FREE eBook

LEARNING ironpython

#ironpython

Table of Contents

About

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: ironpython

It is an unofficial and free ironpython ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official ironpython.

Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

Chapter 1: Getting started with ironpython

Remarks

This section provides an overview of what ironpython is, and why a developer might want to use it.

It should also mention any large subjects within ironpython, and link out to the related topics. Since the Documentation for ironpython is new, you may need to create initial versions of those related topics.

Examples

Installation or Setup

Detailed instructions on getting ironpython set up or installed.

Install IronPython

Just download the newest version from http://ironpython.net and follow the instructions from the msi package. This package will setup everything you need to start working with ironpython.

Using the IronPython command line

To use the IronPython command line, open `ipy.exe` or `ipy64.exe`. Both files are located in the path that was selected during installation. By default they will be located at `C:\Program Files\IronPython 2.7\`.

Then start writing your statements directly in the IronPython command line.

For example: print 'Hello World'

OR

For ironpython 3: print ('Hello World')

Read Getting started with ironpython online: https://riptutorial.com/ironpython/topic/951/getting-started-with-ironpython

Chapter 2: Creating Windows Forms with IronPython

Examples

Hello Word example using Windows Forms

First, references will be added to the CLR assemblies that will be used.

```
import clr
clr.AddReference('System.Windows.Forms')
```

Next the names we will use are imported.

```
from System.Windows.Forms import Application, Form
```

A class will be created for the Hello World form using `Form` as its subclass.

```
class HelloWorldForm(System.Windows.Forms.Form):
    def __init__(self):
        self.Text = 'Hello World'
        self.Name = 'Hello World'
```

The text attribute of the form sets the title bar's text.

To run the application, we create an instance of the `HelloWorldForm`.

```
form = HelloWorldForm()
Application.Run(form)
```

The `Application` class provides static methods and such as starting and stopping an application. The `Run` static method runs the form on the current thread.

Read Creating Windows Forms with IronPython online: https://riptutorial.com/ironpython/topic/2619/creating-windows-forms-with-ironpython

Chapter 3: Difference between Python and IronPython

Examples

Using .Net assemblies from Python code

With IronPython you can access any .net assembly which is compiled using the same or a lower version than the IronPython core.

Example: Importing a a .net assembly and class

```
from System import Math
```

Example: Using an imported class:

```
from System import Math
print Math.Abs(-123)
```

You can also load additional assemblies by using the builtin `clr` module.

```
import clr
clr.AddReference('Sample') # Sample.dll inside of the working directory.
```

Than just use it as any other .net or python library.

IronPython is written in pure c#

IronPython is completly written using managed .net (c#) code. So all `builtin` python methods and libraries (such as `next()`, `int()`, etc.) are writtin in .net.

This example shows the implementation of `len()` for a list of different types (only a few):

```
....

public static int len([NotNull]List/*!*/ list) {
    return list.__len__();
}

public static int len([NotNull]PythonTuple/*!*/ tuple) {
    return tuple.__len__();
}

public static int len([NotNull]PythonDictionary/*!*/ dict) {
    return dict.__len__();
}
```

....

If we would need some other type to count the length off, just add them in `Builtin.cs` and it will be available automatically.

Using generics within IronPython

IronPython enables to use generic classes and methods from the .net framework. Generics can be used with the same syntax as accessing an index. For passing more than one type-parameter, they must be separated with a comma:

```
l = Dictionary[int, str]()
```

That way we create a dictionary where keys only accepts `integers` and the values must be a `string` .

A sample usage could look like this

```
from System.Collections.Generic import List
lst = List[str]()
lst.Add('Hello')
lst.Add('World')
for l in lst:
    print
```

Output

> Hello
>
> World

When adding new items, type checking will also be performed:

```
lst = List[str]()
lst.Add(123)
```

> Traceback (most recent call last):
>
> File "<stdin>", line 1, in
>
> TypeError: expected str, got int

Read Difference between Python and IronPython online: https://riptutorial.com/ironpython/topic/1059/difference-between-python-and-ironpython

Chapter 4: Overview

Examples

What is IronPython?

IronPython is an open-source implementation of the Python programming language which is tightly integrated with the .NET Framework. IronPython can use the .NET Framework and Python libraries, and other .NET languages can use Python code just as easily.

Read Overview online: https://riptutorial.com/ironpython/topic/4239/overview

Credits

S. No	Chapters	Contributors
1	Getting started with ironpython	BendEg, Chandu, Community, D. Alveno
2	Creating Windows Forms with IronPython	D. Alveno
3	Difference between Python and IronPython	BendEg
4	Overview	D. Alveno