

PepE: Python Embedded Programs in Eiffel

Daniel Rodríguez

2004-12-29

Contents

1	Introduction	2
1.1	Is static better than dynamic?	2
1.2	Eiffel + Python = True	2
2	Extending and embedding	3
3	Embedding	3
4	The interpreter	3
5	Python types	3
6	Exceptions	3
7	asi	3
8	Downloading	3
9	Requirements	3
10	Installing	3

Abstract

This document describes an Eiffel programming library named PepE wich permits to embed Python programs in Eiffel code. Embedding Python in Eiffel includes: executing Python programs from Eiffel code, setting and inspecting Python variables from Eiffel and converting Python objects to Eiffel objects.

1 Introduction

1.1 Is static better than dynamic?

There is a hard and old debatt out there among programmers. It is about the kind of programming language that is better suited for professional use. The struggle is about statically- or dynamically-typed programming languages.

Statically-typed languages offer better compile-time control and are preferred by programmers that put a value in issues as type correct systems and like to have full contol on the system they are developing. These programmers mean that static typing increses the reliability of the system because type errors can already be found at compile time. Another advantage that they have is that static typing results in faster compiled systems that does not need run-time environements to execute.

On the other side: are those who appreciate the flexibility offered by dynamic typing, they find practical use in changing the system whenever they want, without the necessity of compiling the whole or part of the system after each change. Maybe they get the feeling that they can come into results quickly with dinamic typing.

The truth is that the debatt above is a silly one. You don't have to choose between either kind of language, there is not necessary to do that. Depending on the problem to solve and its context, the professional programmer may choose the most adequate kind of language to solve the problems in the current task.

Statically-typed programming languages are, in my opinion, the right choice when you are developing modules implemented in reusable libraries. A tipical application consists of many modules that handles different aspects of the application function. Each one of these modules is a candidate to be included in a library of reusables components. A very important module in a application is the one that implements the application core object model; other ones handles error processing, logging, communication, user interface and other tasks which are common in many systems. Each one of these modules has to be implemented with a statically-typed language to guarantee correctness and potential reuse.

1.2 Eiffel + Python = True

Ok, we now know that you may think about programming languages as you do with tools in a handcraft. As if they were

Eiffel is a strong typed language and in my opinion: the best among these, no doubt. It has a very robust type system and a lot of other features as clean syntax and design by contract that does it a given choice for a serious programmer.

Among weak typed ones, Python is a modern object-oriented scripting language with a large list of features and a huge library support. It is also a very good representant of weakly-typed programming languages.

2 Extending and embedding

Python and Eiffel are able to communicate with external languages using C as “lingua franca”. It means also that they can communicate with each other. Eiffel integrates with C by means the “external” mechanism and Python does it using the Python C-API.

It is possible to run a Python sequence of statements from Eiffel code and afterwards inspect values of Python variables from the Eiffel side. It is also possible to load Python modules programmatically in Eiffel or run Python statements entirely by means of Eiffel calls. All these cases are examples of what we shortly call embedding Python in Eiffel. This is the way to integrate Python in Eiffel, also integration in one of two possible directions.

The other way is to call Eiffel code from Python. To do that, you must: first, create dynamic libraries in Eiffel; second, wrap them in a specific way to handle parameters and return values to enable them to be loaded in a Python environment. There are tools that will help you in doing that¹.

3 Embedding

Design

4 The interpreter

5 Python types

6 Exceptions

How to use it

ssssssssssssssssssssssss

7 asi

8 Downloading

9 Requirements

10 Installing

¹Swig is tool to wrap C in another languages