Практикум по алгоритмизации и программированию на Python

Иван Хахаев, 2009

Методические указания для учителей и преподавателей.

Введение. Почему Python?

Выбор в пользу того или иного языка программирования является следствием огромного количества факторов — от требований эффективного использования ресурсов вычислительной системы до наличия в нужное время подходящей книжки.

Поэтому, чтобы избежать непродуктивного и спорного сравнения различных языков программирования друг с другом (к тому же, такое сравнение провести крайне трудно ввиду их огромного количества и разнообразия параметров сравнения), рассмотрим только аргументы в пользу выбора языка Python (общепринятое произношение — «Питòн», хотя допускается и «Пàйтон»).

1. Python — сравнительно «молодой» язык. Создавая его в 1990-1991 годах, его автор Гвидо ван Россум (Guido van Rossum) учёл все достоинства и недостатки предшествующих языков программирования.

2. Python имеет достаточно долгую историю развития и использования (почти 20 лет). В настоящее время Python поддерживается обширным международным сообществом разработчиков.

3. Python — развивающийся язык, используемый в реальных проектах. Это означает, что его изучение не пройдёт напрасно.

4. Средства для работы с Python относятся к категории свободно распространяемого программного обеспечения (СПО). Это гарантирует во-первых, от каких-либо претензий относительно использования «интеллектуальной собственности», а во-вторых, от превращения Python в обозримом будущем в «мёртвый» язык (вспомните про «популярный» Turbo Pascal).

5. Python имеет обширную область применения. Так, на Python создаются расширения к графическому редактору GIMP, на Python можно программировать в офисном пакете OpenOffice.org, на Python пишутся сценарии для пакета 3D-моделирования Blender, на Python написаны системы управления контентом Plone и MoinMoin Wiki, Python активно используется при создании компьютерных игр.

6. Python — интерпретируемый язык, что очень удобно при обучении программированию. Интерпретатор Python входит в большинство дистрибутивов GNU/Linux (и разумеется, в ПСПО для школ).

7. Существует множество средств, облегчающих процесс создания программ на Python. Это и специализированные лексические анализаторы, и редакторы для программистов (например, Kate и Bluefish), и интегрированные среды разработки (IDE).

8. Наконец, многие средства для работы с Python являются кросс-платформенными, а в конструкциях языка поддерживаются многобайтные кодировки (Unicode), поэтому программы на Python легко переносятся с одной среды функционирования на другую.

Требования к программной конфигурации.

Для успешного проведения практикума по алгоритмизации и программированию на Python на рабочих местах должны быть установлены собственно Python (версия не ниже 2.4), библиотеки Tkinter и NumPy, среды разработки на Python — IDLE, Eric или Geany, а также какие-либо эмуляторы терминалов — xterm, rxvt и т.п.

В сборке от ALT Linux следует проверить наличие в системе следующих пакетов

- geany
- eric
- xterm
- python
- python-base
- python-doc
- python-module-numpy
- python-modules
- python-modules-encodings
- python-modules-tkinter
- python-tools-idle

(Какие-то пакеты будут установлены по зависимостям при установке Python, Eric и Geany с помощью менеджера пакетов, остальные нужно установить «вручную».)

При создании программ (этот процесс обозначается звучным словом «разработка») удобно одновременно видеть текст программы и результаты её выполнения. Хорошо также, если при этом по-разному выделяются ключевые слова, названия функций и их аргументы, а также сразу же показываются строки, содержащие ошибки. Кроме того, бывает полезно выполнять программу по шагам и при этом следить за значениями каких-то переменных. Все эти возможности реализуются в так называемых «Интегрированных средах разработки» - Integrated Development Environment (IDE).

Современные IDE, входящие в дистрибутивы Linux, могут работать с разными языками программирования. Существует IDE, лучше всего приспособленные для работы с одним конкретным языком, которые с другими языками работают, так

сказать, «факультативно». Кроме того, существуют IDE, которые одинаково успешно обеспечивают работу с самыми разными языками, как в режиме интерпретатора, так и в режиме компилятора.

В зависимости от версии ALT Linux удобно пользоваться либо Geany, либо Eric. Далее будут рассмотрены особенности работы в обоих IDE.

Основные понятия и определения (глоссарий)

В дальнейшем часто придётся использовать термины, связанные с процессом разработки и функционирования программ, поэтому здесь приведён краткий «словарик», чтобы уже не возвращаться к проблеме определений. Для составления этого «словаря» использованы в основном материалы сайта «Глоссарий.Ру» (http://glossary.ru).

Алгоритм

Алгоритм — точное предписание исполнителю совершить определённую последовательность действий для достижения поставленной цели за конечное число шагов.

Данные

Данные — сведения:

- полученные путём измерения, наблюдения, логических или арифметических операций; и
- представленные в форме, пригодной для постоянного хранения, передачи и (автоматизированной) обработки.

Тип данных

Тип данных — характеристика набора данных, которая определяет:

- диапазон возможных значений данных из набора;
- допустимые операции, которые можно выполнять над этими значениями;
- способ хранения этих значений в памяти.

Различают:

- простые типы данных: целые, действительные числа, символы, строки, логические величины;
- составные типы данных: массивы, файлы и др.

Программа

Программа — согласно ГОСТ 19781-90 — данные, предназначенные для

управления конкретными компонентами системы обработки информации в целях реализации определённого алгоритма.

Алгоритмический язык (язык программирования)

Язык программирования — искусственный (формальный) язык, предназначенный для записи алгоритмов. Язык программирования задаётся своим описанием и реализуется в виде специальной программы: компилятора или интерпретатора.

Транслятор языка программирования

Транслятор — в широком смысле — программа, преобразующая текст, написанный на одном языке, в текст на другом языке.

Транслятор — в узком смысле — программа, преобразующая: программу, написанную на одном (входном) языке в программу, представленную на другом (выходном) языке.

Транслятор языка программирования — программа, преобразующая *исходный текст* программы на языке программирования в *машинный язык* вычислительной системы, на которой эта программ должна выполняться.

Интерпретатор

Интерпретатор — транслятор, способный параллельно переводить и выполнять программу, написанную на алгоритмическом языке высокого уровня.

Компилятор

Компилятор — программа, преобразующая текст, написанный на алгоритмическом языке, в программу, состоящую из машинных команд. Компилятор создаёт законченный вариант программы на машинном языке.

Константа

Константа — в программировании — элемент данных, который занимает место в памяти, имеет имя и определённый тип, причём его значение никогда не меняется.

Переменная

Переменная — в языках программирования — именованная часть памяти, в которую могут помещаться разные значения переменной. Причём в каждый момент времени переменная имеет единственное значение (или единственный набор значений). В процессе выполнения программы значение переменной может изменяться.

Тип переменных определяется типом данных, которые они представляют.

Подпрограмма

Подпрограмма — самостоятельная часть программы, которая разрабатывается независимо от других частей и затем вызывается по имени.

Функция

Подпрограмма, которая на основе некоторых данных (аргументов функции) вычисляет значение некоторой переменной («функция возвращает значение»).

Идентификатор

Идентификатор — символическое имя переменной или подпрограммы, которые однозначно идентифицируют их в программе.

Выражение

Выражение — конструкция на языке программирования, предназначенная для выполнения вычислений. Выражение состоит из операндов, объединённых знаками операций. Различают арифметические, логические и символьные выражения.

Операнд

Операнд — константа, переменная, функция, выражение и другой объект языка программирования, над которым производятся операции.

Арифметическая операция

Арифметическая операция — вычислительная операция над числами.

Во многих языках программирования определены двуместные арифметические операции: сложения, вычитания, умножения, деления, деления нацело, вычисление остатка от деления.

Логическая операция

Логическая операция — операция над логическими («булевскими») операндами, принимающими значения «Истина» или «Ложь». Наиболее распространёнными являются следующие операции:

- многоместное логическое сложение;
- многоместное логическое умножение;
- одноместное логическое отрицание.

(«Многоместная» операция означает, что в ней может два и более

операндов, а в «одноместной» или «унарной» операции участвует только один операнд).

Операция отношения

Операция отношения производит сравнение двух величин. Результат операции отношения является «булевской» переменной, принимающей значение «Истина» (True или логическая 1) или «Ложь» (False или логический 0).

Массив (массив данных)

Совокупность, как правило, однотипных данных, каждое из которых идентифицируется с именем массива и индексом (индексами).

В зависимости от количества индексов массивы бывают одномерные (линейные), двухмерные и т.д.

Индекс

Номер (или номера, если массив данных многомерный), добавляемый к имени массива, чтобы идентифицировать каждый элемент данного массива.

Например, `a[1,3]` означает, что определён элемент двухмерного массива `a` с индексом 1,3 (строка — 1, столбец — 3).

Присваивание

Операция записи значения в переменную. В каждом языке программирования определён *оператор присваивания*. Если в переменную записывается новое значение, старое стирается.

Цикл

Цикл (циклические вычисления) означают многократное выполнение одних и тех же операций. В зависимости от задачи различаются циклы с переменной (со счётчиком, с известным количеством повторений) и циклы с условием (цикл повторяется, пока не выполнится условие завершение цикла).

Зацикливание

Для циклов с условием — ситуация, при которой условие завершения цикла никогда не выполняется.

Использование IDE Geany.

Один из вариантов IDE для работы с Python – кросс-платформенный пакет

Geany, который можно запустить с помощью K-меню («K-меню/Разработка/Geany» или «K-меню/Прочее/Разработка/Geany»). Внешний вид окна Geany при первом запуске показан на рис. 1.

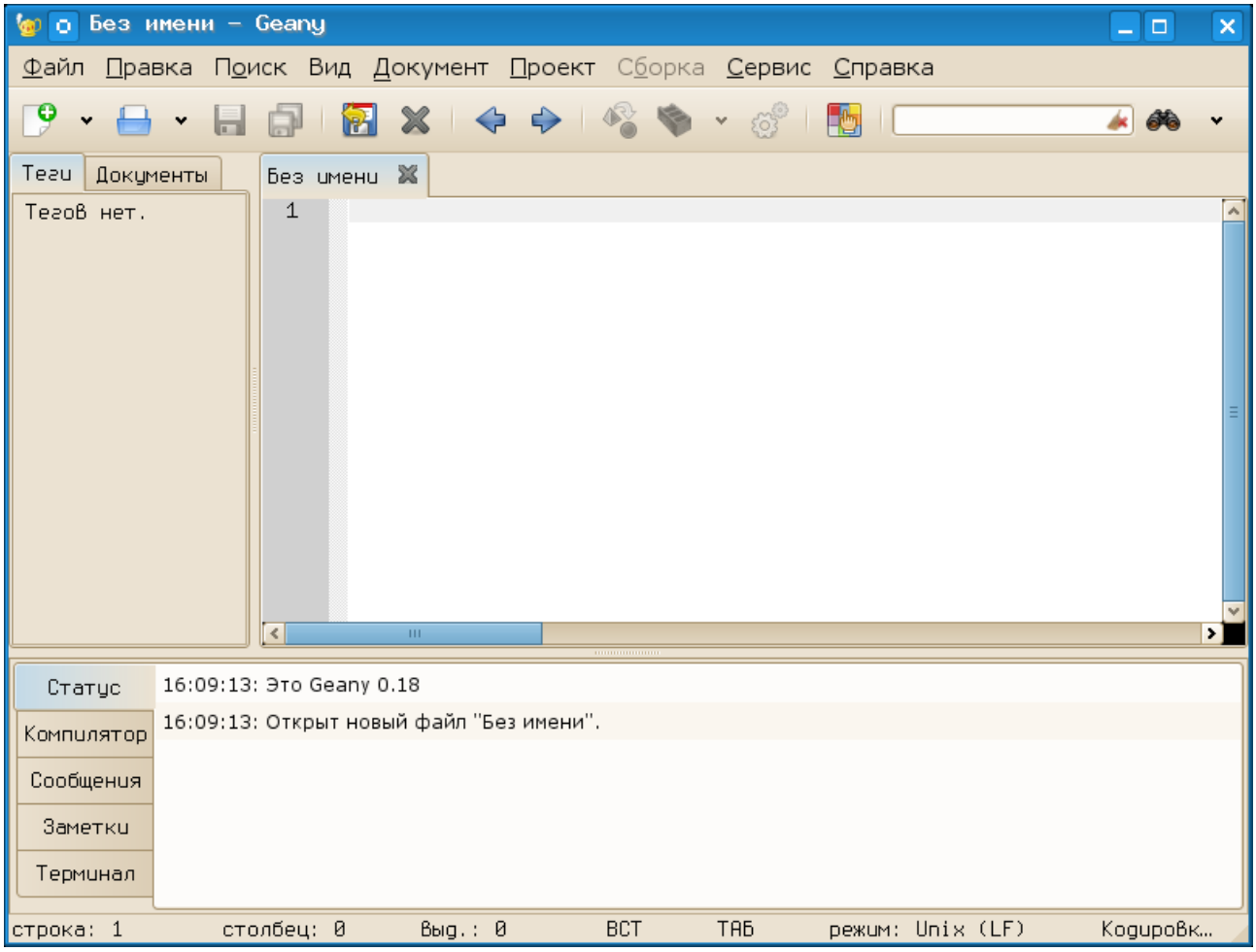

Рисунок 1. Внешний вид IDE «Geany»

Центральная часть окна предназначена для размещения вкладок с текстами программ, в нижней части размещается несколько вкладок. Некоторые из этих вкладок будут полезны в дальнейшем, а некоторые при работе с Python (например, вкладка «Компилятор») не нужны. В панели инструментов часть кнопок недоступна (кнопки имеют серый цвет). Пока нет текста программы, для этих кнопок «нет работы».

В нижней части окна находится несколько вкладок. Вкладка «Статус» фиксирует историю открытия и закрытия файлов с текстами программ, а на вкладке «Компилятор» отображаются сообщения компилятора при компиляции исходного текста (компиляция запускается кнопкой «Скомпилировать текущий файл» в панели инструментов). Понятно, что вкладка «Компилятор» требуется

только для работы с компилируемыми программами (C, Pascal и т. п.).

Вкладка «Заметки» может использоваться как своего рода «блокнот» для записи каких-то замечаний и пожеланий, а вкладка «Терминал» является встроенным в Geany эмулятором терминала (Virtual terminal Emulator – VTE). Однако в дальнейшем будет использоваться «внешний эмулятор терминала – xterm.

Первоначальная настройка

Первоначальная настройка полезна для создания комфортной «среды обитания» в IDE. Разумеется, в каждом конкретном случае важны личные предпочтения, а здесь рассмотрим параметры настройки и разумные значения параметров для работы с Python в рамках «Практикума».

Для вызова диалога настроек используется команда главного меню «Правка/ Настройки» или комбинация клавиш <CTRL>+<ALT>+<P>. Диалог содержит несколько вкладок (рис. 2) и начинается всё с общих настроек.

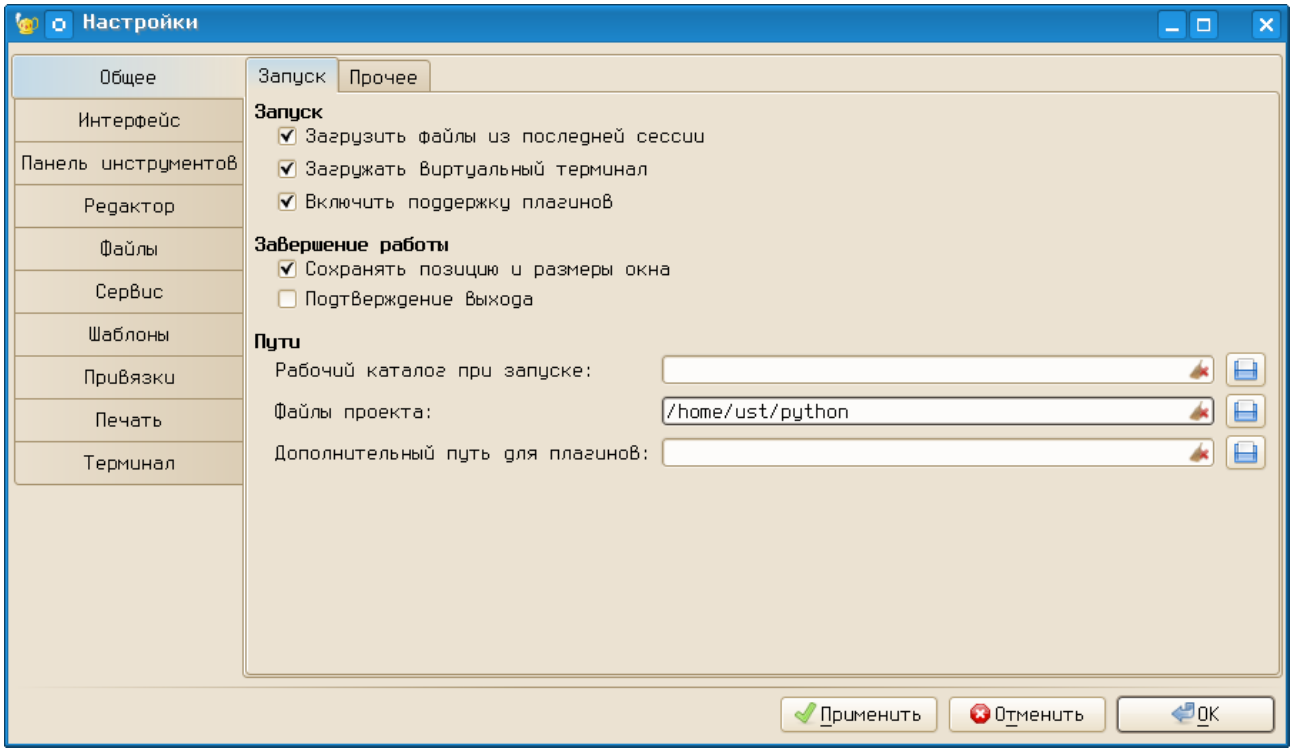

Рисунок 2. Диалог настройки Geany — общие настройки

На вкладке «Общие» диалога настройки Geany имеет смысл установить режимы «Загрузить файлы из последней сессии» и «Сохранять позицию и размеры окна». Что касается режима «Загружать виртуальный терминал» то при использовании внешнего эмулятора терминала (в нашем случае – xterm) он не

нужен. Отключение этого режима приведёт к исчезновению вкладки «Терминал» в нижней части окна Geany, а также к исчезновению вкладки «Терминал» в диалоге настроек Geany при его последующем вызове.

Режим «Включить поддержку плагинов» нужен для автоматического вызова интерпретаторов и компиляторов языков программирования. Если для плагинов не указан дополнительный путь, Geany будет использовать пути из системы. Рабочий каталог и каталог для файлов проекта можно установить (набрав путь в строке ввода или нажав на кнопку «Открыть папку» справа от строки ввода), но это не обязательно при включённом режиме Загрузить файлы из последней сессии».

За внешний вид Geany отвечают три вкладки диалога настроек – «Интерфейс», «Панель инструментов» и «Редактор». На рис. 3 показана вкладка «Интерфейс».

На вкладке «Интерфейс» имеет смысл только настроить шрифты для области редактора и других панелей. Диалог выбора шрифтов (рис. 4) вызывается нажатием на кнопку с названием шрифта.

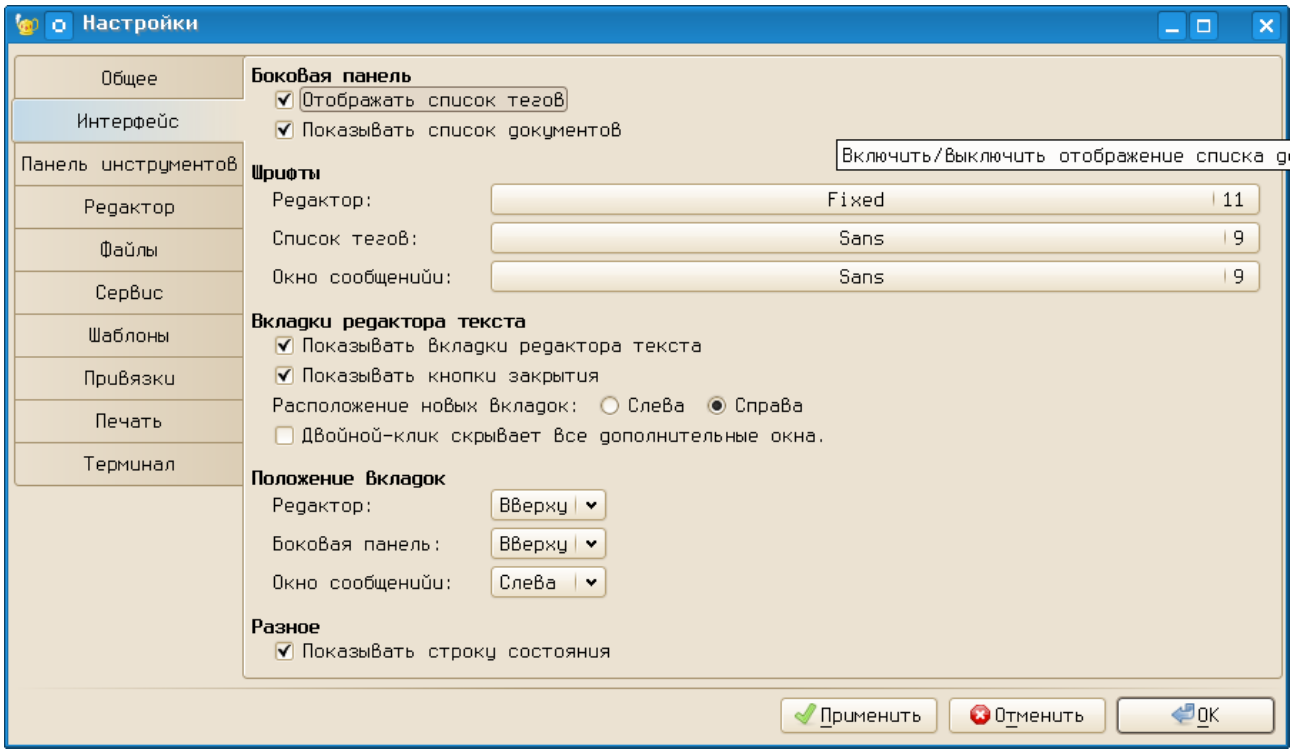

Рисунок 3. Настройки внешнего вида IDE Geany

Шрифты каждый выбирает для себя, в частности, автор предпочитает для редактирования программ использовать моноширинные шрифты (в именах которых содержится слово «Fixed»).

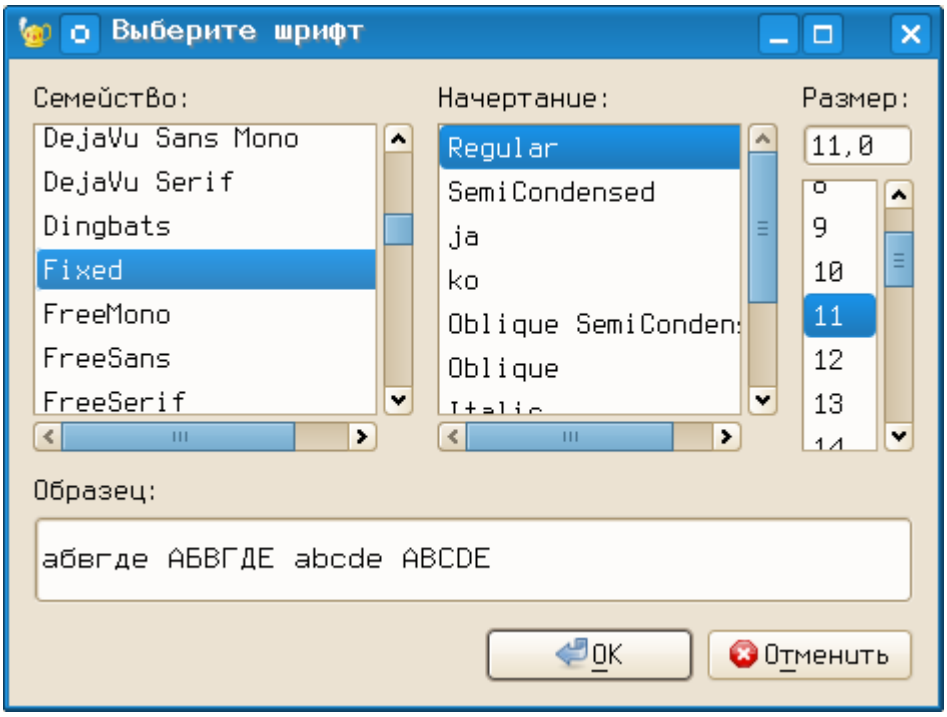

Рисунок 4. Выбор шрифтов в Geany

Боковая панель при использовании режимов, показанных на рис. 3, будет иметь две вкладки – «Теги» и «Документы». На вкладке «Документы» показывается список открытых документов, а на вкладке «Теги» при работе с языками программирования отображаются имена переменных с номерами строк, в которых эти переменные определяются, а также имена использованных в программе функций с номерами строк, в которых эти функции определяются.

На вкладке «Панель инструментов» (рис. 5) очевидно должен быть включён режим «Показывать панель инструментов». Включение режима «Добавить панель инструментов в меню» приведёт к тому, что панель инструментов станет «продолжением» строки главного меню. В большинстве случаев это нецелесообразно, поэтому такой режим включать не рекомендуется.

Для внешнего вида панели инструментов разумно выбрать вариант «Только иконки», а размер иконок выбрать по вкусу.

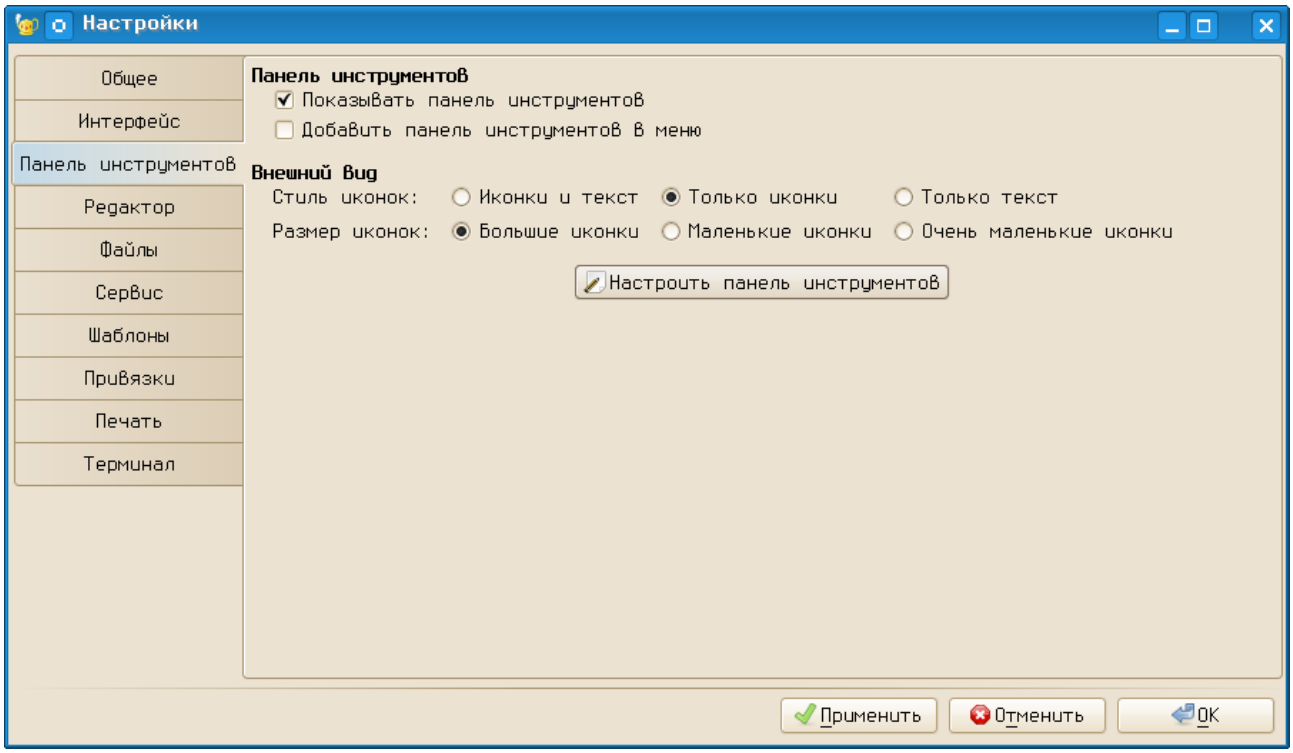

Рисунок 5. Настройка панели инструментов в Geany

Если есть желание изменить набор видимых инструментов Geany, можно использовать диалог настройки панели инструментов (кнопка «Настроить панель инструментов»). Диалог настройки показан на рис. 6.

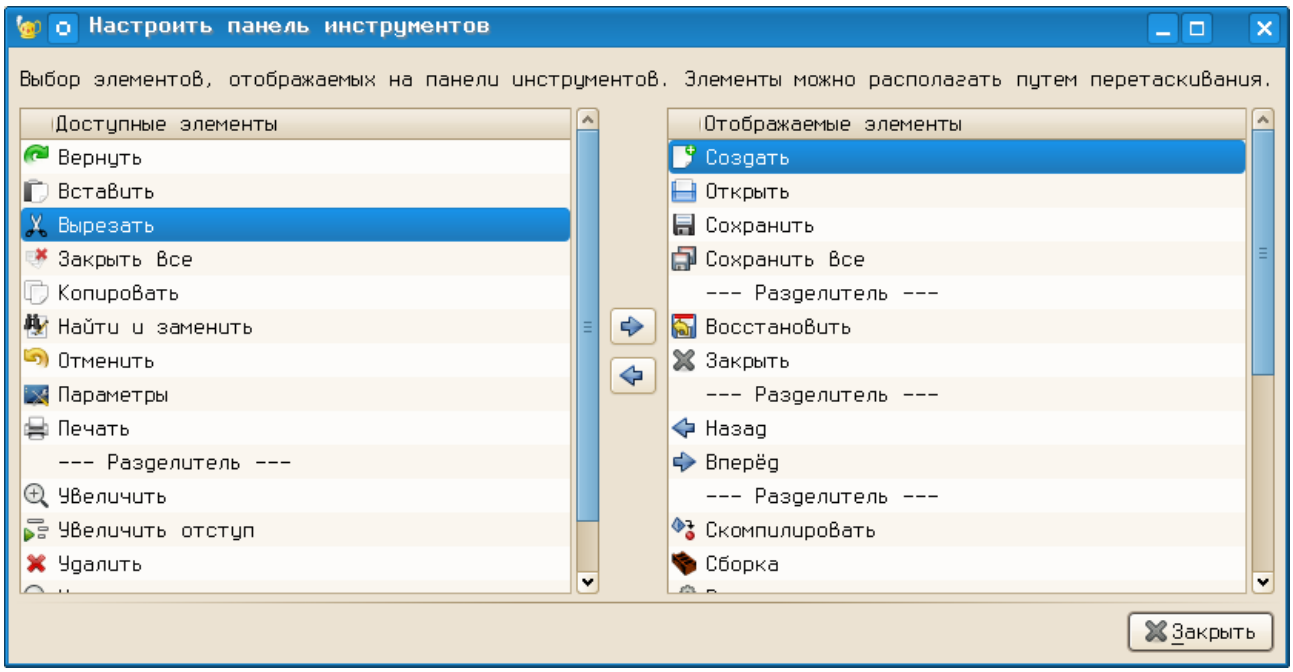

Рисунок 6. Диалог настройки панели инструментов Geany

В этом диалоге можно перемещать доступные элементы из левой панели в область отображаемых элементов (правую панель) кнопкой «Стрелка вправо» и убирать отображаемые элементы из панели кнопкой «Стрелка влево» (кнопки находятся в центре окна диалога). Изменение порядка кнопок, как следует из подсказки этого диалога, достигается путём перетаскивания выбранных элементов.

На панели инструментов Geany (рис. 1) имеется строка ввода слова для поиска и кнопка поиска, но отсутствуют часто используемые кнопки «Вырезать», «Копировать» и «Вставить». Поэтому при начальной настройке имеет смысл убрать строку поиска из панели и добавить нужные кнопки, расположив их примерно так, как показано на рис. 7.

Важно не забыть закрыть диалог настройки панели инструментов (кнопка «Закрыть» и применить сделанные изменения (кнопка «Применить» на вкладке «Панель инструментов» диалога настроек Geany).

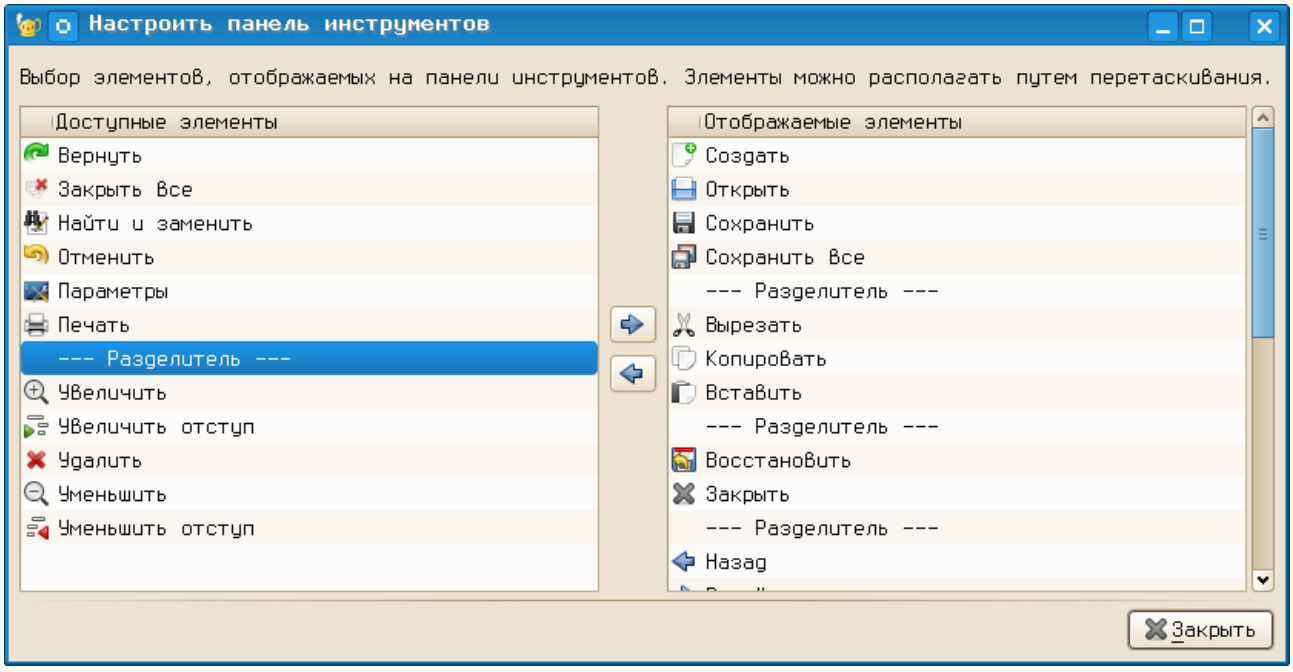

Рисунок 7.Изменение панели инструментов Geany

Вид окна программы после изменений показан на рис. 17.

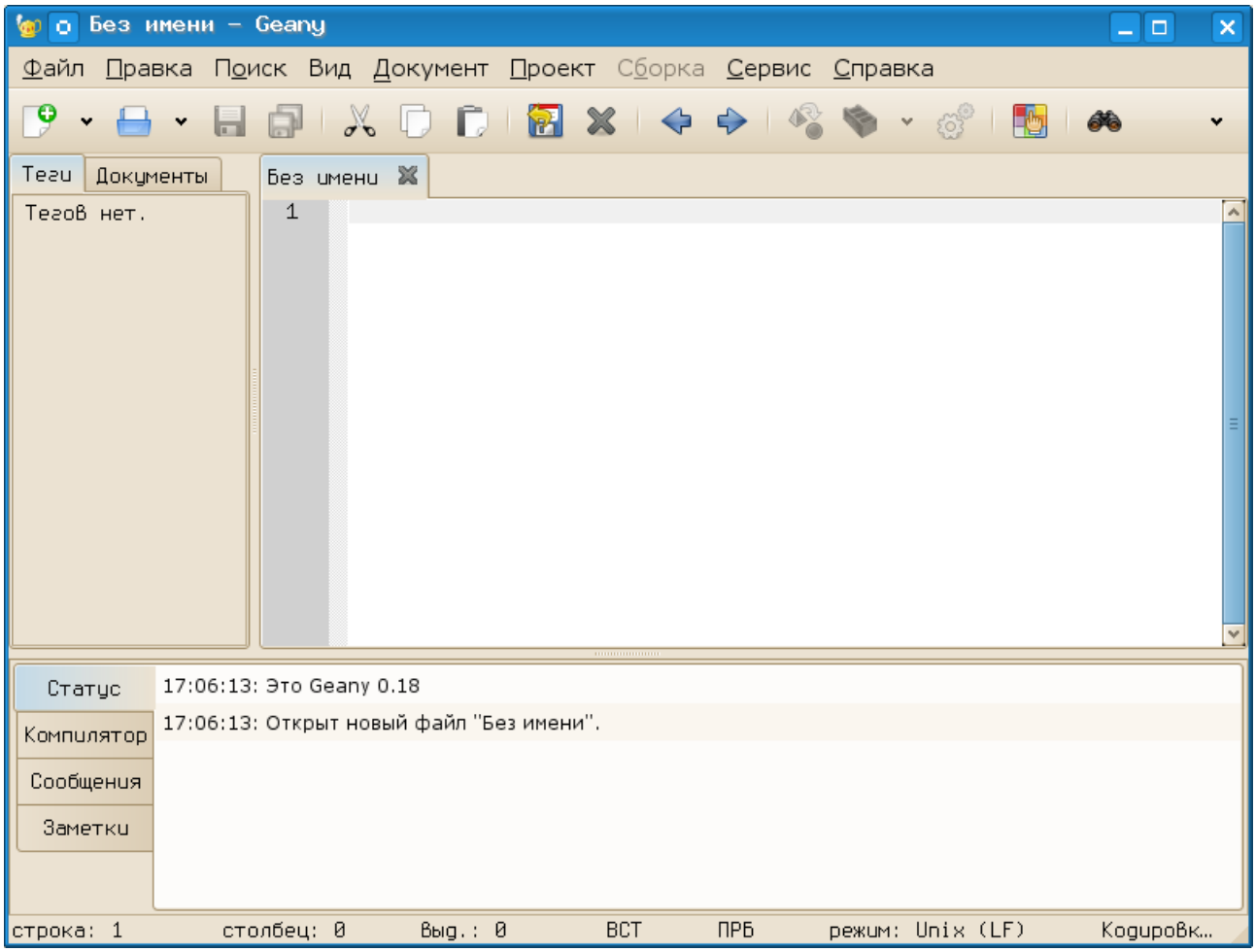

Рисунок 8. Geany с отключённым терминалом и модифицированной панелью инструментов

Вкладка для настроек редактора (рис. 9) в свою очередь, является многостраничным диалогом.

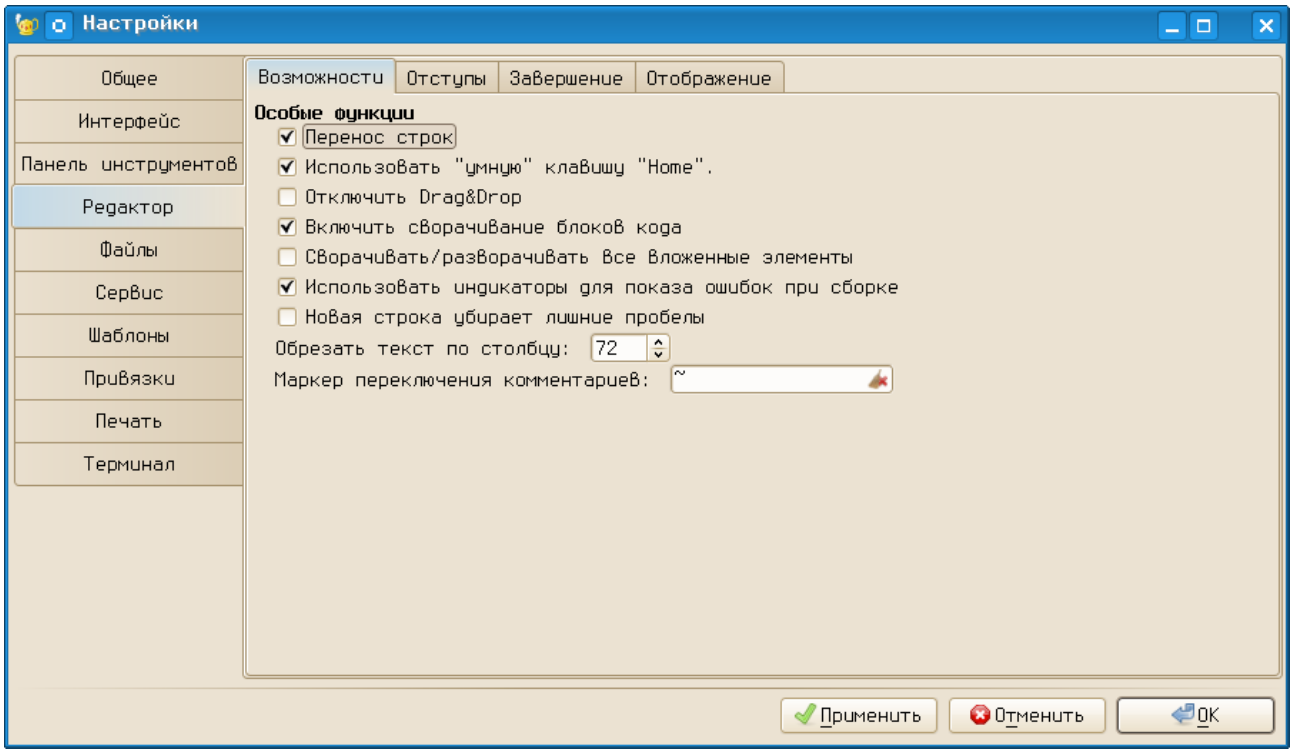

Рисунок 9. Настройки редактора в Geany

На листе «Возможности» следует проследить, что включены режимы «Перенос строк» и «Включить сворачивание блоков кода». На листе «Отступы» (рис. 10) нужно установить ширину отступа в 4 символа (в соответствии с правилами Python) и тип отступа – «Пробелы». Режим «Отступ при помощи клавиши Tab» можно оставить без изменений.

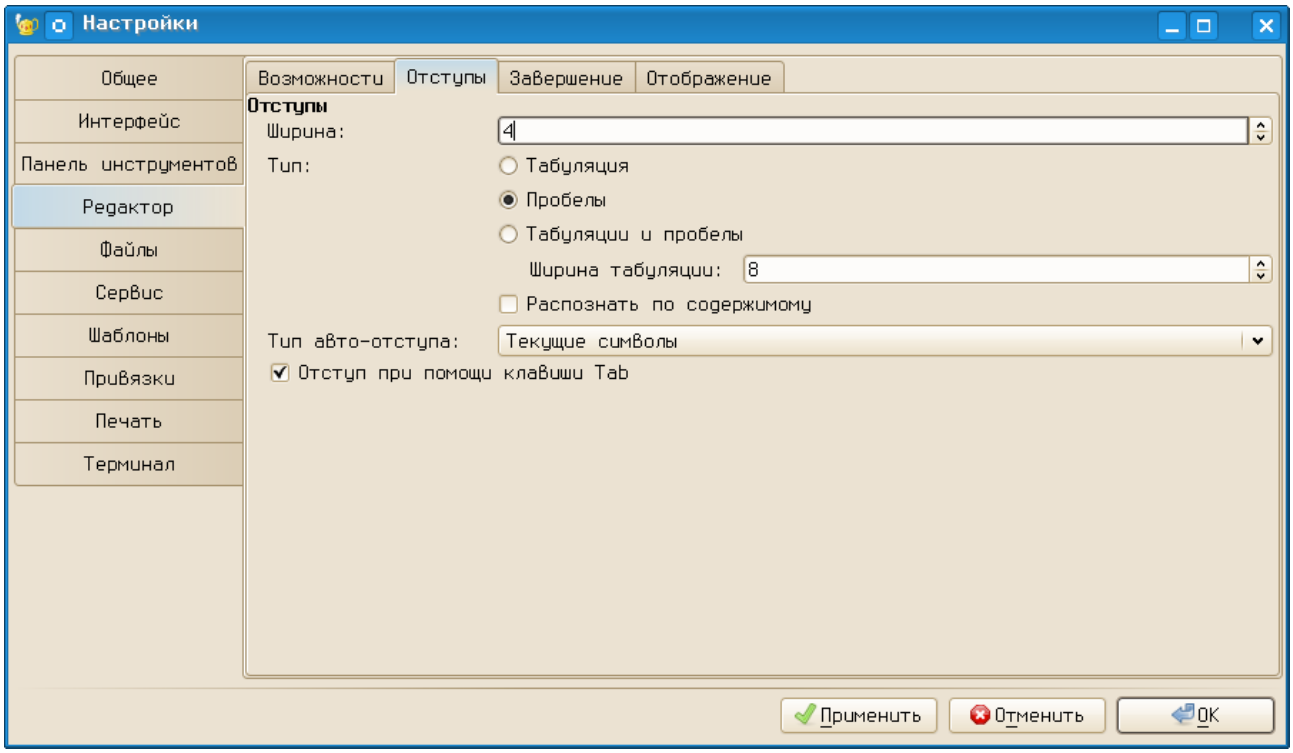

Рисунок 10. Настройки отступов для работы с Python в Geany

На листе «Завершение» (рис. 11) при желании можно настроить режимы авто-завершения слов и автоматического создания парных скобок, однако начальные настройки являются достаточно разумными и без особой необходимости их менять не нужно.

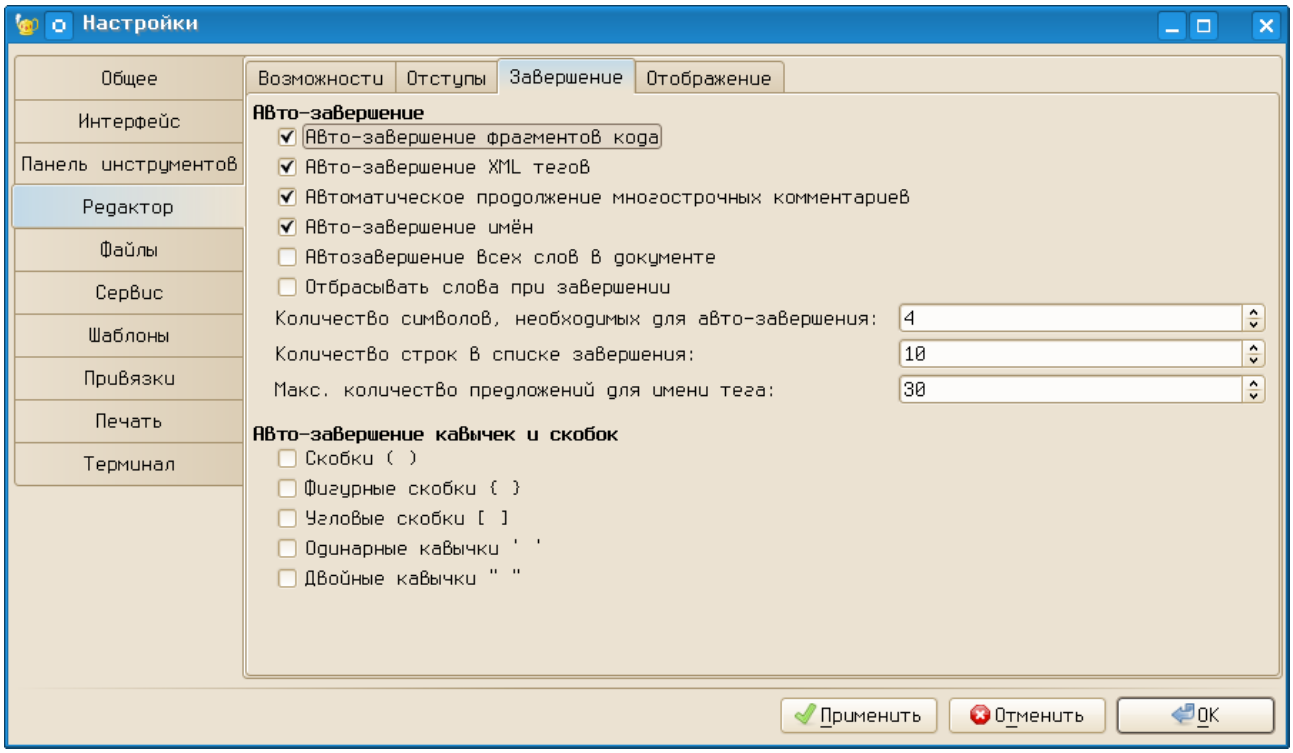

Рисунок 11. Настройки авто-завершения в редакторе Geany

На листе «Отображение» в настройках редактора (рис. 12) нужно обязательно включить режимы «Показывать индикаторы отступа», «Показывать чистые пробелы» и «Показывать номера строк». Кнопка «Цвет маркера длинной строки» открывает диалог выбора цвета для GTK (рис. 13). Этим цветом будет обозначена вертикальная линия в окне редактора (условная правая граница текста).

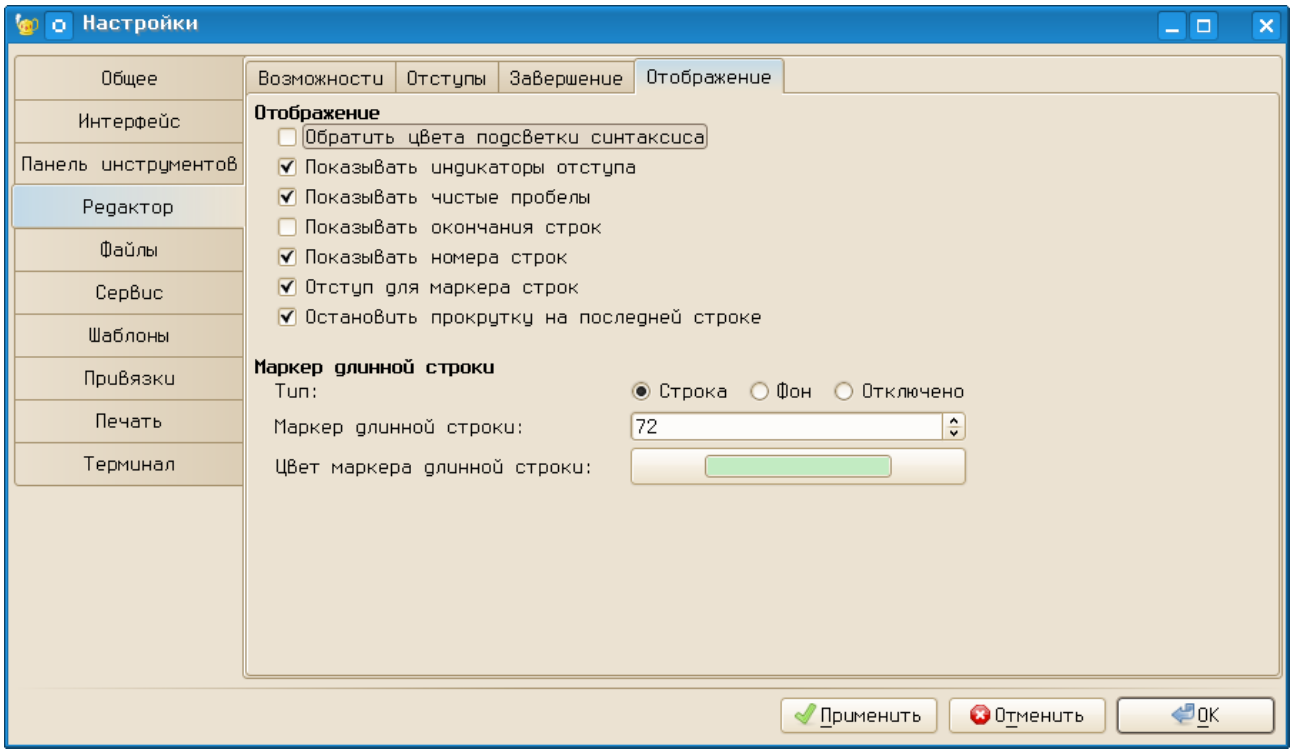

Рисунок 12. Настройка отображения структуры программы в редакторе Geany

В диалоге выбора цвета можно «крутить» треугольник по цветному кольцу, перетаскивая мышью чёрный отрезок и перетаскивать мышью чёрный кружочек в пределах треугольника. Результат тут же будет показан в палитрах HSV («Тон-Насыщенность-Значение») и RGB («Красный-Зелёный-Синий»), а также в виде HTML-эквивалента («Называние цвета»). И наоборот, можно в помощью полей ввода в диалоге установить нужный цвет и он будет показан в треугольнике соответствующей ориентации. Использование кнопки «пипетка» позволяет получить цвет с любой точки экрана. Под цветным кольцом с треугольником показывается сравнение выбранного цвета с текущим.

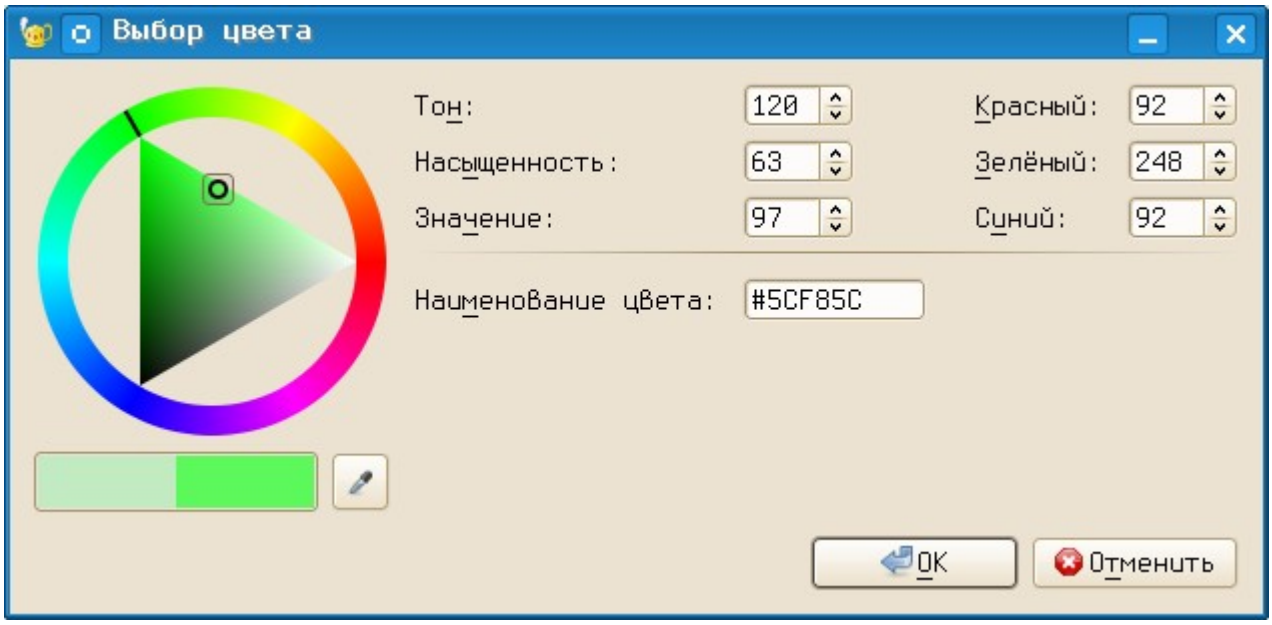

Рисунок 13. GTK-диалог выбора цвета для обозначения правой границы текста в Geany

На вкладке «Файлы» диалога настроек Geany (рис. 14) определяются режимы обработки файлов. Здесь можно ничего не менять, только следует обратить внимание на то, что для всех сохраняемых файлов устанавливается одинаковая кодировка. Geany корректно открывает файлы в различных кодировках, но сохраняет всегда в одной и той же.

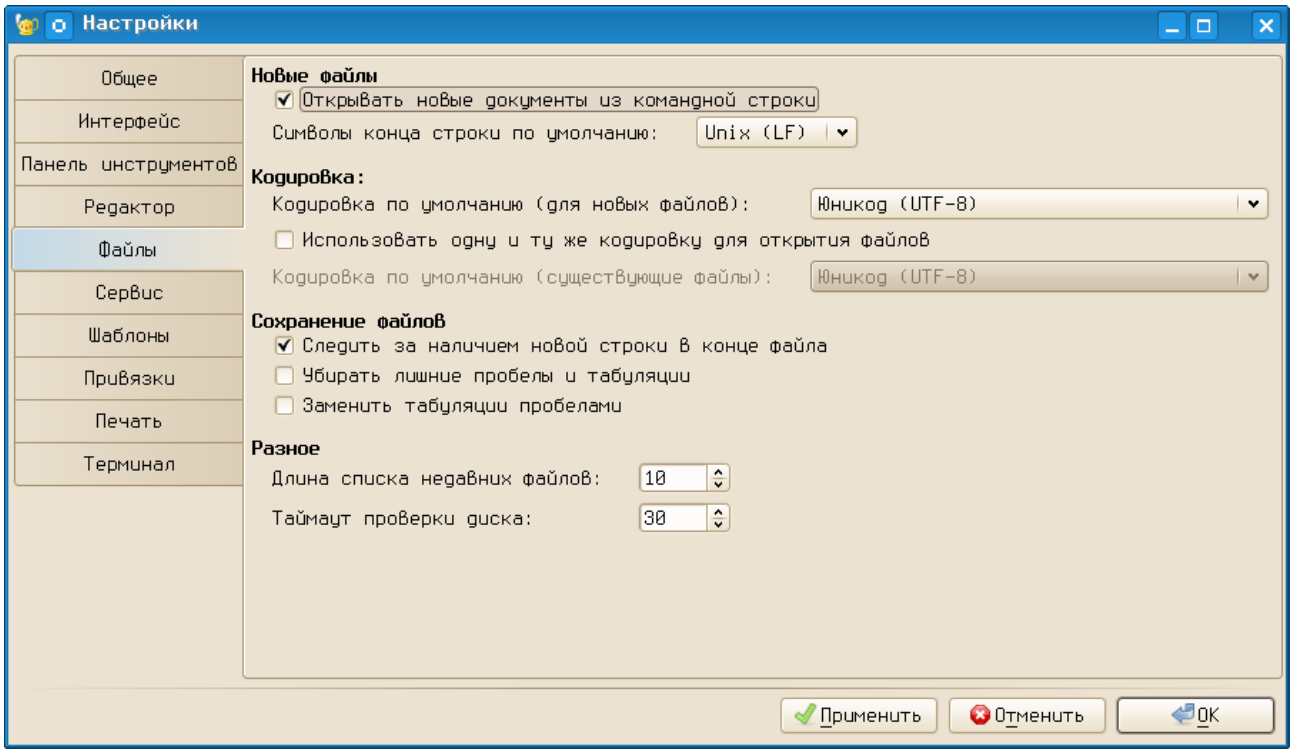

Рисунок 14. Настройка обработки файлов в Geany

На вкладке «Сервис» определяются внешние программы (утилиты), которые используются Geany для обработки исходных текстов (рис. 15).

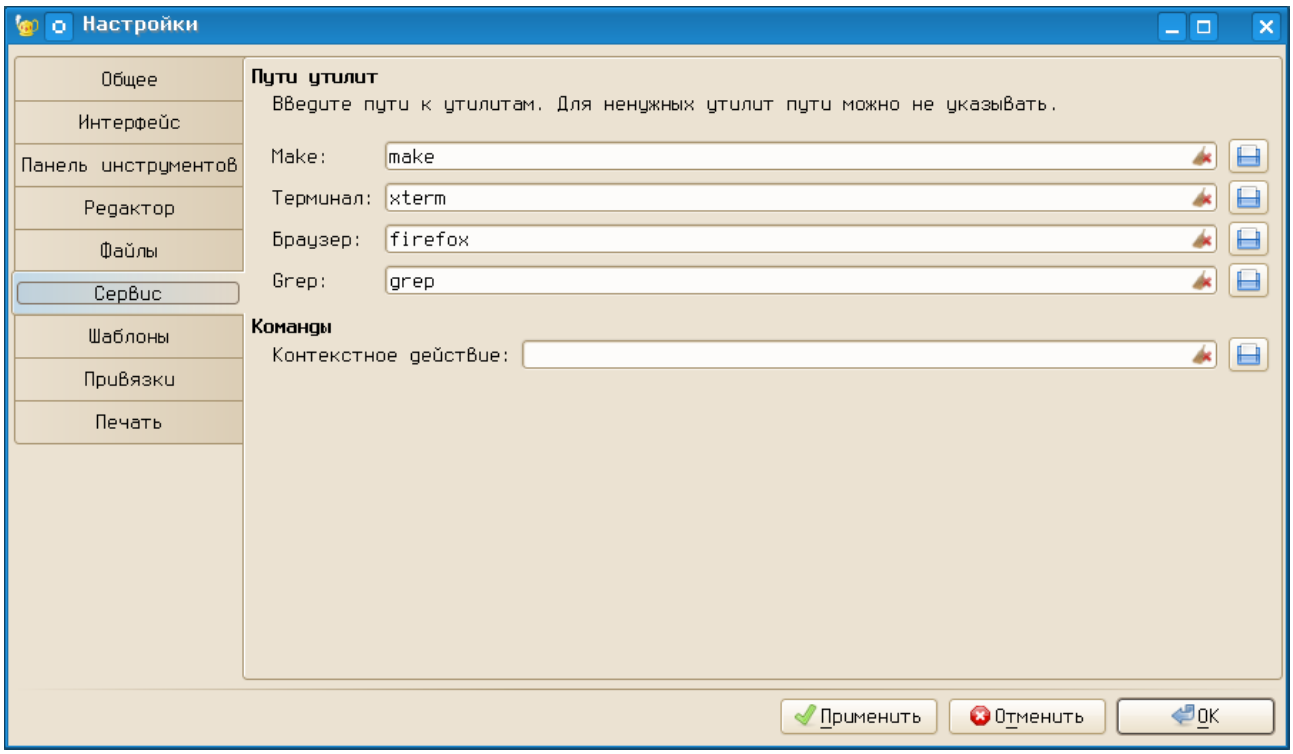

Рисунок 15. Подключение внешних программ

Как раз здесь и определяется программа, которая будет использоваться в качестве эмулятора терминала. Как уже отмечалось выше, будет использоваться xterm.

Оставшиеся вкладки («Шаблоны», «Привязки» и «Печать») диалога настроек Geany пока не представляют интереса.

Подключение документации и её использование

Очень важно иметь возможность быстро открыть документацию. После установки пакета python-doc оригинальная документация находится в каталоге `/usr/share/doc/python-doc-x.y.z`, где `x.y.z` – версия Python (например, `/usr/share/doc/python-doc-2.5.2`)

Самый простой путь подключения документации – запустить любой браузер (Konqueror или Firefox), выбрать в главном меню команду «Файл/Открыть файл...» и в диалоге выбора файлов в иерархии корневого каталога найти `/usr/share/doc/python-doc-x.y.z/index.html`. После этого в браузере можно добавить этот адрес в закладки.

Общий вид документации по Python в браузере показан на рис. 16.

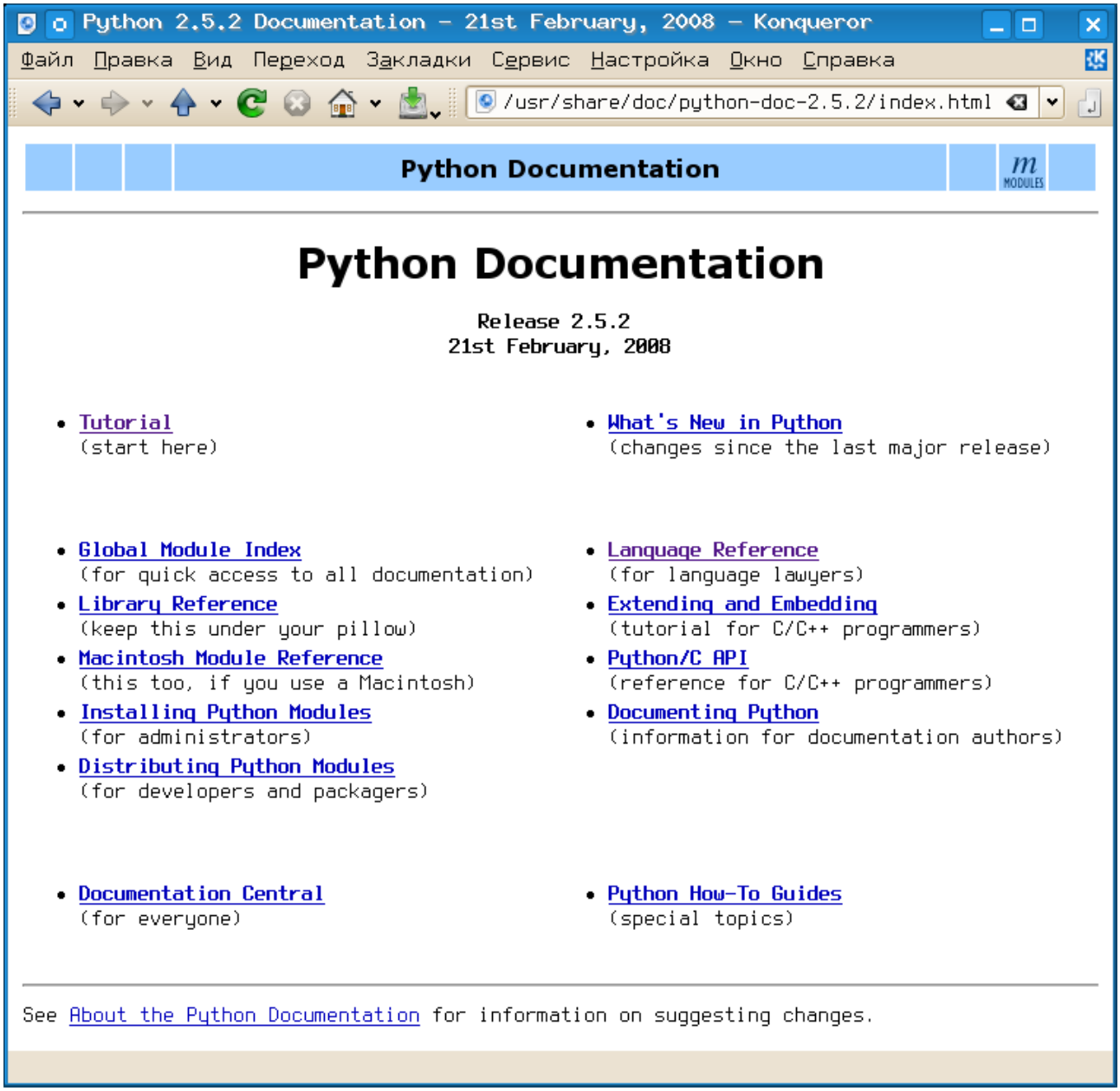

Рисунок 16. Документация по Python (главная страница)

Русский перевод этого руководства доступен на сайте python.ru в PDF- и PostScript-вариантах.

Для начинающих эта документация практически бесполезна, поэтому рекомендуется пользоваться прилагаемым к «Практикуму» списку литературы.

Сохранение и открытие файлов, запуск выполнения программ

Текст программы пишется в окне редактора, и пока он не сохранён, на ярлычке активной вкладки имя файла выделено красным цветом (рис. 17). Кроме того, для вновь набранного текста программы отсутствует подсветка синтаксиса, поскольку анализатор синтаксиса пока «не знает» языка, на котором написан этот файл.

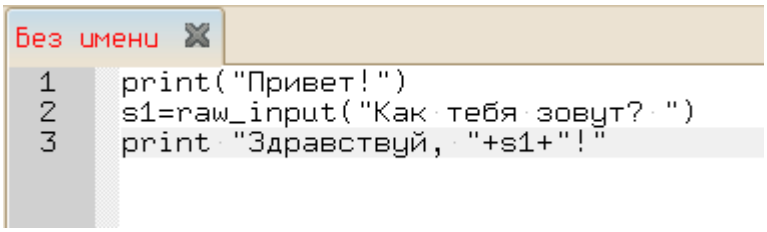

Рисунок 17. Текст программы в окне редактора до сохранения

Для сохранения файла используется команда главного меню «Файл/Сохранить» (или «Файл/Сохранить как...» при первом сохранении), что равносильно использованию комбинации клавиш <CTRL>+<S>.

Выбор этой команды открывает GTK-диалог сохранения/открытия файла (рис. 18). Для получения возможности сохранения файла в каталоге по выбору пользователя нужно щёлкнуть по «стрелочке» слева от пояснения «Просмотреть другие папки». Тогда в диалоге будут показаны дополнительные панели – «Маста» и «Имя». В панели «Места» имеется список наиболее части используемых мест в файловой системе, а в панели «Имя» показан список каталогов и файлов в выбранном месте. Для переключения на новое место нужен одиночный щелчок левой кнопкой мыши в панели «Места», а для открытия папки (каталога) в панели «Имя» нужен двойной щелчок левой кнопки мыши независимо от настроек пользовательской среды в сеансе.

Если выделить каталог в панели «Имя» одиночным щелчком левой кнопкой мыши, то его можно добавить в список мест в помощью кнопки «Добавить», расположенной под панелью «Места».

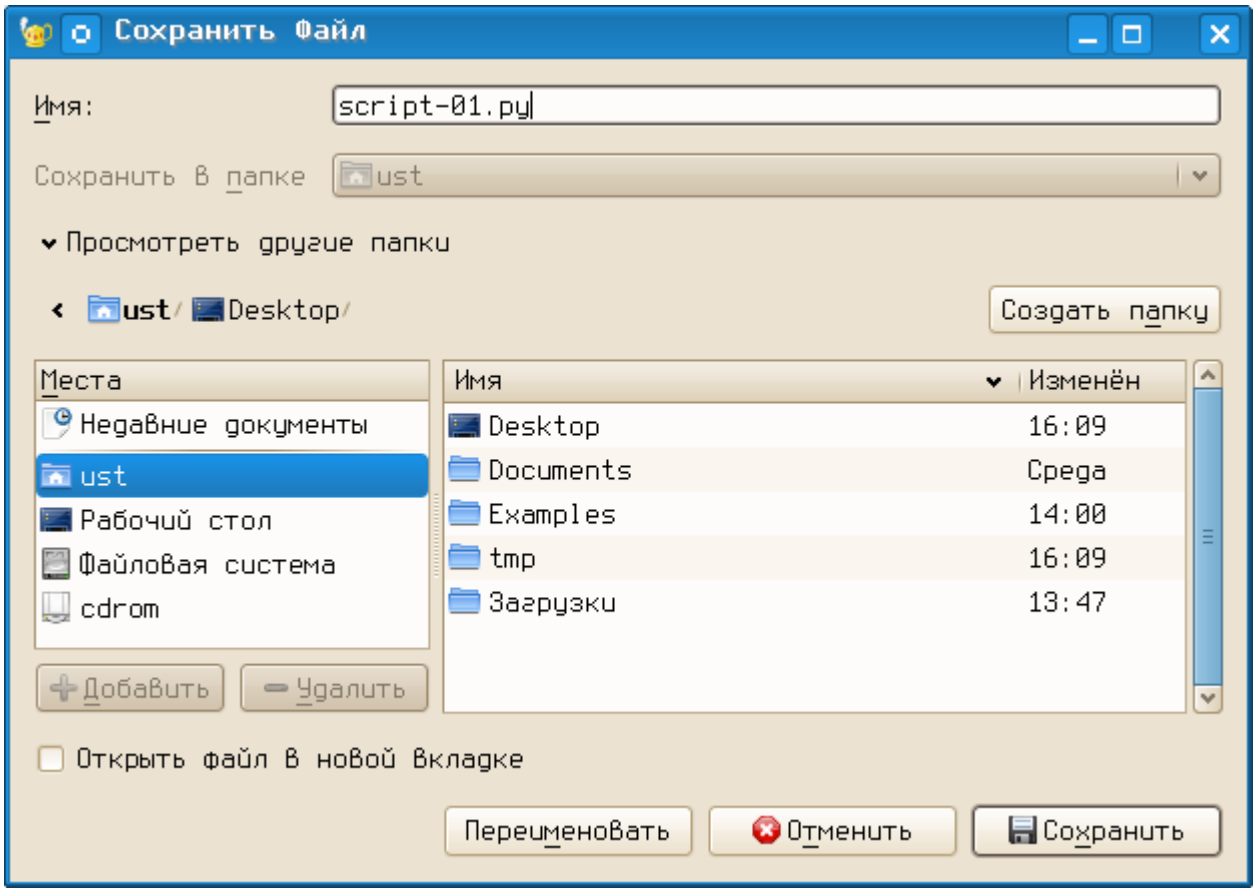

Рисунок 18. GTK-диалог сохранения (открытия) файла

При сохранении необходимо указать имя и «расширение» файла (для текстов на Python – «`.py`», как показано на рис. 18), иначе не будет работать анализатор синтаксиса.

Нажатие на кнопку «Сохранить» закрывает диалог сохранения и в редакторе видим подсветку синтаксиса, а также список переменных в панели «Теги» (рис. 19).

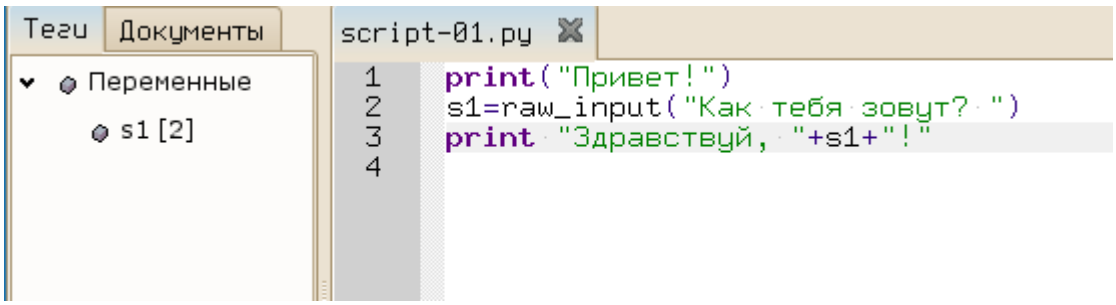

Рисунок 19. Подсветка синтаксиса и теги для программы на Python

В списке тегов число в квадратных скобках справа от имени переменной означает номер строки, в которой первый раз определена эта переменная.

Для открытия файла удобно использовать список последних файлов (команда «Файл/Недавние файлы»).

Для запуска программы на выполнение (точнее, на трансляцию и выполнение интерпретатором Python) используется клавиша <F5> или кнопка «Запустить или посмотреть текущий файл» в панели инструментов.

Обработка ошибок

Самая первая ошибка при создании программы на Python – пропуск строки с указанием кодовой страницы. Именно такая ошибка допущена в рассмотренном выше примере. На рис. 20 показано сообщение интерпретатора после запуска этого примера.

Рисунок 20. Сообщение интерпретатора об ошибке, связанной с отсутствием указания кодовой страницы

(Закрыть окно терминала можно нажатием на <ENTER>).

В случае незакрытых кавычек или скобок Geany отмечает цветом символ, не имеющий пары (рис. 21).

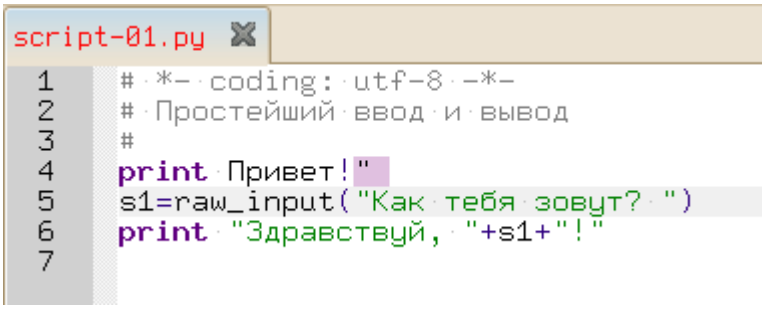

Рисунок 21. Выделение незакрытой кавычки

Ошибки в именах функций, ключевых словах и составных операторах при запуске программы на выполнение также сопровождаются сообщениями интерпретатора.

Для каждой ошибки указывается номер строки и (по возможности) что именно вызвало ошибку.

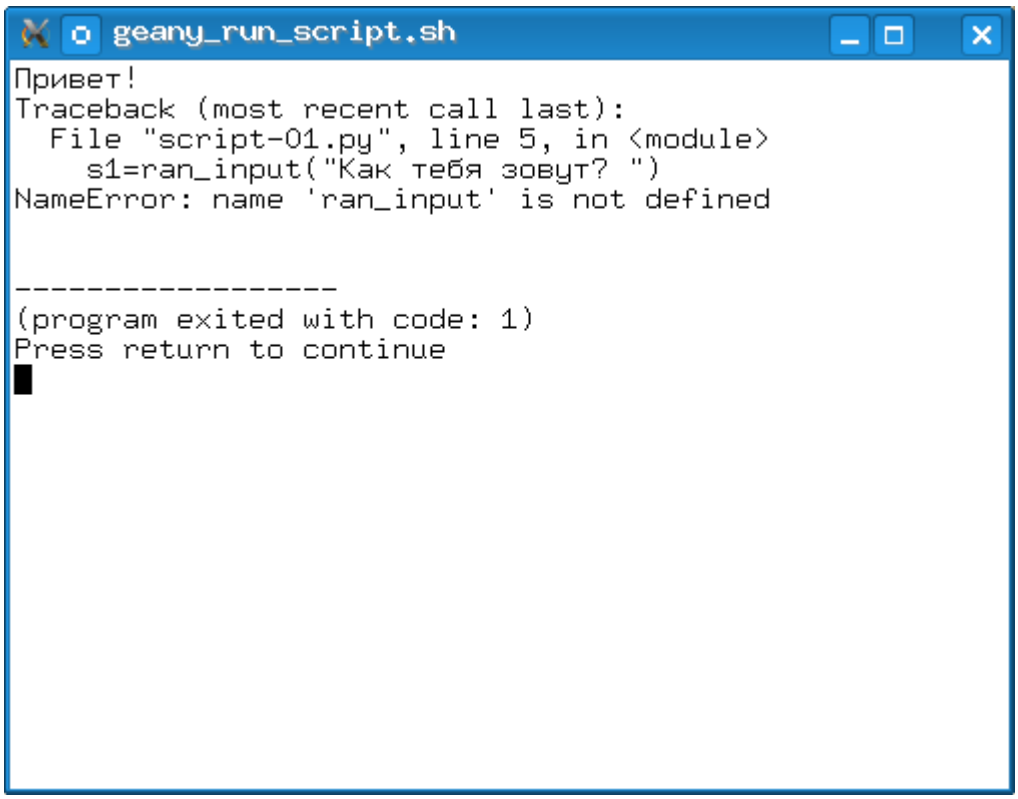

Рисунок 22. Синтаксическая ошибка: неверное имя функции

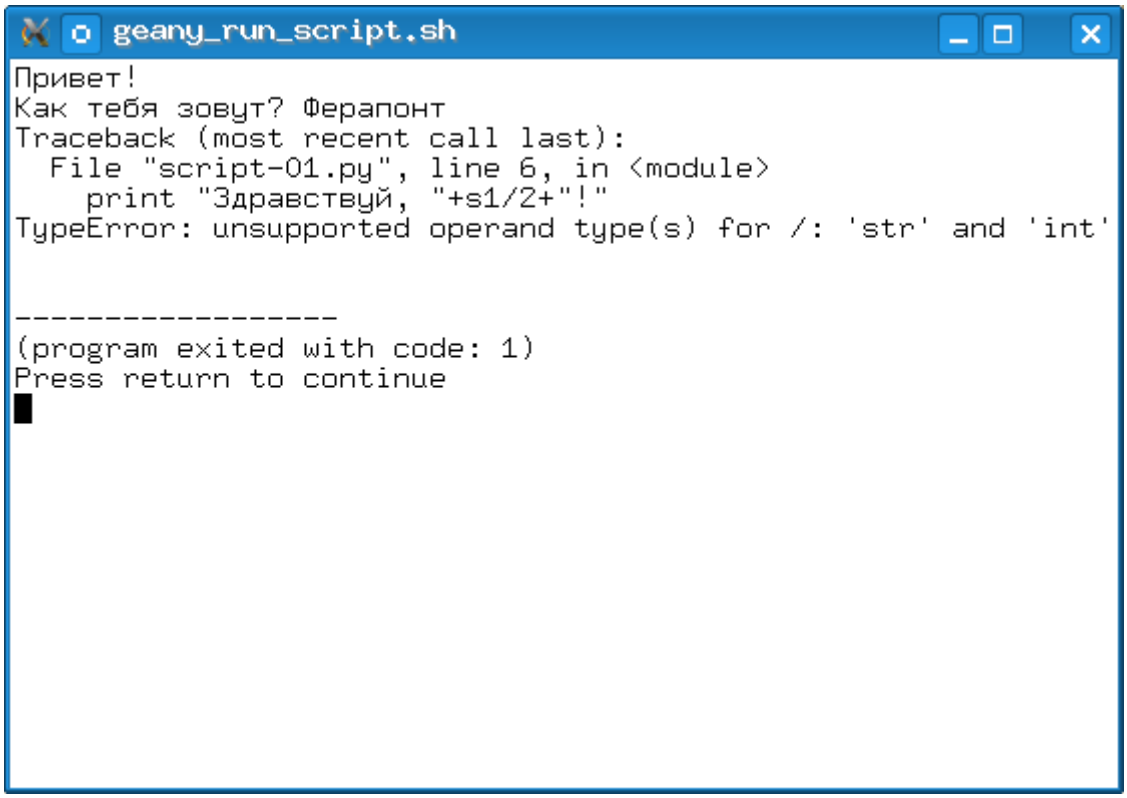

Рисунок 23. Ошибка времени выполнения: несоответствие типов (попытка поделить строку на число)

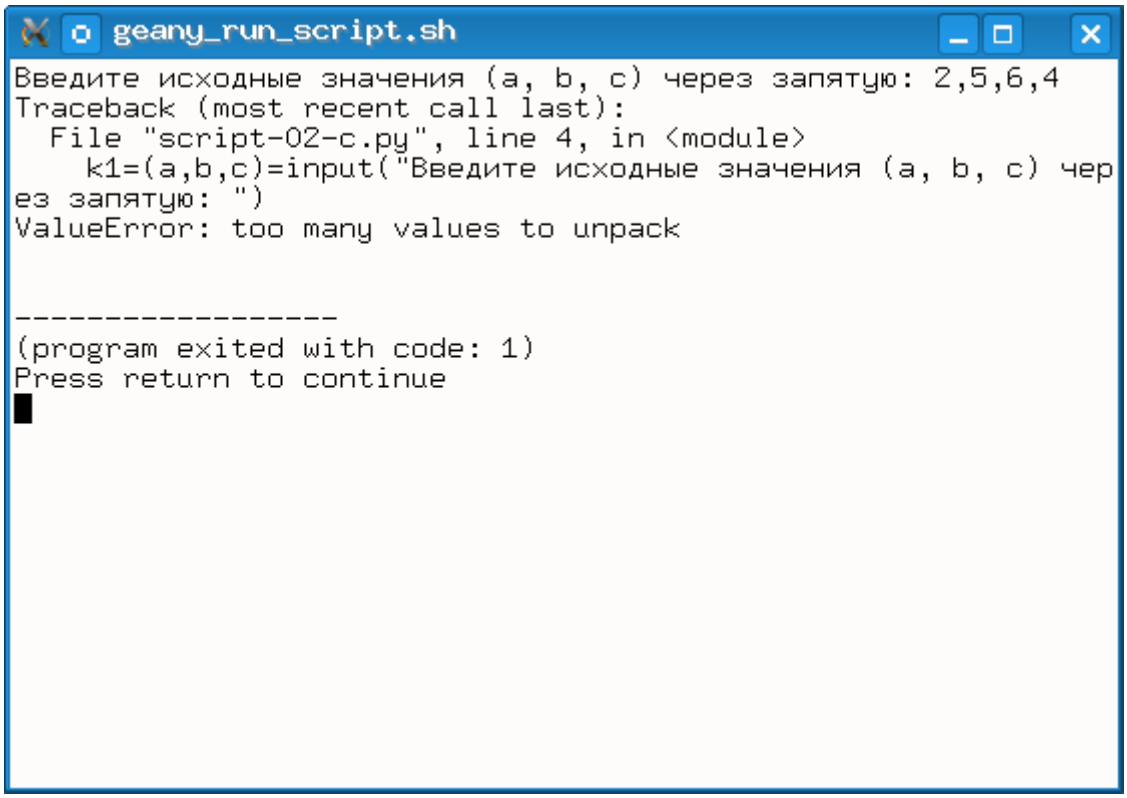

Рисунок 24. Ошибка времени выполнения: несоответствие количества элементов последовательности

В случае синтаксических ошибок и ошибок времени выполнения («исключения» – «exception» в терминах Python) программа завершается с кодом 1.

Если программа завершается с кодом 0 (в терминале сообщение «`(program exited with code: 0)`», но работает неверно, то такая ошибка называется «семантической» и по сути является ошибкой в алгоритме. Такие ошибки выявляются только при тестировании программы на контрольных примерах.

Использование IDE Eric.

Внешний вид окна Eric при первом запуске показан на рис. 25.

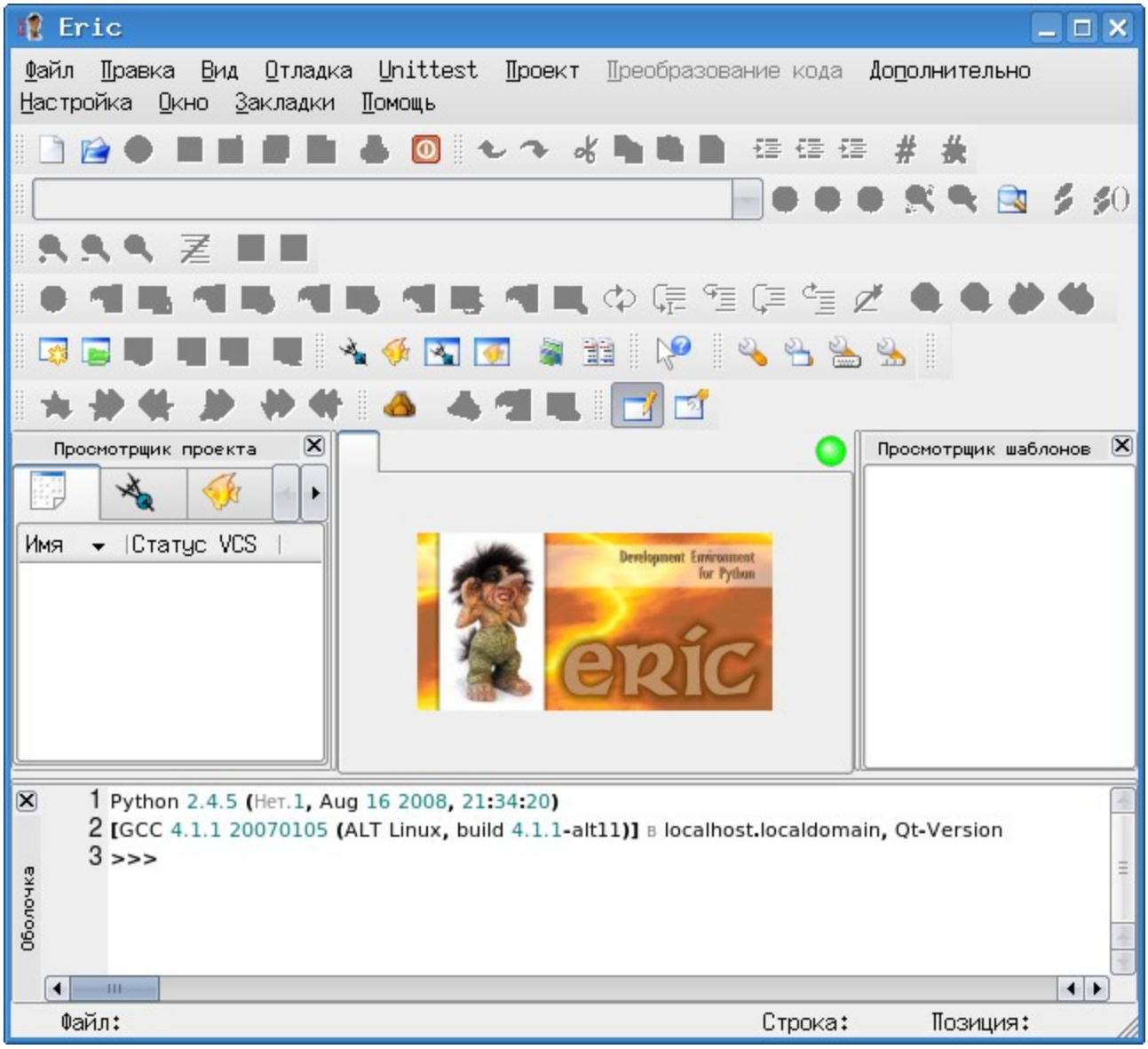

Рисунок 25. Внешний вид IDE «Eric»

Центральная часть окна (в которой находится картинка с троллем Эриком) предназначена для размещения вкладок с текстами программ, нижняя часть — окно выполнения программы (панель «Оболочка»). В окне Eric много различных панелей инструментов, и очень многие кнопки недоступны (закрашены серым). Пока нет текста программы, для этих кнопок «нет работы».

Первоначальная настройка

Можно упросить вид окна, убрав лишние области («панели»), такие как панель «Просмотрщик проекта» слева от изображения Эрика и панель «Просмотрщик шаблонов» справа от изображения Эрика.

Нажатие кнопки с изображением пустого листа в левой части самой верхней панели инструментов (кнопка «Новый») приведёт к созданию нового документа, а внешний вид окна слегка изменится, и некоторые кнопки в панелях инструментов станут активными (рис. 26).

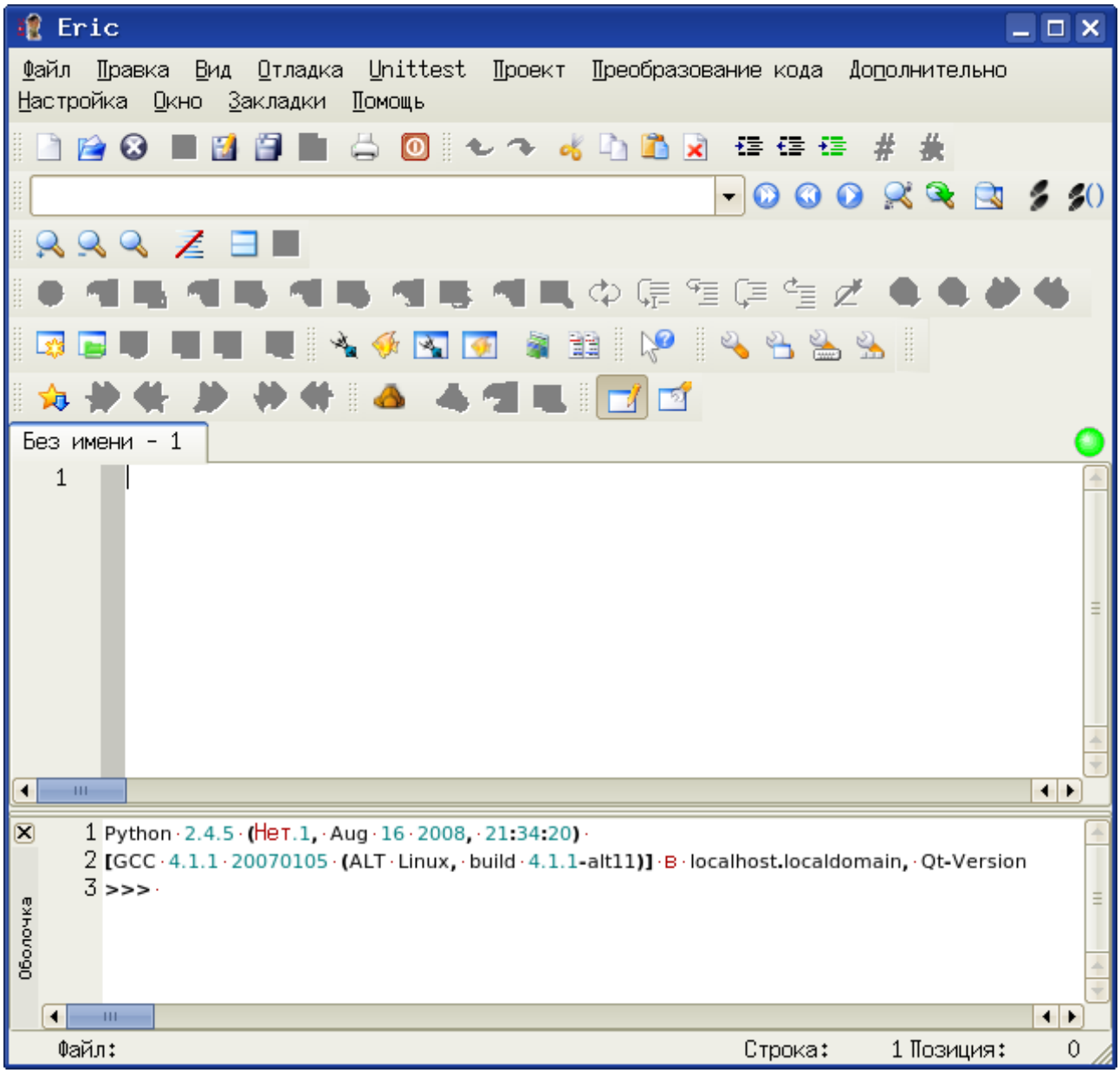

Рисунок 26. Упрощённый Eric — редактор и оболочка

Ещё больше упростить внешний вид можно, убрав лишние панели инструментов. Если открыть пункт главного меню «Окно», то во вложенном меню «Панели инструментов» целесообразно оставить включёнными (с «галочками») панели «Закладки», «Профили», «Редактировать» и «Файл» («галочки» ставятся и убираются щелчком левой кнопкой мыши). Тогда окно IDE Eric приобретёт совсем простой вид (рис. 27, показан пример кода при настроенной подсветке синтаксиса.).

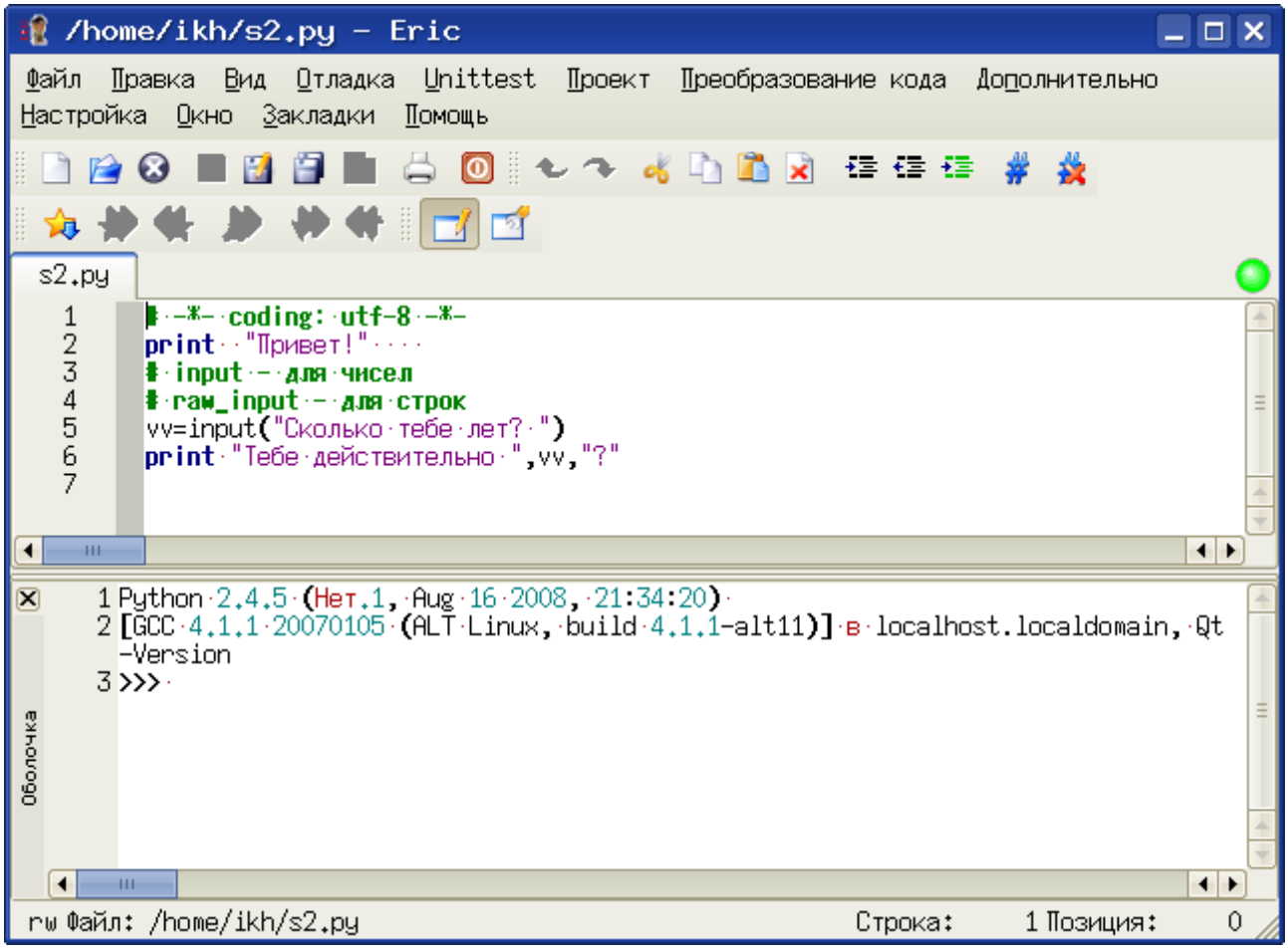

Рисунок 27. Максимально упрощённый вид IDE Eric

Если хочется что-то изменить во внешнем виде программы, можно использовать настройки предпочтений («Настройка/Предпочтения...» в главном меню окна Eric). Настроек очень много (рис. 28), но имеет смысл пока изменять только основные настройки редактора (как показано на рис. 28) и стиль редактора (рис. 29).

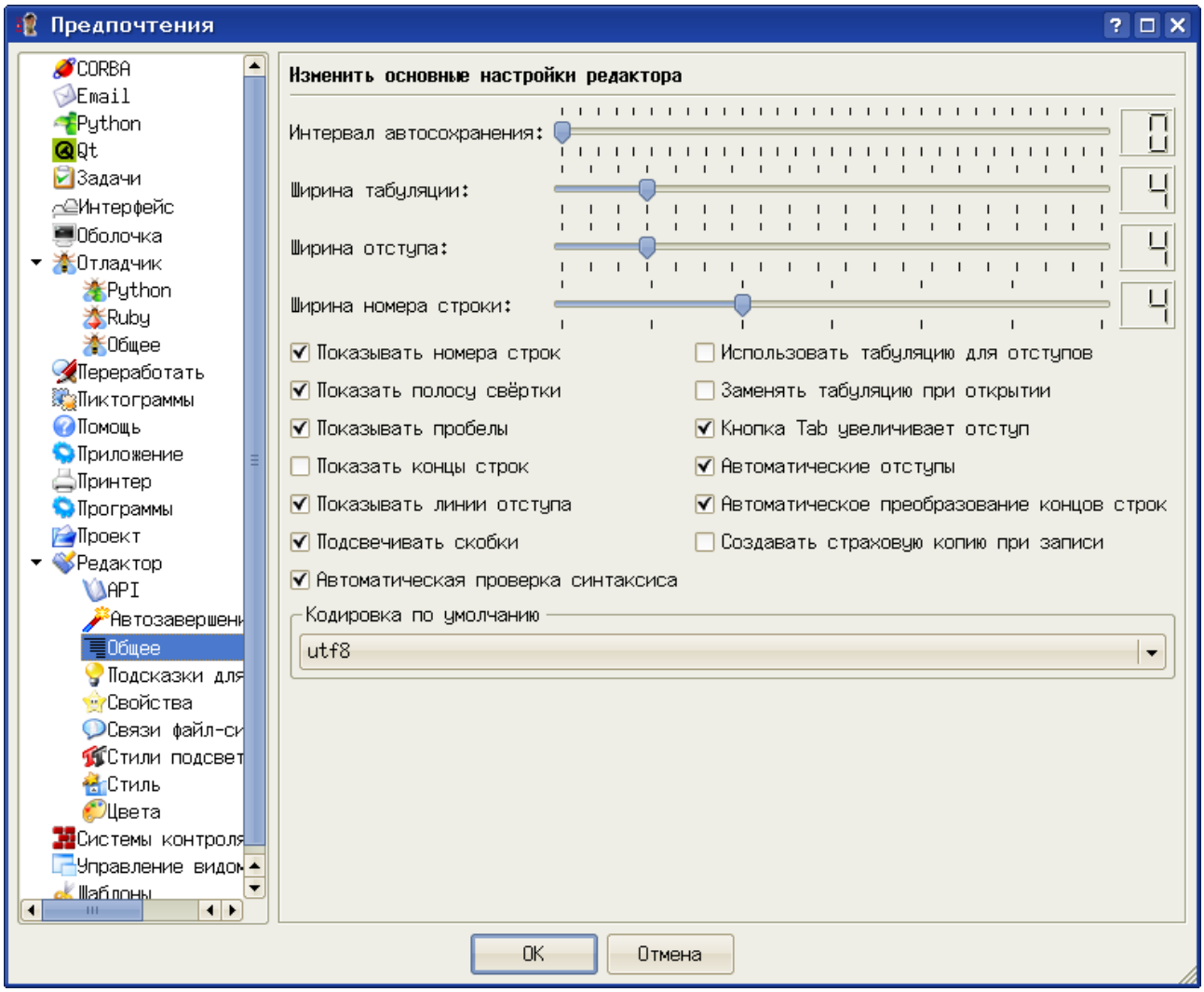

Рисунок 28. Основные настройки редактора в IDE Eric

В диалоге настройки основных свойств редактора полезно установить режимы показа номеров строк, полосы свёртки, пробелов и линий отступа. Полезно также включить режим подсветки скобок и автоматической проверки синтаксиса (в этом случае известные Eric слова и конструкции Python будут выделяться цветом, а неизвестные — не будут), Также полезно использование режима автоматических отступов, которые обсудим позже.

В диалоге настройки стилей редактора (рис. 29) кнопки «Шрифт для номеров строк» и «Моноширинный шрифт» открывают диалоги выбора шрифта, в которых можно выбрать наиболее приятный для пользователя шрифт. Здесь каждый выбирает для себя, в частности, автор предпочитает для редактирования программ и для вывода сообщений IDE использовать моноширинные шрифты (в именах которых содержится слово «Fixed»).

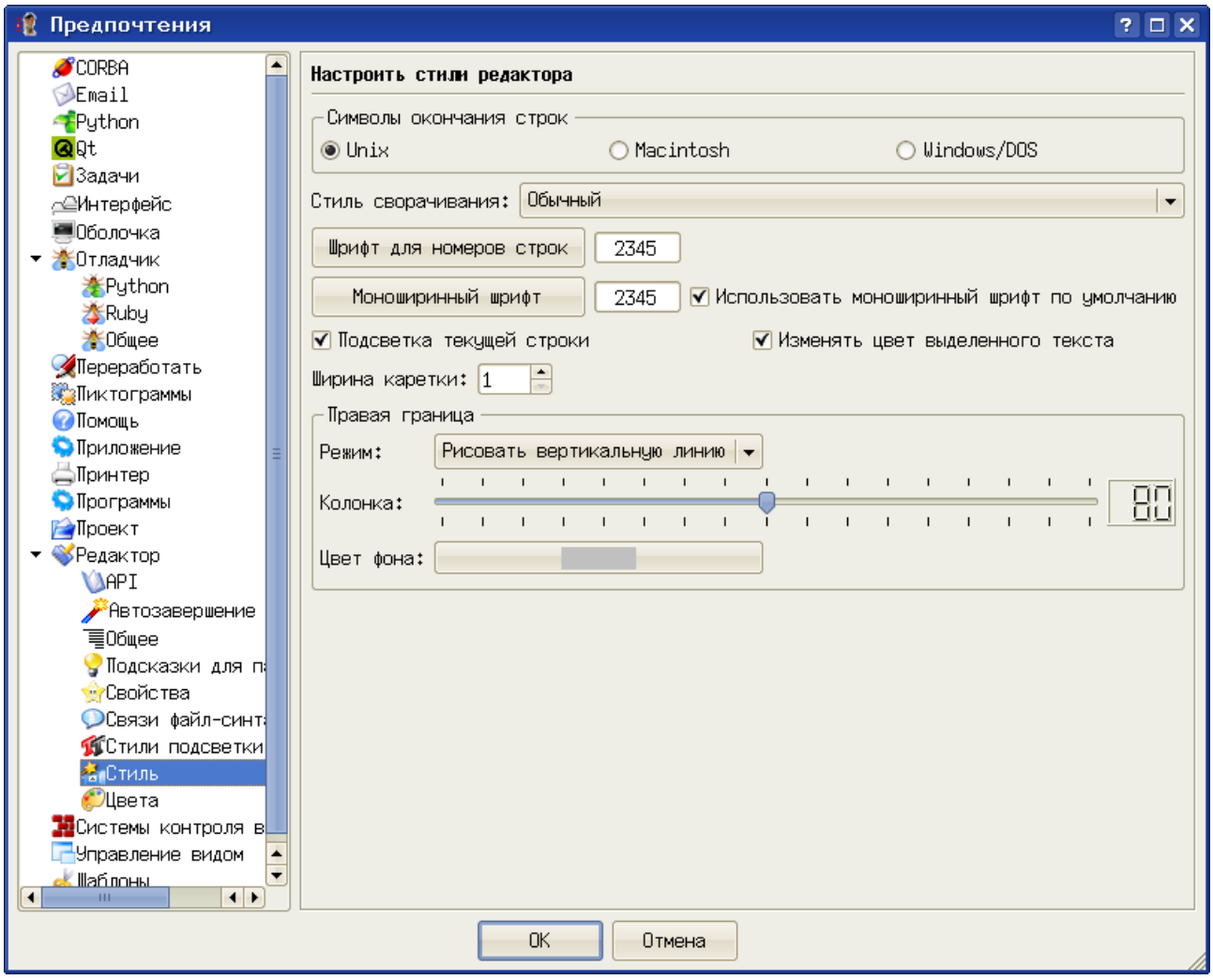

Рисунок 29. Настройка стиля редактора в IDE Eric

Если есть желание изменить вид, размер и цвет подсветки для элементов языка, можно использовать диалог настройки стилей подсветки в «Настройках предпочтений», выбрав вариант «Python» из списка языков, известных лексическому анализатору (рис. 30).

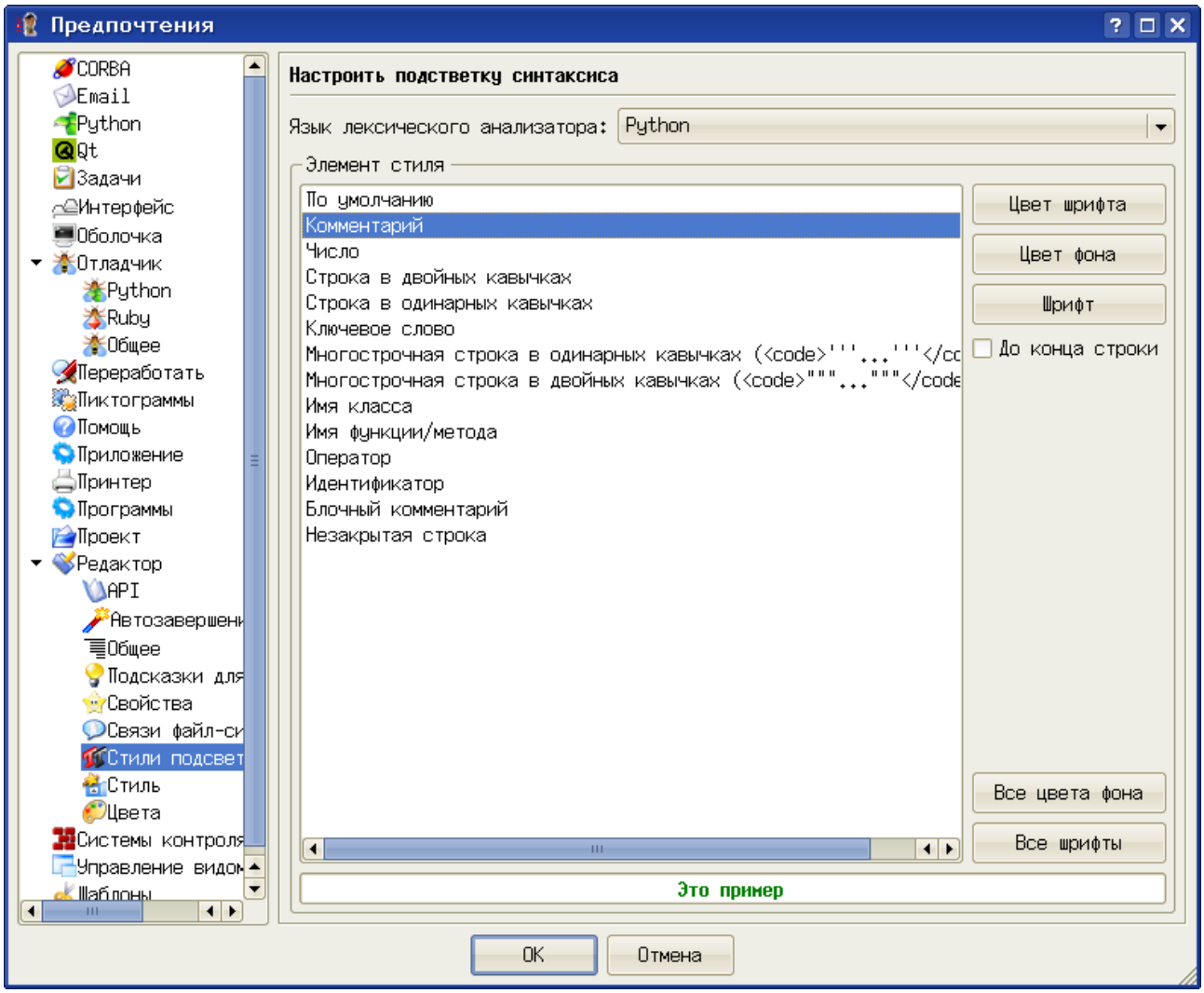

Рисунок 30. Диалог настройки подсветки синтаксиса

Подключение документации и её использование

Очень важно правильно настроить параметры помощи и документации (диалог «Помощь», рис. 31). Здесь указываются варианты просмотрщика помощи IDE Eric, браузера и программы просмотра файлов PDF, а также путь к каталогу с документацией по Python, Нужно заметить, что этот каталог должен существовать, то есть документация по Python (пакет python-doc или что-то похожее) должна быть установлена в системе.

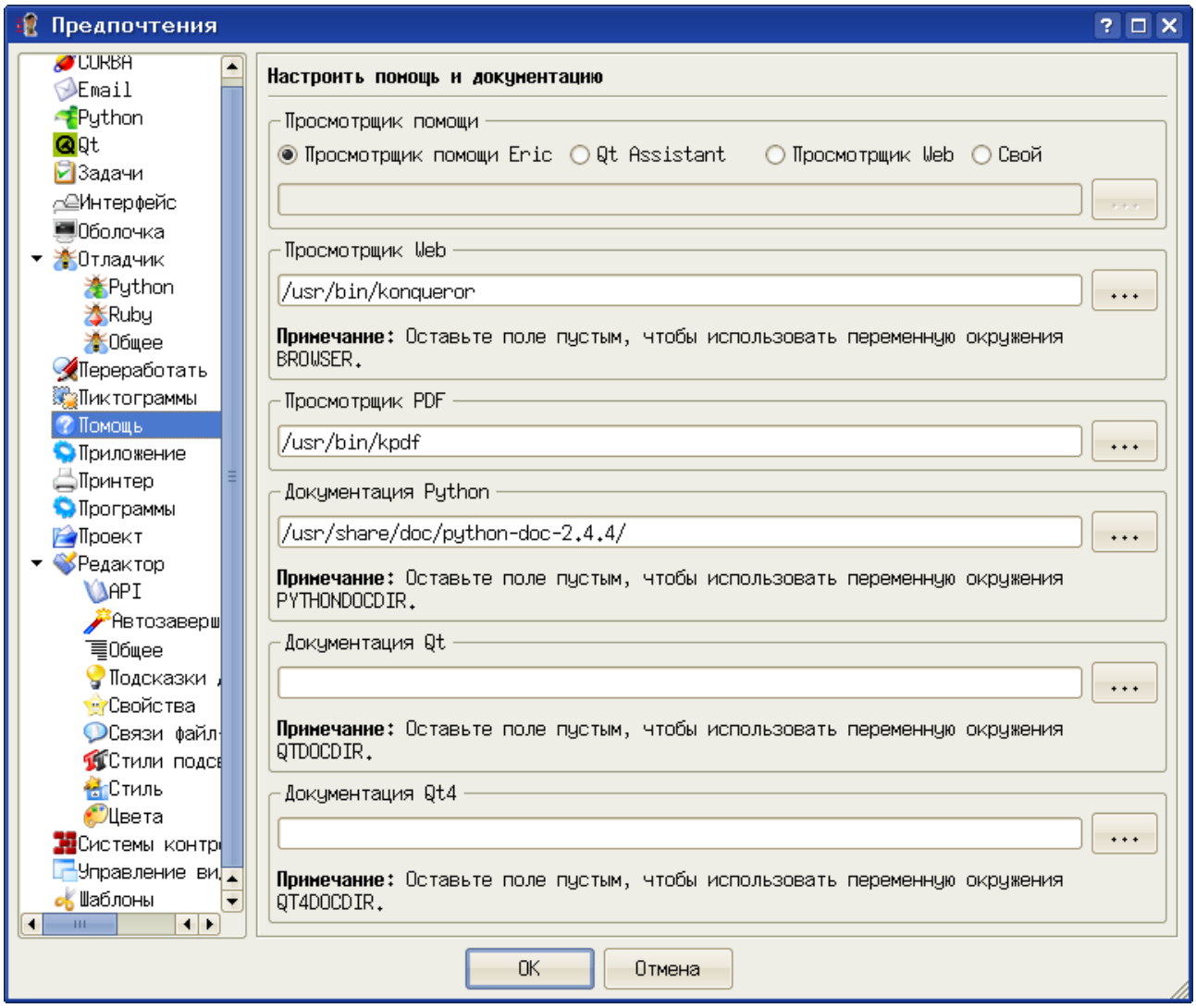

Рисунок 31. Настройка помощи и документации

Программы для просмотра web и файлов pdf можно выбрать по вкусу, узнав предварительно у администратора (или другого специалиста) как называются эти программы и где они находятся в системе.

Каталог документации по Python также нужно указывать реально существующий.

После завершения настроек попробуем воспользоваться системой помощи IDE Eric (меню «Помощь», рис. 32).

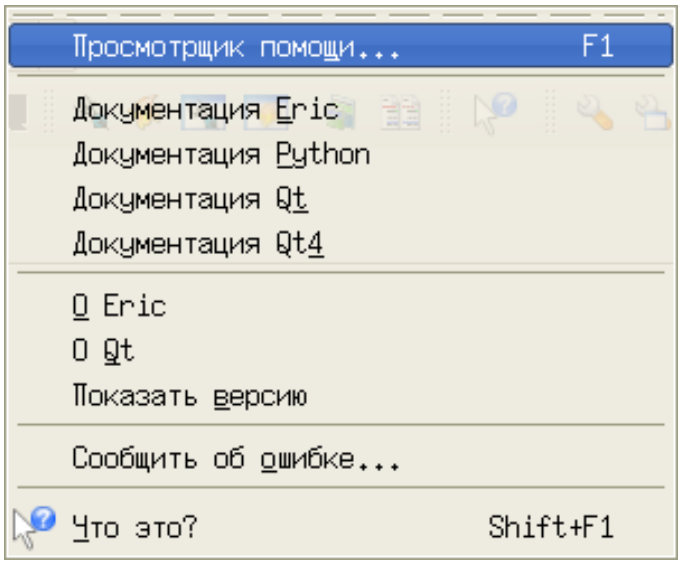

Рисунок 32. Меню «Помощь» IDE Eric

Выбор пункта «Просмотрщик помощи...» приводит к появлению пустого окна, в котором можно просматривать любые html-файлы. Назначение этого элемента пока непонятно.

Выбор пункта «Документация Eric» открывает документацию по пакету Eric (рис. 33), которая на данном этапе тоже не нужна.

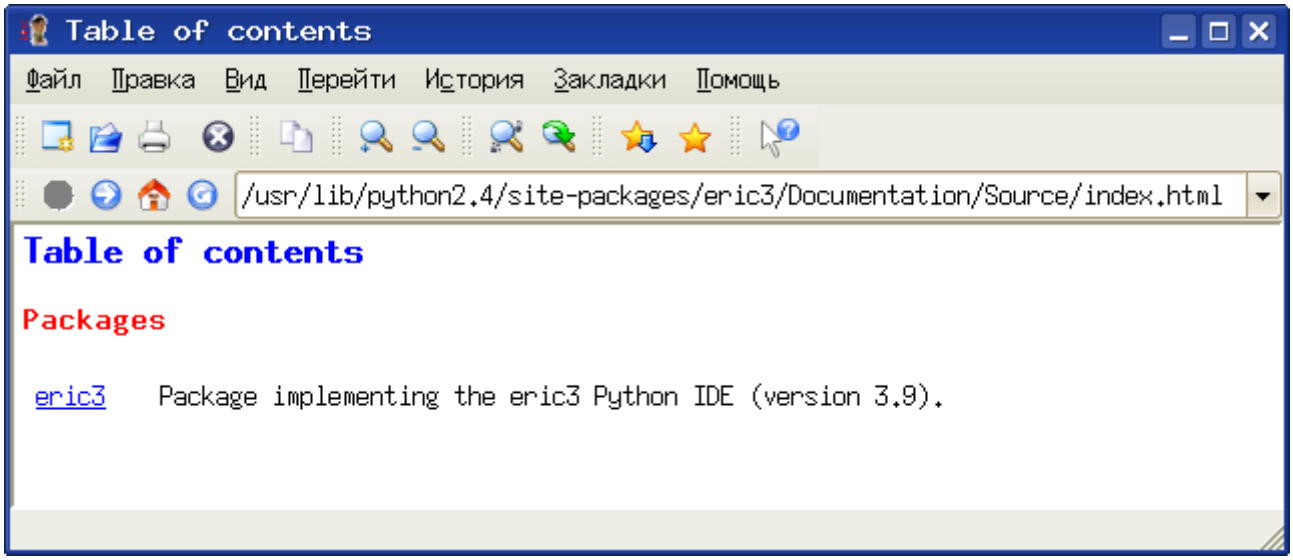

Рисунок 33. Окно документации Eric

Выбор пункта «Документация Python» открывает html-руководство по Python, написанное автором языка, но, к сожалению, на английском (рис. 34).

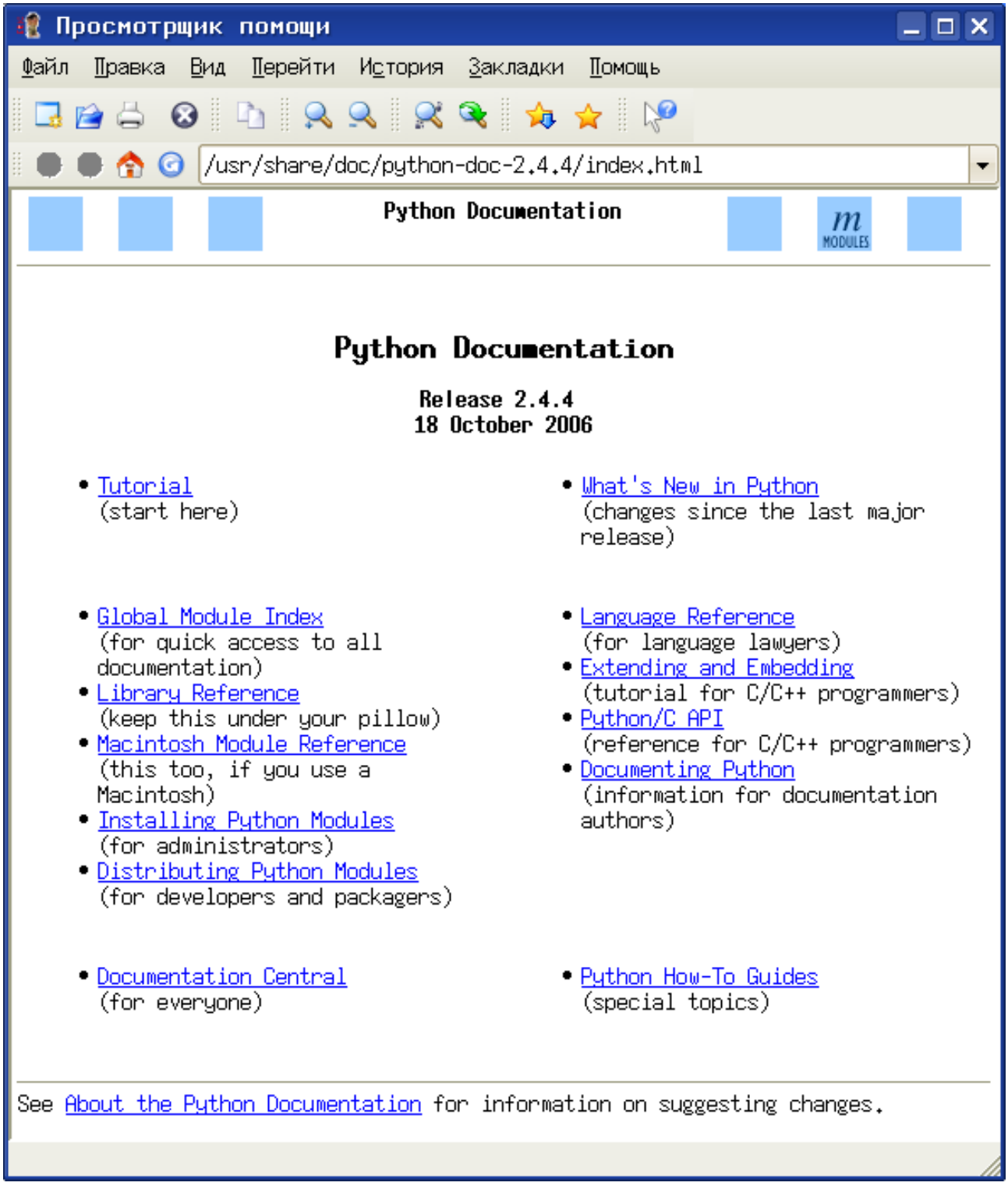

Рисунок 34. Документация по Python в IDE Eric

Сохранение и открытие файлов, запуск выполнения программ

Текст программы пишется в окне редактора, и пока он не сохранён, на ярлычке активной вкладки показан значок с изображением листа бумаги и карандаша (рис. 35).

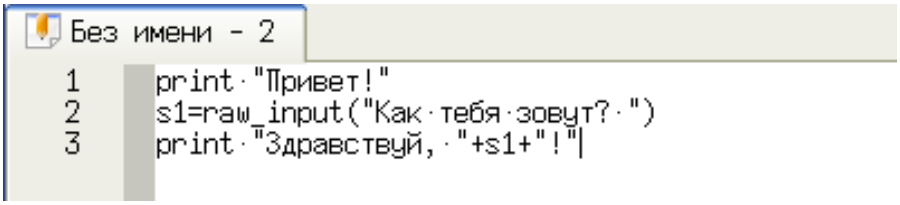

Рисунок 35. Текст программы в окне редактора Eric до сохранения

Для сохранения файла используется команда главного меню «Файл/Сохранить» (или «Файл/Сохранить как...» при первом сохранении), что равносильно использованию комбинации клавиш <CTRL>+S.

Выбор этой команды открывает диалог сохранения/открытия файла (рис. 36). Для открытия папки (каталога) в этом диалоге нужен двойной щелчок мышью независимо от настроек пользовательской среды в сеансе.

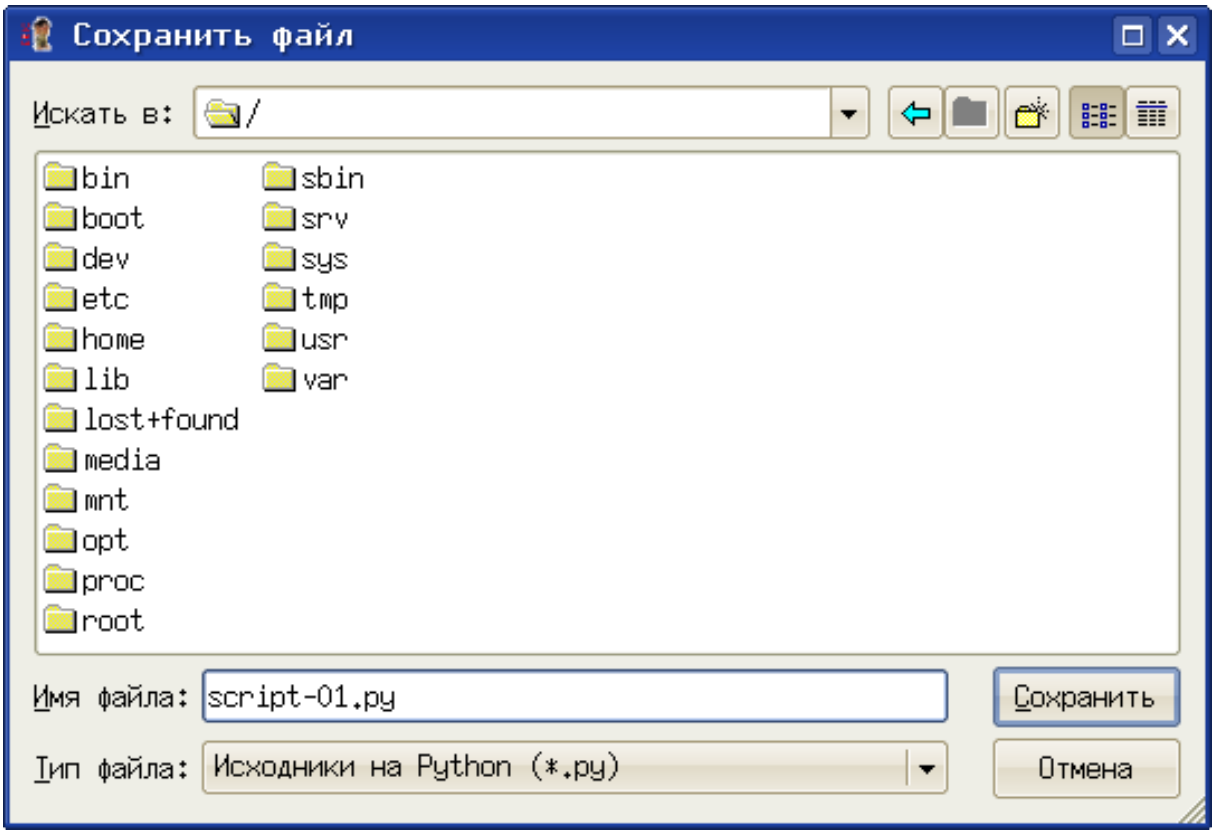

Рисунок 36. Диалог сохранения (открытия) файла в Eric

Для открытия файла удобно использовать список последних файлов (команда «Файл/Открыть недавние файлы»).

Для запуска программы на выполнение (точнее, на трансляцию и выполнение интерпретатором Python) используется клавиша <F2>, нажатие на которую приводит к открытию диалога запуска программы (сценария, скрипта — рис. 37).

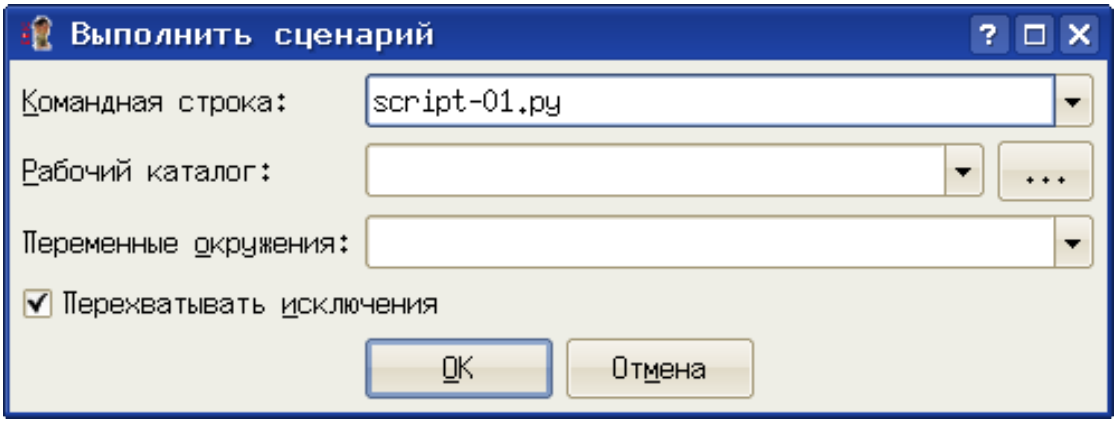

Рисунок 37. Диалог запуска выполнения программы в Eric

Здесь можно указать имя файла программы (программа не обязательно

должна быть открыта в окне редактора) и при необходимости, аргументы скрипта. Однако для программы, открытой в активной вкладке редактора, можно не указывать имя, а сразу нажать <ENTER>. Таким образом, кратчайший вариант запуска выполнения программы в IDE Eric — последовательное нажатие клавиш <F2> и <ENTER>.

Если перед запускам пропущен (забыт) этап сохранения изменений — Eric напомнит и позволит сохранить последнюю версию программы и выполнить её или выполнить предыдущий вариант программы (рис. 38).

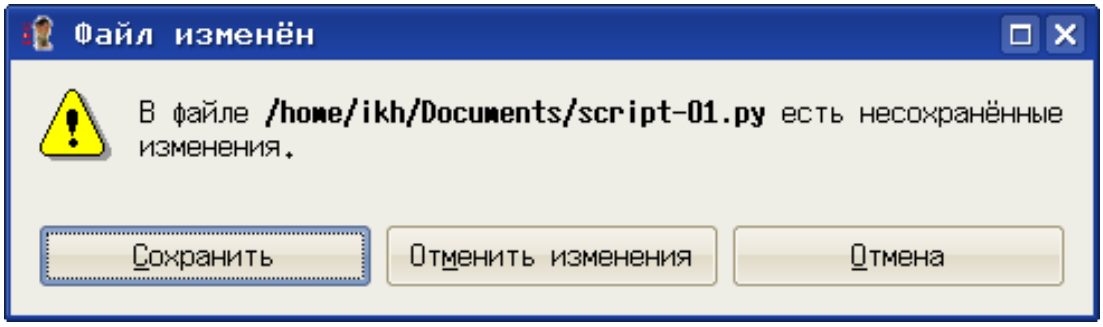

Рисунок 38. Пример реакции IDE Eric на запуск несохранённого варианта программы

Обработка ошибок

Синтаксические ошибки в IDE Eric выделяются подсветкой строки с ошибкой красным цветом (или другим цветом, поведение можно настроить) и изображением «жучка» («бага») слева от номера строки (рис. 39). Реакция на ошибку происходит сразу после перехода на новую строку в редакторе.

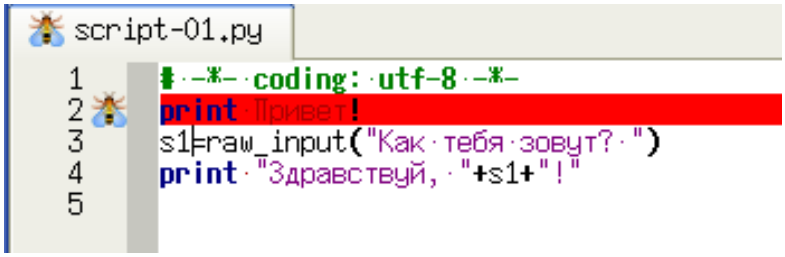

Рисунок 39. Обработка синтаксической ошибки в IDE Eric

Если ошибка не «проявилась» в процессе написания текста программы (например, допущена ошибка при редактировании существующего текста), Eric выдаст соответствующее сообщение при попытке запуска такой программы (рис.

40).

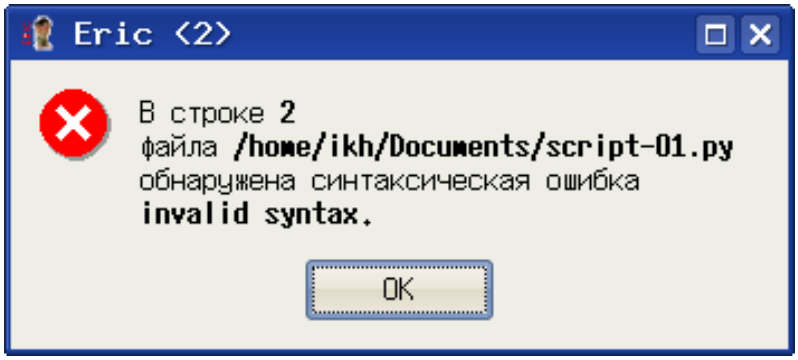

Рисунок 40. Оповещение Eric о синтаксической ошибке

Ошибки времени выполнения в Python называются «исключениями» (exception) и появляются при попытке выполнения недопустимых операций (типа деления на 0), вводе данных с несоответствующим типом или количеством элементов, а также при попытке использования функции с неверным типом или количеством аргументов.

Такие ошибки проявляются только при выполнении программы, причём уже в процессе выполнения. IDE Eric в таких случаях также выдаёт соответствующее сообщение (рис. 41) .

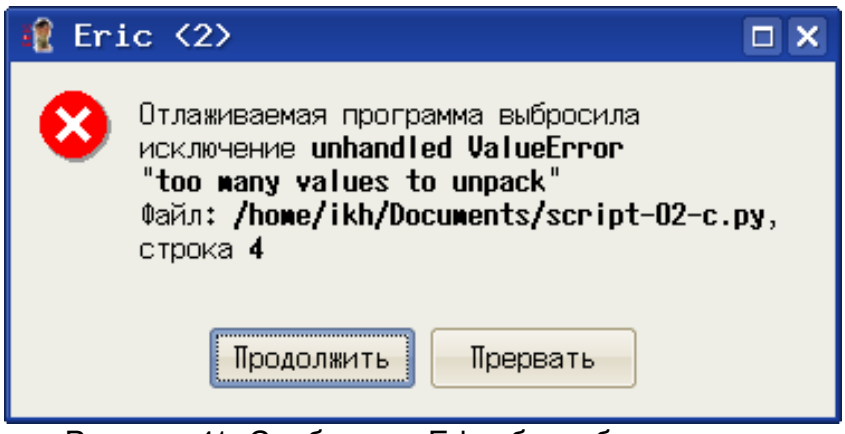

Рисунок 41. Сообщение Eric об ошибке времени выполнения («исключении»)

Сообщение, показанное на рис. 41, вызвано попыткой ввести в кортеж больше значений, чем должно в нём содержаться.

Семантические ошибки в IDE Eric (как и в любой другой среде разработки) никак не обнаруживаются и не обрабатываются, их можно обнаружить только по

результатам выполнения контрольных примеров.

Особенности работы с приложениями Tk и Tkinter

При запуске приложений `Tk` и `Tkinter` (графических) Geany открывает окно терминала и графическое окно. Для завершения работы программы нужно сначала зхакрывать графическое окно, а затем — окно терминала.

При работе в Eric можно получить неожиданный и неприятный эффект «зависания» при попытке закрытия приложения `Tkinter` кнопкой, вызывающей метод `quit` для экземпляра объекта `Tk` при запуске этого приложения из IDE Eric. Дело в том, что метод `quit` закрывает «родительское» окно `Tk` и останавливает интерпретатор Python, а при использовании IDE Eric остановить интерпретатор Python оказывается невозможно, поскольку он запущен процессом-родителем (которым является IDE Eric...). Поэтому при запуске примеров для `Tkinter` из IDE Eric настоятельно рекомендуется обратить внимание, что закрывать окно приложения `Tkinter` следует кнопкой закрытия окна пользовательской среды (крестик в правом углу в строке заголовка окна).

Ещё одна особенность касается отрисовки шрифтов в приложениях `Tkinter` в версиях Python до 2.5.x, которые выглядят, мягко говоря, некрасиво. Шрифт надписей для других интерфейсных элементов `Tkinter` можно исправить, а в области рисования (виджет `canvas`) так ничего и не удаётся сделать. Для исправления вида шрифтов интерфейсных элементов в `Tkinter` версий до 2.5.x можно сделать следующее.

1. Создать в «домашнем каталоге» (/home/<username>) с помощью любого текстового редактора (KWrite или Kate) файл с именем `.Xresources` (имя должно начинаться с точки!)
2. Записать в этом файле одну-единственную строку:

```
Tk*font: -*-terminus-*-r-*-*-12-*-*-*-*-*-*-*
```

3. Сохранить файл и перезагрузить сеанс работы (выйти из сеанса и войти снова)

Эта строка содержит название желаемого шрифта, записанное так, как оно формируется в программе `xfontsel`.

Использование примеров скриптов.

В состав данного комплекса включены примеры программ (скриптов) на Python, обеспечивающих решение задач, описанных в «Практикуме...». Можно существенно сэкономить время на занятиях, если использовать эти примеры в качестве основы для индивидуальных заданий. Особенно это качается больших (по количеству строк) программ для работы с графикой.

Все примеры, естественно, являются свободно распространяемым программным обеспечением (если это можно так назвать) на условиях GNU GPL2 и

выше.