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. Python относится к категории свободно распространяемого программного обеспечения (СПО). Это гарантирует во-первых, от каких-либо претензий относительно использования «интеллектуальной собственности», а во-вторых, от превращения его 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, Python является кросс-платформенным и поддерживает

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, а также среды разработки на Python — IDLE или Eric.

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

- eric
- 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 Eric с помощью менеджера пакетов Synaptic, остальные нужно установить «вручную»)

Весь дальнейший материал ориентирован в основном на IDE Eric.

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 Eric.

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, однако любая IDE лучше всего приспособлена для работы с одним конкретным языком, а с другими работает, так сказать, «факультативно».

Для работы с Python лучше всего приспособлена IDE, именуемая Eric, которую можно запустить с помощью K-меню («K-меню/Разработка/Eric»). Внешний вид окна Eric при первом запуске показан на рис. 1.

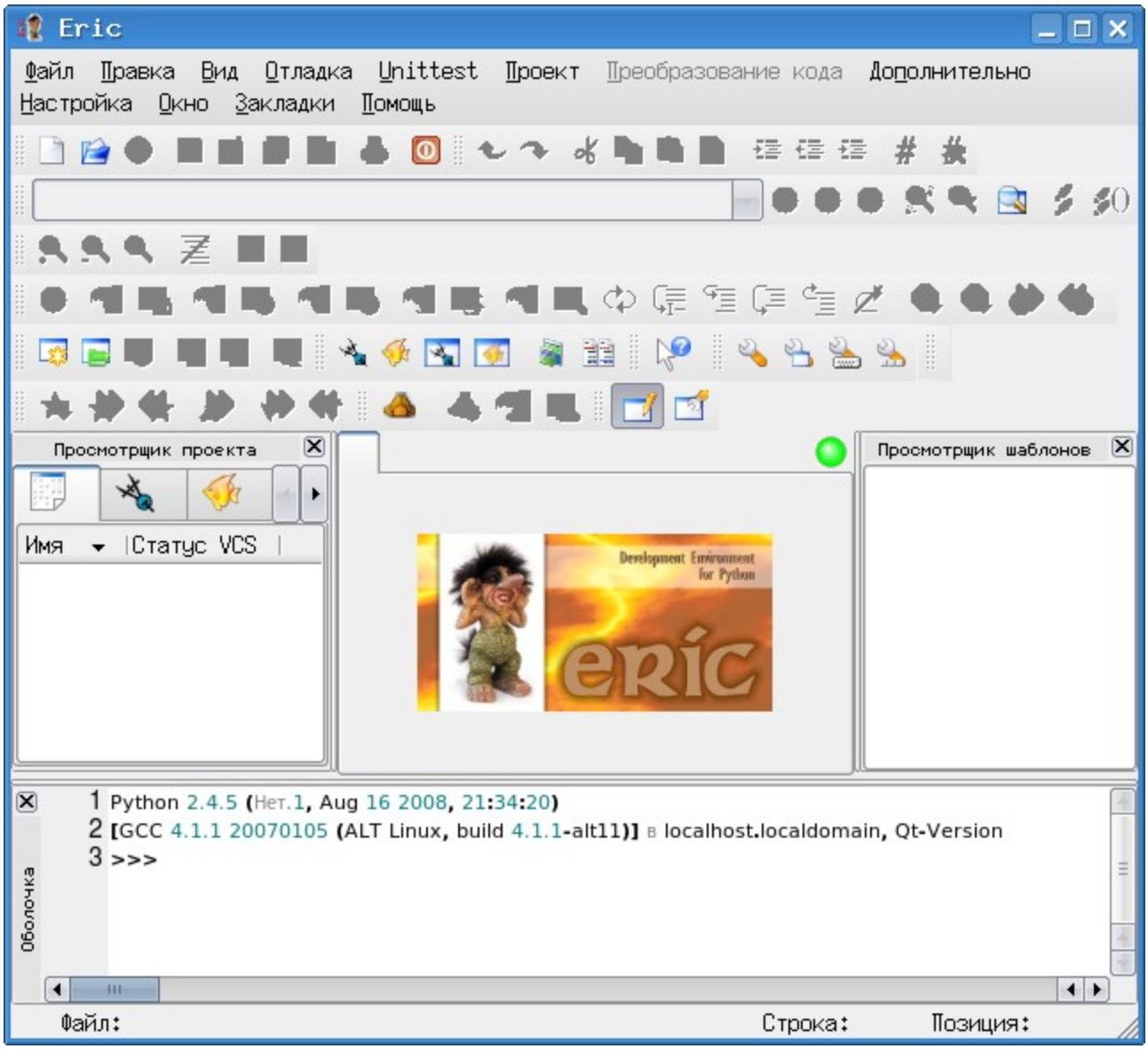

Рисунок 1. Внешний вид IDE «Eric»

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. 2).

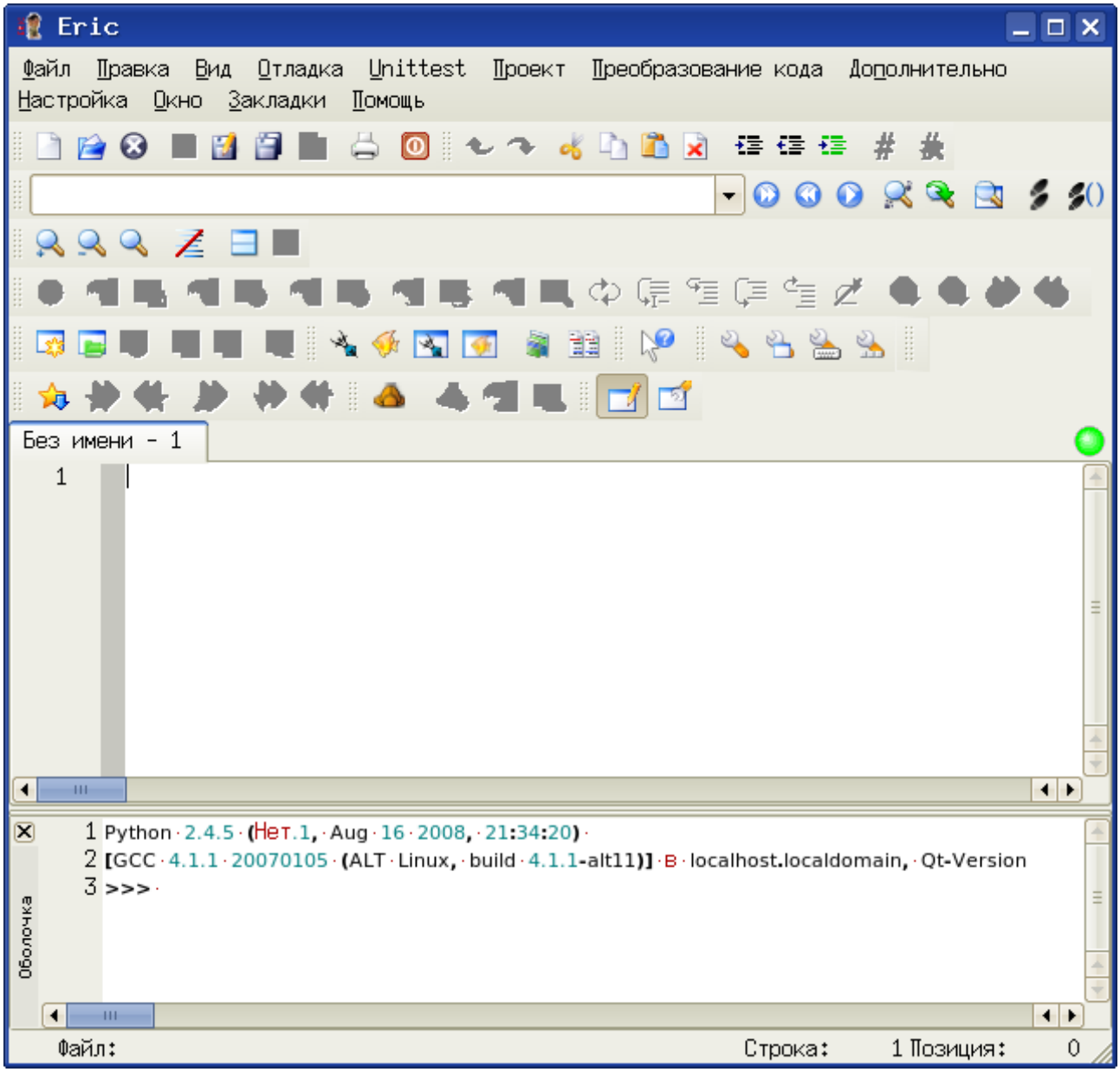

Рисунок 2. Упрощённый Eric — редактор и оболочка

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. 3, показан пример кода при настроенной подсветке синтаксиса.).

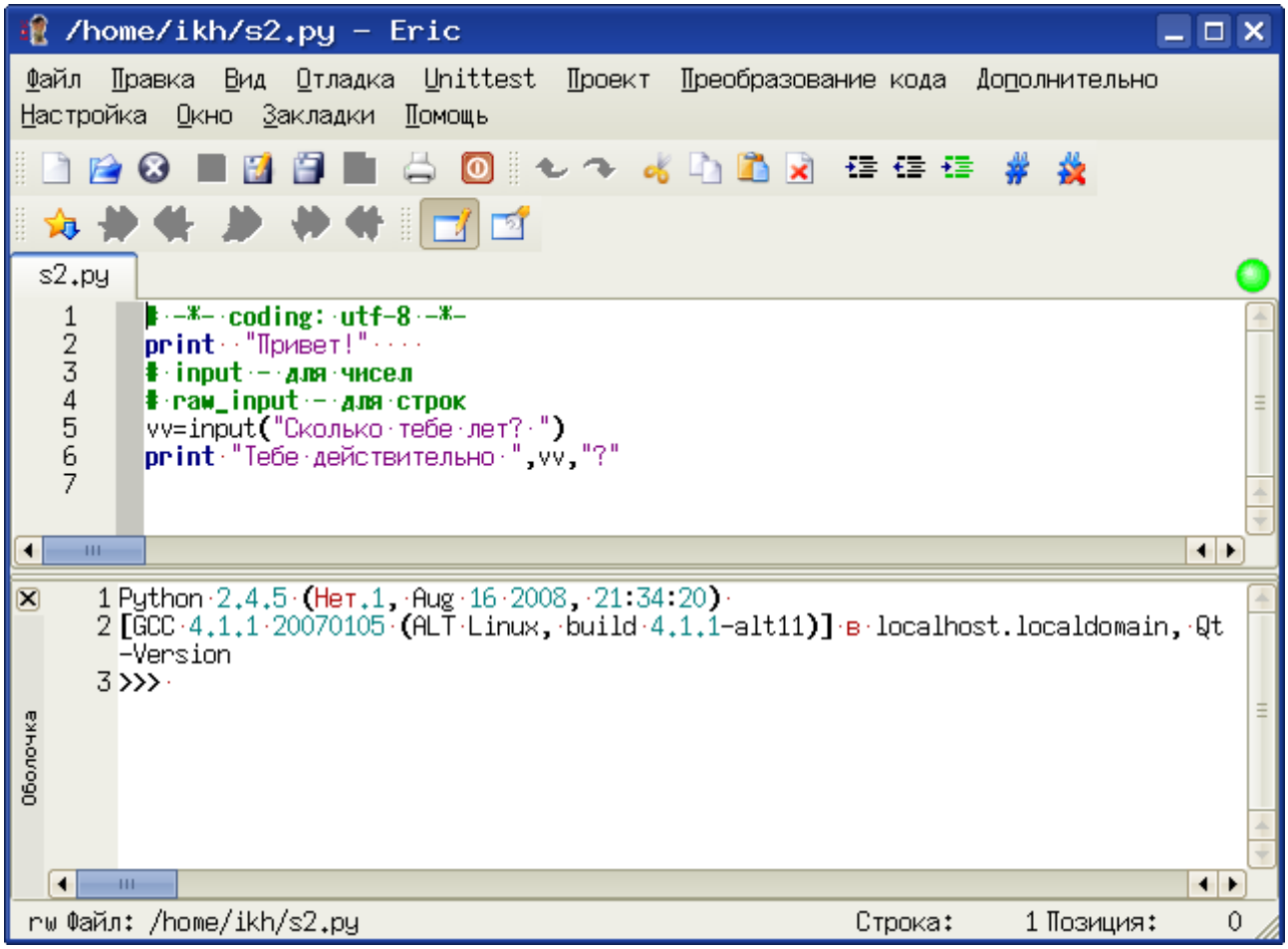

Рисунок 3. Максимально упрощённый вид IDE Eric

Если хочется что-то изменить во внешнем виде программы, можно использовать настройки предпочтений («Настройка/Предпочтения...» в главном меню окна Eric). Настроек очень много (рис. 4), но имеет смысл пока изменять только основные настройки редактора (как показано на рис. 4) и стиль редактора (рис. 5).

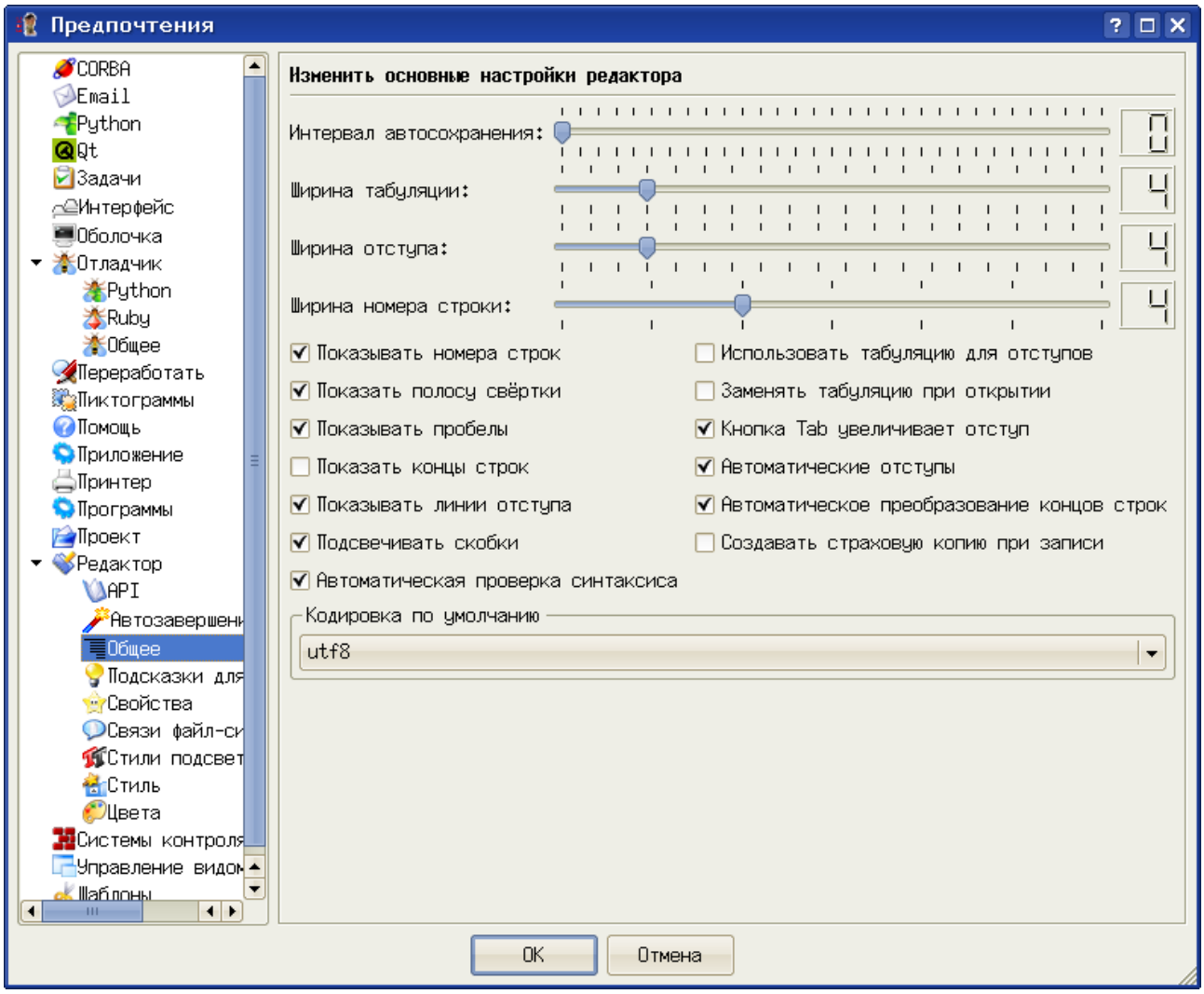

Рисунок 4. Основные настройки редактора в IDE Eric

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. 5) 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»).

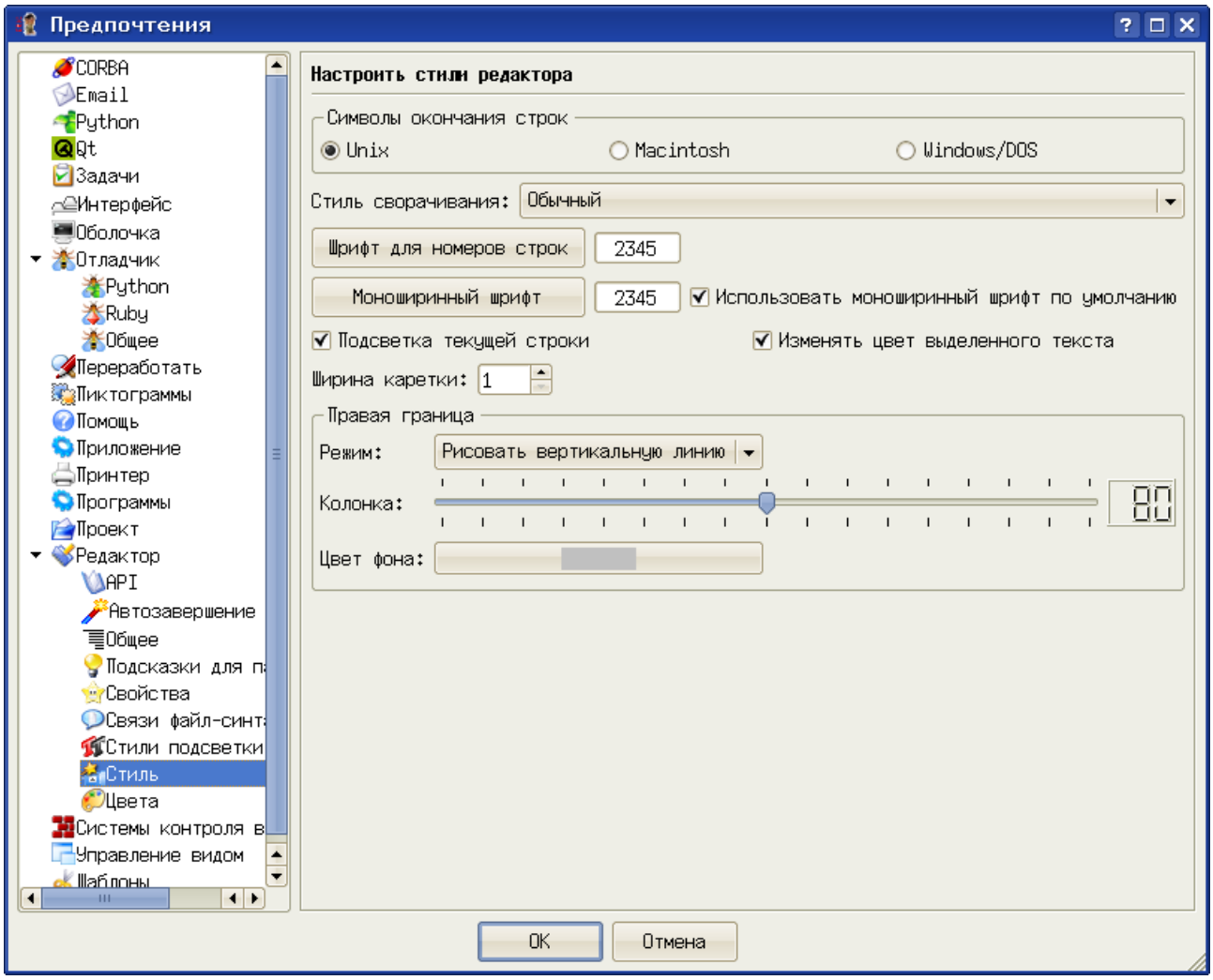

Рисунок 5. Настройка стиля редактора в IDE Eric

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. 6).

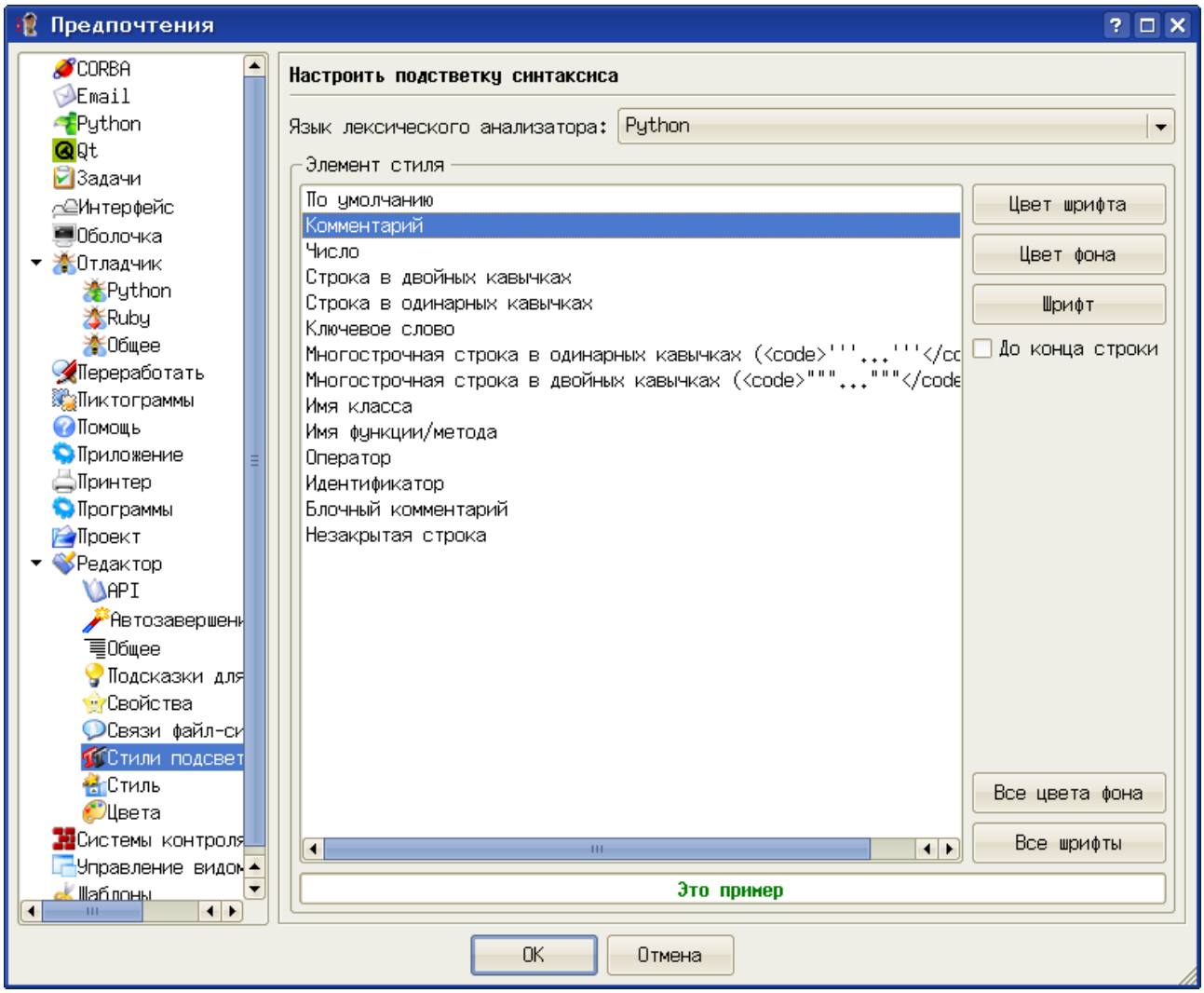

Рисунок 6. Диалог настройки подсветки синтаксиса

Подключение документации и её использование

Очень важно правильно настроить параметры помощи и документации (диалог «Помощь», рис. 7). 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.

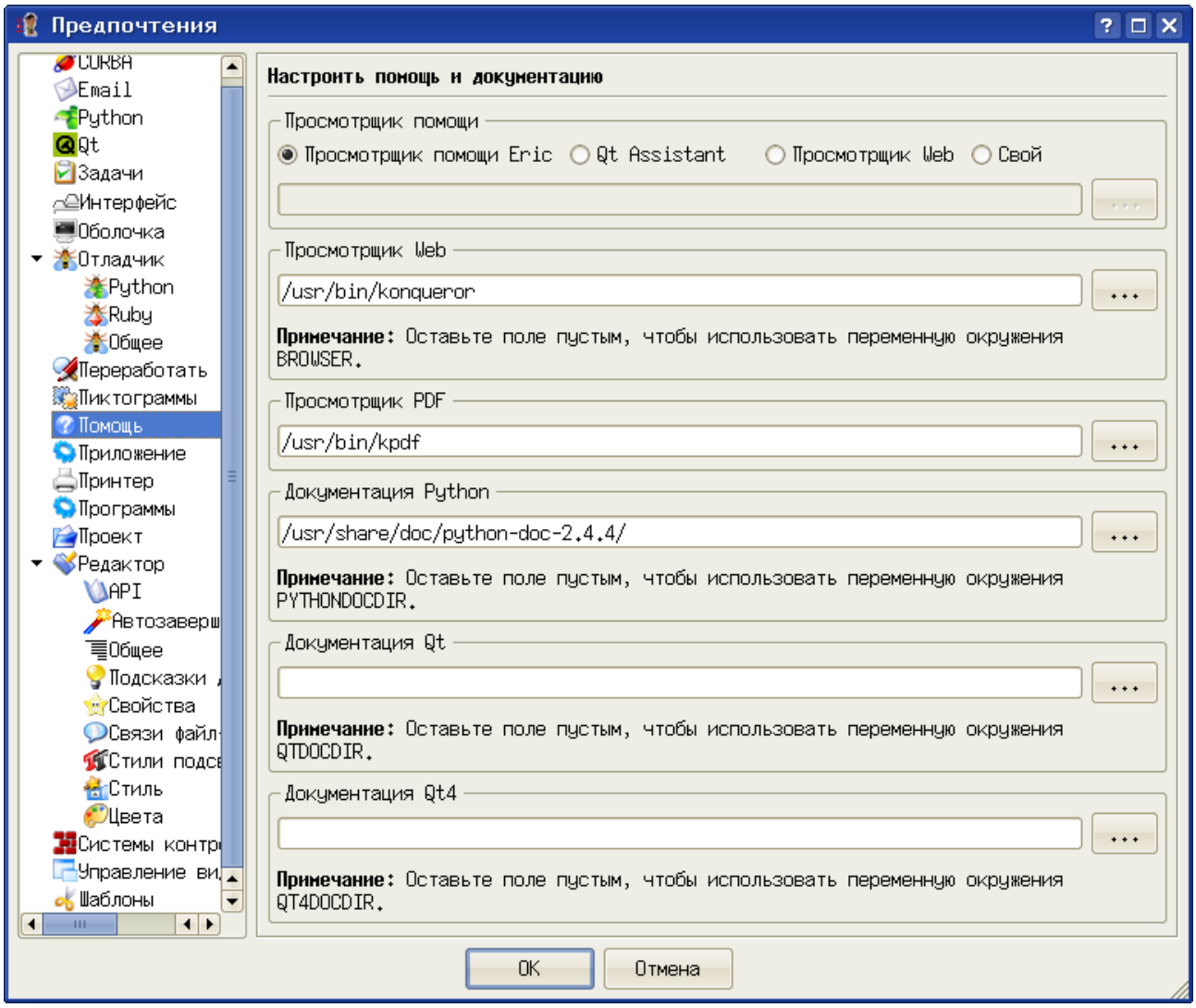

Рисунок 7. Настройка помощи и документации

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. 8).

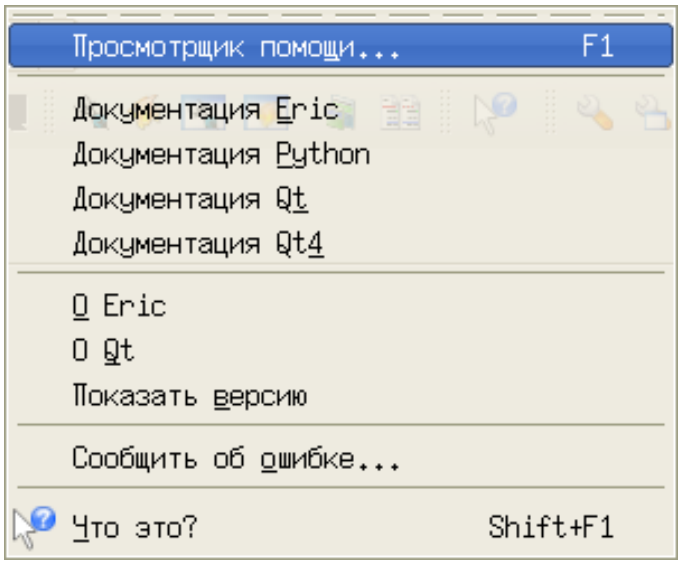

Рисунок 8. Меню «Помощь» IDE Eric

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. 9), которая на данном этапе тоже не нужна.

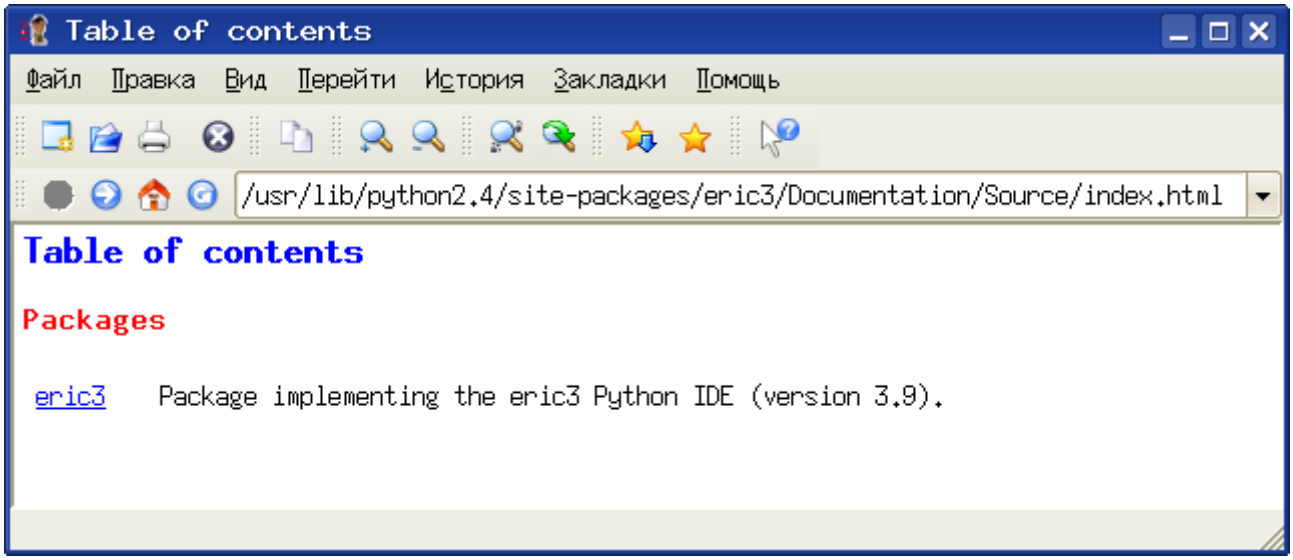

Рисунок 9. Окно документации Eric

Выбор пункта «Документация Python» открывает html-руководство по Python, написанное автором языка, но, к сожалению, на английском (рис. 10).

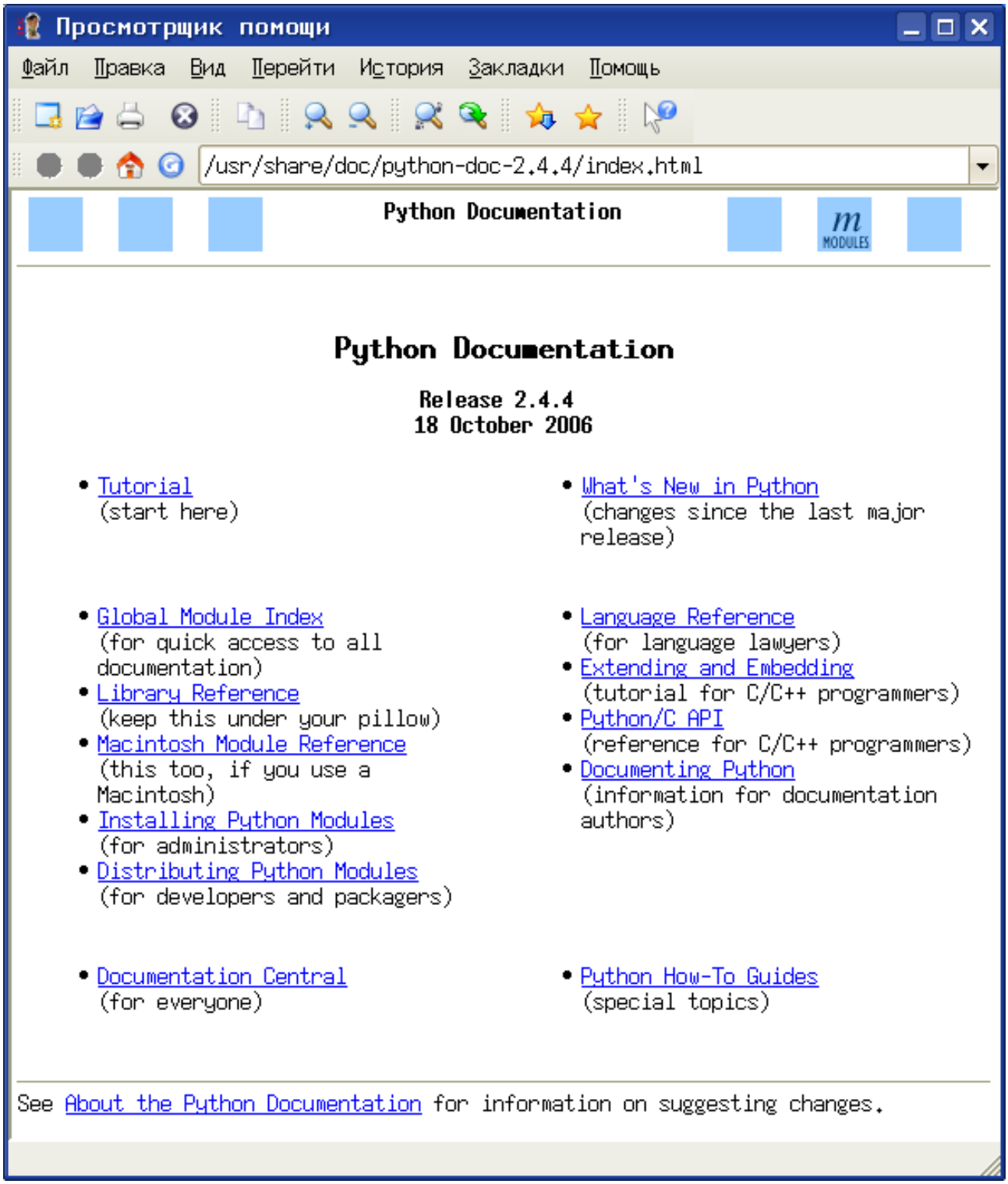

Рисунок 10. Документация по Python в IDE Eric

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

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. 11).

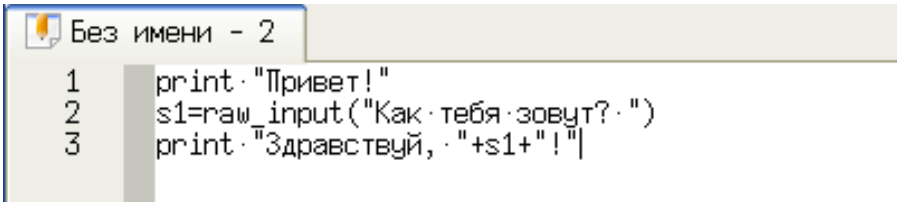

Рисунок 11. Текст программы в окне редактора до сохранения

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 диалог сохранения/открытия файла (рис. 12). Для открытия папки (каталога) в этом диалоге нужен двойной щелчок мышью независимо от настроек пользовательской среды в сеансе.

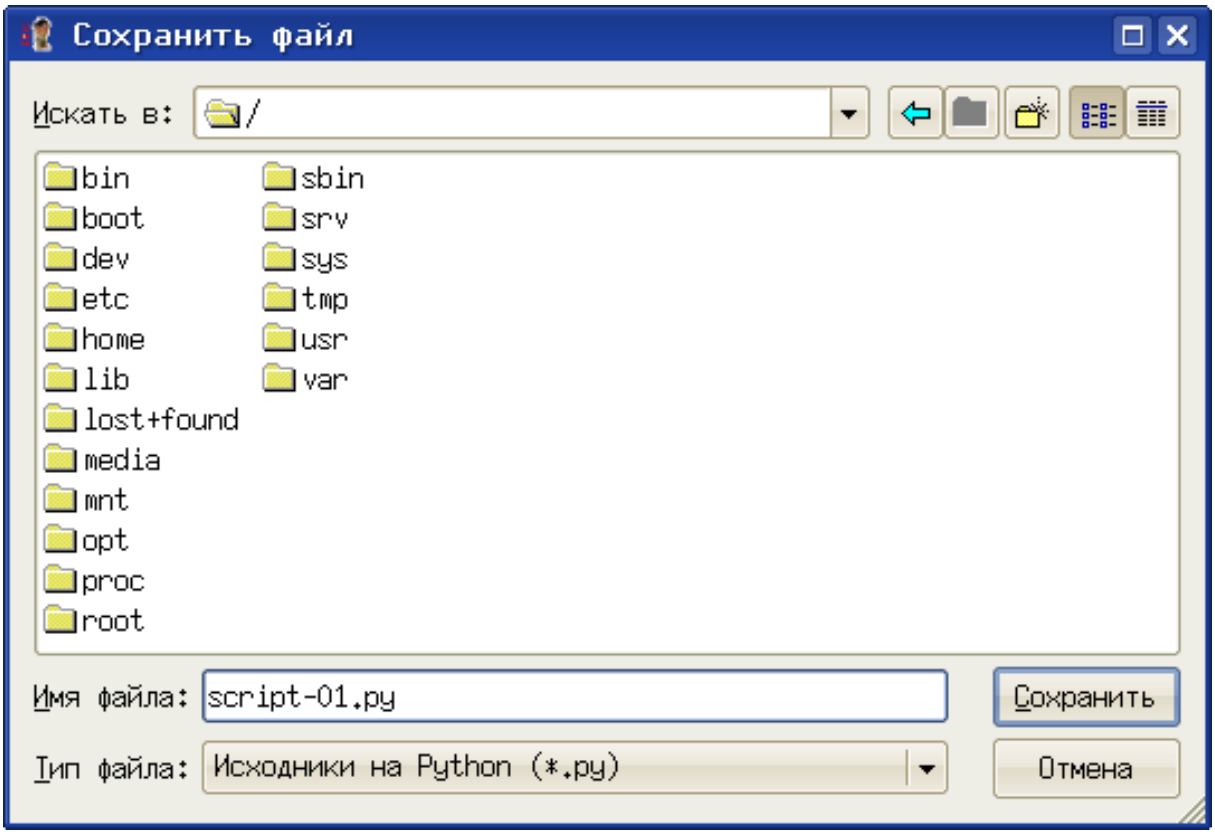

Рисунок 12. Диалог сохранения (открытия) файла

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. 13).

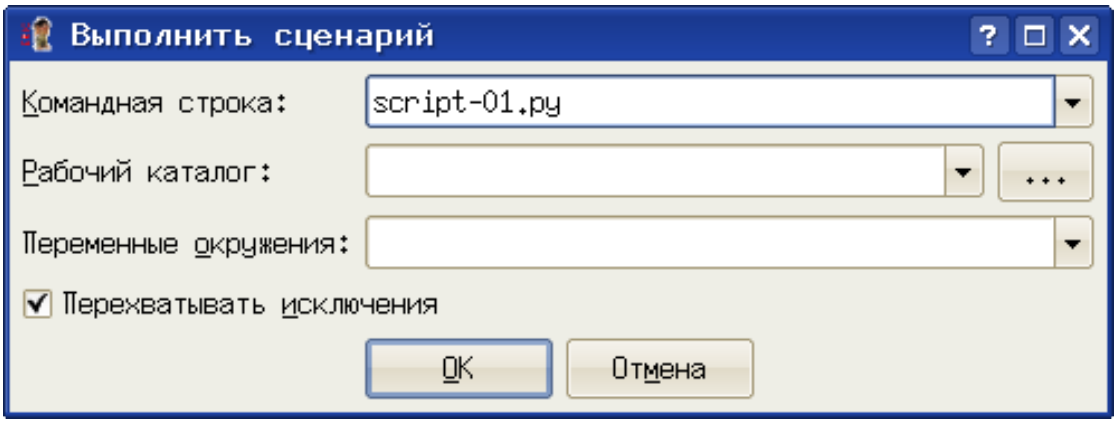

Рисунок 13. Диалог запуска выполнения программы

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. 14).

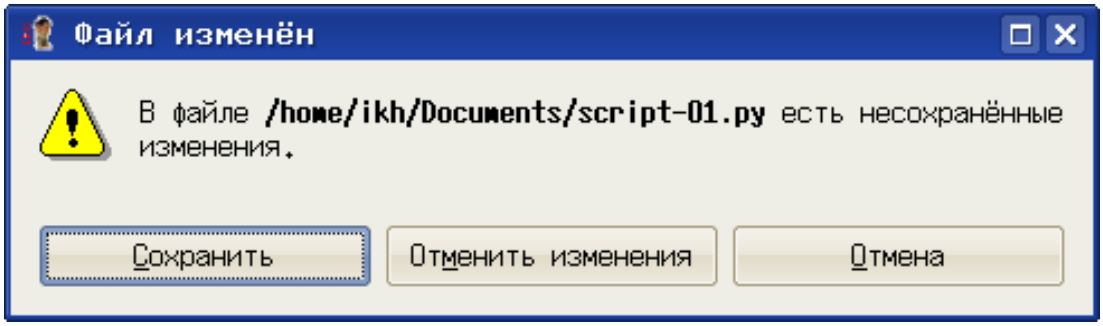

Рисунок 14. Пример реакции IDE Eric на запуск несохранённого варианта программы

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. 15). Реакция на ошибку происходит сразу после перехода на новую строку в редакторе.

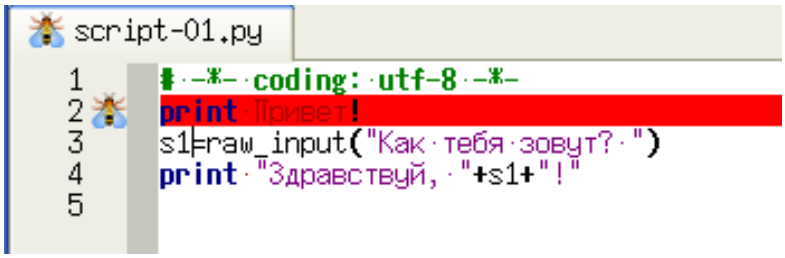

Рисунок 15. Обработка синтаксической ошибки в IDE Eric

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. 16).

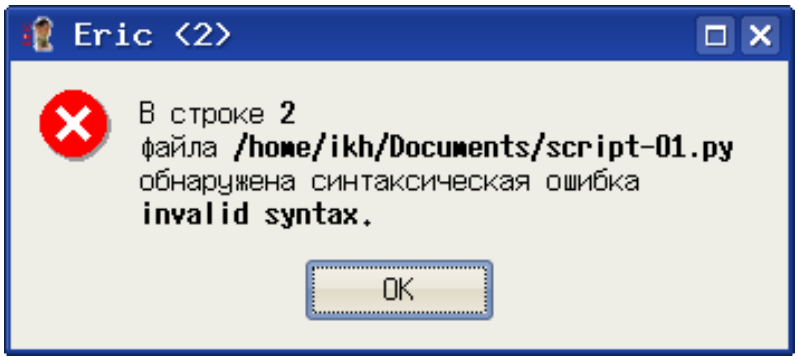

Рисунок 16. Оповещение о синтаксической ошибке

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. 17) .

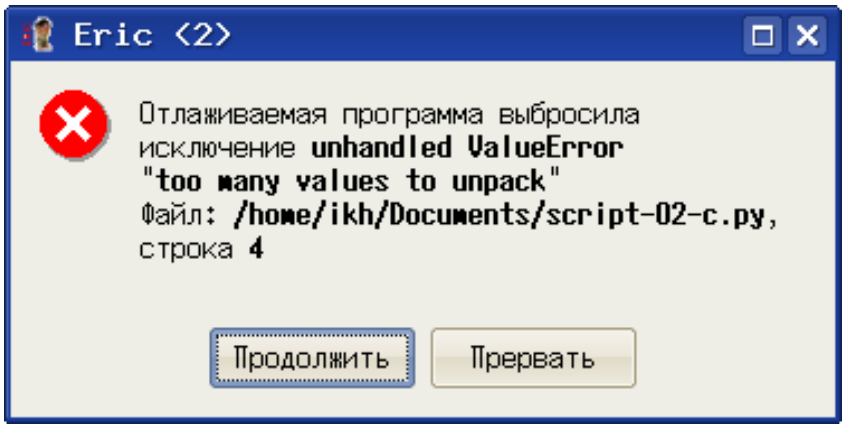

Рисунок 17. Сообщение об ошибке времени выполнения («исключении»)

Сообщение, показанное на рис. 17, 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 Tkinter

Неожиданный и неприятный эффект «зависания» можно получить 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).

Вторая особенность касается отрисовки шрифтов в приложениях Tkinter. В версиях Python до 2.5.x эти шрифты выглядят, мягко говоря, некрасиво, причём в области рисования (виджет `canvas`) так ничего и не удаётся сделать. Однако шрифт надписей для других интерфейсных элементов `Tkinter` можно исправить следующим образом.

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.