Введение в Python и Eric

Иван Хахаев, 2009

Простой ввод и вывод

Приступим, наконец к созданию программ. При работе с компьютером пользователь всегда что-то сообщает компьютеру (вводит данные), компьютер с этими данными производит какие-то операции и показывает результат этих операций (выводит результат). Даже если пользователь щёлкает мышкой по разным местам экрана, он всё равно сообщает компьютеру данные (координаты позиции указателя мышки на экране).

Пока что будем работать с простейшими вариантами ввода и вывода данных — вводить данные будем с клавиатуры, а выводить на экран в окне выполнения программы.

Напишем в окне редактирования текст, показанный на рис. 1.

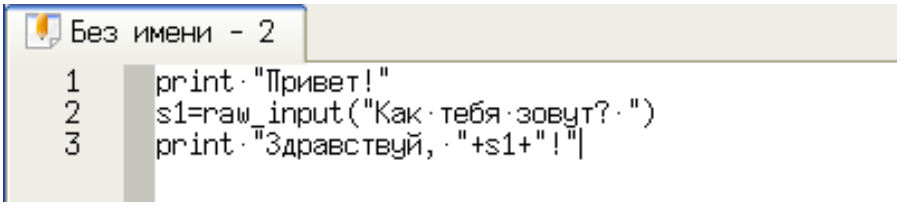

Рисунок 1. Текст первой программы на Python

Значок с изображением листа бумаги и карандаша на ярлычке вкладки над текстом означает, что текст редактируется и последние изменения не сохранены. Перед попыткой выполнения написанной программы необходимо сохранить её текст.

Для сохранения текста программы в файле найдём в верхней панели инструментов кнопку «Сохранить» и щёлкнем по ней левой кнопкой мыши, вызвав диалог сохранения файлов (рис. 2). Заметим, что аналогичный эффект даёт выбор команды «Файл/Сохранить» в главном меню или использование комбинации клавиш <CTRL>+<S>.

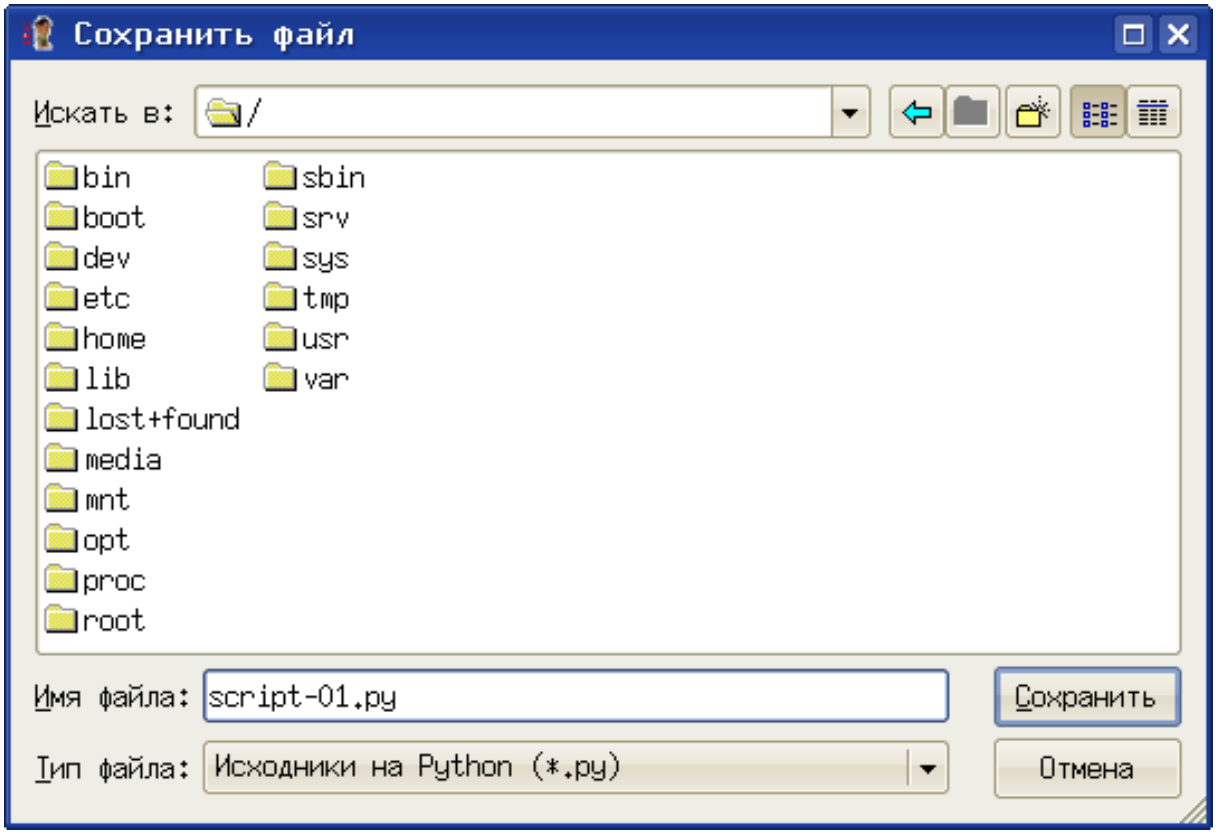

Рисунок 2. Диалог сохранения файлов в Eric

В строке «Имя файла:» впишем имя (например, `script-01.py`), выберем каталог (папку), в которую можно записывать файлы (например, папку `Documents` в «Домашнем каталоге») и нажмём на кнопку «Сохранить»(для входа в папку нужно дважды щёлкнуть левой кнопкой мыши по ней). После сохранения текста программы в файле увидим изменения в редакторе среды Eric (рис. 3).

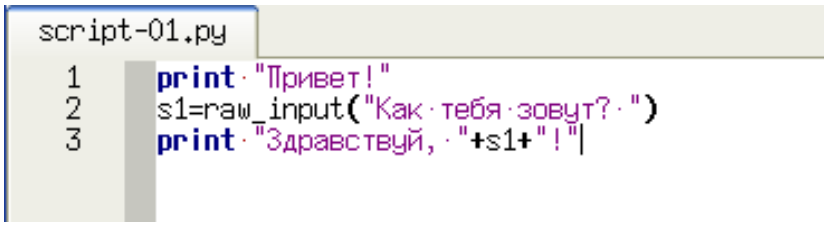

Рисунок 3. Результат работы лексического анализатора

Когда мы объявили, что данный текст является программой на Python (указав `.py` в имени сохраняемого файла), Eric произвёл разбор этого текста, нашёл в нём слова и конструкции, характерные для языка Python и сделал подсветку синтаксиса. 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. 4).

Обратим внимание, что количество активных кнопок в панелях инструментов IDE Eric

увеличилось после того, как был набран и сохранён текст программы.

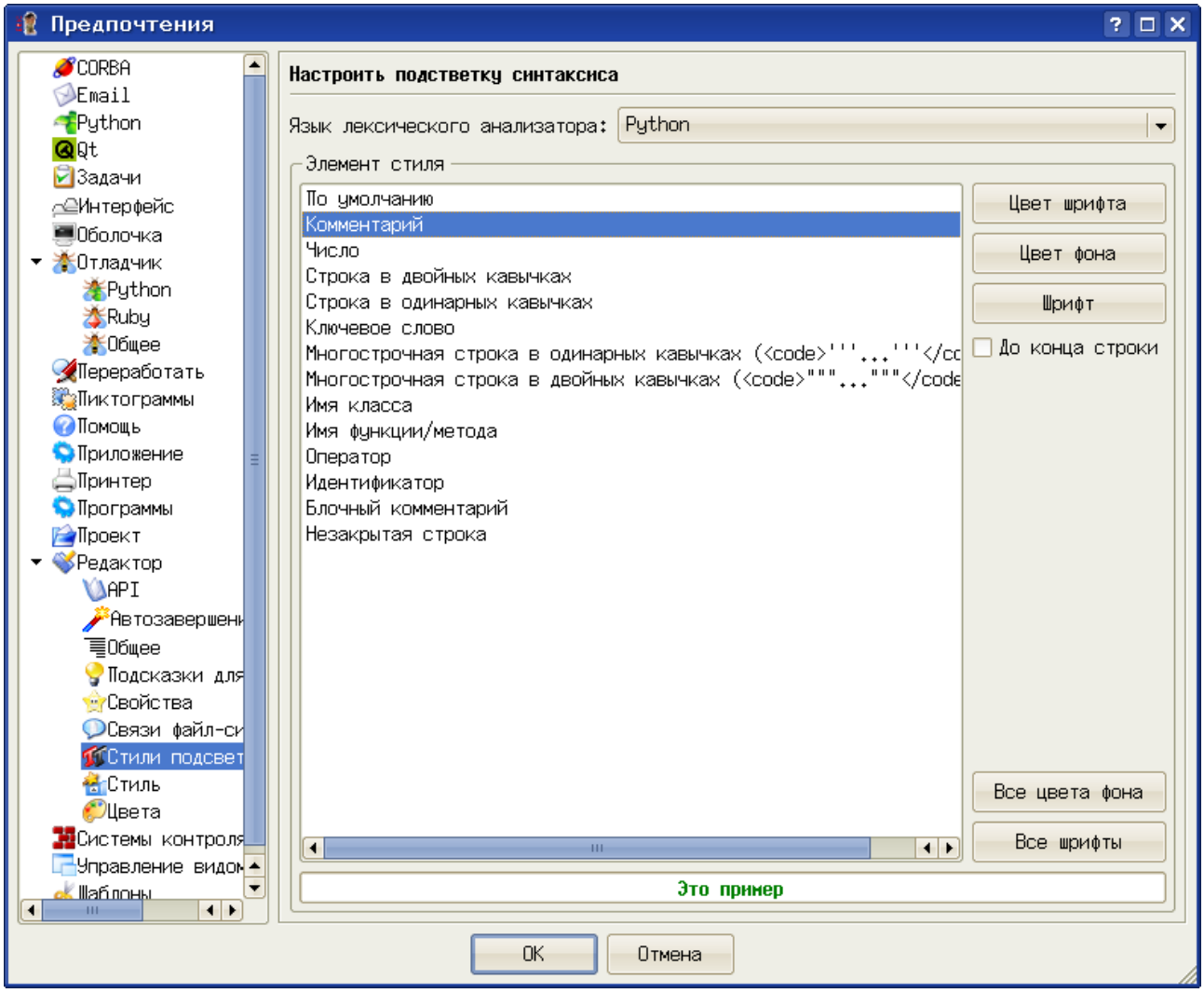

Рисунок 4. Диалог настройки подсветки синтаксиса

Наконец, настало время выполнить нашу программу. Для запуска программы на выполнение нажмём клавишу <F2> и получим диалог выполнения сценария (рис. 5).

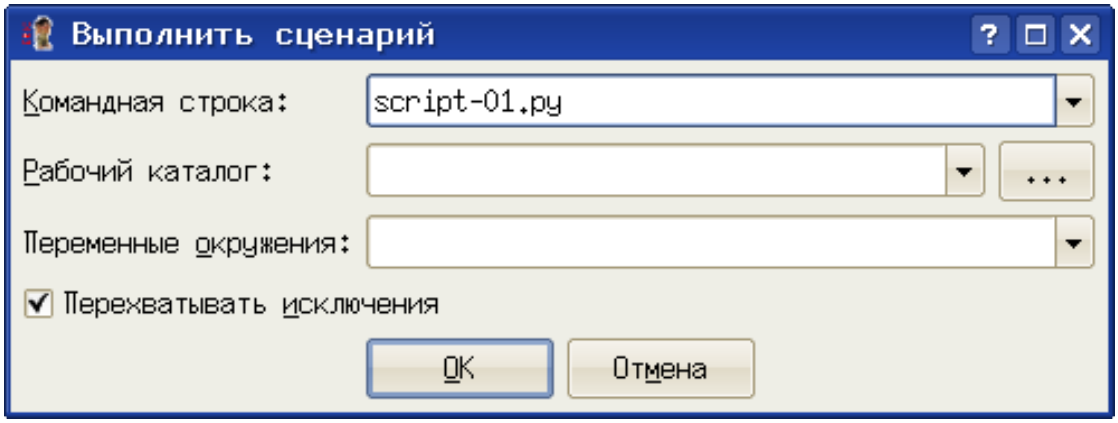

Рисунок 5. Диалог выполнения сценария

В первой строке диалога («Командная строка:») напишем имя нашей программы (`script-01.py`), нажмём на «OK» и в окне выполнения (панель «Оболочка») увидим результат (рис. 6).

Однако для упрощения и ускорения запуска программы можно не писать имя программы, а просто нажать <F2> и <ENTER>.

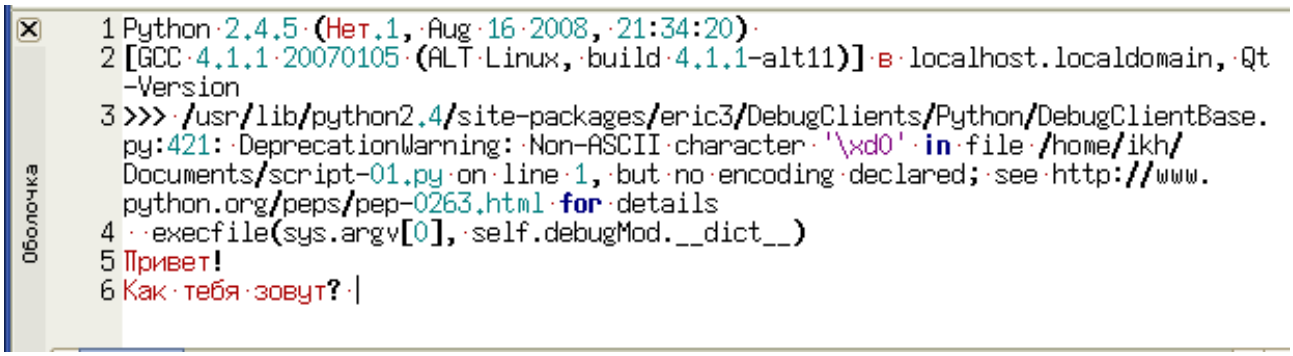

Рисунок 6. Результат первого запуска программы

Программа что-то сделала, вывела вопрос «Как тебя зовут?» и остановилась, причём после вопросительного знака мигает курсор. Это означает, что программа ждёт реакции пользователя, в данном случае — ввода какого-то имени. Напишем в окне выполнения ответ на вопрос (например, слово «Yodha» и нажмём клавишу <ENTER>. Программа продолжит и закончит работу (рис. 7).

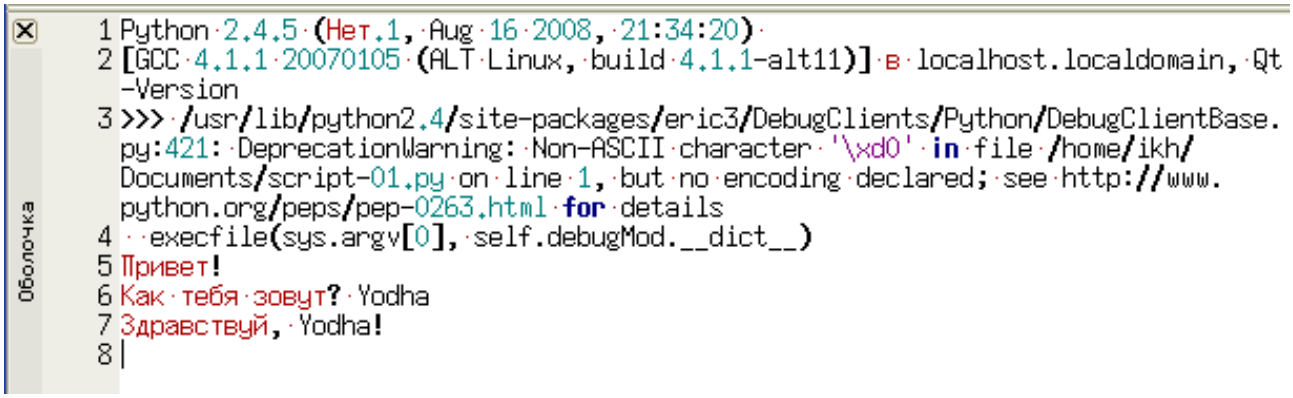

Рисунок 7. Демонстрация работы программы

Как убедиться в том, что программа закончила работу? В восьмой строке снова мигает курсор, ожидая ввода. Нажмём клавишу <ENTER> и через секунду-другую увидим, что в девятой строке появился значок «>>>», который называется «приглашение оболочки Python». Таким образом Python сообщает, что он готов к дальнейшим действиям.

Чтобы разобраться в происходящем, нужно одновременно смотреть на текст программы и на результат её выполнения (вот и пригодилась IDE!). Разберём по строчкам текст программы и также по строчкам — результат её работы (номера строчек написаны слева на сером поле).

Итак, первая строка программы:

```
print "Привет!"
```

В этой строке записана инструкция (команда) «Напечатать строку «Привет!»» (print — печатать, отсюда пошло слово «принтер»). Строка — это произвольный набор букв, цифр и всяких других значков, которые можно найти на клавиатуре. Строки обязательно должны записываться в двойных или одиночных кавычках (чуть позже проверим, что произойдёт, если кавычки убрать). Компьютер (а точнее, Python) честно выполнил эту команду (см. строку номер 5 в окне выполнения).

Вторая строка программы:

```
s1=raw_input("Как тебя зовут? ")
```

Здесь Python получает указание написать строку «Как тебя зовут?» (6-я строка в окне выполнения), дождаться, пока пользователь что-то на этот вопрос ответит и нажмёт клавишу <ENTER>, а то, что ответил пользователь, сохранить в переменной `s1` в виде строки.

Дело в том, что процесс написания компьютерной программы похож на решение задачи по физике — вначале пишется решение в общем виде, в котором различные величины (переменные) обозначаются буквенно-цифровыми сочетаниями, а потом подставляются конкретные значения (данные). Такое решение (программа) оказывается правильным при самых разных данных. Кстати, в компьютерных программах, также как и в задачах по физике, кроме переменных могут использоваться константы. Только в физике константы связаны с единицами измерений и определёнными явлениями природы, а программах константы просто остаются неизменными в ходе выполнения программы.

Инструкция `raw_input()` означает, что Python ждёт, что пользователь введёт строку (последовательность букв, цифр и других значков с клавиатуры).

На самом деле в этой строке программы записано ***присваивание*** результата выполнения

инструкции `raw_input()` переменной `s1` (иначе говоря, до этой строки переменной `s1` не существовало, а после выполнения этой строки переменная `s1` появилась и означает то, что сообщил компьютеру пользователь, причём при каждом следующем запуске это могут быть самые разные строки).

Наконец, третья строка программы

```
print "Здравствуй, "+s1+"!"
```

означает, что должна быть напечатана строка, состоящая из слова «Здравствуй, », к которому дописано содержимое (значение) переменной `s1`, к которому в свою очередь дописывается восклицательный знак (7-я строка окна вывода). Здесь «+» означает не арифметические сложение, а ***конкатенацию*** (сцепление) строк (его можно рассматривать как «сложение» в том смысле, в котором слова «складываются» из букв). Понятно, что арифметические свойства операции сложения к «сложению» строк никак не подходят.

Итак, мы обнаружили, что трём строкам программы по странной случайности соответствуют три строки в окне выполнения. Кроме того, выяснилось, что знак «>>>» в окне выполнения сообщает о готовности Python выполнять инструкции (команды, программы).

Спрашивается, что же означают «протестные предупреждения» и упоминания каких-то «peps» в 3-й строке окна выполнения? Это будет всегда?

И ещё. А что же у нас в 1-й и 2-й строках окна выполнения?

Отвечать начнём с конца.

В первой и второй строках окна выполнения Python рассказывает о себе (информация для специалистов). После этого рассказа Python в 3-й строке выдаёт приглашение оболочки, поэтому содержимое 3-й строки — это уже реакция на наши действия.

Однако мы не совершали никаких действий, кроме запуска программы, состоящей из трёх разобранных выше строк. Неужели в этих строках содержится что-то, вызывающее «протестные предупреждения» (DeprecationWarning)? Ключом к разгадке послужит продолжение строки предупреждения после слов DeprecationWarning: - «`Non-ASCII character ... in file .... script-01.py on line 1`». Это означает, что в первой строке программы Python наткнулся на символы, не входящие в стандартную таблицу символов для обмена информацией в Америке (American Standard Characters for Information Interchange — ASCII). Очевидно, что символами, не являющимися стандартными для Америки, оказываются русские буквы (символы кириллицы), используемые как аргумент функции `raw_input()`. Чтобы избавиться от таких «протестных предупреждений», слегка изменим текст программы, добавив в него магические слова (рис. 8).

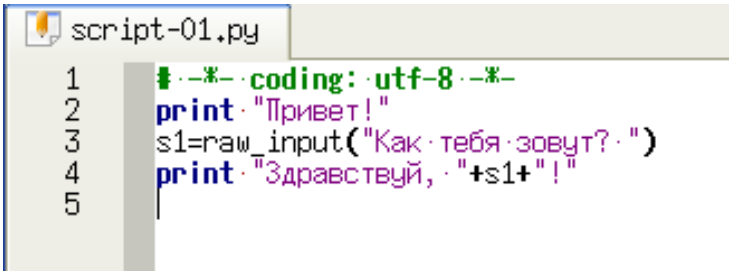

Рисунок 8. Модифицированная программа

Теперь в первой строке содержится указание, что для этой программы используется набор символов UTF-8, а не ASCII. В начале этой строки стоит символ «#», который для Python означает ***комментарий***. Комментарий не воспринимается как часть программы, но и не игнорируется вовсе. Комментарии полезны для понимания смысла программы, поэтому в дальнейшем будем их активно использовать.

Сохранив и перезапустив эту программу (вызвав диалог выполнения сценария кнопкой <F2>), увидим, что в окне выполнения все лишние непонятные слова исчезли (рис. 9.

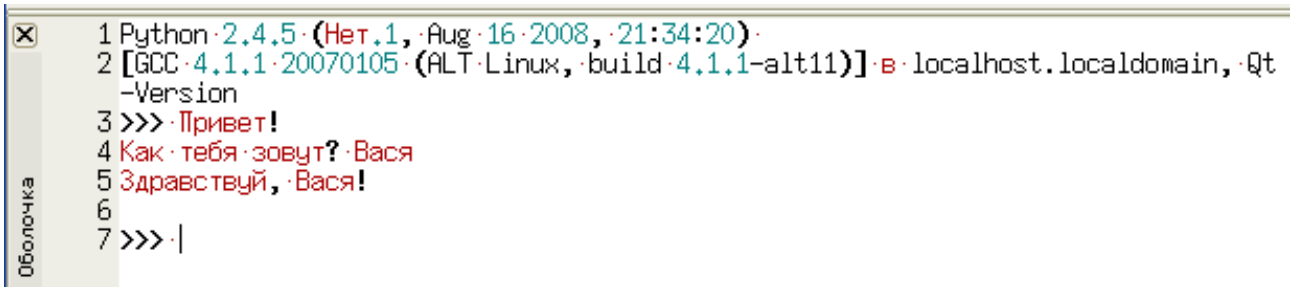

Рисунок 9. Результат работы модифицированной программы

Как видим, программе можно сообщать и строки, содержащие символы кириллицы и теперь (после указания используемого набора символов) всё нормально работает.

Проведём ряд экспериментов.

Сначала попробуем в ответ на вопрос «Как тебя зовут?» ввести какое-то число (в самом деле, может же существовать робот с именем «1234»?).

Легко убедиться, что программа в этом случае работает нормально.

Вопрос для самостоятельного изучения: А что будет, если вместо числа программе ввести арифметическое выражение типа «2+3»? А почему?

Следующий эксперимент. В строке

```
print "Привет!"
```

уберём кавычки и запустим программу на выполнение, нажав <F2> и <ENTER>.

Сначала получим предупреждение о том, что текст программы перед запуском на выполнение не был сохранён (рис. 10).

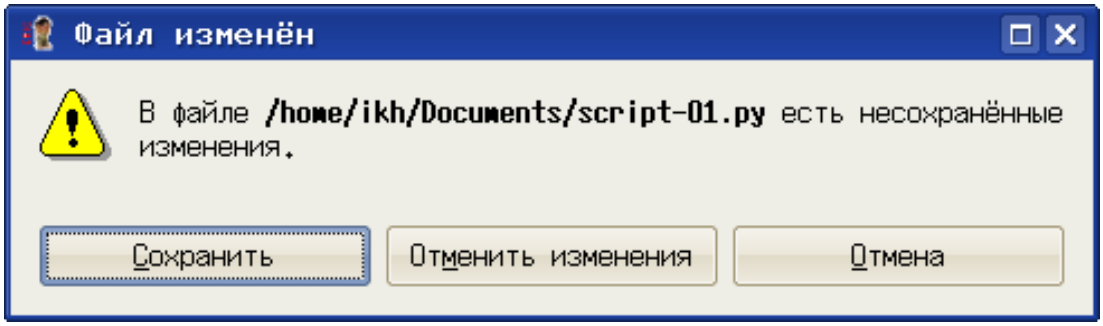

Рисунок 10. Предупреждение о запуске программы с несохранёнными изменениями

Здесь мы имеем возможность сохранить изменения, вернуться к прежнему варианту (кнопка «Отменить изменения» или отменить выполнение программы (кнопка «Отменить»). Но мы смело жмём на кнопку «Сохранить»... и получаем сообщение об ошибке (рис. 11).

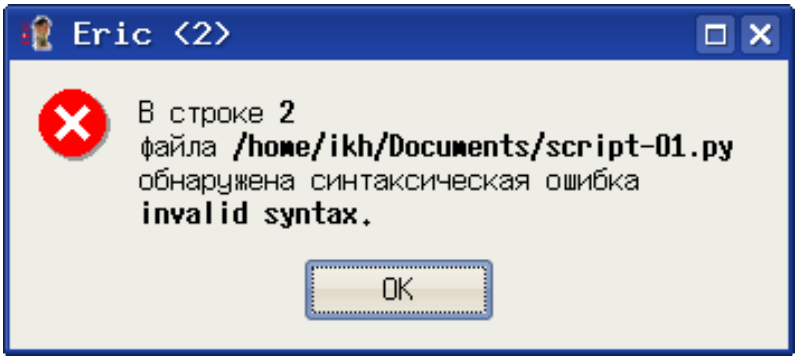

Рисунок 11. Сообщение об ошибке в IDE Eric

В окне редактора IDE Eric строка с ошибкой подсвечивается и рядом с номером строки появляется значок «жучка» - bug (рис. 12).

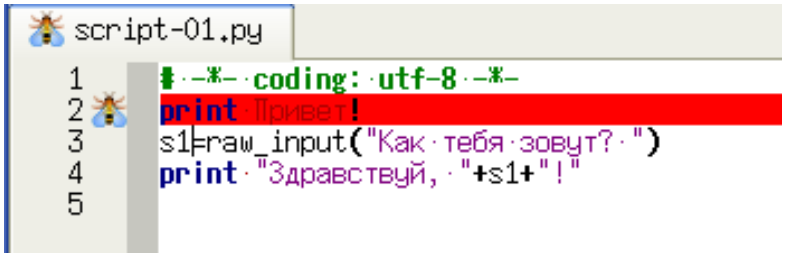

Рисунок 12. Указание на ошибку в IDE Eric

Никаких других подробностей об ошибке Eric не сообщает. Остаётся догадываться (или выяснять на основе работающих примеров), что для вывода строки с помощью инструкции `print` нужно эту строку заключать в кавычки.

Здесь мы столкнулись с ***синтаксической ошибкой***, в результате Python нас «не понял».

Была использована конструкция, не разрешённая правилами языка. В этом случае программа даже не запускается на выполнение.

Есть и другие варианты ошибок — ***ошибки времени выполнения*** (run-time), которые в Python называются «исключения» и ***семантические ошибки***. Ситуации, в которых они возникают и методы устранения этих ошибок мы рассмотрим позже.