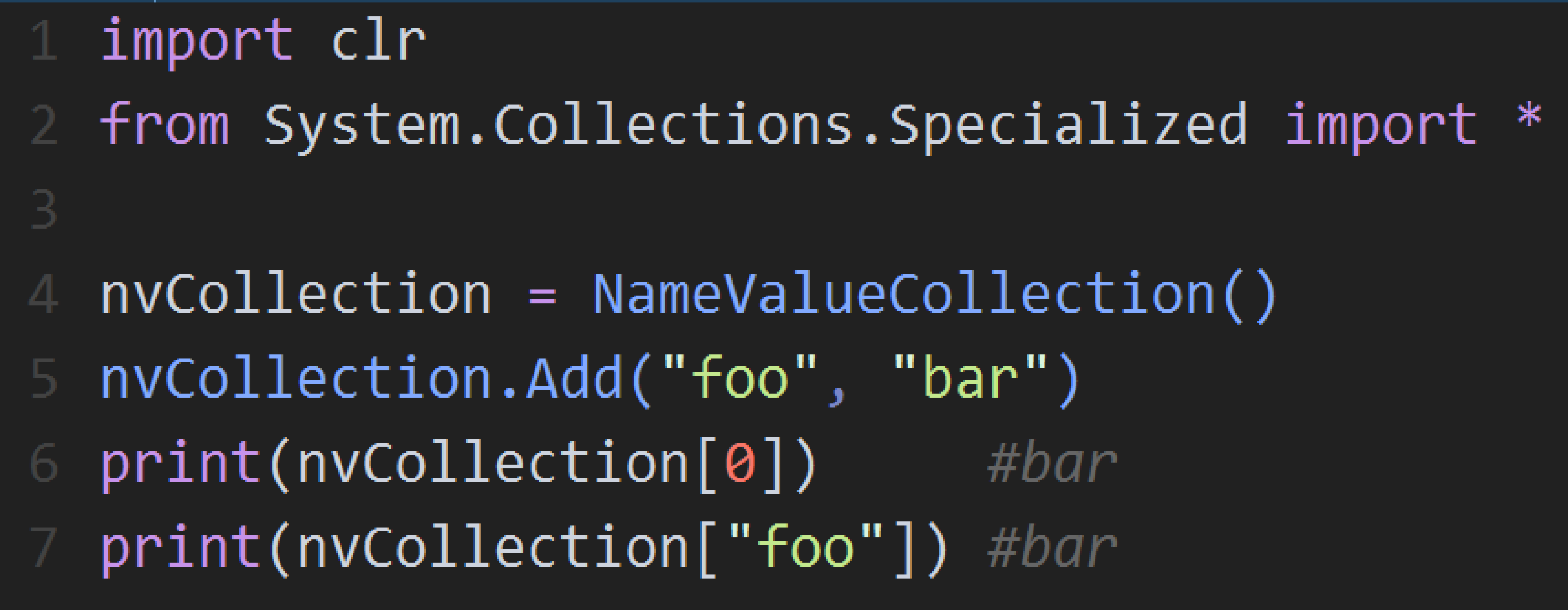
Generics

- Can bind a generic type to an identifier before use
- Same goes for methods

```
import clr
from System.Collections.Generic import *

strList = List[str]
instance = strList()
instance.Add("foobar")
```

Fields, Properties, and Indexers

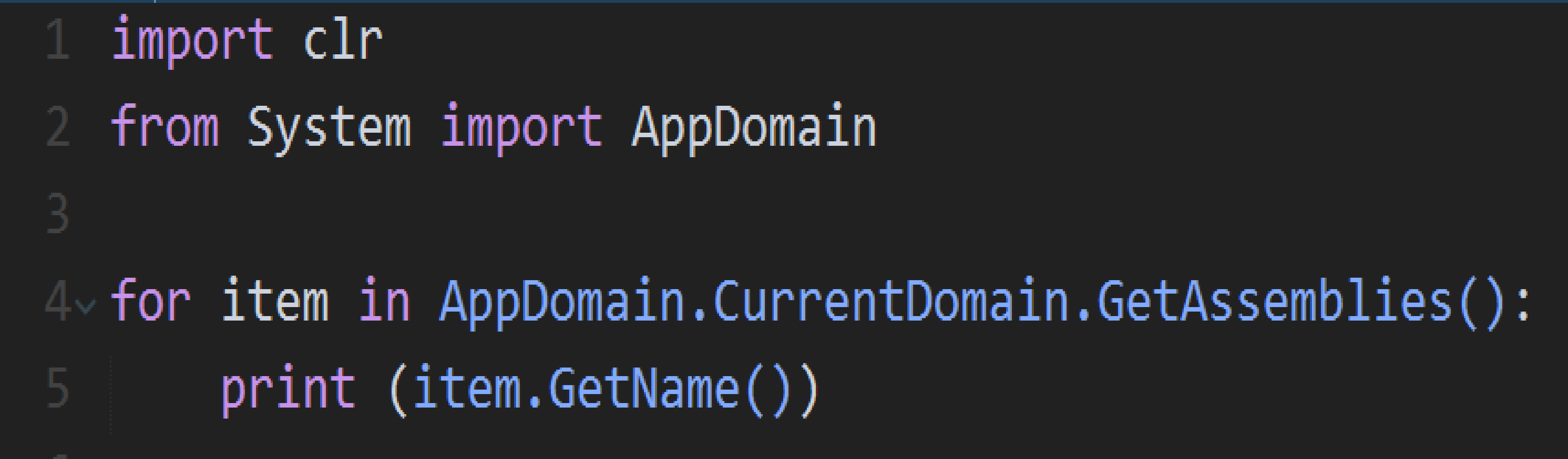
- .NET fields and properties are treated like normal Python fields
- Overloaded indexers are supported



```python
import clr
from System.Collections.Specialized import *

nvCollection = NameValueCollection()
nvCollection.Add("foo", "bar")
print(nvCollection[0])       #bar
print(nvCollection["foo"]) #bar
```

Collections

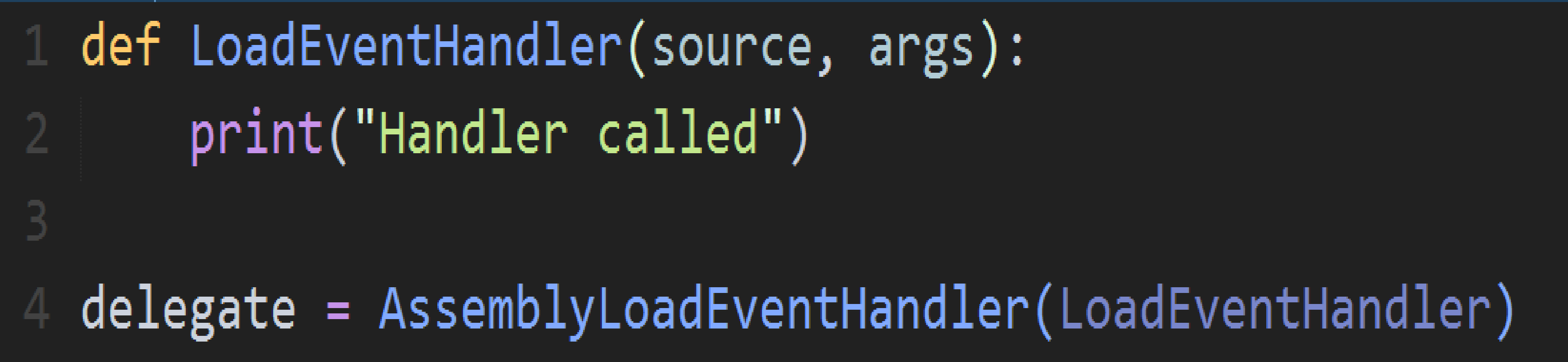
- Managed objects which implement IEnumerable may be enumerated from within Python



```python
import clr
from System import AppDomain

for item in AppDomain.CurrentDomain.GetAssemblies():
    print (item.GetName())
```

Delegates

- Delegates can be handed Python functions without issue
- As long as the Python function has the same parameter count as the delegate



```python
def LoadEventHandler(source, args):
    print("Handler called")

delegate = AssemblyLoadEventHandler(LoadEventHandler)
```

Events

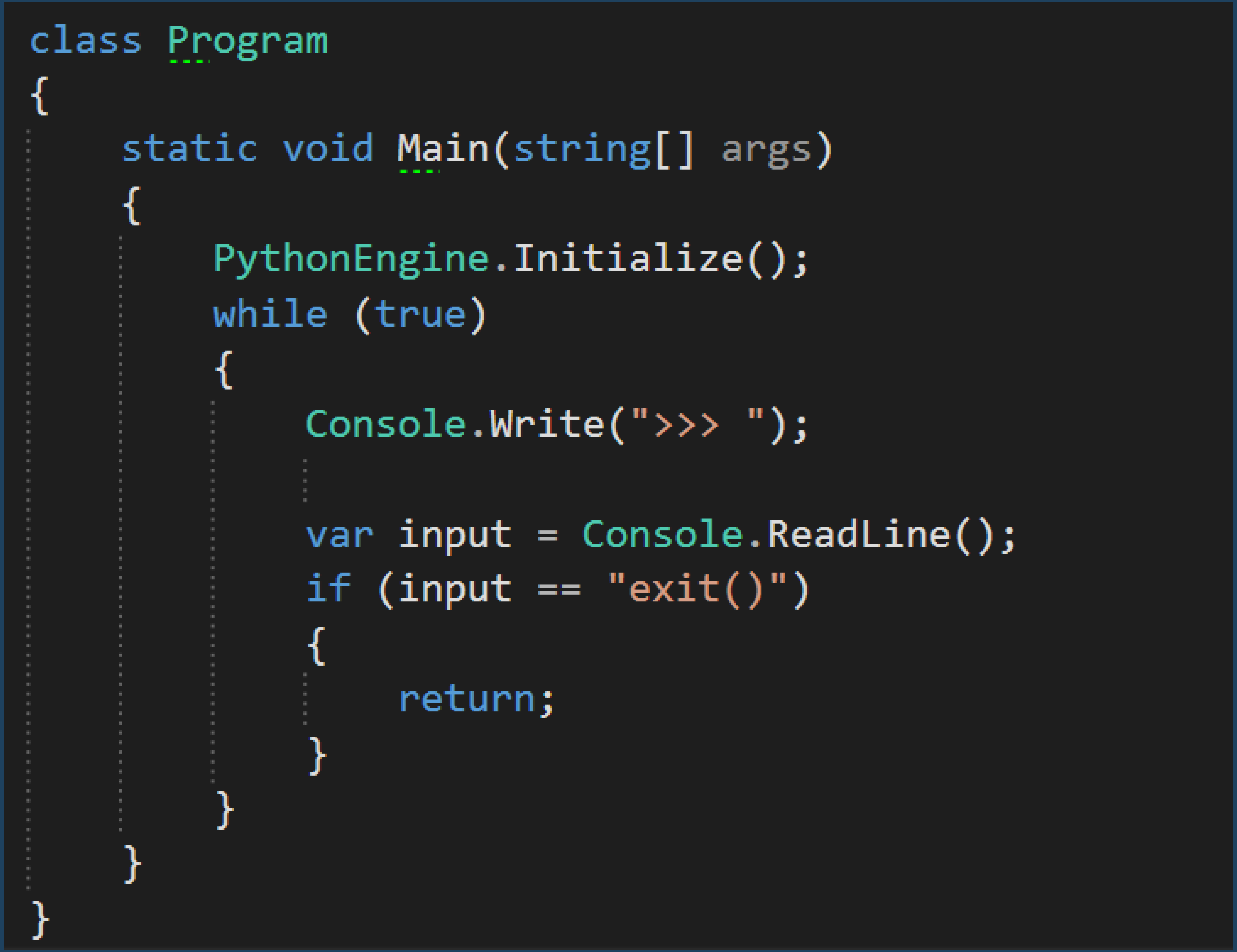
- Register event handlers like you would in C#
- Also supports implicit delegate casting

```
button = System.Windows.Forms.Button()

def ClickHandler(source, arg):
    print("I got clicked.")

button.MouseClick += ClickHandler
```

Embedding Python in your C#

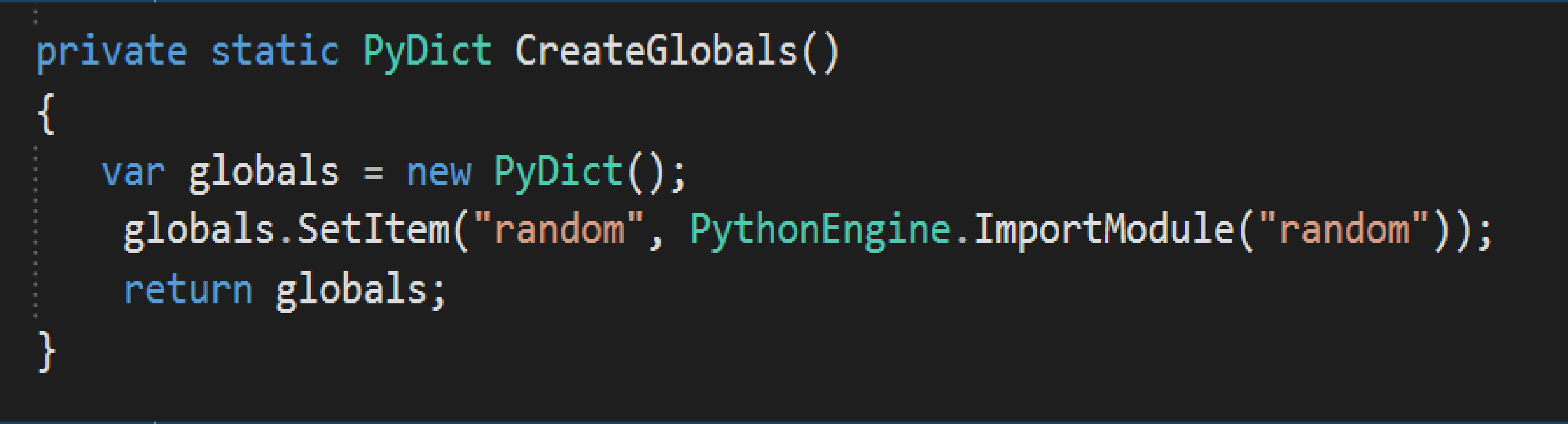
- Run code strings through the python engine
- Import modules through the python engine
- Point the engine to a file and run it directly

Building a REPL



```
class Program
{
    static void Main(string[] args)
    {
        PythonEngine.Initialize();
        while (true)
        {
            Console.Write(">>> ");

            var input = Console.ReadLine();
            if (input == "exit()")
            {
                return;
            }
        }
    }
}
```

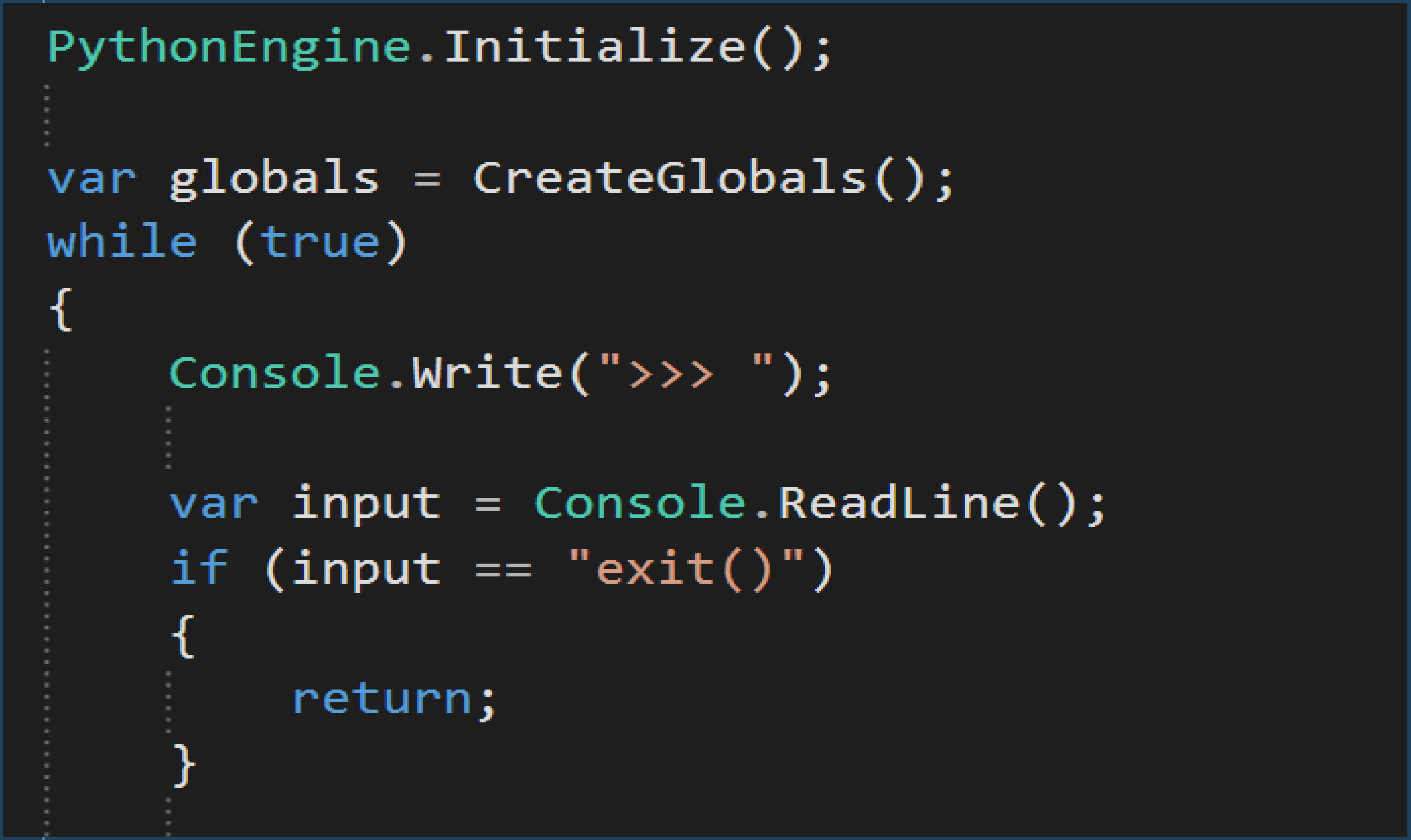
Create Globals



```
private static PyDict CreateGlobals()
{
   var globals = new PyDict();
    globals.SetItem("random", PythonEngine.ImportModule("random"));
    return globals;
}
```

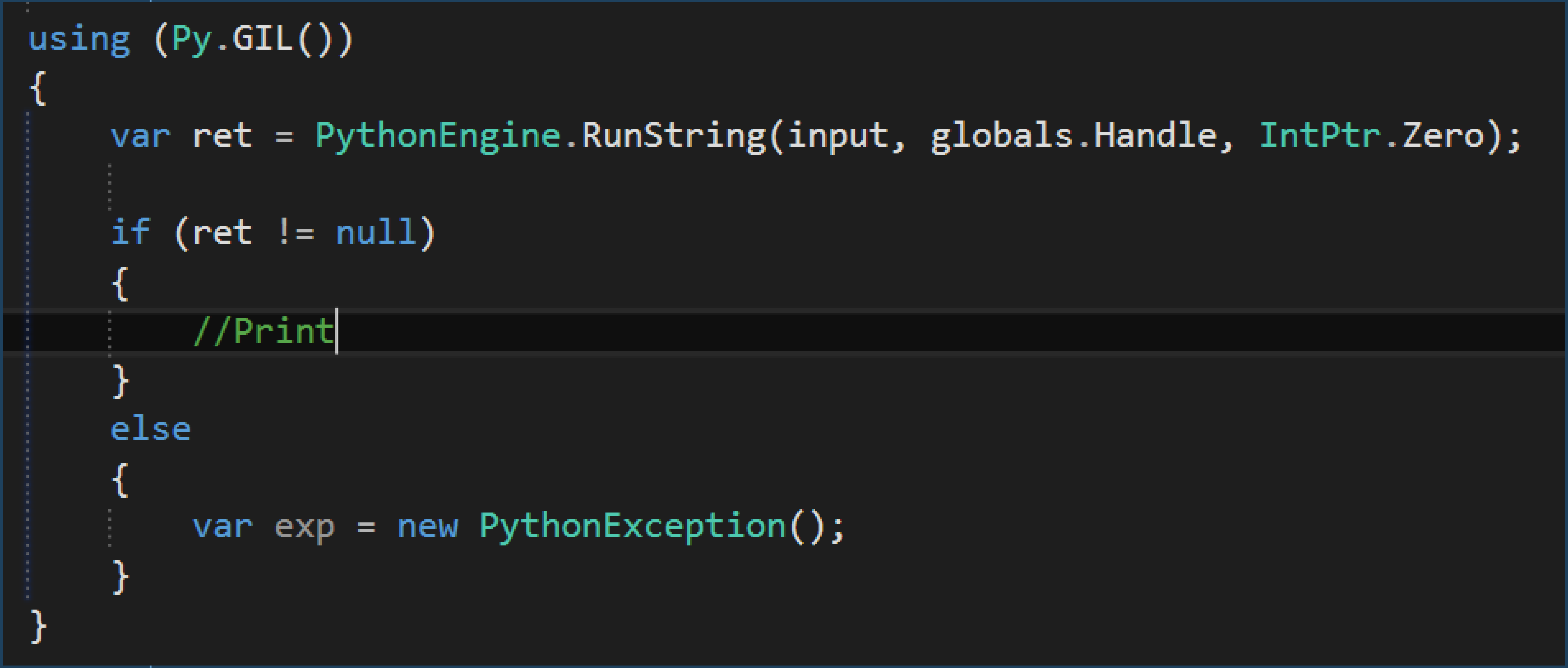
This is also how you would sandbox your scripts.

Get Our Globals

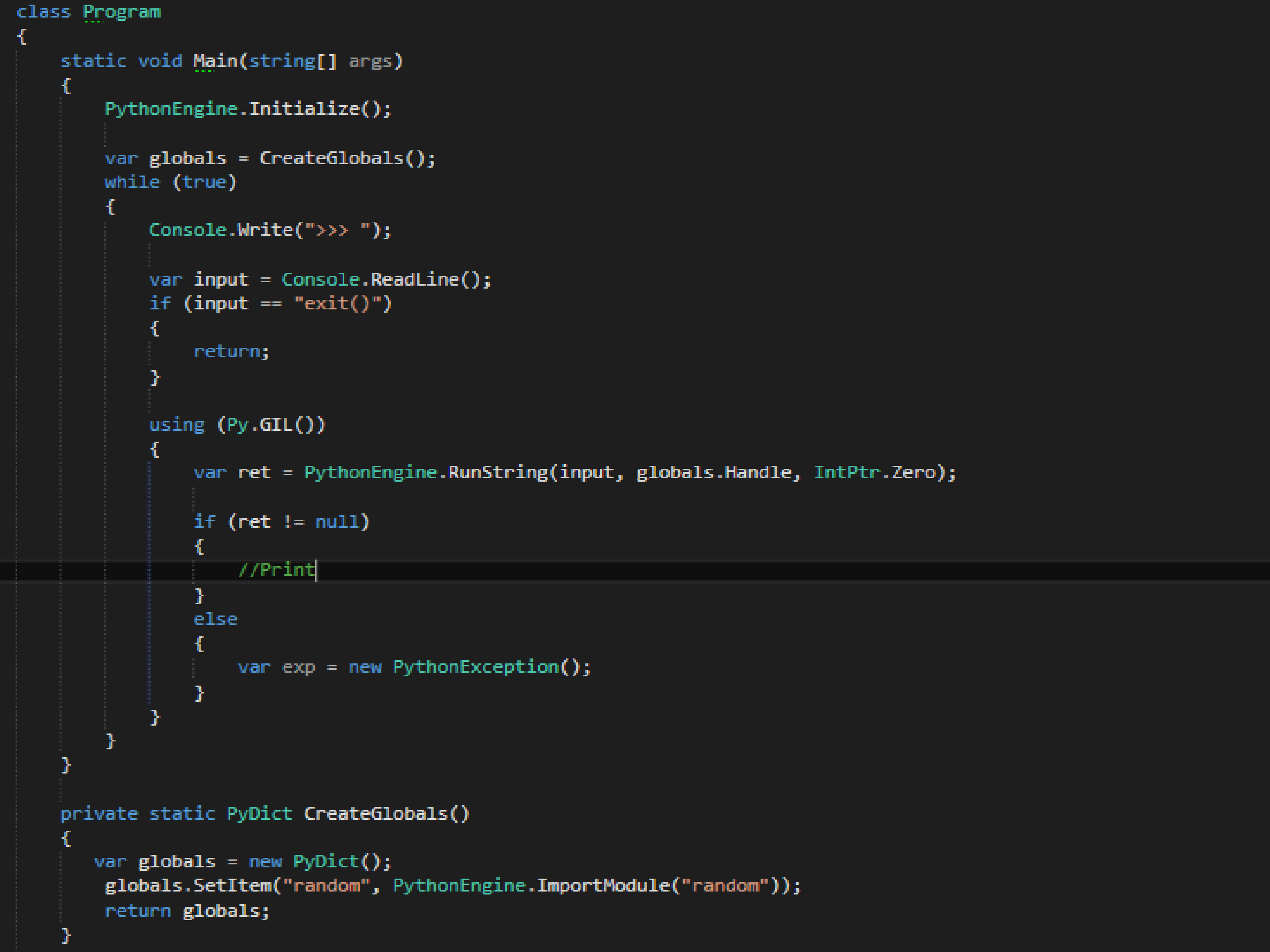


```
PythonEngine.Initialize();

var globals = CreateGlobals();
while (true)
{
    Console.Write(">>> ");

    var input = Console.ReadLine();
    if (input == "exit()")
    {
        return;
    }
```

Run Entered Code

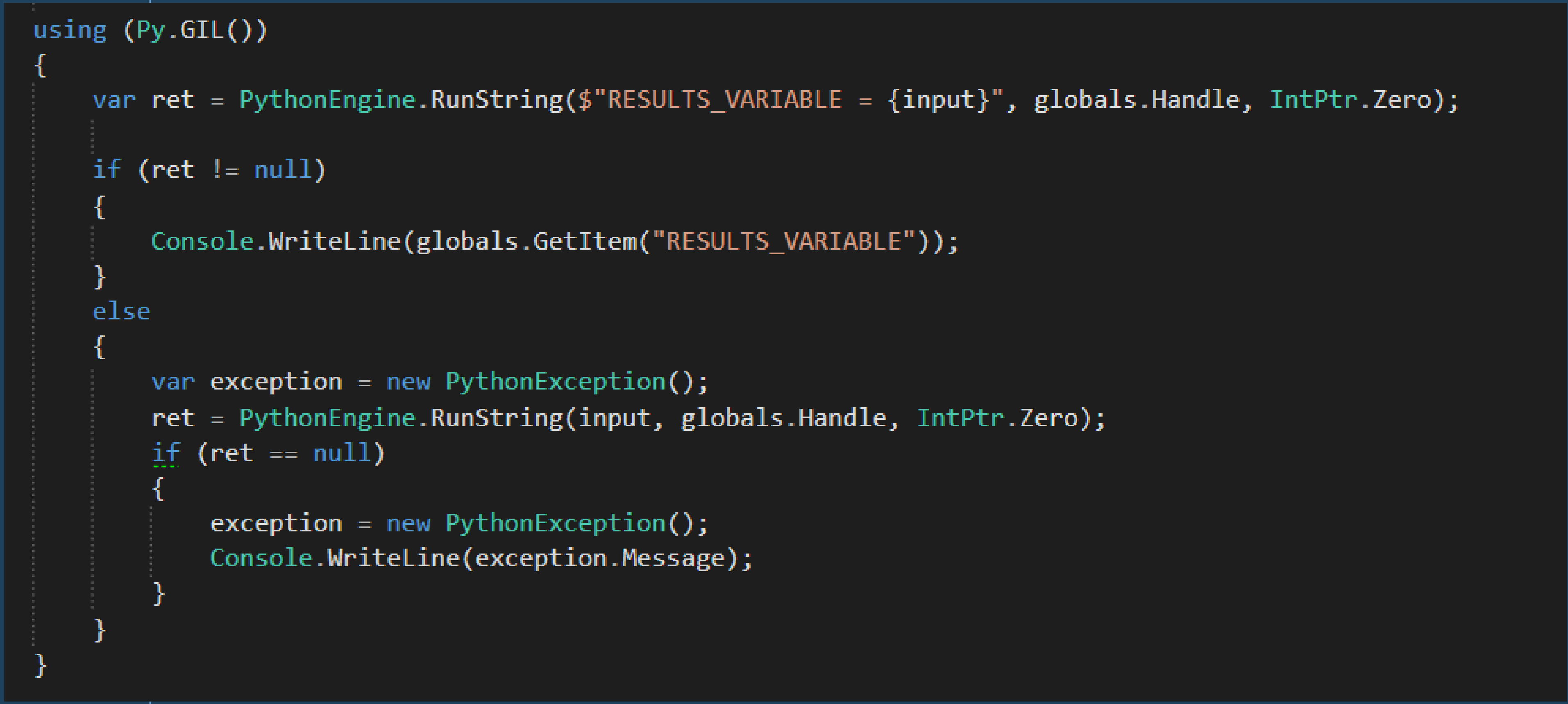


```
using (Py.GIL())
{
    var ret = PythonEngine.RunString(input, globals.Handle, IntPtr.Zero);

    if (ret != null)
    {
        //Print
    }
    else
    {
        var exp = new PythonException();
    }
}
```



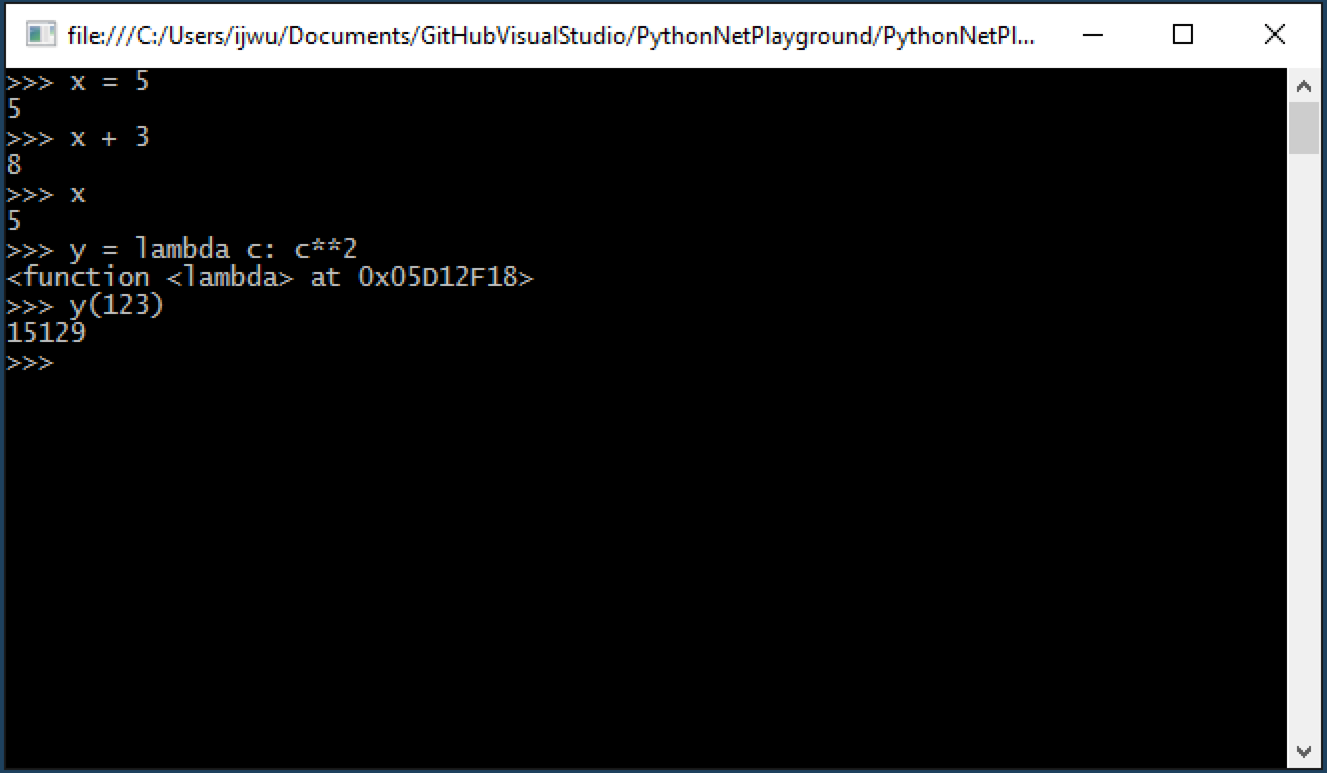
```
class Program
{
    static void Main(string[] args)
    {
        PythonEngine.Initialize();

        var globals = CreateGlobals();
        while (true)
        {
            Console.Write(">>> ");

            var input = Console.ReadLine();
            if (input == "exit()")
            {
                return;
            }

            using (Py.GIL())
            {
                var ret = PythonEngine.RunString(input, globals.Handle, IntPtr.Zero);

                if (ret != null)
                {
                    //Print
                }
                else
                {
                    var exp = new PythonException();
                }
            }
        }
    }

    private static PyDict CreateGlobals()
    {
       var globals = new PyDict();
        globals.SetItem("random", PythonEngine.ImportModule("random"));
        return globals;
    }
```

Getting Our Results



```
using (Py.GIL())
{
    var ret = PythonEngine.RunString($"RESULTS_VARIABLE = {input}", globals.Handle, IntPtr.Zero);

    if (ret != null)
    {
        Console.WriteLine(globals.GetItem("RESULTS_VARIABLE"));
    }
    else
    {
        var exception = new PythonException();
        ret = PythonEngine.RunString(input, globals.Handle, IntPtr.Zero);
        if (ret == null)
        {
            exception = new PythonException();
            Console.WriteLine(exception.Message);
        }
    }
}
```

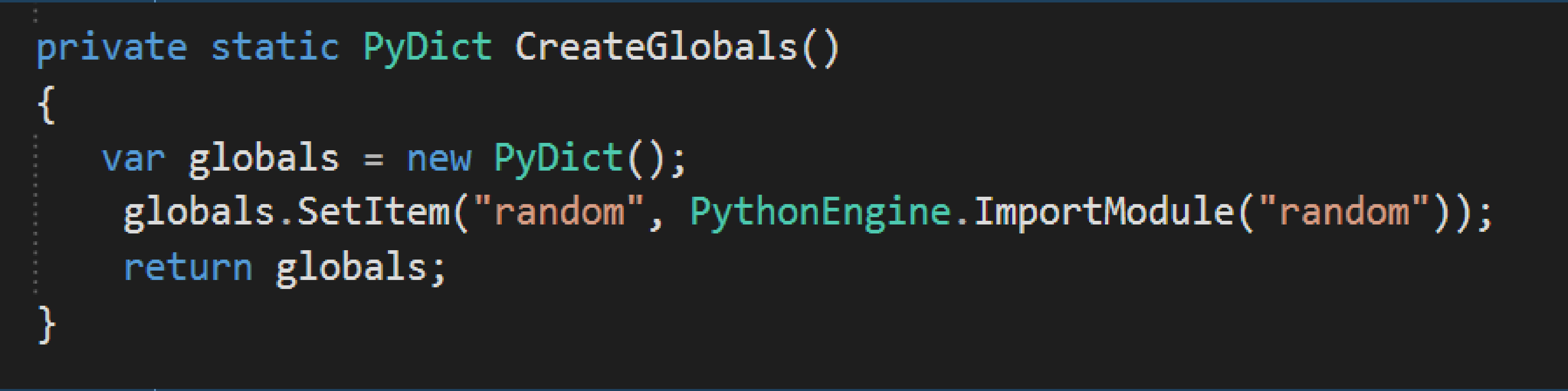
Results


file:///C:/Users/ijwu/Documents/GitHubVisualStudio/PythonNetPlayground/PythonNetPl...

```
>>> x = 5
5
>>> x + 3
8
>>> x
5
>>> y = lambda c: c**2
<function <lambda> at 0x05D12F18>
>>> y(123)
15129
>>>
```

Sandboxing

- We've already done this.
- Remember this code?

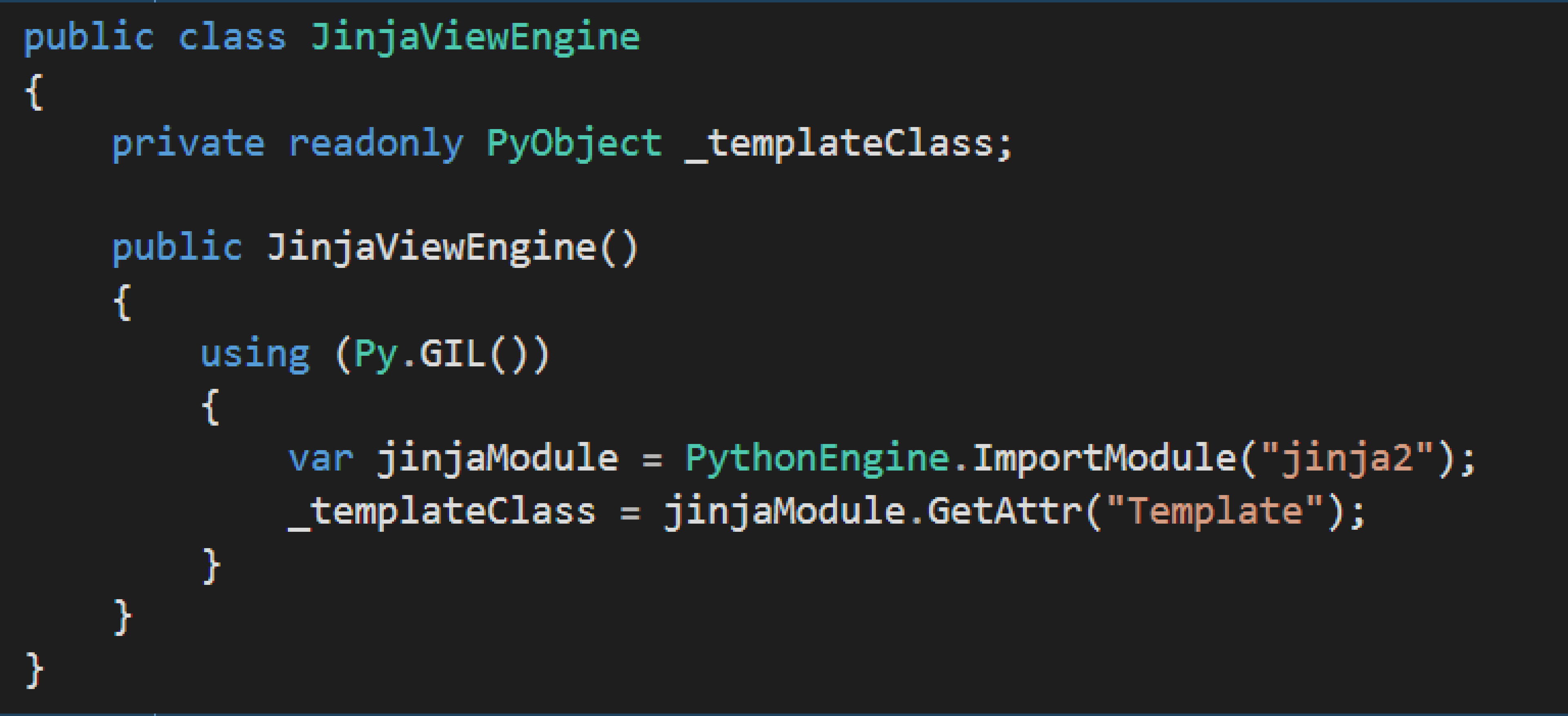


```
private static PyDict CreateGlobals()
{
    var globals = new PyDict();
     globals.SetItem("random", PythonEngine.ImportModule("random"));
     return globals;
}
```

Let's use Jinja2 in ASP.NET MVC

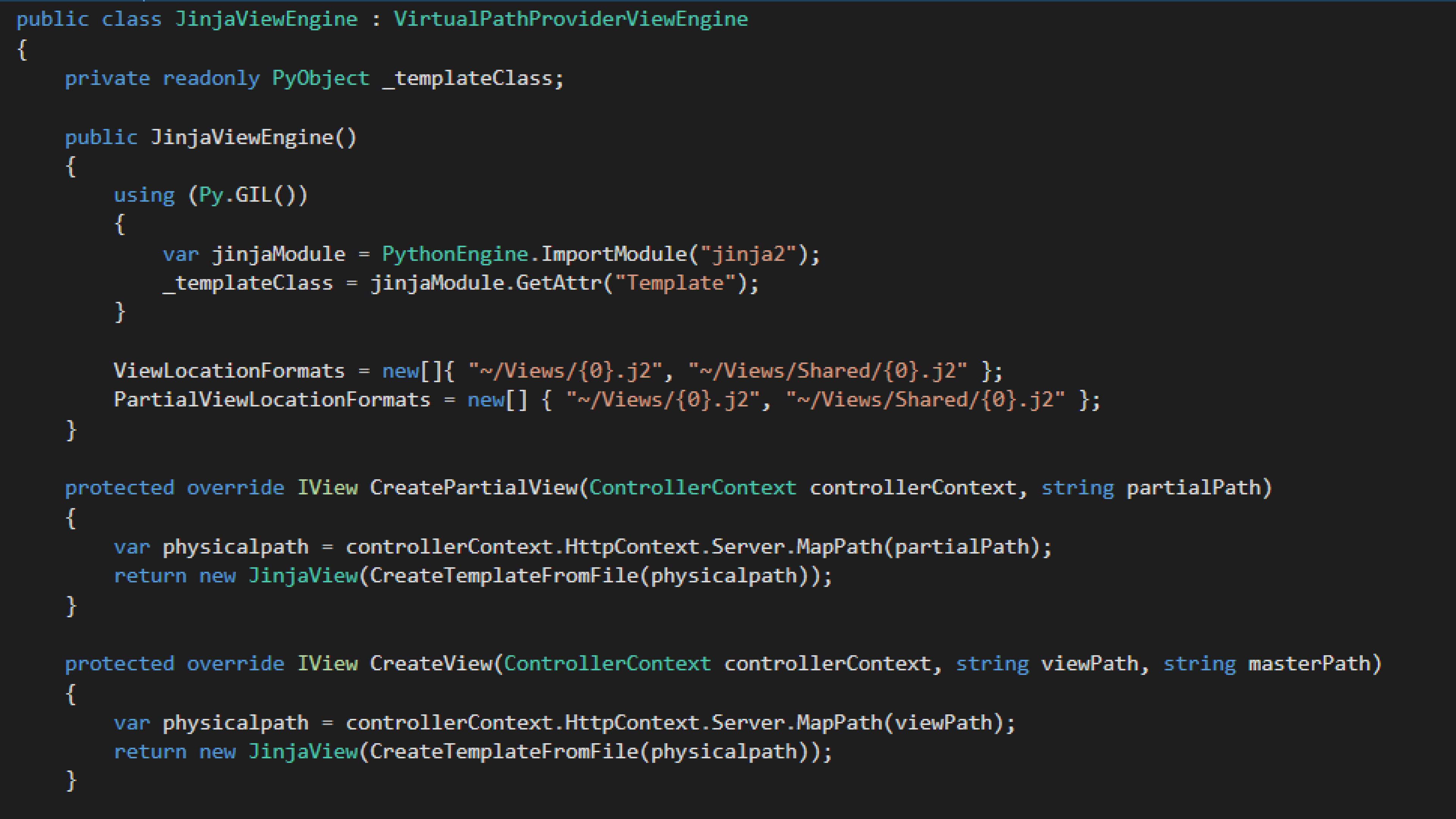
- Replacing Razor view engine with a Jinja2 view engine
- Use Python to render our Jinja2 views
- Jinja2 or an equivalent is not available in .NET
- Why not?

Let’s make a view engine



```
public class JinjaViewEngine
{
    private readonly PyObject _templateClass;

    public JinjaViewEngine()
    {
        using (Py.GIL())
        {
            var jinjaModule = PythonEngine.ImportModule("jinja2");
            _templateClass = jinjaModule.GetAttr("Template");
        }
    }
}
```

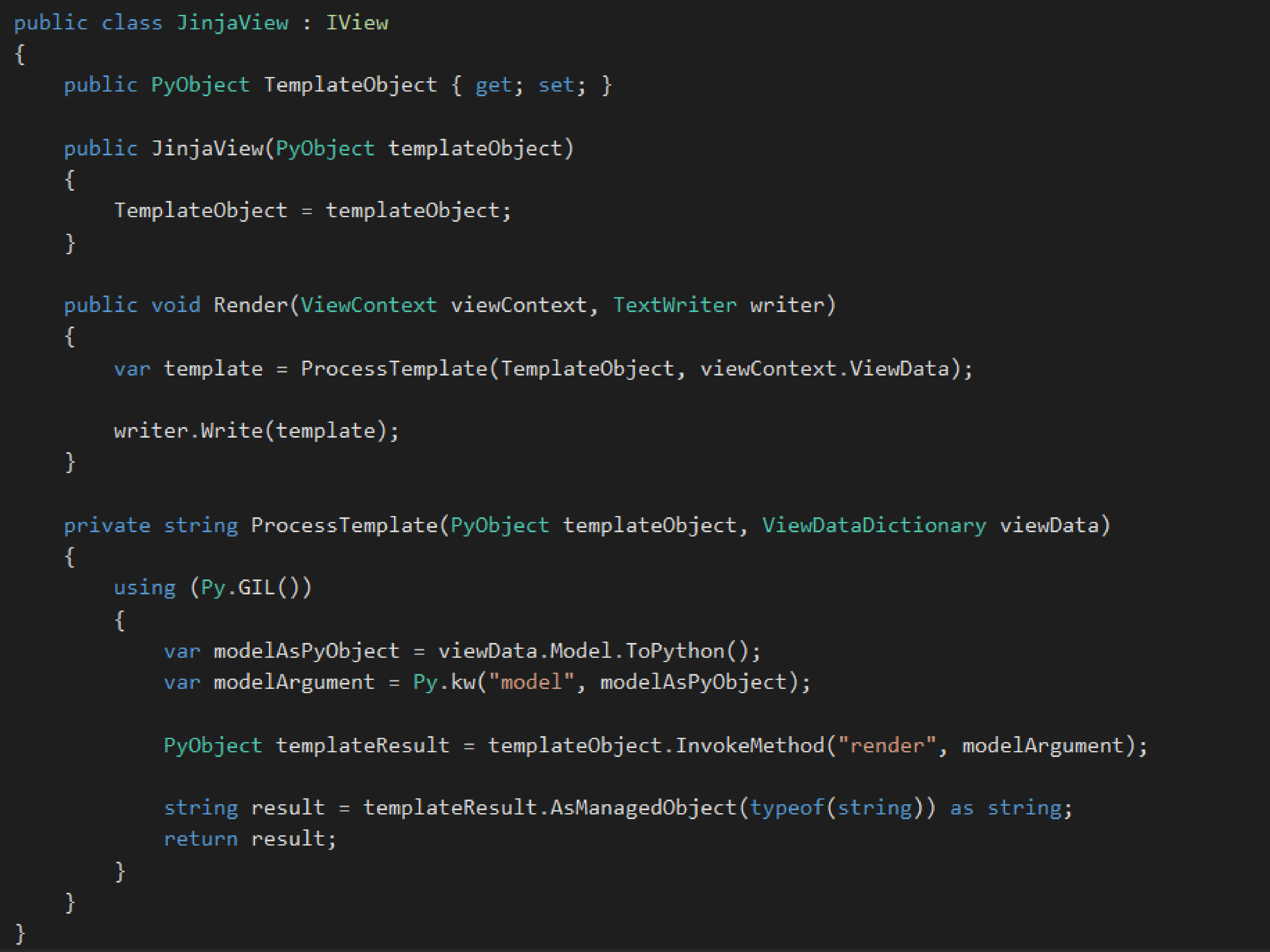
Let's make it official



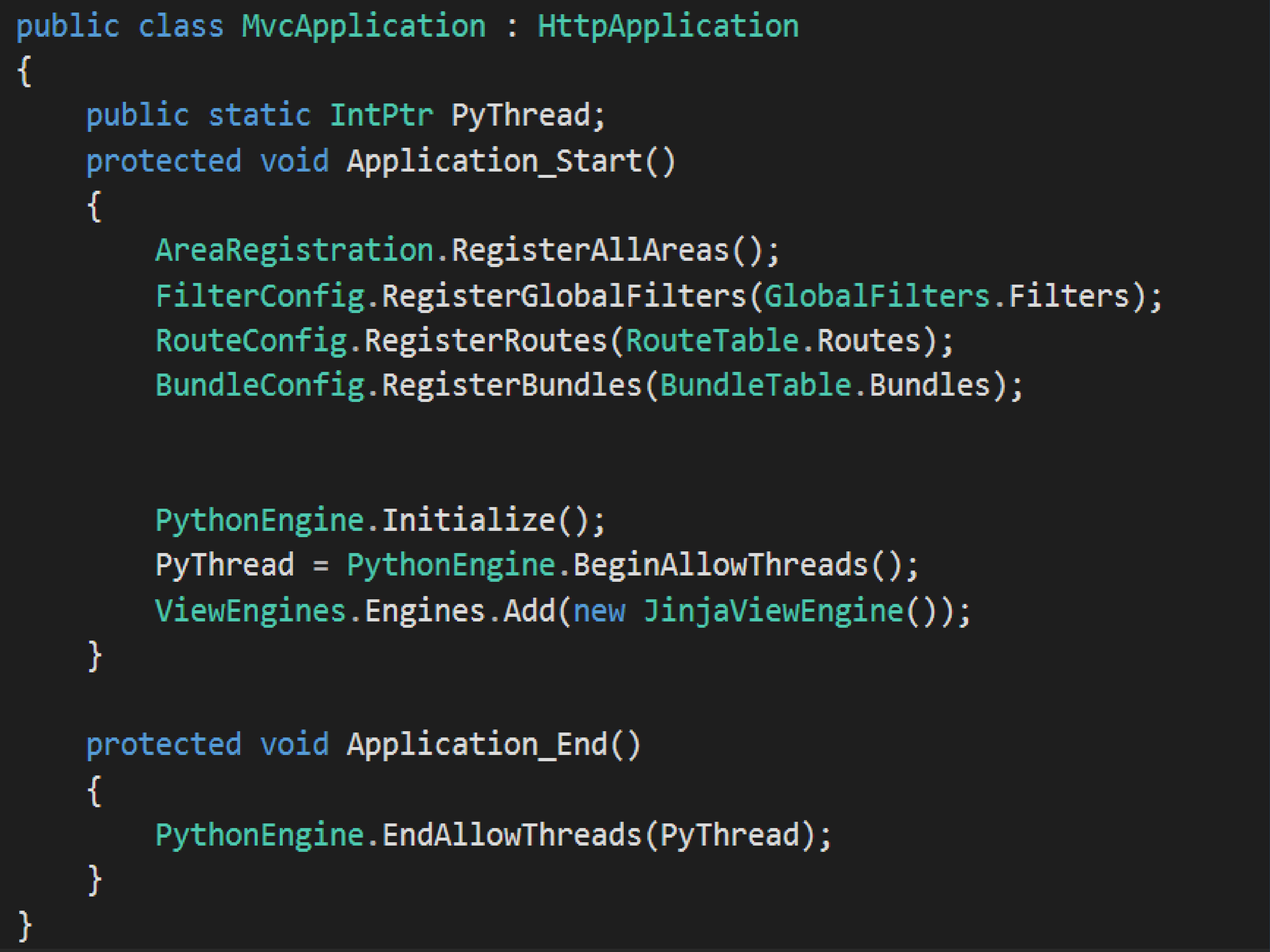
```
public class JinjaViewEngine : VirtualPathProviderViewEngine
{
    private readonly PyObject _templateClass;

    public JinjaViewEngine()
    {
        using (Py.GIL())
        {
            var jinjaModule = PythonEngine.ImportModule("jinja2");
            _templateClass = jinjaModule.GetAttr("Template");
        }

        ViewLocationFormats = new[]{ "~/Views/{0}.j2", "~/Views/Shared/{0}.j2" };
        PartialViewLocationFormats = new[] { "~/Views/{0}.j2", "~/Views/Shared/{0}.j2" };
    }

    protected override IView CreatePartialView(ControllerContext controllerContext, string partialPath)
    {
        var physicalpath = controllerContext.HttpContext.Server.MapPath(partialPath);
        return new JinjaView(CreateTemplateFromFile(physicalpath));
    }

    protected override IView CreateView(ControllerContext controllerContext, string viewPath, string masterPath)
    {
        var physicalpath = controllerContext.HttpContext.Server.MapPath(viewPath);
        return new JinjaView(CreateTemplateFromFile(physicalpath));
    }
```

Helpers

```
private PyObject CreateTemplateFromFile(string physicalpath)
{
    if (string.IsNullOrWhiteSpace(physicalpath))
    {
        throw new ArgumentException("Path string is null", nameof(physicalpath));
    }

    if (!File.Exists(physicalpath))
    {
        throw new FileNotFoundException("Template file not found");
    }

    var templateString = File.ReadAllText(physicalpath);
    using (Py.GIL())
    {
        PyObject templateInstance = _templateClass.Invoke(templateString.ToPython());
        return templateInstance;
    }
}
```

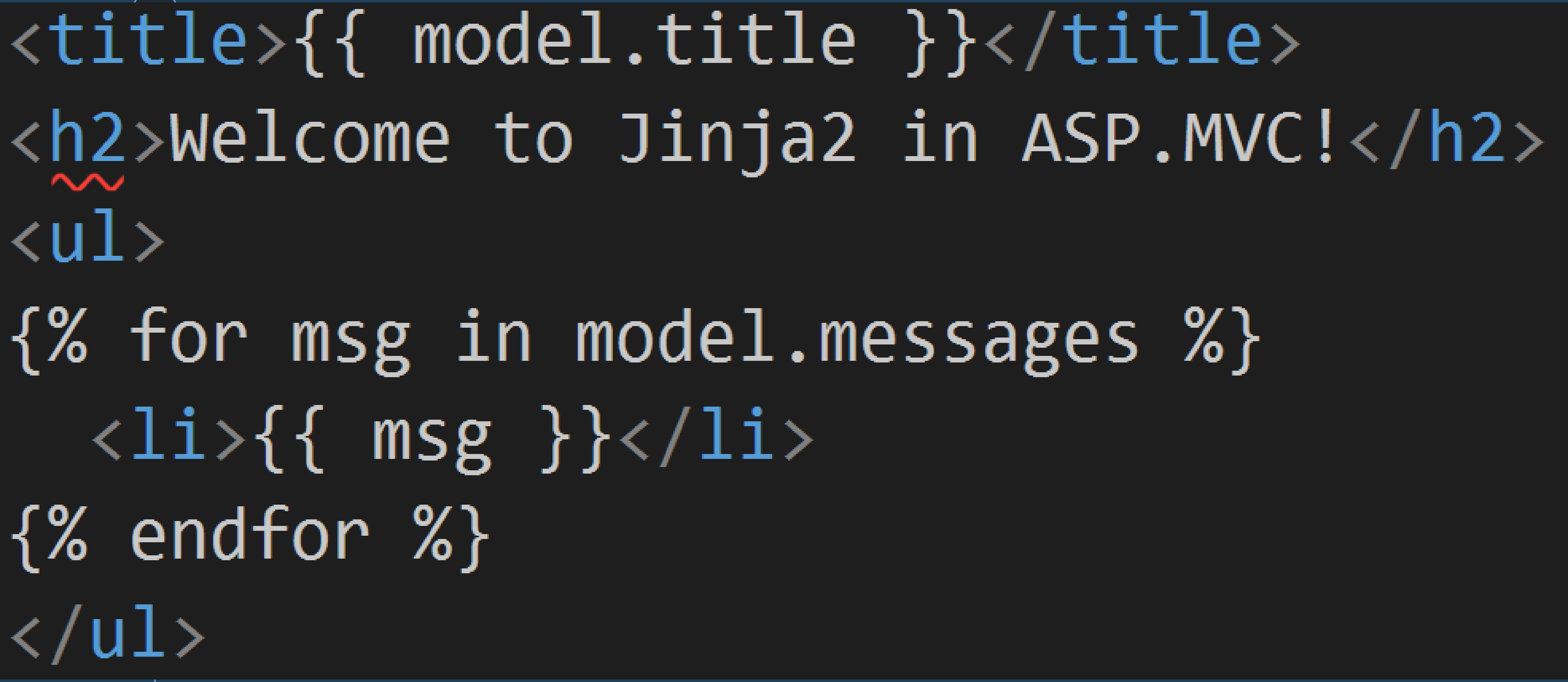


```
public class JinjaView : IView
{
    public PyObject TemplateObject { get; set; }

    public JinjaView(PyObject templateObject)
    {
        TemplateObject = templateObject;
    }

    public void Render(ViewContext viewContext, TextWriter writer)
    {
        var template = ProcessTemplate(TemplateObject, viewContext.ViewData);

        writer.Write(template);
    }

    private string ProcessTemplate(PyObject templateObject, ViewDataDictionary viewData)
    {
        using (Py.GIL())
        {
            var modelAsPyObject = viewData.Model.ToPython();
            var modelArgument = Py.kw("model", modelAsPyObject);

            PyObject templateResult = templateObject.InvokeMethod("render", modelArgument);

            string result = templateResult.AsManagedObject(typeof(string)) as string;
            return result;
        }
    }
}
```



```
public class MvcApplication : HttpApplication
{
    public static IntPtr PyThread;
    protected void Application_Start()
    {
        AreaRegistration.RegisterAllAreas();
        FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
        RouteConfig.RegisterRoutes(RouteTable.Routes);
        BundleConfig.RegisterBundles(BundleTable.Bundles);


        PythonEngine.Initialize();
        PyThread = PythonEngine.BeginAllowThreads();
        ViewEngines.Engines.Add(new JinjaViewEngine());
    }

    protected void Application_End()
    {
        PythonEngine.EndAllowThreads(PyThread);
    }
}
```

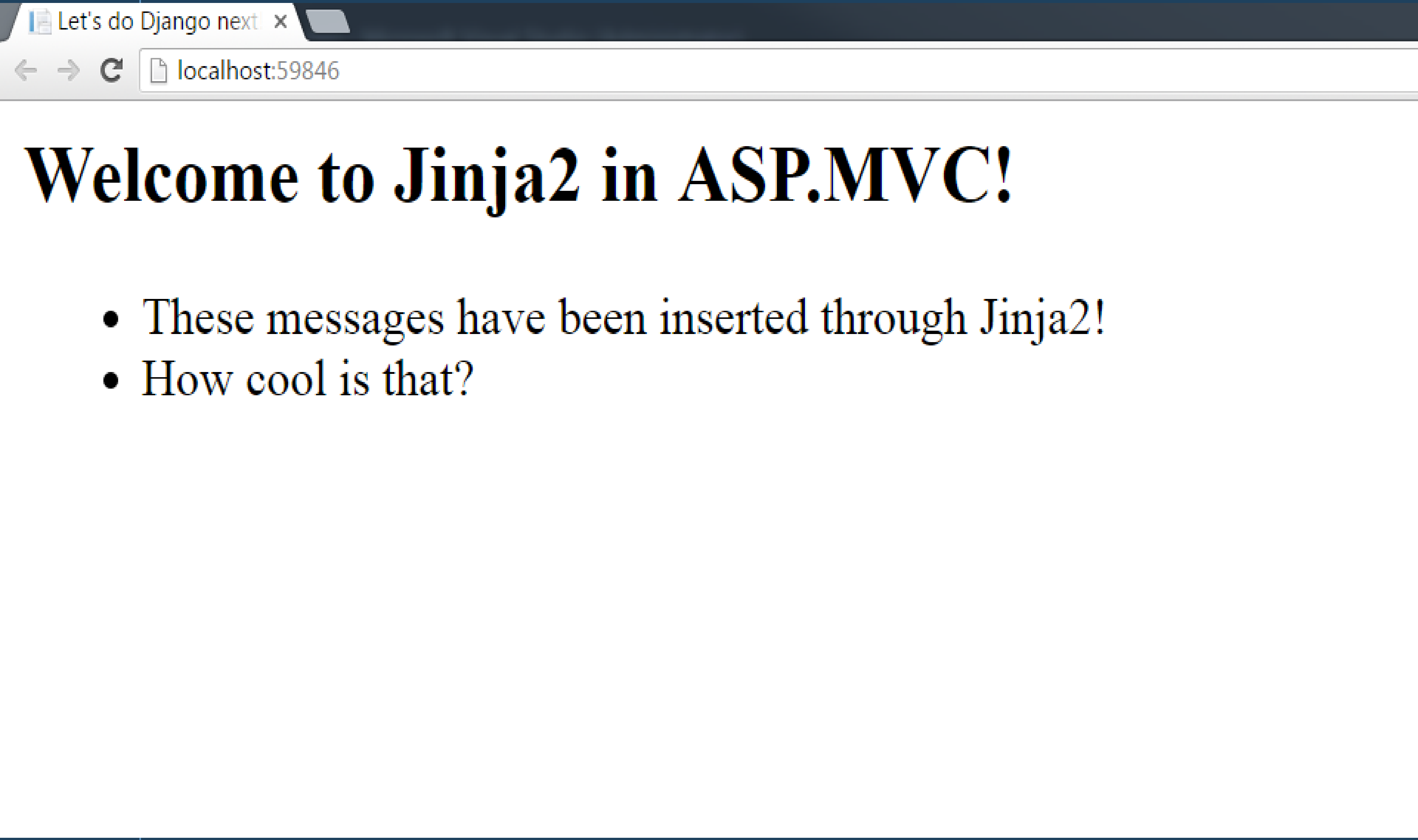
Our Template



```
<title>{{ model.title }}</title>
<h2>Welcome to Jinja2 in ASP.MVC!</h2>
<ul>
{% for msg in model.messages %}
  <li>{{ msg }}</li>
{% endfor %}
</ul>
```

Getting in Control

```csharp
public class HomeController : Controller
{
    public ActionResult Index()
    {
        return View(
            new {
                title = "Let's do Django next!",
                messages = new List<string>
                {
                    "These messages have been inserted through Jinja2!",
                    "How cool is that?"
                }
            }
        );
    }
}
```


Let's do Django next
localhost:59846

Welcome to Jinja2 in ASP.MVC!

- These messages have been inserted through Jinja2!
- How cool is that?

Contact Me

- Feedback Appreciated!
- hmfarran@gmail.com
- github.com/ijwu