Григорий Бакунов

Интерпретируй это

Точно так же, как мастер по дереву с помощью нескольких инструментов, досок и фурнитуры делает мебель, программист создает новую программу. Согласитесь, неудобно было бы, например, шлифовать доски киянкой, а выпиливать рубанком. Вот и программисту для эффективной работы тоже важно пользоваться подходящим инструментом.

Неумолимые цифры статистики утверждают, что на сегодняшний день существует более четырех тысяч языков программирования. Некоторые из них уже отжили свой век, другие же в самом начале пути к популярности. Бывает и так, что язык умирает только родившись. Без всяких сомнений, на самой вершине языкового олимпа находятся C++ и Java. На этих языках написано более 80% современных программ. И если C++ более востребован среди «низкоуровневых» программистов, таких как разработчики операционных систем и приложений, для которых критична скорость исполнения, то Java прочно укоренилась в сфере бизнес-приложений. Однако оба этих языка — даже не рубанки, а скорее большие многофункциональные станки, использование которых потребует огромных затрат на обучение. Именно поэтому существуют более простые языки программирования, которые, соответственно, намного доступнее, но наделены меньшими возможностями. В их числе можно отметить популярный среди программистов старшего поколения Perl, завоевавший признание среди веб-программистов PHP, а также Python, являющийся одним из самых легких в изучении.

Отличительные черты Python

Язык программирования Python был задуман как своеобразная альтернатива языку Basic. Главной задачей при его разработке стало создание четкого, стройного и понятного языка для начинающих. Один из ведущих разработчиков Python, Гвидо ван Россум, собрав лучшие черты других языков, на их основе написал первую версию интерпретатора Python, которая развивается и совершенствуется до сих пор при посильной помощи группы энтузиастов. Строго говоря, Python относится к классическим языкам программирования, и разобраться с ним будет достаточно просто каждому, кто хоть раз писал даже самую маленькую программу.

В чем же главное отличие Python от традиционных языков, таких как вышеупомянутые C++ или Java? Python — интерпретатор, а значит программа, написанная на этом языке, не требует предварительного процесса компиляции. Поэтому программы на Python занимают значительно меньше места, хотя и требуют наличия установленного в системе интерпретатора. Синтаксис этого языка во многом похож на Java, но он более гибкий. Это позволяет использовать его не только как средство традиционного объектно-ориентированного программирования, но и обрабатывать с его помощью данные, как это делает, например, Lisp.

Работа с интерпретатором

Одной из самых важных особенностей Python является интерактивный интерпретатор. Запустить его достаточно просто. Достаточно в текстовой консоли или эмуляторе терминала выполнить команду «python». Сразу же после того как вы нажмете «Enter», появится приглашение интерпретатора:

```
python
Python 2.3.4 (#1, Feb  2 2005, 12:11:53)
[GCC 3.4.2 20041017 (Red Hat 3.4.2-6.fc3)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Бесспорно, с основами работы интерпретатора сможет разобраться даже ребенок. Для начала давайте посчитаем, сколько кубиков со стороной 10 см войдет в коробку размером 10x30x30:

```
>>> (30 * 30 * 10) / (10 * 10 * 10)
9
>>>
```

Так можно поступать и с остальными математическими операциями. Однако после каждого действия компьютер не сохраняет результат, а только выводит его на экран, что довольно неудобно при сложных вычислениях. В таких случаях программисты помещают данные в специальные контейнеры — переменные.

Например, для того чтобы не запоминать, сколько кубиков входит в коробку, поместим это число в переменную:

```
>>> v_korobke = (30 * 30 * 10) / (10 * 10 * 10)
>>> v_korobke
9
```

Теперь мы легко можем посчитать, сколько кубиков находится и в шести коробках:

```
>>> v_korobke * 6
54
>>>
```

Если вы хотите сохранить последовательность действий, можно записать набранные строчки в файл. В этом случае единственное отличие от интерпретатора состоит в том, что для вывода результатов необходимо воспользоваться командой «print»:

```
#!/usr/bin/python
v_korobke = (30 * 30 * 10) / (10 * 10 * 10)
print v_korobke * 6
```

Первая строка в этом файле служебная. Именно по ней операционная система определит, что для выполнения этой программы необходимо запустить интерпретатор языка Python.

До сих пор мы говорили об очень простых действиях. Попробуем перейти к чему-нибудь более серьезному. Создадим файл shop.py следующего содержания:

```
1. #!/usr/bin/python
2. v_korobke = (30 * 30 * 10) / (10 * 10 * 10)
3. print "Сколько коробок было в магазине?"
4. korobok_bylo = int ( raw_input() )
5. kubikov_bylo = korobok_bylo * v_korobke
6. print "Сколько коробок стало?"
7. korobok_stalo = int ( raw_input() )
8. kubikov_stalo = korobok_stalo * v_korobke
9. rezultat = kubikov_bylo — kubikov_stalo
10. if rezultat > 0 :
11.         print "Продано %d кубиков, хорошо торгуем"
% rezultat
12. elif rezultat < 0 :
13.         print "Стало больше на %d кубиков, плохо
считаем." % (-rezultat)
14. else:
15.         print "Ничего не продали, выходной, однако"
```

Разберем по порядку наиболее интересные строки. В третьей, к примеру, используется пока незнакомый вам вариант оператора print. До этого с его помощью мы печатали только цифры, теперь же понятно, что оператором print можно вывести на экран и произвольную строку, ограниченную кавычками. В четвертой строке в переменную korobok_bylo вносится число коробок, указываемое пользователем. Сложное с виду выражение на самом деле оказывается достаточно простым: raw_input() ожидает от пользователя ввода, а int указывает на то, что введенные данные должны быть целым числом. В девятой строке в переменную rezultat помещается разность между количеством кубиков «до» и «после». В строках 10–15 программа сообщает пользователю результат вычислений. Далеко не всегда удобно, если результат только числовой, поэтому наша дружелюбная к пользователю программа сообщает результаты тремя разными способами. В десятой строке происходит проверка на положительность результата, то есть доказывается, что при «втором» подсчете кубиков стало меньше, следовательно, переменная rezultat больше нуля. В строку 11 программа попадает только в случае положительного результата этой проверки. Если внимательно присмотреться к тексту внутри кавычек строки 11, можно обратить внимание на странное сочетание «%d» — это знак подстановки. Сюда Python подставит значение переменной, указанной после символа «%» за пределами кавычек. В нашем случае это переменная rezultat. Строку 12 следует читать так: «если предыдущее условие не выполняется и переменная rezultat меньше нуля, то:». Таким образом, программа попадает в строку 13 только если результат «торговли» отрицательный. Обратите внимание на символ «-» перед переменной rezultat в строке 13. Этот символ инвертирует знак числа в переменной. Так как в переменной rezultat в данном случае отрицательное число (например, -9), мы отбрасываем «-» для удобства пользователя. Строка 14 отличается от строки 12 отсутствием условия, то есть следующая после нее строка выполняется в том случае, если не верны предыдущие два утверждения. Простейшая логика подсказывает нам, что это может быть только в том случае, если в переменной rezultat находится число 0.

Также следует обратить внимание на отступы в строках после символов двоеточия. Это так называемые смысловые блоки. В других языках программирования смысловые блоки выделяются специальными ключевыми словами (например, begin и end в языке Pascal или "{" и "}" в C++). Однако создатели Python решили, что такие отступы намного наглядней. Чтобы показать это на примере, немного расширим вывод результатов программы, если значение переменной rezultat больше нуля. Между 11 и 12 строками вставим еще одну следующего содержания:

```
11.75 print "Продано %d процентов кубиков." %((1 —
float (kubikov_stalo) / kubikov_bylo)*100)
```

Обратите внимание на слово «float» перед переменной kubikov_stalo — оно подчеркивает, что обращаться с ней надо как с числом с плавающей точкой, то есть, возможно, содержащим дробную часть.

Заключение

Конечно же, данная статья рассматривает Python на самом примитивном уровне. Возможности этого языка программирования намного шире. Он позволяет писать и полноценные пользовательские приложения. Как нетрудно было убедиться, синтаксис самого языка, а также его использование очень просты и доступны даже слабо подготовленному программисту, что немало способствует быстрому росту его популярности.