

# 独创性声明

本人声明所呈交的论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得新疆农业大学或其他教育单位的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

研究生签名：刘志凯

时间：2015年5月29日

## 关于学位论文使用授权的说明

本人完全了解新疆农业大学有关保留、使用学位论文的规定，即：新疆农业大学有权保留并向国家有关部门或机构送交论文的复印件和电子文档，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文，允许论文被查阅和借阅。本人授权新疆农业大学将学位论文的全部或部分内容编入有关数据库进行检索，可以公布（包括刊登）论文的全部或部分内容。

(保密的学位论文在解密后应遵守此协议)

研究生签名：刘志凯

时间：2015年5月29日

导师签名：张太

时间：2015年5月29日



# 基于 Web 的 Python 编程环境研究

## 摘 要

随着互联网的广泛应用，编程环境已逐渐从本地过渡到 Web，经过十几年的发展，基于 Web 的编程环境已经具有良好的性能和用户基础，形成了稳定的产业链，并成为一个值得关注的重要领域。为满足人们日益增长的编程需求，开发出针对不同领域和要求的编程环境，已经成为未来的发展趋势。

首先，本文描述了基于 Web 的 Python 编程环境的基本概念，研究背景和研究意义，介绍了国内外基于 Web 的编程环境的发展现状，阐述了基于 Web 的 Python 编程环境的实现方法，分析并比较了两款开源 Python 编程环境 Skulpt 和 repl.it 的性能、特点和体系结构。

其次，本文详细介绍了基于 Web 的 Python 编程环境 CSPython 的设计方法和实现过程。具体描述了 9 大模块的主要功能和实现细节：底层 Web 框架搭建模块，前端页面展示模块，代码编辑模块，语法、关键字和内置函数模块，浏览器接口模块，Python 标准库加载模块，代码转换模块，代码编译运行模块，远程代码分享模块。

最后，本文设计了一套完整的测试方案，对 Skulpt、repl.it 和 CSPython 三种编程环境进行了测试评估，测试结果表明，CSPython 无论在功能完整性还是使用性能上都具有较好的表现，适合在实际应用中推广。

**关键词：**web；python；编程环境；代码转换；远程代码分享

# Study on Python Programming Environment Based on Web

## Abstract

With the extensive application of the internet, programming environment has been a gradual transition from the local to the web. After ten years of development, web-based programming environment already has a good performance and user base, forming a stable industrial chain and becoming an important field of concern. In order to meet people's growing programming requirements, developed for different fields and different request of programming environment has become the development trend of the future.

Firstly, this paper describes the basic concepts of web-based python programming environment, the research background and the research significance. Also, this paper describes the domestic and international development status of the web-based programming environment. And then, this paper expounds the implementation method of the web-based python programming environment. At the meantime, it analyzes and compares the performance, characteristics and system structure of two open source python programming environment: Skulpt and repl.it.

Secondly, this paper describes the design method and implementation process in details of CSPython, the web-based python programming environment. Specifically analysis the main function and implementation details of its nine main modules: constructions of the underline web framework module, front page display module, code editing module, syntax, keywords, and built-in function modules, browser interface module, python standard library loading module, code conversion module, compile and run module, remote code sharing module.

Finally, this paper designed a complete set of test scheme to test the performance of three programming environment-Skulpt, repl.it and CSPython. Test result shows that the CSPython have a better performance in terms of functional integrity or using performance, and it is suitable for promotion in practical application.

**Key words:** web-based; python; programming environment; code conversion; remote code sharing

# 目 录

<b>第 1 章 绪论</b>	<b>1</b>
1.1 研究背景	1
1.2 国内外研究状况	3
1.3 研究意义	5
1.4 论文的主要工作	6
1.5 基于 Web 的编程环境文献分析	6
1.6 论文的内容安排	7
<b>第 2 章 基于 Web 的 Python 编程环境系统概述</b>	<b>9</b>
2.1 基于 Web 的 Python 编程环境的定义	9
2.2 基于 Web 的 Python 编程环境的特点	10
2.3 常见基于 Web 的 Python 编程环境介绍	11
2.4 本章小结	14
<b>第 3 章 基于 Web 的 Python 编程环境设计与实现</b>	<b>15</b>
3.1 基于 Web 的 Python 编程环境的设计特点	15
3.2 基于 Web 的 Python 编程环境 CSPython 的体系结构	15
3.3 底层 Web 框架搭建模块的设计与实现	17
3.4 前端页面展示模块的设计与实现	19
3.5 代码编辑模块的设计与实现	22
3.6 语法、关键字和内置函数模块的设计与实现	25
3.7 浏览器接口模块的设计与实现	26
3.8 Python 标准库加载模块的设计与实现	31
3.9 代码转换模块的设计与实现	32
3.10 代码编译运行模块的设计与实现	34
3.11 远程代码分享模块的设计与实现	38
3.12 本章小结	39
<b>第 4 章 基于 Web 的 Python 编程环境测试与评估</b>	<b>41</b>

4.1 测试目的 .....	41
4.2 实验环境 .....	41
4.3 编程环境功能测试 .....	42
4.4 编程环境性能实验设计 .....	42
4.5 实验结果分析及讨论 .....	43
4.6 本章小结 .....	44
<b>第5章 结论与展望 .....</b>	<b>46</b>
5.1 研究工作总结 .....	46
5.2 研究展望 .....	47
<b>参考文献.....</b>	<b>48</b>
<b>致谢 .....</b>	<b>48</b>
<b>作者简历.....</b>	<b>52</b>

## 第 1 章 绪论

### 1.1 研究背景

自从人类步入 21 世纪以来,科学技术得到了前所未有的发展,互联网的普及和计算机的广泛应用更是为人们的日常生活带来了极大的便捷。据中国互联网信息中心的官方数据统计,截止至去年,我国网民已有 6 亿多人,较上一年又有大幅度的提高<sup>[1]</sup>。互联网的影响力已经达到了前所未有的高度,其中网民的上网方式也呈现多元化的特点。值得一提的是,手机上网比例首次超过传统 PC 上网比例,移动互联网带动整体互联网发展,可以说互联网与人们的日常生活已经密不可分,这一切都要归功于科技界的幕后英雄——程序员。

程序员是指从事程序设计、开发与维护的专业人员,对 IT 从业者来说,程序员这一名词再熟悉不过了,它的历史可以追溯到世界上第一种编程语言的出现。所谓编程语言,俗称编写计算机程序的语言,总的来说可以分成机器语言、汇编语言、高级语言三大类。由于当时的编程语言过于复杂,它与人类语言的差别极大,人们无法理解,所以我们称之为机器语言。

随着时代的变迁,编程语言也有了长足的发展。据 TIOBE2014 年 11 月编程语言排行榜统计,排在前十的编程语言分别是 C, Java, Objective-C, C++, C#, PHP, Python, JavaScript, Perl, Visual Basic.NET。其中前 6 种语言在近几年的排名中趋于稳定,而后 4 种语言的排名有上升的趋势。以上这些编程语言之所以能排名靠前,说明它们在程序设计中占据重要的地位,并深受程序设计人员的喜爱。

在 2014 年 7 月的一份报告中表明,在美国众多的著名高校中,有很多计算机院系都把 Python 语言当做计算机编程的入门语言。排在前 10 位的计算机系中就有 8 个在教授 Python,这足以说明 Python 这门编程语言的重要性<sup>[2]</sup>。

Python 作为一门编程语言,以其简单易用,层次清晰,功能丰富,运行高效而闻名。说其简单易用,是因为在 Python 中变量都不用声明,可以拿来就用,而且相对于其它语言(比如 C、Java),它的语法规则更加贴近人类的思维,便于理解;说其层次清晰,

是因为在 Python 中创造性的提出将代码的缩进强制性的规定为程序块的分界符。这种划分方式一方面使得程序间的层次更加清晰、分明，另一方面让程序员们都养成良好的编程习惯，更加利于日后代码的阅读与维护；说其功能丰富，是因为 Python 本身的标准库就很丰富，基本满足了不同用户的需求。而在此基础上 Python 还有许多获得官方认可的第三方扩展库（多达几千个），这些都是 Python 不断发展壮大的保障；说其运行高效，是因为 Python 是一种解释性的脚本语言，但更确切地说是先编译后解释的脚本语言。Python 在第一次运行时是要编译的，运行结束后会将结果写入 pyc 文件，当第二次运行时可以直接读取 pyc 文件，这样就会节省编译的时间，提高运行效率。

作为编程初学者，Python 无疑是第一选择，如果你想运行 Python 程序（这里以 windows 系统为例），传统的方法是下载 Python 安装包（Python 官方 IDE，在命令行模式下运行），然后在本地进行安装。这个流程看起来很简单，但对于编程初学者来说却存在许多问题。一方面比起编程环境的安装过程，编程初学者更加关心的是能有一个可以直接运行 Python 代码的编程环境，也就是说如果安装过程出现问题，就不能运行 Python 代码，那也就意味着你将失去一个学习和掌握一门出色编程语言的机会；另一方面假设你的安装过程没有问题，但这时如果你要运行大一点的 Python 程序代码，光有一个 Python Shell 是不够的，你可能还会需要安装像 PyCharm 这样的 Python 集成开发环境。到这里你可能会觉得只要依次安装这些开发环境，问题不就解决了？其实不然，问题远没有你想象的这么简单。假设你已经在你公司的电脑上安装了 Python 编程环境，如果你也想在家里做 Python 编程，那你必须再次安装 Python 编程环境，试想如果有一天你既不在公司也不在家里，而你又要做 Python 编程该怎么办？这时你可能会想如果能有一个方法，既不用安装 Python 编程环境，又能随时随地的做 Python 开发该有多好！根据以上背景，本文采用基于 Web 的 Python 编程环境，使编程人员免去安装和部署 Python 开发环境的过程，只要你能有一个能上网的终端（台式机，笔记本，平板电脑等），你就可以通过浏览器来访问 Python 的编程环境，从而能全身心的投入到 Python 编程中。

目前 Python 在发展过程中存在 Python2.x 和 Python3.x 两个版本，Python3.x 较 Python2.x 有了很多改动，不能向下兼容，不过许多 Python2.x 开发的平台都已陆续移植到 Python3.x 上来，而且 Python3.x 的开发社区也非常活跃，可以预计在不远的将来

Python3.x 将会完全取代 Python2.x，因此本文提到的基于 Web 的 Python 编程环境具体指的是 Python3.x。

通过以上分析可以看出，Python 是一门非常棒的编程语言，适合编程初学者使用；基于 Web 的 Python 编程环境使用方便，无需安装，只需通过浏览器访问即可。

1.2 国内外研究状况

1.2.1 国外基于 Web 的编程环境发展状况

2008 年 10 月 6 日，Remy Sharp 开发了 JS Bin，即支持实时更新 JavaScript、CSS 和 HTML 的环境，这被认为是最早的基于 Web 的编程环境。近些年来随着科学技术的发展，国外涌现了大批基于 Web 的编程环境，这些 Web 编程环境用途和规模也不尽相同，大体上分为以下几种：

- (1) 用于学习编程语言，适合编程初学者学习测试，代表产品有 CodePad, Skulpt, CodeSkulptor, Python Fiddle 等。
- (2) 用于学习 Web 开发，适合 Web 开发者学习测试，代表产品有 Codepen.io, CSSdeck.com, Dabblet.com, Fiddle Salad 等。
- (3) 用于移动 App 项目开发，适合移动 App 项目开发者学习和测试，代表产品有 Icenium.com。
- (4) 通用 IDE，是一个通用的在线开发环境，支持多种编程语言，代表产品有 Cloud9, Codenvy, Codiad 等。

国外主要基于 Web 的编程环境如下表所示：

表 1-1 国外主要基于 Web 的编程环境

序号	名称	支持语言	用途
1	Compilr	C,C++,Java,Python 等	商用
2	Dabblet	HTML,CSS	测试工具
3	jsdo.it	JavaScript,HTML5,CSS	学习交流
4	Thimble	HTML,CSS	学习交流
5	Jsfiddle	JavaScript,HTML,CSS	测试工具
6	CodeMirror	多种语言实时高亮	学习交流
7	eXo Cloud IDE	Java,Groovy,Ruby,PHP,Python 等	商用
8	JS Bin	JavaScript,CSS	测试工具



续表 1-1 国外主要基于 Web 的编程环境

序号	名称	支持语言	用途
9	eCoder	PHP, JavaScript	学习交流
10	Kodingen	PHP,Perl,Python 等	商用
11	EditArea	多种语言实时高亮	学习交流
12	Codeanywhere	HTML,PHP, JavaScript,CSS,XML	商用
13	Drawter	HTML,CSS	学习交流
14	Maqetta	HTML5	学习交流
15	ShiftEdit	PHP, Python,Rerl, JavaScrip 等	学习交流
16	Squad Editor	C/C++ CSS HTML Java JavaScript 等	商用
17	Cloud9 IDE	Ruby,Python,PHP,等	商用
18	CodePad	C/C++,PHP,Python 等	测试工具
19	IdeOne	C/C++,Java,PHP,Python 等	测试工具

1.2.2 国内基于 Web 的编程环境发展状况

国内对于 Web 编程环境的研究相对滞后，一是因为国内对于 Web 编程环境的研究起步较晚，近些年来才刚刚兴起；二是由于国内与国外教育理念的差异，国外教育以学生为中心，重在培养学生学习兴趣，而国内教育以老师为中心，老师只重视结果而忽视学习的过程，学生则处于被动接受老师灌输知识的地位，这无疑阻碍了基于 Web 的编程环境的研究与发展。

目前国内编程开发环境大多采用本地安装的方式，用于教学的在线编程环境少之又少，所以开发基于 Web 的编程环境无论是用于教学实践还是用于研究测试，都将为传统的教育模式或业界研究人员带来新鲜的血液。这种在线编程的模式不仅简单轻便，因为它只需运行在浏览器中。而且适合学生作业的协同批改，因为它能为每个程序代码生成唯一的 URL 地址。当然这些优点的前提是它拥有大多编程环境所拥有的功能。

1.2.3 开源的基于 Web 的编程环境发展状况

现如今越来越多的软件选择以开源的形式发布出来，而这些开源的软件往往能发展得更好，因为在这些开源软件身后大都会有开源社区的支持，这些社区内有许多来自世界各地编程高手，他们会持续的对开源软件进行更新和维护。

目前开源的基于 Web 的编程环境主要有两大类。第一类是针对某个特定编程语言的编程环境，这其中比较出名的有 Skulpt；第二类是针对大多数编程语言的通用编程环

境和代码编辑器，这其中比较出名的有 repl.it, EditArea 等。

表 1-2 主要开源的基于 Web 的编程环境

序号	名称	分类
1	Skulpt	Python2
2	eCoder	PHP, JavaScript
3	Maqetta	HTML5
4	EditArea	多种语言实时高亮
5	repl.it	Python2

### 1.3 研究意义

随着 Internet 技术的发展，人们的日常生活已经离不开网络，而对于程序员来说，我们的世界也已经离不开各种程序了，电脑，智能手机，各种电子设备、系统等。夸张点说，程序员正在接管世界，而世界也因为网络得以互联互通。可以预计的是，人类社会与网络的关系将会更加密切，所以基于 Web 的编程将是未来发展的趋势，它将大大简化编程过程，提高开发效率。另外，本文选取的 Python 语言以其实用灵活，功能强大等特点成为近年来众多程序员研究和使用的首选。

本文通过阅读大量专业文献，仔细研究国内外主要成果，在此基础上设计并开发出基于 Web 的 Python 编程环境，认真地描述了它的设计特点，详细的阐述了它的体系结构，包括各个功能模块在实现上的技术细节和方法，随后对它进行了一系列合理实用的性能和功能测试，进一步的论证了它在理论上和实际中的意义，为基于 Web 的编程环境的开发和应用提供了有价值的研究资料和数据。

虽然在基于 Web 的 Python 编程环境研究方面，国外要比国内成熟得多，也出现了针对学习测试和商用两类不同的产品，那么本文设计和研究的意义何在？一方面是因为目前大部分的编程环境还只支持 Python2，即便是支持 Python3，对于学习测试的编程环境来说，它们只适用于一些小的程序，不能满足运行大的程序的需求；另一方面对于商业用途的编程环境来说，用户需要定期向其支付一定的费用，并且只能分得有限容量的网络存储。

综上所述，本文设计并开发的基于 Web 的 Python 编程环境无论是在理论研究，还是在实际应用上都具有重要的研究意义。

## 1.4 论文的主要工作

本文首先对现有基于 Web 的编程环境相关理论和方法以及存在的主要问题进行了深入分析,重点研究了 Python 语言在基于 Web 的编程环境中的应用。在此基础上针对浏览器在处理方式上的特殊需求及应用环境,采用将 Python 代码转换成 JavaScript 脚本代码的方法。该方法通过引入 brython.js 文件来执行转换后的 JavaScript 代码,实现在浏览器中运行 Python 代码,最后重点介绍了基于此方法的原型系统的设计与实现。

## 1.5 基于 Web 的编程环境文献分析

### 1.5.1 集成开发环境

所谓集成开发环境,就是指拥有编程环境的工具。Borland 公司首次提出集成开发环境这个概念,而 Basic 则是第一个使用集成开发环境的编程语言。文献<sup>[3-12]</sup>介绍了如何通过集成开发环境来开发相关项目。

### 1.5.2 Python 语言

Python 语言由 Guido van Rossum 在 1989 年底开发,第一个公开发行人版发行于 1991 年<sup>[13]</sup>,它的源代码遵循 GPL(GNU General Public License)协议<sup>[14]</sup>。其中 G Van Rossum 和 FL Drake 于 2003 年出版了 Python 参考手册<sup>[15]</sup>,对这门编程语言进行了系统的介绍,而 A Downe 则向人们介绍如何像 Python 一样思考<sup>[16]</sup>。JM Zelle 在 2004 年提出将 Python 语言在计算机科学与编程中推广<sup>[17]</sup>,TE Oliphant 于 2009 年讨论将 Python 语言应用在科学计算上<sup>[18]</sup>。当然,Python 语言不仅在计算机学科中使用广泛,比如机器学习<sup>[19]</sup>,还包括分子生物学和生物信息学中的应用<sup>[20-21]</sup>,在 2009 年 S Bird 与 E Klein 等人已经讨论使用 Python 来进行自然语言处理<sup>[22]</sup>。

### 1.5.3 JavaScript

JavaScript 是一个人们熟知的脚本语言,由网景公司的 Brendan Eich 开发。JavaScript 第一次问世是在 1995 年,它被应用在网景公司的 Netscape Navigator 浏览器中<sup>[23]</sup>,从此 JavaScript 就与 HTML 页面结缘,在近十几年的时间里,成为最受欢迎的脚本语言之一

[24-26]。其中 Douglas Crockford 是资深的 JavaScript 架构师,他同时还发明了 JSON、JSLint、JSMIn 和 ADSafe, 由他编写的《JavaScript: The Good Parts》(包括英文版和中文版)一直深受好评[27-32]。

#### 1.5.4 Web 框架

Web 框架是一种用来支持动态网站、网络应用程序及网络服务的开发框架。由于 Web 框架可提高网站开发效率的原因,使得它成为程序员必须掌握的技能之一。具体来说 Web 框架有很多种,按语言种类来分可以分为 PHP、JavaScript、Python、Ruby、.net、Java 等。Supaartagorn 和杨彦侃等研究了基于 PHP 框架的系统应用[33-34];王映、周林和曹宇等探讨了如何在 JavaScript 框架下开发 Web 应用[35-37];Forcier 和范文星等描述了基于 Python 的 Django 框架[38-39],而 DiPierro 和 Massimo 则介绍了另一种基于 Python 的企业及框架 Web2py[40];王准和张辉等描述了基于 Ruby On Rails 框架的 Web 开发[41-42];张帆、赵伟和史国滨等系统的介绍了基于.NET 技术的 Web 开发[43-45];廖福保、张宇和陈辉分别对 Java 主流框架 Spring、Hibernate 和 Struts 做了详细的讲解和说明[46-48]。

#### 1.5.5 代码编译器

一个好的代码编译器通常由两大主要功能:代码编辑与代码运行。CodeMirror 是一款开源的代码编辑器,支持多语言的语法高亮[49]。Skulpt、repl.it 和 Brython 都是在浏览器中运行 Python 代码的开源项目,不同的是前两者只支持 Python2,而 Brython 则支持 Python3[50-52]。

### 1.6 论文的内容安排

#### 第一章:绪论

详细介绍基于 Web 的编程环境的研究背景,具体分析基于 Web 的编程环境的研究意义,主要讨论目前国内外相关研究成果的文献和发展状况。

#### 第二章:基于 Web 的 Python 编程环境系统概述

介绍基于 Web 的 Python 编程环境的定义和主要特点,对 Skulpt 和 repl.it,两款开

源的基于 Web 的 Python 编程环境的体系结构和运行特点进行重点介绍。

### 第三章：基于 Web 的 Python 编程环境设计与实现

着重介绍基于 Web 的 Python 编程环境 CSPython 的 9 大模块：底层 Web 框架搭建模块，前端页面展示模块，代码编辑模块，语法、关键字和内置函数模块，浏览器接口模块，Python 标准库加载模块，代码转换模块，代码编译运行模块，远程代码分享模块。详细阐述这些模块的实现细节和关键技术。

### 第四章：基于 Web 的 Python 编程环境测试与评估

设计一套完善的测试方案对 Skulpt、repl.it 和 CSPython 三种编程环境进行测试，用以评估 CSPython 的性能。通过实验结果论证了 CSPython 在功能完整性和使用性能上的优势。

### 第五章：结论与展望

对全文的主要工作内容进行详细总结，并对下一步的研究重点进行展望。

## 第 2 章 基于 Web 的 Python 编程环境系统概述

### 2.1 基于 Web 的 Python 编程环境的定义

所谓编程环境或者说是开发环境，指的就是一个软件工具，它为用户提供程序开发所需的必要功能组件。这样的编程环境一般包括代码编辑器、编译器、调试器和图形用户界面工具，因此所有具备这一特性的软件或软件套件都可以叫做编程环境。如微软的 Visual Studio，Borland 的 C++ Builder、Delphi 等。显而易见，基于 Web 的 Python 编程环境就是一个为用户提供 Python 开发所需必要功能组件的软件工具，所不同的是，这个编程环境是架构在 Web 端，而不是在本地客户端。通俗一点的说，它是一个通过 Web 浏览器来编译和运行 Python 代码的软件工具。

由于基于 Web 的 Python 编程环境是在传统的 Python 编程环境基础之上发展来的，所以它继承了传统的 Python 编程环境的一些基本特性，它与后者在功能上有许多相似之处，不同之处在于，它在使用过程中会因为基于 Web 的编程环境的特殊性而受到各个方面的制约，包括浏览器的支持，页面的设计，代码的解析等等问题，因此在该编程环境的使用上有很强的针对性。综上所述，本文从以下三个方面来继续详细阐述基于 Web 的 Python 编程环境所固有的特性：编程环境的设计特性、基于 Web 的特殊性、独特的个性化功能特性。

**编程环境的设计特性：**由于编程环境的底层架构在 Web 端，所以它在设计上具有一定的特殊性。它具体体现在整个编程环境的作用范围、快速性、准确性、稳定性、高效性、安全性和可交互性等方面。这些特性可以通过项目整体框架的选取，各个功能模块的合理安排，底层编译器的设计，代码风格的控制和前端页面的美化等特点来具体的衡量。

**基于 Web 的便捷性：**基于 Web 的便捷性体现在相比传统的 Python 编程环境是否能更加灵活更加方便的完成编程工作，为用户带来更便捷的用户体验。这种便捷性主要体现在几个方面，第一，它能够自行提供开发环境，使得用户可以摆脱传统编程环境的束缚。随着互联网的迅速普及，电脑，pad，手机等网络终端设备成为人们生活中的必需

品, 这些终端设备五花八门, 功能各异, 但它们却有一样是相同的, 就是都拥有一个畅游网络的浏览器, 基于 Web 的 Python 编程环境正是借助浏览器来完成编译运行 Python 代码的功能, 免去在没有安装 Python 编程环境的设备上安装软件的苦恼。第二, 实现远程代码分享功能, 用户可以通过 Web 工具与互联网上的其它用户更方便的交流与协作。如今的时代是开放的, 也是相互关联的, 作为程序员的我们更是如此, 你不必再为困扰了你许久的技术难题而孤军奋战, 网络让我们能与更多的开发者建立联系, 让我们能方便的协同工作, 一起分享编程带来的喜悦, 一起解决让你百思不得其解的难题, 而这些都将提高开发效率, 节省工作时间。

独特的个性化功能特性: 独特的个性化功能特性反映在基于 Web 的 Python 编程环境设计思想和应用上。光有一个有建设性的想法是不够的, 它只是成功的基础, 还需要将想法付诸行动并在实际中应用。本文设计并开发的编程环境着眼于实际问题, 通过整合现有资源, 改进传统编程手段, 创造出高效便捷的编程环境。这个编程环境集代码编辑、编译、运行、代码存储与分享等为一体, 在应用上能达到准确、高效等指标。

## 2.2 基于 Web 的 Python 编程环境的特点

基于 Web 的 Python 编程环境在设计之初就是秉着为用户提供方便高效、安全可靠的原则来设计的。与市场 and 网络上那些编程环境为用户提供界面花哨、种类繁多但大部分用户都不会去用的功能不同, 本编程环境只为用户提供一些常用的功能, 而这些功能对于程序开发来说也已经足够用了。基于 Web 的 Python 编程环境提供了一个简约直观又不失美观的操作界面, 该编程环境的操作方法也非常简单, 它内置一个著名的代码编辑器, 在代码编译方面由于所处 Web 平台的特殊性, 与传统的 Python 编译方式不同, 采用将 Python 代码转换成 JavaScript 脚本代码的方式, 再通过浏览器来解析。最后编写的代码支持以链接的形式进行保存。这种做法即解决了编译 Python 代码的难题, 同时在效率上也有一定的提高。一方面由于浏览器最擅长解析的就是 HTML 和 JavaScript, 而将 Python 代码转化成 JavaScript 脚本代码也可以实现, 而且这也是浏览器比较擅长的, 在程序执行的效率上也有较好的体现。另一方面对于用户而言, 代码的保存变得有趣多了, 这种保存为链接的形式为用户提供协同作业的机会。最后, 相比本地编程环境, 基

于 Web 的编程环境占用的系统开销相对较小, 因为这部分“压力”其实已经转移到网站的服务器端, 而且用户存储的数据只要通过适当的算法来保存, 数据量也是在可控的范围内。总之基于 Web 的 Python 编程环境的主要特点是: 用户访问速度快, 编程环境运行准确稳定、系统开销小。

基于 Web 的 Python 编程环境的主旨就是用最小的系统开销来实现最常用的功能需求, 因此它在设计上也是遵循简单、实用、稳定、高效的原则。具体的性能要求有如下几点:

- (1) 支持多款浏览器, 以实现基于 Web 的特性。
- (2) 安全性和稳定性。
- (3) 可移植性和可扩展性强。
- (4) 系统开销低。
- (5) 程序运行准确高效。
- (6) 开发平台简单易维护。

相对应的它的功能要求有如下几点:

- (1) 支持几种常用的编辑功能。
- (2) 支持多用户的并发访问操作。
- (3) 支持代码关键字的高亮显示。
- (4) 支持语法纠错。
- (5) 支持 Python 代码的编译运行和保存。
- (6) 支持智能缩进。
- (7) 支持多人协同操作。

## 2.3 常见基于 Web 的 Python 编程环境介绍

### 2.3.1 Skulpt 简介

Skulpt 是由 Scott Graham 创建并发起的一个开源项目, 从 2010 年开始由 Brad Miller 接管。它用 JavaScript 来实现编译运行 Python, 所以它是一个完全基于浏览器的 Python 运行环境, 你可以直接输入 Python 代码并运行, 无需任何预处理、无需服务器端支持。



可以说 Skulpt 是一个完全独立于本地环境的 Python 运行环境,目前它只支持 Python 2.x, 你可以在 <http://www.skulpt.org> 发现更多的实例。另外它的项目源码可以在 [github](https://github.com/skulpt/skulpt) 上下载, 它的网址是 <https://github.com/skulpt/skulpt>, 上面还有许多帮助文档能帮助你对其有更多的了解, 具体说来 Skulpt 有以下特性:

- (1) 无需安装额外软件, 直接在浏览器中运行。
- (2) 源代码完全开放。
- (3) 提供技术支持的社区长期处于活跃状态。
- (4) 有完整的开发人员手册。
- (5) 源代码有较好的注释。
- (6) 运行流畅, 体积轻便小巧。

正因为 Skulpt 的小巧轻便, 使其在许多领域得到应用, 莱斯大学于 2014 年在 Coursera 在线教育平台上开设的一门 Python 语言编程课, 使用的编程环境就是以 Skulpt 为基础设计而成的, 在实际使用中取得了良好的效果。

当然, Skulpt 还在不断完善中, 还有许多地方需要改进:

(1) 从 CPython 的标准库中扩充 Skulpt 的标准库, 以包括更多的模块。到目前为止, Skulpt 拥有 `math`, `random`, `turtle`, `time` (部分) `random` (部分) `urllib` (部分) `unittest`, `image`, `DOM` (部分) 和 `re` (部分) 函数库。

(2) 随着时间的推移, 越来越多的用户希望在 Skulpt 中加入更加高级的 Python 模块, 其中包括 `matplotlib`、`Tkinter` 和 `numpy`。

(3) 扩展和清理外部函数 API, 这些 API 是实现部分标准库的关键。

(4) 做到支持 Python3 的语法, 最好能设置一个标志位来选择运行 Python2 和 Python3。

(5) 做到支持 DOM 访问, 并把它作为标准库的一部分。

(6) 目前 Skulpt 内置的类型 (`list`, `tuple`, `string` 等) 没有子类化。使这些内置类型子类化可以消除一些 Skulpt 中已知的错误。

(7) 扩大和提高综合语言覆盖面。目前 Skulpt 在这方面非常符合 80/20 法则 (二八定律, 也叫帕累托法则, 即少数 (20 %) 是至关重要的, 大多数 (80 %) 是微不

足道的)。Skulpt 已经覆盖了所使用代码 80% 的语言特性（或者高达 90%），但是有许多内置类型没有被覆盖。

（8）改变代码执行模式，使其可以逐行或逐步执行。

（9）增加调试功能。

### 2.3.2 repl.it 简介

提到 repl.it 这个名字，首先你会想到 repl（英语：Read-Eval-Print Loop，简称 REPL），它是“读取-求值-输出”循环的缩写，这个词常常用于指代一个 Lisp 的交互式开发环境，但也能指代命令行的模式和例如“APL、BASIC、Clojure、F#、Haskell、J、Julia、Perl、PHP、Prolog、Python、R、Ruby、Scala、Smalltalk、Standard ML、Tcl、JavaScript”这样的编程语言所拥有的类似的编程环境。

当然这个 repl.it 可能正是想做到这一点，具体说来它是一个交互式编程环境，底层基于 jsREPL。它支持的语言如下：

表 2-1 repl.it 编程环境支持的语言

支持语言种类	语言名称
网页语言	JavaScript, CoffeeScript, Kaffeine, Move, JavaScript.next
深奥的语言	Bloop, Brainfuck, LOLCODE, Unlambda, Emoticon
经典的语言	Quick Basic, Forth
严肃的语言	Scheme, Lua, Python, Ruby (beta)

repl.it 是一个开源项目，由于它支持十几种语言，所以把它作为代码测试来用是一个不错的选择。不同于 Skulpt 的是，repl.it 需要构建自己的运行环境，具体步奏如下：

#### 1. 到 github 上克隆代码

```
git clone git://github.com/replit/repl.it.git
```

```
cd repl.it
```

```
git submodule update --init --recursive
```

#### 2. 安装依赖环境

```
node.js,npm,CoffeeScript,Pygments
```

#### 3. 运行

repl.it 捆绑了一个服务器和编译器，输入如下命令：

```
./server.js 8080
```

然后在浏览器中输入网址：<http://localhost:8080/index.html>，就会进入到 repl.it 的编程环境界面。

可以说 repl.it 也是一个非常简单高效的编程环境，虽然其目前还只支持 Python2，但它在设计理念和功能实用性上值得本文借鉴和学习。

2.4 本章小结

本章主要介绍了基于 Web 的 Python 编程环境的定义和特点，着重分析了两款成熟的基于 Web 的 Python 编程环境。从以往的用户反馈和未来的发展趋势来看，这两款基于 Web 的 Python 编程环境都有很大的发展空间，特别是在市场占有和用户体验上也都有巨大的发展潜能。在尝试多种途径搜集资料，翻阅大量文献，然后经过认真严格的筛选后，本文从以下角度考虑，选择对它们的性能和特点进行多层次和全方位的比较：

表 2-2 Skulpt 和 repl.it 的比较

性能特点	Skulpt	repl.it
是否需要安装	否	是
是否需要数据库	否	否
开发语言	Python2, JavaScript	Python2, JavaScript
支持语言	Python2	Python2/ QBasic/JavaScript 等
编译方式	JavaScript	JavaScript
代码保存类型	无	链接
适用系统	Linux/windows/Mac/Android	Linux/windows/Mac/Android
编译错误提示	有	有
支持调试功能	否	否
是否开源	是	是
使用难易程度	较易	适中

## 第 3 章 基于 Web 的 Python 编程环境设计与实现

本文以开源在线编译器 Skulpt、Brython 和开源在线代码编辑器 CodeMirror 为原型，设计并实现了一个基于 Web 的 Python 编程环境：CSPython。它采用了本章所提供和设计的 Python 编译方式，实现了 Python 代码与 JavaScript 脚本代码的转换，以及增加对 Python3 标准库的支持和远程代码分享等功能。本章首先介绍 CSPython 的主要特点、体系结构和各个模块的主要功能，然后详细描述了 CSPython 中几个主要模块具体的实现细节。

### 3.1 基于 Web 的 Python 编程环境的设计特点

CSPython 编程环境主要有以下特征：

- (1) 轻量级的应用，体积轻便，功能完善。
- (2) 运行环境范围广，可以适用于所有现代通用浏览器中。
- (3) 基于 Web 模式。
- (4) 支持常用编辑操作，包括剪切、复制、粘贴、保存等功能。
- (5) 支持智能代码编辑，自动识别关键字、关键字高亮显示、支持快捷键智能缩进。
- (6) 实现对 Python3 代码的编译运行，提供运行提示信息，包括错误信息，错误行行号。
- (7) 支持常用 Python3 标准库。

### 3.2 基于 Web 的 Python 编程环境 CSPython 的体系结构

基于 Web 的 Python 编程环境 CSPython 的体系结构主要有九个模块，实现了在浏览器中运行 Python 的功能。整个工程项目编写的代码层次清晰，功能明确（主要是 Python 代码，还包括一些 Html 页面），而且不需要额外的外部依赖环境，可以独立运行于各大主流操作系统上。

CSPython 的主要功能模块有：底层 Web 框架搭建模块，前端页面展示模块，代码编辑模块，语法、关键字和内置函数模块，浏览器接口模块，Python 标准库加载模块，代码转换模块，代码编译运行模块，远程代码分享模块。

(1) 底层 Web 框架搭建模块：基于 Web 的编程环境 CSPython 的底层采用开源的 Web 应用框架——Django，它包括一个模式系统，对象相关的映射和用于动态创建管理界面的结构，而且该框架是由 Python 编写的，所以有关 Python 的接口和 API 文档都非常完善。

(2) 前端页面展示模块：整个页面由 HTML 语言实现，页面布局采用 CSS+DIV，其中还包含大量 JavaScript 和 Python 代码。

(3) 代码编辑模块：采用非常有名的语法高亮显示引擎 CodeMirror。CodeMirror 是用 JavaScript 来实现，并且应用在浏览器中的多功能文本编辑器。它是专门用来编辑代码的，与此同时它还拥有许多可选的语言模式和插件，以便来实现更高级的编辑功能。

(4) 语法、关键字和内置函数模块：定义 CSPython 中支持的语法、关键字和内置函数。

(5) 浏览器接口模块：介绍了 DOM（文档对象模型）接口，负责与浏览器交互。

(6) Python 标准库加载模块：介绍了如何从 Python 标准库中导入模块。

(7) 代码转换模块：将 Python 代码转换成 JavaScript 代码。

(8) 代码编译运行模块：通过在页面中加载 brython() 函数来实现 Python 代码的编译和运行。

(9) 远程代码分享模块：将用户的代码以链接的形式保存。

CSPython 系统架构图如下所示：

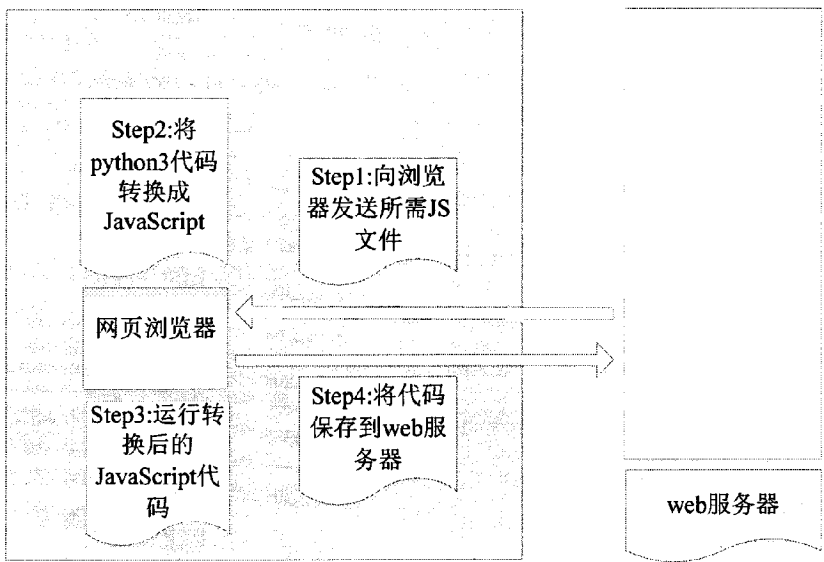


图 3-1 CSPython 系统架构图

3.3 底层 Web 框架搭建模块的设计与实现

要设计和开发 CSPython 编程环境项目首先要选取合适的 Web 框架，Django 是由 Python 开发的一个免费的开源网站框架，可以用于快速搭建高性能、优雅的网站。Django 框架遵循 MVC(Modern,view,controler)开发模式，它包含多种 Web 开发组件，同时拥有一个独立的轻量级 Web 服务器，为快速开发 Web 应用提供便利，具体说来它具有以下特点：

- （1）简单的对象关系映射：支持 SQL 语句，一条记录对应一个对象，而对象之间也很容易建立关联。
- （2）出色的后台管理系统：从创建的第一个 APP 起，Django 就为用户自动生成了一个出色的后台管理系统，相比一般 Web 网站后台也毫不逊色。
- （3）灵活的 URL 设计：虽然用到了正则表达式，但只要仔细研究会发现，它其实非常灵活，只要拥有一定的基础，都能开发出出色的网站。
- （4）简单的 APP 应用程序：它就像在手机上安装一款应用程序一样简单，如果不喜欢可以直接删掉，而且对整个平台没有较大的影响。
- （5）友好的错误提示界面：Django 为用户提供非常详尽的错误提示界面，使用户

可以准确的定位错误所在位置。相比其它平台，Django 的提示界面也更加友好。

(6) 多语言支持：Django 可以实现不同语言的国家或地区在浏览网站时显示本地语言的功能。

在这个模块中，设计并实现了 CSPython 的 Web 框架，由于 Django 框架的诸多优点，使得整个框架的搭建过程变得更容易，具体开发过程如下：

### 1. 安装 Python

本项目使用的是 Ubuntu14.04，系统已经预装了 Python。在命令提示符下输入 Python3 可以验证是否安装成功，并可查看 Python 版本，如果系统没有预装 Python，则需要手动下载安装。

### 2. 安装 Django

由于本系统没有自带 Django，所以选择手动下载然后安装。首先，下载 Django-1.6.5-final.tar.gz 压缩文件，解压缩之后运行 `setup.py install` 即可。

在 Python 交互窗口中输入 `import django` 会导入 Django 模块，然后输入 `django.Version` 可以查看 Django 版本。

### 3. 安装数据库

Django 支持 PostgreSQL、SQLite 3、MySQL 和 Oracle 四种数据库。本项目使用 MySQL 数据库，可以从 <http://www.djangoproject.com/r/Python-mysql/> 来下载和安装以下环境：python-mysql、python-mysqldb 和 myspl-python。

### 4. 创建工程项目

为基于 Web 的 Python3 编程环境 CSPython 创建名为 mysite 的工程项目：

```
django-admin.py startproject mysite
```

项目结构如下：

```
mysite/  
    __init__.py  
    manage.py  
    settings.py  
    urls.py
```

## 5. 运行开发服务器

运行命令：`python manage.py runserver` 将启动 Django 服务器。现在用浏览器访问 `http://127.0.0.1:8000/`，如果看到 Django 的欢迎页面，这表明它已经开始工作了，这样一个 Web 环境就搭建好了，接下来需要做的就是补充并完善整个项目的功能。

虽然 Django 本身提供了一个适合快速开发的 Web 服务器，但考虑到后期平台因用户数量的增多带来的高并发问题，决定将其移植到 Apache 服务器上，具体操作也很简单，只需安装和配置相关文件即可。Apache 服务器全称为 Apache HTTP Server，它是一个开源的网页服务器，并且可以运行在绝大多数的计算机操作系统上，由于其运行快速，稳定性高，支持多平台，拥有较高安全性，它已经成为世界上最流行的 Web 服务器。

## 3.4 前端页面展示模块的设计与实现

如果说底层 Web 框架搭建模块是骨架，那么前端页面展示模块就是骨架外面的脸面，而往往好的页面布局能给人眼前一亮的感觉。CSPython 编程环境的页面展示采用简单明了的设计布局。该模块实现了用户与编程环境的数据交互，代码的编写和与之相关的一些操作，在外观上达到了界面友好的特点，在功能和结构布局上能做到层次分明。其实一个好的编程环境说到底就是一个实现编译、运行功能的工具，在这个原则上开发出一个界面简洁，层次结构分明，功能齐全的编程环境就是本项目一直以来的追求。要做到这点，具体要实现以下功能：

- (1) 整个编程界面风格简洁明了，能突出各个功能区的特点。
- (2) 在代码编辑区能正确完整的完成代码编辑功能，包括代码的剪切、复制、粘贴、关键字高亮显示、智能代码缩进、行号显示，支持代码编辑器选择不同的主题。
- (3) 在结果输出区能正确的显示编译结果，包括编译错误的提示信息。
- (4) 在性能方面能做到对各操作作出快速反应，具有良好的交互。

整个页面的布局示意图如下所示：



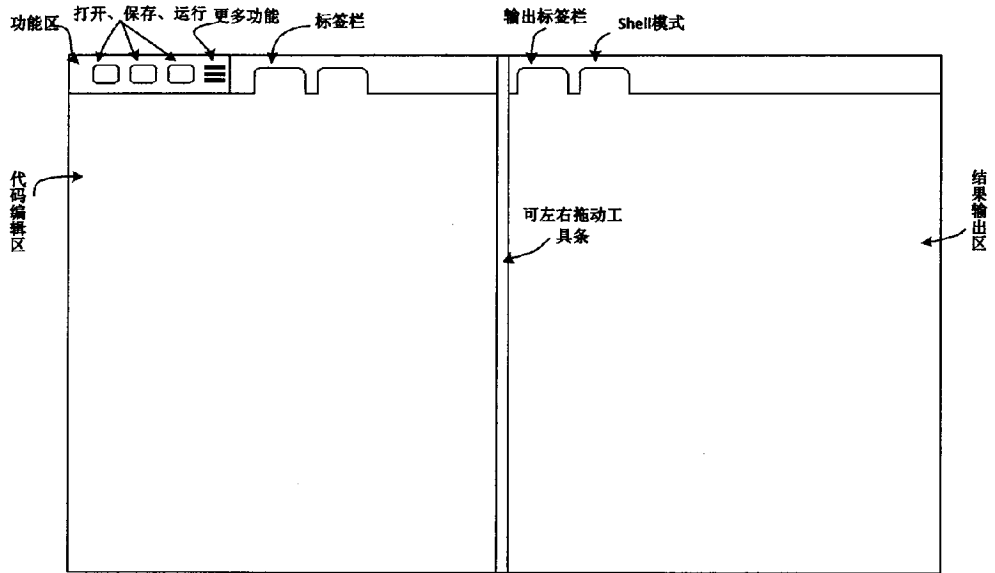


图 3-2 基于 Web 的 Python 编程环境页面布局示意图

在这个页面布局示意图中可以清楚的看到整个编程环境被分成了三个部分：

1. 功能区
2. 代码编辑区
3. 结果输出区。

其中每个区域内又包含多个功能控件按钮，下面就每个功能区和对应的功能控件做详细的介绍和说明。

（1）功能区：在这个区域内包含了对 Python3 程序代码和整个编程环境的全部操作，具体的功能有：

- 打开：当鼠标单击打开按钮后会弹出一个“打开文件”对话框，该功能允许用户在代码编辑区打开一个本地计算机上以.py 为后缀名的 Python 文件，这样可以方便用户直接导入已经存在的 Python 文件。
- 保存：当用户单击保存按钮后，系统后台会做出两步响应。第一步响应是该编程环境将用户在代码编辑区编写的程序代码保存到系统指定的目录中去；第二步响应是在作出第一步响应的同时系统将生成对应保存有用户编写的程序代码的页面，该页面由相应的算法来保证它的唯一性，这其实就是生成了一个链接地址，而这个链接地址就指向保存有用户程序代码的页面，这样做的好处是为用户提供了一个相互分享和交流的便

利。

- 运行：顾名思义，当这个运行按钮被按下后，会触发该编程环境对代码编辑区的程序代码进行编译，相应的编译结果会显示在输出标签栏的结果输出区，在结果输出区中如果程序代码运行正确会得到相应正确的结果，如果程序代码运行错误会得到相应的错误提示信息。

- 新建标签：如图所示，在示意图的左侧代码编辑区有两个标签栏，如果用户想重新打开一个新的标签栏，可以单击“新建标签”按钮，这样用户想要编写新的程序代码就不用再重新打开一个新的页面了。

- 更多功能：其实“更多功能：是功能区的一个扩展，里面也包含部分功能区里的功能，另外它还包括一些帮助文档和程序代码实例，以帮助用户更好的来理解和学习 Python 这门编程语言。

(2) 代码编辑区：代码编辑区也是 CPython 编程环境的一个主要组成部分，CPython 编程环境的代码编辑区位于整个编程环境的左侧，下图是代码编辑区示意图：

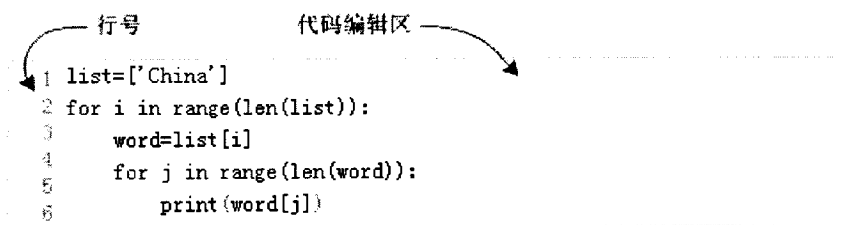


图 3-3 代码编辑区示意图

(3) 结果输出区：

运行上图的程序代码，得到如下运行结果：

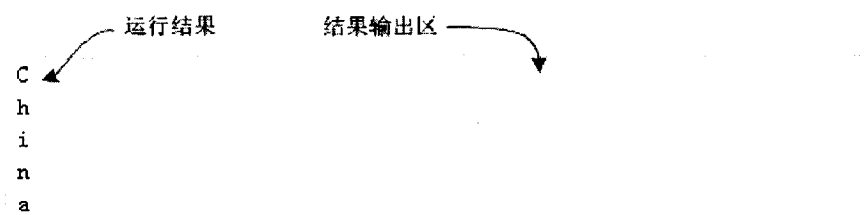


图 3-4 结果输出区

值得一提的是，在结果输出区上还有一个 shell 模式标签栏，这个 shell 模式就是我们熟悉的标准 Python shell 命令行模式。

### 3.5 代码编辑模块的设计与实现

在上一小节中简单介绍了 CPython 编程环境的编码编辑区的主要布局，在本节中将着重介绍如何在页面中嵌入代码编辑器。

目前网络上流行的在线代码编辑器不下十几种，它们大都支持在线代码的编辑和语法高亮，然而有一款在线代码编辑器却一直受到众多程序员的热捧，它就是 CodeMirror，原因很简单，因为许多有名的在线代码编辑器都是以它为基础来设计和实现的，所以说它也是一款追求自由的软件。CodeMirror 拥有丰富的 API 和 CSS 主题系统，这些都可以自由定制，以此来满足不同的用户需求，并为应用程序扩展新的功能。

作为一款出色的在线代码编辑器，CodeMirror 具有以下特点：

1. 支持的语言超过 90 种，而且能够“开箱即用”。
2. 一个强大的，可组合的语言模式系统。
3. 支持自动完成功能。
4. 支持代码折叠功能。
5. 可配置的快捷键。
6. 支持 Vim, Emacs 和 SublimeText 的键位绑定。
7. 拥有查询和替换接口。
8. 能匹配括号和标签。
9. 支持拆分视图。
10. 集成 Linter (JavaScript 语法检查)。
11. 混合字体大小和样式。
12. 拥有各种不同的主题。
13. 根据内容调整。
14. 支持行内和块内插入小工具。
15. 行号沟槽可编程。
16. 能识别代码块。
17. 支持双向文本（能正确处理从左向右和从右向左书写的脚本）。

18. 许多其它的方法和插件。

目前 CodeMirror 兼容的浏览器如下表所示:

表 3-1 CodeMirror 支持浏览器情况一览表

浏览器名称	适用版本
Firefox	4.0 及以上
Chrome	所有版本
Safari	5.2 及以上
Internet	8.0 及以上
Opera	9.0 及以上

从上表可以看出 CodeMirror 实现了对大多数主流浏览器的支持, 只要用户选择对应的浏览器版本都能获得良好的编程体验。

CodeMirror 这个名称从字面上来理解就是“代码镜子”的意思, 而事实上它就是通过“textarea 文本框”来对程序代码进行美化的插件, 一个简单的 CodeMirror 实例如下所示:

```

1 <!-- Create a simple CodeMirror instance -->
2 <link rel="stylesheet" href="lib/codemirror.css">
3 <script src="lib/codemirror.js"></script>
4 <script>
5   var editor = CodeMirror.fromTextArea(myTextarea, {
6     lineNumbers: true
7   });
8 </script>

```

图 3-5 CodeMirror 运行界面

要实现这样的功能, 具体的调用方法如下所示:

#### 1. 加载核心文件

```
<link rel="stylesheet" href="lib/codemirror.css">
```

```
<script src="lib/codemirror.js"></script>
```

上面两行分别加载了两个文件: codemirror.css 和 codemirror.js, 这两个文件是 CodeMirror 里面最核心的文件, 所以无论是实现哪种语言的高亮显示效果都要加载这两个文件。

#### 2. HTML 示例代码内容如下:

```
<textarea id="demotext">
```

```
<!-- Create a simple CodeMirror instance -->
<link rel="stylesheet" href="lib/codemirror.css">
<script src="lib/codemirror.js"></script>
<script>
    var editor = CodeMirror.fromTextArea(myTextarea, {
        mode: "text/html"
    });
</script></textarea>
```

其中 “ myTextarea ” 为你的编辑器 dom 元素 ID ， 一般使用 document.getElementById( “code” )方法获取文本框的内容。

3. 函数调用代码如下：

```
<script>
    var editor = CodeMirror.fromTextArea(document.getElementById("demotext"), {
        mode: {
            name: "Python",
            version: 3,
            singleLineStringErrors: false
        },
        lineNumbers: true,
        indentUnit: 4,
        matchBrackets: true,
        autoMatchParens: true,
        indentWithTabs: true,
        smartIndent: true,
        autofocus: true
    });
</script>
```

由于 CSPython 编程环境是 Python3, 所以 mode 的 name 属性设置为 Python, version 属性为 3, 同时该编辑器设置 lineNumbers 为 true, 表示可以显示行号; 设置 indentWithTabs 为 true, 表示支持 tab 缩进; 设置 indentUnit 为 4, 代表 tab 缩进为 4 字符; 设置 matchBrackets 为 true, 表示支持括号匹配; 设置 smartindent 为 true, 表示支持智能缩进; 设置 autofocus 为 true, 表示支持自动聚焦。当然如果想设置更丰富的显示效果和功能, 可以添加更多属性和插件。

### 3.6 语法、关键字和内置函数模块的设计与实现

CSPython 编程环境在设计上借鉴了 Brython 的设计思路, 定义了以下 Python 语法:

- (1) 空格将作为语句块的一部分 (注: 因为 Python 中的缩进很重要)。
- (2) 列表用 [] 或者 list() 来创建, 元组用 () 或者 tuple 来创建, 字典用 {} 或 dict() 来创建, 集合用 set() 来创建。

(3) generators(), 例如: `foo(x for x in bar if x>5)`。

(4) 三元操作符: `x = r1 if 条件 else r2`。

(5) 函数可与固定参数、默认值、变量参数和变量关键字参数任意组合:

`def foo(x,y=0,*args,**kw)`。

(6) 参数列表或字典在函数调用中的取值: `x = foo(*args,**kw)`。

(7) 类中的多继承。

(8) decorators (装饰器)。

(9) import:

`import foo`

`from foo import X`

`import foo as bar`

`from foo import X as Y`

`from foo import *`。

CSPython 编程环境支持的 Python3 关键字和内置函数:

- 关键字: as, assert, break, class, continue, def, del, elif, else, except, False, finally,

for, from, global, if, import, is, lambda, None, pass, return, True, try, while, with, yield。

- 内置函数: abs(), all(), any(), ascii(), bin(), bool(), bytes(), callable(), chr(), classmethod(), delattr(), dict(), dir(), divmod(), enumerate(), eval(), exec(), filter(), float(), frozenset(), getattr(), globals(), hasattr(), hash(), hex(), id(), input(), int(), isinstance(), iter(), len(), list(), locals(), map(), max(), min(), next(), object(), open(), ord(), pow(), print(), property(), range(), repr(), reversed(), round(), set(), setattr(), slice(), sorted(), str(), sum(), super(), tuple(), type(), zip(), \_\_import\_\_()。

## 3.7 浏览器接口模块的设计与实现

### 3.7.1 DOM API 介绍

为了与浏览器交互, CSPython 编程环境兼容了 DOM (文档对象模型) 接口。该接口是独立于语言之外的, 所有在 DOM API 中描述的操作都依赖于定义在浏览器模块中的两个对象: document (文档) 和 window (窗口)。

document 实现在 DOM API 中定义的 Document 接口。例如它支持下面的方法:

- document.getElementById(elt\_id)

该方法将返回一个 id 等于 elt\_id 的 DOM 元素。

- document.createElement(tagName)

该方法将返回一个新创建的元素, 例如新建一个超链接:

- link = document.createElement('A')

document.appendChild(elt)

将元素 elt 添加到 document 中。

### 3.7.2 创建一个 document

Brython 是用来开发 Web 应用程序的, 所以用户可以通过 HTML 页面来交互。一个页面是由各种元素 (文本、图像、声音等等) 组成, 这些元素可以通过以下两种方式加载在页面中:

1. 使用标签来编写 HTML 代码

```

<html>

<body>

<b>I</b> love <a href="http://www.Python.org">Python</a>

    for Web browsers

</body>

</html>

```

## 2. 使用内置模块 browser.html 来编写 Python 代码

```

<html>

<body>

<script type="text/Python">

    from browser import document

    from browser.html import A,B

    document <= B("I")+ "love "

    document <= A("Python",href="http://www.Python.org")

</script>

</body>

</html>

```

### 3.7.3 访问元素

可以通过不同的方式来访问一个元素，最常用的方法是使用它的标识符，即它的属性 id，例如一个 input 字段定义为：

- <input id = "data">

这样就可以通过它的 id 来得到这个字段的值：

- from browser import document  
data = document["data"]

document 在浏览器模块中定义并且指向 HTML 文档。它就像一个 dictionary(字典)，它的 keys(键值)就是元素在页面中的标识符。如果元素中没有这个指定的键值，该程



序将会抛出一个 `KeyError` 异常。

相应的也可以得到所有给定类型的元素，例如所有超文本链接，只要使用如下语法：

```
● from browser import html

links = document[html.A]
```

最终，页面中所有的元素都可以通过一个 `get()` 方法来查找：

```
● elt.get(name=N)
```

按降序返回所有 `elt` 中 `name` 等于 `N` 的元素。

```
● elt.get(selector=S)
```

按降序返回所有 `elt` 中 `CSS selector` 等于 `S` 的元素。

3.7.4 元素属性和方法

一个页面中元素的属性和方法跟元素的类型有关，这点在许多 Internet 网站上都可以找到。由于这些属性和方法的名字会因浏览器的不同而略有区别，Brython 定义了许多附加的属性以便可以在任何环境中发挥作用，这些属性如下表所示：

表 3-2 Brython 附加属性一览表

名称	类型	描述	R:只读 R/W:读和写
text	String	在元素中的文本	R/W
html	String	在元素中的 HTML 代码	R/W
Left,top	Integers	该元素相对于页面左上角边界的 位置	R
children	List	该元素在文档树中的子节点	R
parent	DOMNode	该元素的双亲（与文档无关）	R
Class_name	String	元素的类的名字（标签属性： class）	R/W
clear	function	删除该元素的所有后代	L
remove	function	从该元素子节点列表中移出 子节点	R

给元素添加一个子节点，使用 “`<=`” 操作符（可以把它想做向左分配的箭头），示例代码如下：

```
● from browser import document, html

document['zone'] <= html.INPUT(Id="data")
```

一个元素的子节点的迭代可用常用的 Python 语法:

- for child in element:

(...)

要去除一个元素可以使用关键字 “del” :

- zone = document['zone']  
del zone

与一个 SELECT 对象有关联的 options 集合有一个 Python 列表接口(这里的 options 泛指元素或对象) :

- 通过它的索引来访问 option: option = elt.options[index]
- 在该索引位置插入一个 option: elt.options.insert(index,option)
- 在列表的末尾插入一个 option: elt.options.append(option)
- 删除一个 option: del elt.options[index]

### 3.7.5 使用 JavaScript 对象和库

本文通过 Brython 来使用 JavaScript 库和对象。默认情况下, Brython 在全局 JavaScript 命名空间中只有两个名字:

1. brython(): 在页面中加载的 run 函数。
2. \_\_BRYTHON\_\_: 一个 Brython 内部使用的对象,用来存储脚本执行所需的对象。

因此,在默认情况下,一个 JavaScript 程序是不能访问 Brython 对象的。例如,不使用常规的 JavaScript 语法,而是在 Brython 脚本中定义了一个 echo()函数,让它来响应某个元素在页面中的事件,代码如下:

- <button onclick="echo()">

因为 JavaScript 不能直接访问 Brython 定义的 echo()函数,解决办法是给该元素添加一个 id:

- <button id="mybutton">

然后在该元素和(echo()函数对应的)“click”事件之间建立一个链接:

- document['mybutton'].bind('click',echo)

另一种方法是通过在 browser 模块中把 echo 定义为 window 对象的属性，从而强制将它添加到 JavaScript 的命名空间：

- from browser import window  
window.echo = echo

但是不推荐这种方法，因为它增加了在页面中使用定义在 JavaScript 程序或库中的名称的风险。

一个 HTML document 可以使用 JavaScript 脚本或库函数，也可以使用 Python 脚本或库函数，但是 Brython 却不能直接使用 JavaScript 对象，例如，lookup 属性使用 \_\_class\_\_ 属性，但是 JavaScript 对象中却不存在该属性。

要想在 Python 脚本中使用 JavaScript 对象，必须通过 JavaScript 的内置模块 JSObject 转换，示例代码如下：

```
<script type="text/JavaScript">
    circle = {surface:function(r){return 3.14*r*r}}
</script>
<script type="text/Python">
    from browser import document
    from JavaScript import JSObject
    document['result'].value = JSObject(circle).surface(10)
</script>
```

如果一个 JavaScript 函数是一个对象的构造函数，即它可以通过 JavaScript 代码中的关键字 new 来调用，那么通过 JavaScript 的内置模块 JSConstructor() 将该函数转换后，它就可以在 Brython 中使用了，示例代码如下：

```
<script type="text/JavaScript">
function Rectangle(x0,y0,x1,y1){
    this.x0 = x0
    this.y0 = y0
    this.x1 = x1
```

```

    this.y1 = y1

    this.surface = function(){return (x1-x0)*(y1-y0)}
}
</script>
<script type="text/Python">
    from browser import alert
    from JavaScript import JSConstructor
    rectangle = JSConstructor(Rectangle)
    alert(rectangle(10,10,30,30).surface())
</script>

```

### 3.8 Python 标准库加载模块的设计与实现

要完成 Python 模块或包的导入，我们采用与 CPython 相同的机制：为解决“import X”的问题，该程序需要在多个地方查找这个文件，首先会在标准库中查找，这些标准库包括：

1. libs/X.js
2. Lib/X.py
3. Lib/X/\_\_init\_\_.py
4. Lib/site-packages/X.py
5. Lib/site-packages/X/\_\_init\_\_.py

然后，如果这些目录下的文件中都查找不到，那么在包含 X.py 和 X/\_\_init\_\_.py 的目录下的脚本来实现 import 模块。

由于浏览器不能直接访问到文件系统，要查找一个文件必须通过 Ajax 来调用，如果在指定的路径中没有查找到该文件，它将返回一条错误信息。

但是，这种方法对于必须要导入很多模块的脚本来说需要耗费很多时间，比如说要完成“import random”（导入随机函数模块），至少还需要导入关联的 44 个模块。为了提高性能，提出了几个解决方案。

1. 通过在 HTML 页面中引用 py\_VFS.js 文件来加载标准库：

```
<script src="/src/py_VFS.js"></script>
```

在这种情况下，在标准库中的查找就在于检查这个模块是否在这个脚本中被引用。如果引用了这个模块，该程序就找到了这个模块然后运行，而不用再通过一个 Ajax 调用来实现导入。

这种方法的优点是加快了从标准库中导入模块的速度，但它的缺点是 py\_VFS.js 文件非常大（大约 2MB），一旦用户修改了标准库的内容（这种做法不推荐，但可以用来调试），用户必须用 Python 脚本“scripts/make\_VFS.py”来生成一个新的 py\_VFS.js 文件。

2. 将模块映射到 py\_VFS.js 文件，然后将 url 路径保存到表中：

如果表中包含这个模块的名字，系统会到这个指定位置由一个 Ajax 调用来实现模块的导入。

这个方法的唯一缺点是，如果用户修改了模块在标准库中的位置，用户就必须通过 Python 脚本“scripts/make\_dist.py”生成一张新表。

如果要禁用此选项并强制通过 Ajax 调用来查找所有可能的地方，需要调用 brython() 函数，并将 static\_stdlib\_import 选项设置为 false。

需要注意的是除了脚本顶部的编码声明外，所有的模块都要用 utf-8 编码。

### 3.9 代码转换模块的设计与实现

本文的目的是设计并开发出基于 Web 的 Python 编程环境，要解决的关键问题是如何在浏览器中运行 Python 程序代码，目前主流的做法是将 Python 代码转换成 JavaScript 代码。

众所周知，Python 和 JavaScript 都是脚本语言，有许多脚本语言所共有的特性，但是二者却也有着本质的区别，JavaScript 是一种客户端的脚本语言，主要应用于浏览器，而 Python 由于其“优雅”、“明确”、“简单”的设计而广受欢迎，多用于教育、科学计算和 Web 开发。另外，所谓基于 Web 的编程环境，本质上就是指在由浏览器解析的网页中进行 Python 编程。而浏览器最擅长解析的就是 HTML 代码和 JavaScript 脚本

代码，所以将 Python 代码转换成 JavaScript 代码似乎是一种简单可行的方法，但实际上却并不是那么简单。

整理了目前网上将 Python 代码编译为 JavaScript 的工具，如下表所示：

表 3-3 Python 转 JavaScript 工具

名称	项目地址	开发语言
Brython	<a href="http://brython.info/">http://brython.info/</a>	Python3, JavaScript
PythonJS	<a href="https://github.com/PythonJS/PythonJS">https://github.com/PythonJS/PythonJS</a>	Python2
Pyjs	<a href="http://pyjs.org/">http://pyjs.org/</a>	Python2
Skulpt	<a href="http://www.skulpt.org/">http://www.skulpt.org/</a>	Python2, JavaScript

还有很多类似的工具没有一一列出，由上表可以看出，大部分工具都可以实现 Python2 代码转换为 JavaScript 代码，只有 Brython 实现的是 Python3 代码转换为 JavaScript 代码。另外，除了 Brython，其他的工具在转换的过程中都将额外生成许多编译文件，而且单从转换效果来看也不尽如人意，总结起来可以说是精确度不够，性能不高。

基于以上几点，本文采用 Brython 这一开源项目工具，来实现将 Python3 代码转换成 JavaScript 代码，在原有功能的基础上，本文尝试一种新的方法，将 Python 代码加上 script 属性，示例代码如下：

```
function run_Python(editor,result)
{
    var ele = document.getElementById("tempPython");
    if(ele)
    {
        document.body.removeChild(ele);
    }
    //清空控制台
    document.getElementById(result).value = "";
    //添加 import
    var val = "import sys\n";
```

```

    val += "from browser import document,prompt\n"
    val += "def raw_input(data):\n";
    val += "    return prompt(data)\n"
    val += "def write(data):\n";
    val += "    document[''+result+''].value += str(data)";
    val += "\n"+"sys.stdout.write = write";
    val += "\n" + "sys.stderr.write = write";
    //获取 script 值
    val += "\n" + editor.getValue();
    var py_script = document.createElement('script');
    py_script.setAttribute("id","tempPython");
    //将这部分脚本隐藏
    py_script.style.visibility = 'hidden';
    py_script.type = 'text/Python3';
    //设置 Python script
    py_script.textContent = val;
    document.body.appendChild(py_script)
    //运行 brython()函数，查找有 script 标签的 Python 代码
    brython({debug:1,py_tag:'script'});
    //移除 Script
    document.body.removeChild(py_script)
}

```

### 3.10 代码编译运行模块的设计与实现

按照 CSPython 编程环境的设计特点，要在页面中编译并运行 Python 代码，必须在页面中加载 brython()函数，可以通过以下两种方式来添加：

1. 在页面的 body 载入时加载

```
<body onload="brython([options])">
```

2. 将该 `brython()` 函数写入 `x.js` 文件，然后在页面中引入该 `x.js` 文件

```
<script src="...../static/js/x.js"></script>
```

其中 `options` 选项可以是一个 JavaScript 对象，它的值可以是：

- `debug`：它是一个整数，表示调试级别。
- `0`（默认值）：表示没有调试。当使用 `0` 来调试应用程序时，可以稍微加快执行速度。
- `1`：将错误信息打印在浏览器的控制台上（或由 `sys.stderr` 指定的输出流）。
- `2`：将转换成 JavaScript 的 Python 代码输出在浏览器的控制台上。
- `10`：将转换成 JavaScript 的 Python 代码和导入的模块一起输出在浏览器的控制台上。
- `static_stdlib_import`：boolean 类型，表示是否需要使用 `stdlib_paths.js` 脚本中的静态映射表来从标准库中导入模块或包，默认值是 `true`。
- `Pythonpath`：包含被导入模块所在路径的列表。
- `ipy_id`：默认情况下 `brython()` 函数会运行页面中的所有脚本。这个选项指定了一个列表，列表中是一个 `id` 值，就是说 `bython()` 函数会把这个 `id` 对应的文本内容当做 Python 代码来运行，示例代码如下：

```
brython({debug:1, ipy_id:['hello']})
```

该例子中将会运行 `id` 属性为“hello”的元素的内容，并且设置的调试级别为 `1`。

下面详细介绍编译和运行 Python 代码的步骤：

### 1. 读取 Python 代码

该部分由在 `py2js.py` 中的 `brython(debug_mode)` 函数来完成。如果要读取的 Python 代码来自于一个外部文件，它会通过一个 Ajax 调用来查找。这个函数创建的环境变量包括：

- `__BRYTHON__.$py_src`：它是一个通过模块名称来索引的对象，它的值是模块的源代码。
- `__BRYTHON__.debug`：调试级别。



- `__BRYTHON__.exception_stack`: 一个在解析或运行过程中产生的错误列表。
- `__BRYTHON__.imported`: 一个 JavaScript 对象, 将导入的模块名称映射到匹配的模块对象。
- `__BRYTHON__.modules`: 一个 JavaScript 对象, 将模块名称映射到模块对象。
- `__BRYTHON__.vars`: 一个 JavaScript 对象, 将模块名称映射到在模块中定义的变量的路径。

## 2. 构造 Python 代码树

该部分由在 `py2js.py` 中的 `__BRYTHON__.py2js(source,module)` 函数来完成。这个函数将会调用以下方法:

- `$tokenize(<i>source</i>)`: 对在 Python 源代码和树构建过程中的令牌进行语法分析, 返回树的根。
- `transform(<i>root</i>)`: 转换树, 为转换成 JavaScript 做准备。
- `$add_line_num()`: 如果调试级别大于 0, 将添加行号。

`py2js` 函数将返回树根。

## 3. 生成 JavaScript 代码

该部分由 `to_js()` 函数来完成。这个函数将会递归调用查找树中所有同名语法元素的方法, 它将返回包含有转换成 JavaScript 代码后的字符串。如果调试级别是 2, 将会把该字符串打印到浏览器的控制台。

## 4. 运行 JavaScript 代码

该部分由 `eval()` 函数来完成。其中 `brython.js` 脚本是由如下几个脚本文件编译生成的:

- `brython_builtins.js`: 用来定义 `__BRYTHON__` 对象, 它是原生 JavaScript 对象 (`Date`, `RegExp`, `Stotage`.....) 和 Brython 之间的一个桥梁。
- `version_info.js`: Brython 版本信息。
- `py2js.js`: 将 Python 代码转换成 JavaScript 代码。
- `py_utils`: 一些实用的函数 (例如: JavaScript 和 Python 之间的类型转换)。
- `py_object.js`: 实现 Python object 类。

- `py_type.js`: 实现 Python type 类。
- `py_builtin_functions.js`: Python 内置函数。
- `js_objects.js`: JavaScript 对象和构造函数的接口。
- `py_import.js`: 实现 `import`。
- `py_float.js`, `py_int.js`, `py_complex.js`, `py_dict.js`, `py_list.js`, `py_string.js`, `py_set.js`: 实

现匹配 Python 的类。

- `py_dom.js`: 实现与 DOM 的交互。

通过 `py2js.js` 来转换和运行一个 Brython 脚本需要经过以下几个步骤:

### 1. 语法分析和树的构建

Python 代码被分成以下几种类型的令牌:

- keyword
- identifier (string, integer, float)
- operator
- period (.)
- colon (:)
- semi colon (;)
- parenthesis / bracket / curly brace
- assignment (equal sign =)
- decorator (@)
- end of line

每一个令牌都会调用 `$transition()` 函数, 并根据当前状态和令牌返回一个新的状态。

源代码中每一个指令都匹配树中的一个节点 (`$Node` 类的一个实例)。如果一行中包含多个指令 (通常由冒号或分号隔开, 例如: `(def foo(x): return x)` 或 `(x=1; print(x))`), 那么这行中就包含多个节点。

每一个语法元素 (标识符、函数调用、表达式、操作符等) 都由一个类来处理, 其中能被处理的错误包括:

- 语法错误

- 标识符错误
- 为结束的字符串
- 括号不匹配
- 非法字符
- Brython 不能处理的 Python 函数

## 2. 树的转换

对于 Python 语法中的某些元素，代表源代码的树必须在开始转换成 JavaScript 前，通过在树的顶部递归调用 `transform()` 方法来修改（增加分支）。

例如，首先由 `assert_condition_` 判断，如果 Python 代码满足条件则产生一个树的分支，然后如果条件不满足（`if not_condition_`）则用 `raise AssertionError` 增加一个孩子分支。

必须通过这种方式转换的元素包括：`assert`，连等赋值（`x=y=0`）和多值赋值（`x,y=1,2`），`class`，`def`，`except`，`for`，`try`。

其中，这一步也用于存储 `global` 定义的全局变量。

## 3. 运行 JavaScript 代码

生成的脚本在运行时可以使用：

- 定义在 `py_object.js` 中的内置类，`py_dict.js`，`py_string.js`，`py_list.js`，`py_set.js`，`py_dom.js`。
- 不能在 Python 中访问的内置函数（它们的名字通常以“\$”开头），它们大多数都定义在 `$py_utils.js` 中。
- 定义在 `py_import.js` 脚本中的函数。

### 3.11 远程代码分享模块的设计与实现

将编程环境与 Web 相结合，一方面可以借助 Web 平台进行快速开发，另一方面就是实现了远程代码分享。

所谓远程代码分享，指的是通过一定的技术手段，将本地用户编写的程序代码发布在 Web 端，从而可以与更多的用户分享的功能。无论是技术上的瓶颈问题，还是编程

上的心得体会都可以发布出去,这样做一方面可以合理利用网络资源,集群体智慧来解决个人难题,节约时间成本;另一方面好的创意和设计往往需要更多思维的碰撞,分享和交流也因此变得更加有意义。

本文提到的“技术手段”,不同于一般的网络社区,论坛等平台,具体指的是借助 Django 应用框架,当用户使用保存功能时通过 uuid 来生成包含用户代码的唯一页面,用户可以将该页面的链接发给其他人,也就是说其他人通过链接可以直接进入编程环境界面。示例代码如下:

```
def save(request):
    ss = request.REQUEST.get('code_area',"") //获取用户编写的代码
    id = str(uuid.uuid1()) //通过 uuid 来生成唯一 id 值
    outfile = open('/home/usr/data/'+id+'.py','w')
    outfile.write(ss) //将用户代码写入指定目录
    outfile.close() //关闭文件

    return HttpResponseRedirect('.....'+id) //返回包含用户代码的页面
```

在这段代码中,有两个地方保存了用户的代码。第一个地方是服务器中指定的一个目录(例如本例中选取“/home/usr/data/”),所保存的文件已生成的 id 值来命名,可以保证文件的唯一性,当然这样作也有弊端,就是不能区分是哪一个用户的文件,解决办法是建立一个用户名与用户 id 值的映射表,这样就可以一一对应了;第二个地方就是用户保存的包含其编写的代码的页面。

其中,uuid(Universally Unique Identifier)是 128 位的全局唯一标识符,通常由 32 字节的字符串表示。本示例中使用的是基于时间戳的 uuid1(),它是由 MAC 地址、当前时间戳、随机数生成,可以保证全球范围内的唯一性。

### 3.12 本章小结

本章介绍了基于 Web 的 Python 编程环境 CSPython 的设计与实现,着重介绍了几个主要模块的具体实现细节。实现了一个基于 Web 的 Python 编程环境 CSPython,该编程环境拥有最基本的 Python 代码编译和运行功能,支持常用编辑操作,包括剪切、复制、

粘贴、保存等功能，支持智能代码编辑，自动识别关键字、关键字高亮显示、支持快捷键智能缩进，支持常用 Python3 标准库，支持 27 种关键字，58 种内置函数，支持在多种浏览器中运行。

## 第 4 章 基于 Web 的 Python 编程环境测试与评估

系统测试是程序设计开发中不可或缺的一个环节,合理的进行系统测试可以详尽客观的对系统性能进行评价,了解系统真实的状态,还可以及早发现系统的缺陷并迅速进行修正,从而避免因此带来的损失。

本文在前几章中对基于 Web 的 Python 编程环境做了详尽的介绍,设计并实现了 CSPython 编程环境,本章主要对 CSPython 编程环境做详细全面的系统性能测试,旨在全面了解该编程环境的各项性能指标。为了达到测试的目的,本文在认真研究和分析了所需的实验方法和指标的基础上,设计了一个详尽的系统性能测试方案,以用来检验该编程环境在特定的使用环境下,能否达到预期效果。为了保证测试结果具有说服力,本文选取 CSPython, Skulpt 和 repl.it 一同测试,作为比较。最后得出测试结果,结果表明本文设计的 CSPython 编程环境在编程功能上和稳定性上都有不错的表现,并论证了在实际应用中的可行性。

### 4.1 测试目的

1. 测试系统的功能完整性和正确性:这部分主要包括测试 CSPython 编程环境对 Python3 语言主要功能的支持,包括对编译、运行、关键字、语法和内置函数的支持。

2. 测试系统的性能:这部分主要包括测试 CSPython 编程环境在编译运行过程中的时间消耗和资源消耗。在同样的运行环境下,通过对 CSPython 编程环境大规模的访问与编程操作,测试系统响应时间和系统资源占用情况。

### 4.2 实验环境

本文根据 CSPython 编程环境的具体特点,设计了针对该编程环境的测试实验环境。本文采用一台戴尔服务器来对 CSPython 编程环境进行系统的功能完整性,系统正确性和系统性能进行测试,服务器具体硬件参数如下表所示:

表 4-1 戴尔服务器系统环境

参数名称	参数属性
型号	Poweredge T110 II(E3-1220/4G/2×500G)
处理器	Xeon E3-1220 3.10GHz
内存	4G
硬盘	2×500G 7200 转
操作系统	Linux RedHat 内核 2.6.18-194.el5, x86_64
编译器	Python 3.4.0

4.3 编程环境功能测试

基于 Web 的 Python 编程环境是一个编译和运行 Python 程序代码的软件工具，它对 Python 代码进行有效地编辑、运行、保存与维护，减少用户对本地编程环境的依赖，为用户提供方便、快捷的编程体验。

本文通过对 CSPython、Skulpt 和 repl.it 三个 Python 编程环境进行系统功能测试，以此来对它们进行更细致的比较。三种编程环境的功能测试结果如下表：

表 4-2 三种编程环境的功能测试结果

功能	CSPython	Skulpt	repl.it
新建、打开、关闭、删除 Python 文件	支持	不支持	支持
关键字高亮显示	支持	支持	支持
智能缩进	支持	支持	支持
显示行号	支持	支持	支持
语法纠错	支持	支持	支持
Python 版本	Python3	Python2	Python2
关键字	支持	部分支持	部分支持
Python 标准库	支持 Python3	支持 Python2	支持 Python2
内置函数	支持	不支持	不支持

4.4 编程环境性能实验设计

之所以开发出基于 Web 的 Python 编程环境，就是要为了使用户脱离本地开发环境，直通过浏览器在网页中编程。但是只有功能上的完整性和正确性还不够，还需要该编程环境能对用户的请求作出快速的响应。本小结的主要任务是设计一套测试方案来系统的测试每个编程环境的系统性能。

测试内容：主要采用断言的方式对 Python 的主要语言功能进行测试，具体包括内

置类，重载，字节，类，小数，装饰器，描述器，字典，import，迭代，列表，数字，集合，字符串，字符串转换，字符串函数。

测试方法：通过多个用户在同一时间段内对待测网站进行并发访问，同时运行测试内容，由此算出系统总体用时。本文采用 ApacheBench（Apache 旗下的一个测试工具，简称 ab）对该系统进行测试。该测试工具可以同时模拟多个并发请求来对服务器进行测试，用命令执行示例如下：

```
ab -n 1000 -c 100 http://localhost:8000/index.html/
```

其中 -n 100 代表总的请求数，-c 10 代表每次发送的请求数，上面这个命令的意思就是启动 ab，向 localhost:8000/index.html/ 发送 1000 个请求，并每次发送 100 个请求——也就是说总共要发送 10 次。

## 4.5 实验结果分析及讨论

由于本实验是三组对比测试实验，为保证每组实验在测试过程中不受外界和其它组实验的干扰，需要做如下准备工作：

1. 确保服务器正常运行且只运行本次实验内容。
2. 及时清除运行缓存，保障每次测试的初始条件相同。

本文在保证满足以上条件的情况下，对三种编程环境进行性能测试，考虑到多用户高并发访问的性能问题，本编程环境测试的总访问量为 3000，每次并发访问数为 100。测试结果如下图所示。其中横坐标表示并发访问次数，纵坐标表示所用时间，单位毫秒。



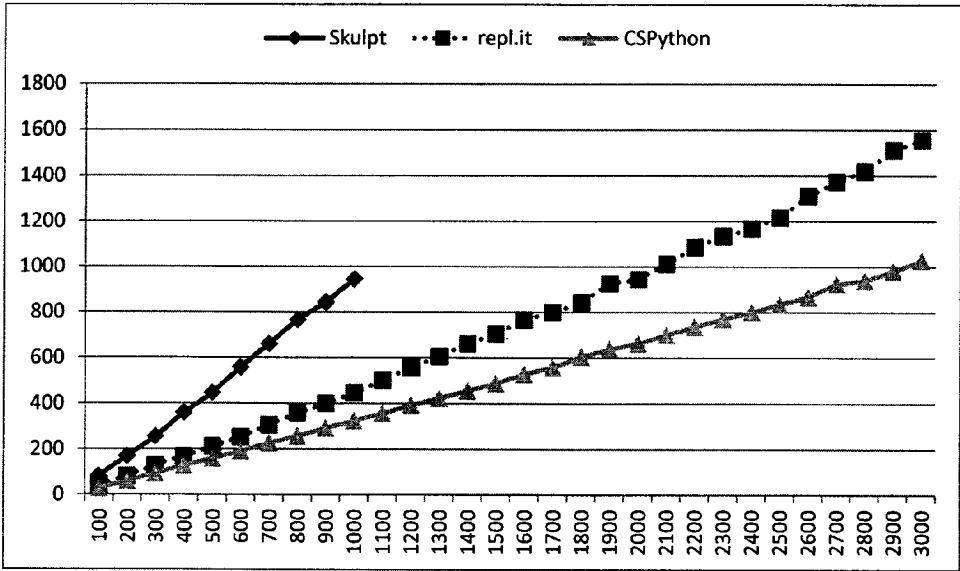


图 4-1 三种编程环境性能测试结果图

由上图可以看出，本文设计的 CSPython 编程环境在性能上远高于 Skulpt，略高于 repl.it。Skulpt 在并发量达到 1000 以上时已经无法访问，而 CSPython 和 repl.it 在 3000 并发量范围内基本可以流畅运行，具体分析有以下几个主要原因。

第一，CSPython 使用 Django 框架搭建，配合高性能的 Apache 服务器，使得本平台在 Linux 系统上运行的更加流畅。

第二，CSPython 使用 DOM 接口。为了与浏览器更好的交互，CSPython 编程环境兼容了 DOM（文档对象模型）接口，通过该接口可以对 HTML 文档中所有元素进行访问。

第三，CSPython 将模块写入 JS 文件，加快导入速度。比起在文件目录中查找模块，CSPython 通过查找保存模块映射的表，以提高开发效率。

由此可以得出结论，CSPython 编程环境在并发测试中有较好的表现，而且 CSPython 体积轻便，功能齐全，实用灵活，运行稳定，适合在中小型规模的程序开发中使用。

4.6 本章小结

本章设计了一系列测试方案，对本文设计的基于 Web 的 Python 编程环境 CSPython

进行了详细的测试，通过三组性能对比实验可知，CSPython 无论在功能的完整性还是并发访问的性能上都有较好的表现，适合在实际程序开发中使用。

## 第 5 章 结论与展望

### 5.1 研究结论

随着互联网的飞速发展，越来越多的开发正在从本地转移到网络，而基于 Web 的 Python 编程环境也成为国内外研究的热点。基于 Web 的 Python 编程环境以其独特的运行特点和强大的运行功能为 Python 程序开发提供了更多的便利，这也成为未来程序项目开发的趋势。本文着眼于当下，通过认真研究前人的成果，仔细翻阅国内外资料，设计并实现了一个基于 Web 的 Python 编程环境 CSPython，然后详细的介绍了 CSPython 各个模块的功能和实现细节，最后设计了一套用于测试其功能完整性与运行性能的方案，证实了该编程环境在实际应用中的价值。

本文具体研究内容包括：

(1) 系统的研究了基于 Web 的编程环境、基于 Web 的 Python 编程环境还有开源编程环境在国内外中的发展现状，分析了基于 Web 的 Python 编程环境在理论研究和实际应用中的巨大发展前景，阐述了基于 Web 的 Python 编程环境的技术特点和实现原理。

(2) 详细的介绍了两个主流的基于 Web 的 Python 编程环境 Skulpt 和 repl.it。总结了这两个编程环境的具体环境要求，一般使用方法，包含的技术特点，当然还有个别需要提高和改进的地方。

(3) 具体的介绍了基于 Web 的 Python 编程环境 CSPython 的 9 大模块，介绍了这些模块的主要功能，包含的算法和实现的细节。在这 9 个模块中，重点介绍了浏览器接口模块，Python 标准库加载模块，代码转换模块和代码编译运行模块。实现了通过浏览器接口来访问 HTML 页面中的元素，采用将 Python 代码转换为 JavaScript 脚本代码的方法，并针对该方法作出了改进，极大的加快了程序运行的速度，提高了效率，从而有效地降低了开发成本。

(4) 经过认真分析 CSPython 编程环境的特点，有针对性的设计了一套完整的测试方案，期间选取另外两个编程环境做对比试验。

研究表明，CSPython 编程环境的运行环境非常宽泛，包括台式机，手提电脑，

Pad 等，通过浏览器都可以运行。具体运行环境包括：

操作系统：Windows XP 及以上，Linux Ubuntu/RedHat/Fedora（桌面版），Mac OS X 10.0 及以上。

浏览器：Firefox 4.0 及以上，Chrome 所有版本，Safari 5.2 及以上，Opera 9.0 及以上。

另外，CSPython 编程环境还具有以下优点：

简单。该编程环境运行门槛低，只需要一款浏览器即可；操作简单，拥有 Python 编程所需所有功能；界面布局简洁，所有功能一目了然。

高效。该编程环境包含一款出色的代码编辑器，内置多种代码编辑功能，能对 Python 代码做出快速响应；编译方面使用将 Python 代码转换成 JavaScript 脚本代码，支持多种内置函数；支持一键分享功能，可快速的将用户编写的代码保存并分享。

稳定。该编程环境采用 Django 框架搭建，它具有完善的 API 文档，提供全套 Web 开发所需的工具，URL 配置和自带的后台管理功能都非常强大，配置性能优异的 Apache 服务器，为用户提供稳定地 Web 开发服务。

综上所述，本文设计并实现的 CSPython 编程环境无论在功能完整性还是运行性能上都具有令人满意的表现，并适合在实际应用中推广。

## 5.2 研究展望

本文设计并实现了基于 Web 的 Python 编程环境 CSPython，实现了通过浏览器在 HTML 页面中运行 Python 代码的功能。但由于时间关系，该编程环境还有许多需要改进和完善的地方。

- （1）对代码转换模块可以进一步改进，以解决在大规模数据中运行效率问题。
- （2）对编程环境的功能可以进一步扩展，比如说增加对图形模块的支持，以满足更广泛的用户需求。
- （3）对代码编译模块可以进一步简化，用更简明方便的算法来提高编译效率。
- （4）继续研究其它相关技术的实现方法，为改进和完善该编程环境做准备。

## 参考文献

- [1] 中国互联网络信息中心(CNNIC).第 35 次中国互联网络发展状况统计报告[R].北京: 2015.
- [2] Philip Guo. Python is Now the Most Popular Introductory Teaching Language at Top U.S. Universities[J/OL].[2014-07-07].<http://cacm.acm.org/blogs/blog-cacm/176450-Python-is-now-the-most-popular-introductory-teaching-language-at-top-us-universities/fulltext>.
- [3] ARCOS J L, ESTEVA M, NORIEGA P, et al. An integrated development environment for electronic institutions [M]. Software agent-based applications, platforms and development kits. Springer. 2005: 121-142.
- [4] CHAFLE G, DAS G, DASGUPTA K, et al. An integrated development environment for Web service composition; proceedings of the Web Services, 2007 ICWS 2007 IEEE International Conference on, F, 2007 [C]. IEEE.
- [5] FEBBRARO O, REALE K, RICCA F. ASPIDE: Integrated development environment for answer set programming [M]. Logic Programming and Nonmonotonic Reasoning. Springer. 2011: 317-330.
- [6] GUHR J T, PICONI J. Integrated development environment for high speed transaction processing WWW applications on heterogeneous computer systems [M]. Google Patents. 2003.
- [7] KERRIGAN M, MOCAN A, TANLER M, et al. The Web service modeling toolkit-an integrated development environment for semantic Web services [M]. The Semantic Web: Research and Applications. Springer. 2007: 789-798.
- [8] KOMPALLI P, PAULY H, FOERG K-H. Rapid application integration using an integrated development environment [M]. Google Patents. 2007.
- [9] KONSYSKI B R, KOTTEMANN J E, NUNAMAKER JR J F, et al. PLEXSYS-84: An integrated development environment for information systems [J]. Journal of Management Information Systems, 1984, 64-104.
- [10] RASURE J R, WILLIAMS C S. An integrated data flow visual language and software development environment [J]. Journal of Visual Languages & Computing, 1991, 2(3): 217-246.
- [11] TISSERANT E, BESSARD L, DE SOUSA M. An open source IEC 61131-3 integrated development environment; proceedings of the Industrial Informatics, 2007 5th IEEE International Conference on, F, 2007 [C]. IEEE.
- [12] ZATLOUKAL K, WAGNER T A. System and method for performing code completion in an integrated development environment [M]. Google Patents. 2007.
- [13] ROSSUM G V. Python programming language [J]. URL <http://www.Python.org>, 1989.
- [14] LICENSE G G P. GNU General Public License [M]. Qt. 1989.
- [15] VAN ROSSUM G, DRAKE F L. Python language reference manual [M]. Network Theory, 2003.
- [16] DOWNEY A. Think Python [M]. " O'Reilly Media, Inc.", 2012.
- [17] ZELLE J M. Python programming: an introduction to computer science [M]. Franklin, Beedle & Associates, Inc., 2004.
- [18] OLIPHANT T E. Python for scientific computing [J]. Computing in Science & Engineering, 2007, 9(3): 10-20.
- [19] PEDREGOSA F, VAROQUAUX G, GRAMFORT A, et al. Scikit-learn: Machine learning in Python [J]. The Journal of Machine Learning Research, 2011, 12:2825-2830.
- [20] SANNER M F. Python: a programming language for software integration and development [J]. J Mol

- Graph Model, 1999, 17(1): 57-61.
- [21] COCK P J, ANTAO T, CHANG J T, et al. BioPython: freely available Python tools for computational molecular biology and bioinformatics [J]. *Bioinformatics*, 2009, 25(11): 1422-1423.
- [22] BIRD S, KLEIN E, LOPER E. Natural language processing with Python [M]. "O'Reilly Media, Inc.", 2009.
- [23] EICH B. Netscape Navigator [J]. Netscape Navigation Corporation, 1996, 24.
- [24] EICH B. JavaScript at ten years [J]. *ACM SIGPLAN Notices*, 2005, 40(9): 129-129.
- [25] EICH B, CLARY B. JavaScript Language Resources [M]. 2003.
- [26] EICH B. JavaScript engine [M]. Google Patents. 2005.
- [27] CROCKFORD D. JavaScript: The Good Parts: The Good Parts [M]. "O'Reilly Media, Inc.", 2008.
- [28] CROCKFORD D. JavaScript: The world's most misunderstood programming language [J]. Douglas Crockford's JavaScript, 2001.
- [29] CROCKFORD D. Json [M]. Technical report, RFC 4627, 2006, 2006. URL <http://www.ietf.org/rfc/rfc4627.txt>. 2012.
- [30] CROCKFORD D. ADsafe: Making JavaScript safe for advertising [M]. 2008.
- [31] CROCKFORD D. JSON in JavaScript [J]. GitHub Social Coding, 2010.
- [32] CROCKFORD D. The application/json media type for JavaScript object notation (json) [J]. 2006.
- [33] SUPAARTAGORN C. PHP Framework for database management based on MVC pattern [J]. Department of Mathematics Statistics and Computer, Ubon Ratchathani University, Thailand, 2011.
- [34] 杨彦侃, 谭心. 一种基于 CodeIgniter 框架科研论文管理系统的研究与实现 [J]. *计算机应用与软件*, 2012, 12: 223-224+251.
- [35] 王映, 于满泉, 李盛韬, et al. JavaScript 引擎在动态网页采集技术中的应用 [J]. *计算机应用*, 2004, 24(2): 33-36.
- [36] 周林, 步丰林. 嵌入式浏览器中 JavaScript 和 DOM 的支持 [J]. *计算机工程*, 2005, 30(B12): 114-117.
- [37] 曹宇, 陈海峰. 基于 JSON, JavaScript, HTML5 和前端存储技术的均衡运算框架 [J]. *实验室研究与探索*, 2014, 33(5): 116-119.
- [38] FORCIER J, BISSEX P, CHUN W. Python Web development with Django [M]. Addison-Wesley Professional, 2008.
- [39] 范文星. 基于 Django 的网络运维管理系统的设计与实现 [J]. *计算机科学*, 2012, S2: 175-177.
- [40] DIPIERRO M. Web2py Enterprise Web Framework [M]. Wiley Publishing, 2009.
- [41] 王淮, 夏阳. 基于 Ruby on Rails 的 WEB 开发新技术 [J]. *微计算机信息*, 2007, 30: 218-220.
- [42] 张辉, 陈祖爵, 于建江. 面向 Ruby on Rails 的多类型 RIA 集成技术研究 [J]. *计算机工程与设计*, 2010, 9: 1912-1915+2123.
- [43] 张帆, 刘刚. 基于 .NET 的农业生产环境信息监测系统 [J]. *计算机工程与设计*, 2013, 34(2): 696-701.
- [44] 赵伟, 王志华, 周兵. 基于 .NET 技术和 MVC 的新架构模式 [J]. *计算机工程与设计*, 2012, 33(7): 2646-2651.
- [45] 史国滨, 王熙. 基于 ASP.NET 的农机监控 WebGIS 系统性能优化 [J]. *安徽农业科学*, 2011, 39(5): 2821-2823.
- [46] 廖福保. 扩展 SpringMVC 模块的 Web 应用 [J]. *实验室研究与探索*, 2012, 10: 70-73.
- [47] 宇 张, 王映辉, 张翔南. 基于 Spring 的 MVC 框架设计与实现 [J]. *计算机工程*, 2010, 36(4): 59-62.
- [48] 陈辉, 赵洪升, 张艳春. Struts+ Spring+ Hibernate 框架的整合实现 [J]. *河南大学学报: 自然科学*

版, 2010, 06: 642-645.

[49] MARIJN H. CodeMirror [OL]. [2014-08-01]. <http://codemirror.net>.

[50] SCOTT G. Skulpt [OL]. [2014-08-01]. <http://skulpt.org>.

[51] AMJAD M. repl.it [OL]. [2014-08-01]. <http://repl.it>.

[52] PIERRE Q. Brython [OL]. [2014-08-01]. <http://brython.info>.

## 致谢

时光荏苒，岁月如梭。转眼间，三年的研究生求学之旅即将结束，坐在我平时学习和生活过的实验室里，无论是奋斗的辛酸还是收获的喜悦都成为永久值得怀念的记忆。新疆农业大学以其优良的学习氛围、严谨的治学方针和丰富多彩的校园生活教我做人，伴我成长，值此论文完成之际，我要向所有关心、爱护和帮助过我的人们表示最诚挚的感谢和最美好的祝愿。

首先，要感谢的是我的导师张太红教授，本论文是在张老师的悉心指导下完成的。三年来张老师渊博的专业知识，严谨的治学态度，诲人不倦的高尚师德对我影响深远。张老师不仅在课堂上给予我和我的同门们帮助与指导，还鼓励我们到社会上去历练，积累宝贵的实践经验。本论文从选提到完成，每一步都离不开张老师的指导，在此我向张老师表示衷心的感谢。

其次，我要感谢学院领导艾尔肯书记、侯涛书记、古丽米拉院长、冯向萍院长、教学秘书靳凤梅老师、杨莹老师，魏星老师、郭斌老师、陈艳红老师、白涛老师、杨舒老师、寇晓斌老师、吴乃宁老师，感谢他们在学习和生活等各方面给予我的关心和帮助。

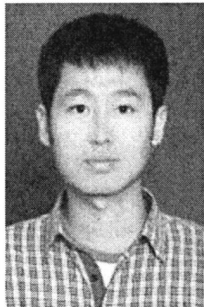
然后，我要感谢时亚南师兄、赵涛师姐，同时也感谢这三年来与我共同奋斗过的同学们，是他们让我的生活充满了阳光，学习充满了乐趣，能成为这个团队的一员是我的荣幸。

接着，我要感谢我的父母，他们的养育之恩我永远铭记于心；我要感谢我的哥哥，他一直是我的榜样。

最后，我要感谢参与我论文查阅和评审的老师们，感谢你们辛勤的劳动，再次送上我真挚的祝福，谢谢你们。



作者简历

姓名	刘志凯	性别	男	民族	汉	
籍贯	黑龙江宝清	政治面貌	中共党员	出生年月	1990. 2	
专业	农业机械化工程					
研究方向	数据库技术					
英语水平	国家六级			学历	硕士	
毕业院校	新疆农业大学			毕业时间	2015 年 7 月	
个人简历	2008 年 9 月—2012 年 6 月: 就读于新疆农业大学计算机与信息工程学院信息管理与信息系统。 2012 年 9 月—2015 年 7 月: 就读于新疆农业大学农业机械化工程专业数据库技术方向研究生。					
专业实践	(1)2012 年 6 月-2012 年 9 月 参加了计算机学院 Java 课程改革培训学习, 参与了部分试题命制工作。 (2) 2012 年 10 月-2012 年 9 月 跟随导师进行课程改革, 参与了导师《数据库及 Java Web 应用程序》课程作业的部分设计工作。 (3)2013 年 4 月-2013 年 6 月 参加并完成了张太红老师布置的《数据库实现》课程作业: “小型数据库 UBase 项目的开发与实现”。 (4) 2013 年 6 月-2013 年 7 月 参加了计算机学院 C 语言课程改革培训学习, 参与了部分试题命制工作。					
发表论文	[1] 刘志凯, 张太红. Django 框架在 Web 开发中的应用[J]. 农业网络信息. 已收录。 [2] 刘志凯, 张太红, 刘磊, 罗鹏. 基于 Web 的 Python3 编程环境[J]. 计算机系统应用. 已录用。					