Boo es un lenguaje de programación orientado a objetos, de tipos estáticos para la Common Language Infrastructure con una sintaxis inspirada en Python y un énfasis en la extensibilidad del lenguaje y su compilador. Sus características incluyen lainferencia de tipos, los generadores, multimétodos, duck typing opcional, macros, clausuras, currificación y funciones de primera clase.

Boo es software de código abierto; tiene una licencia tipo MIT/BSD.

Boo se integra sin fisuras con Microsoft.NET y Mono.

Hola mundo

```
print "Hola Mundo"
```

Función generadora de la Serie de Fibonacci

```
def fib():
    a as long, b as long = 0, 1
    while true:
        yield b
        a, b = b, a + b
for index as int, element in zip(range(5), fib()):
    print("${index+1}: ${element}")
```

Ejemplo simple de Windows Forms con clases, cierres y eventos

```
import System.Windows.Forms
import System.Drawing

class MyForm(Form):
    def constructor():
        b = Button(Text: "Púlsame")
        b.Location = Point(100, 50)
        b.Click += def():
            MessageBox.Show("!has pulsado el botón!")
        self.Controls.Add(b)
f = MyForm()
Application.Run(f)
```

Ejemplo simple de Gtk#

```
import System
import Gtk from "gtk-sharp"
public class MyWindow:
        def constructor():
                w = Gtk.Window("Hola Mundo")
                w.DeleteEvent += ExitWindow
                w.ShowAll()
        def ExitWindow(o, args as DeleteEventArgs):
                Gtk.Application.Quit()
```

```
Gtk.Application.Init()
w = MyWindow()
Gtk.Application.Run()
```

Patrón de diseño asíncrono con un cierre

```
import System

def run():
    print("en ejecución")

print "arrancado"
result = run.BeginInvoke({ print("reclamado") })
System.Threading.Thread.Sleep(50ms)
run.EndInvoke(result)

print "fin"
```

Currificación

```
plusX = { a as int | return { b as int | return a + b }}
print plusX(3)(4)
```

- plusX es una función que toma un entero a, que devuelve otra función que toma un entero b y devuelve a+b.»