

Python for Informatics

Автор: Dr. Charles R. Severance

Содержание курса: <http://pycode.ru/edu/why-python/>

Глава 1. Почему следует научиться писать программы?

Компьютеры способны быстро решать задачи, которые являются трудоемкими для человека. Например, подсчет частоты встречаемости слов в тексте и определение слов, которые встречаются чаще всего. Человек способен решить такую задачу, но это скучная и однообразная работа. Компьютер (или PDA - персональный цифровой помощник) справляется с ней быстро:

```
python words.py
Enter file:words.txt
to 16
```

Это пример выполнения программы, которая в дальнейшем будет разбираться более подробно.

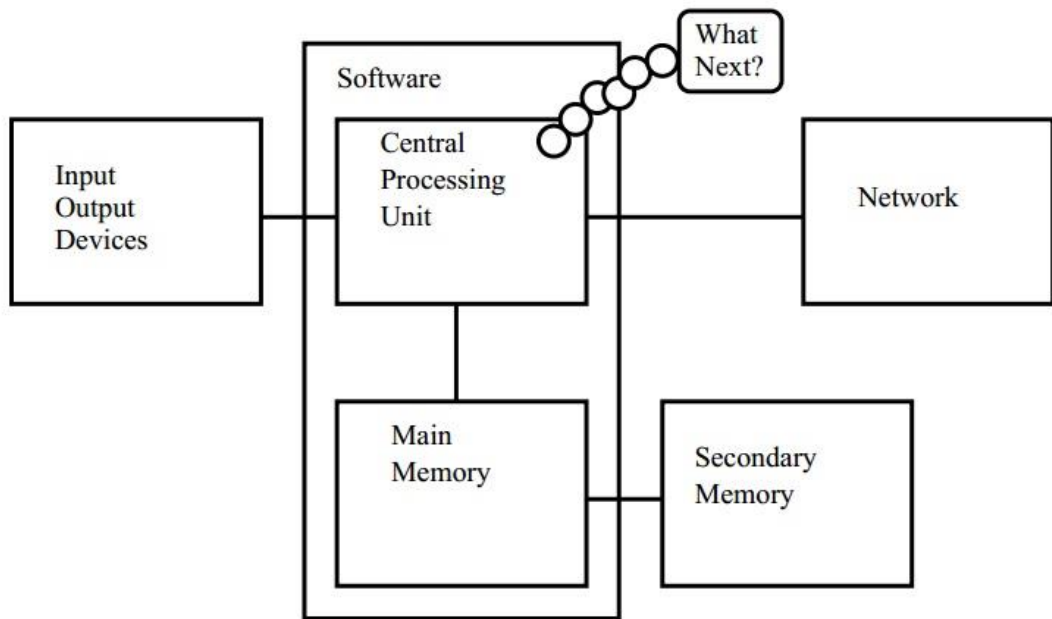
Выучив язык программирования Python, вы сможете передать решение повседневных задач помощнику (компьютеру), таким образом, вы освободите время для решения более интересных проблем.

1.1. Творчество и мотивация

Создание программ, которые будут использоваться другими людьми - это творческое занятие. Огромное количество программ на компьютере конкурируют за ваше внимание и интерес (программисты зарабатывают на этом деньги). Задачей данного пособия является не зарабатывание денег или радость конечного пользователя, а более продуктивное выполнение повседневных задач обработки данных.

1.2. Архитектура компьютера

Если бы вы разобрали компьютер или мобильный телефон и заглянули внутрь, то увидели бы следующие части:



На высоком уровне определения частей следующие:

- *Центральный процессор (CPU)* - часть компьютера, которая создана, чтобы постоянно спрашивать «что дальше»? Если частота вашего компьютера 3.0 ГГц, это означает, что CPU будет спрашивать "что дальше?" три миллиарда раз в секунду.
- *Оперативная память (Main Memory)* используется для хранения информации, которая срочно необходима CPU. Информация, хранящаяся в оперативной памяти, стирается, когда питание компьютера прекращается.
- *Вторичная память (Secondary Memory)* также нужна для хранения информации, но она медленнее оперативной памяти. Особенностью вторичной памяти является возможность хранить информацию после выключения питания компьютера. Примеры вторичной памяти: диски или флеш-память.
- *Устройства ввода/вывода (Input and Output Devices)* – наш монитор, клавиатура, мышь, микрофон, колонки и т.д. Они обеспечивают взаимодействие с компьютером.
- В наше время большинство компьютеров имеют *сетевой адаптер (Network Connection)* для обмена информации через сеть.

1.3. Понимание программирования

В остальной части книги, мы постараемся превратить вас в человека, который является специалистом в области программирования.

В конце вы станете программистом, возможно, не профессиональным программистом, но, по крайней мере, вы будете иметь навыки исследования проблем анализа данных/информации и разработки программ для их решения.

В известном смысле, нужно два навыка, чтобы стать программистом:

- Во-первых, необходимо знать язык программирования (в нашем случае, Python) - вам необходимо знать лексику и грамматику. В новом языке вы должны правильно строить слова и «предложения».
- Во-вторых, вам необходимо «рассказать историю». При написании истории, вы объединяете слова и фразы, передающие сюжет читателю. В программировании, наша программа - это «история», а проблема, которую вы пытаетесь решить – это сюжет.

Однажды выучив язык программирования, такой как Python, вы обнаружите, что сможете легко выучить второй язык программирования, такой как JavaScript или C++. Новый язык программирования имеет много отличий в словаре и грамматике, но как только вы приобретете навыки решения задач, они будут одинаковыми во всех языках программирования.

Вы очень быстро выучите словарь (vocabulary) и выражения (sentences) Python. Но чтобы научиться писать большие программы для решения сложных проблем, понадобится время. Мы изучим программирование подобно изучению письма. Мы начнем читать и разбирать программы, затем напишем простые программы и только потом перейдем к более сложным. В некоторый момент вы «поймаете вашу писательскую музу», самостоятельно увидите шаблоны решения проблемы и сможете написать программу. С этого момента программирование становится очень приятным и творческим процессом.

Мы начнем со словаря и структуры программы на Python. Будьте терпеливы, в первый раз вы учились читать на простых примерах.

1.4. Слова и фразы

В отличие от человеческих языков, словарь Python значительно меньше. Назовем этот словарь списком зарезервированных слов. Существуют слова, которые в Python имеют специальное значение. Когда вы встречаете эти слова в программе на Python, они имеют одно и только одно значение для Python. Позже вы будете писать программы и создавать переменные (*variables*) - собственные слова, которые имеют смысл для вас. У вас будет широкий выбор имен для ваших переменных, но вы не можете использовать в качестве имен переменных зарезервированные слова.

В известном смысле, когда вы дрессируете собаку, вы можете использовать специальные слова, например, «сидеть», «стоять», «принести». Когда вы общаетесь с собакой и не используете какие-либо зарезервированные слова, они только смотрят на вас, пока вы не произнесете зарезервированное слово. К примеру, если вы скажете: «Я желаю большинству людей ходить (*walk*) для улучшения здоровья», то собака услышит нечто подобное: «ла ла ла ла ла ла ходить (*walk*) ла ла ла ла ла ла». Слово ходить (*walk*) является зарезервированным в языке собаки.

Зарезервированные в Python слова:

```
and del for is raise assert elif from lambda return break else
global not try class except if or while continue exec import
pass yield def nally in print
```

В отличие от собаки, Python уже обучен этим словам.

Позже мы изучим зарезервированные слова и то, как они используются, а сейчас сосредоточимся на эквиваленте слову «говорить» (*speak*) в Python:

```
print 'Hello world!'
```

Мы написали нашу первую синтаксически правильную фразу на Python. Наша фраза начинается с зарезервированного слова *print* и продолжается текстовой строкой, заключенной в одиночные кавычки.

1.5. Разговаривающий с Python

Теперь, когда у нас есть слово и простая фраза, которую мы знаем на Python, необходимо научиться общению с Python для проверки наших языковых навыков.

Предварительно нам надо установить Python на компьютер и запустить в интерактивном режиме. Инструкция по установке Python в ОС Windows будет здесь: <http://pycode.ru/edu/why-python/>

В интерактивном режиме получим следующее:

```
Python 2.7.3 (default, Apr 10 2012, 23:31:26) [MSC v.1500 32 bit (Intel)] on win32
```

```
Type "copyright", "credits" or "license()" for more information.
```

```
>>>
```

С помощью символов >>> интерпретатор Python спрашивает: «Что вы хотите, чтобы я сделал дальше?». Теперь Python готов с вами общаться.

Скажем, для примера вы не знали простых слов или фраз для общения с Python. Вы захотели использовать стандартные строки, которые астронавты используют, когда прилетают на другую планету и пытаются поговорить с жителями этой планеты:

```
>>> I come in peace, please take me to your leader
```

```
SyntaxError: invalid syntax
```

Это не очень хорошо. Жители планеты, скорее всего, вас поджарят и съедят на ужин. К счастью, вы захватили с собой в путешествие эту книгу.

Попробуйте еще раз:

```
>>> print 'Hello world!'
```

```
Hello world!
```

```
>>> print 'Привет, мир!'
```

```
Привет, мир!
```

Это выглядит гораздо лучше, поэтому попытайтесь сообщить еще немного:

```
>>> print 'You must be the legendary god that comes from the sky'
```

```
You must be the legendary god that comes from the sky
```

```
>>> print 'We have been waiting for you for a long time'
```

```
We have been waiting for you for a long time
>>> print 'Our legend says you will be very tasty with mustard'
Our legend says you will be very tasty with mustard
>>> print 'We will have a feast tonight unless you say
SyntaxError: EOL while scanning string literal
>>>
```

Разговор шел хорошо, пока вы не совершили маленькую ошибку в языке Python.

На данный момент, вы также должны понимать, что Python удивительно сложный и мощный язык, он очень требователен к синтаксису, используемому для взаимодействия с ним, но Python НЕ обладает разумом. Вы беседуете сами с собой, но с использованием правильного синтаксиса.

В некотором смысле, когда вы используете написанную кем-то программу, общение осуществляется между вами и теми другими программистами, а в качестве посредника выступает Python.

И через несколько глав вы станете одним из тех программистов на Python, которые общаются с пользователями программ.

Перед тем как покинуть наш первый разговор с интерпретатором Python, вам необходимо знать, как правильно сказать «до свидания» при взаимодействии с жителями планеты Python:

```
>>> good-bye
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    good-bye
NameError: name 'good' is not defined
>>> if you don't mind, I need to leave
SyntaxError: invalid syntax
>>> quit()
```

Вы заметили, что для двух первых неудачных попыток ошибки различаются. Вторая ошибка отличается тем, что, *if* - зарезервированное слово и Python подумал, что мы пытаемся что-то сказать, но используем неправильный синтаксис.

Правильный способ сказать «до свидания» на Python, это ввести *quit()* в интерактивной строке приглашения.

1.6. Терминология: интерпретатор и компилятор

Python - высокоуровневый (high-level) язык программирования, созданный, чтобы быть относительно простым для чтения и написания программ человеком, и для компьютерного чтения и выполнения. Другие высокоуровневые языки: Java, C++, PHP, Ruby, Basic, Perl, JavaScript и множество других.

Имеющиеся аппаратные средства внутри CPU не понимают любой из вышеперечисленных языков. CPU понимает язык, который называется машинным (machine-language). Машинный язык является простым, но очень утомительным в написании, т.к. представляет собой только нули и единицы:

```
01010001110100100101010000001111
11100110000011101010010101101101
...
```

Машинный язык кажется довольно простым на поверхности, учитывая, что в нем есть только нули и единицы, но он обладает сложным синтаксисом. Поэтому очень немногие программисты когда-либо писали на машинном языке. Вместо этого мы строим различные трансляторы (переводчики), позволяющие программистам писать на языках высокого уровня, таких как Python или JavaScript. Затем трансляторы преобразуют программы в машинный язык для исполнения их процессором.

Поскольку машинный язык привязан к аппаратному обеспечению компьютера, он не может быть перенесен (portable) на компьютер с другим типом оборудования. Программы, написанные на языках высокого уровня, можно перемещать между разными компьютерами, например, с использованием транслятора на новом компьютере, или с помощью перекомпиляции кода для создания новой версии машинного языка программы.

Трансляторы языков программирования делятся на две основные категории: (1) интерпретаторы и (2) компиляторы.

Интерпретатор (interpreter) читает исходный код программы, написанный программистом, анализирует код, и выполняет инструкции на лету (on-the-

fy). Python является интерпретатором, мы можем написать строку на Python, и он немедленно ее обработает и выдаст результат.

Некоторые строки говорят Python о том, что вы хотите их запомнить в некоторой переменной (variable). Нам необходимо выбрать имя для переменной, позже мы можем воспользоваться этим именем, чтобы получить значение, которое сохранили ранее.

```
>>> x = 6
>>> print x
6
>>> y = x * 7
>>> print y
42
>>>
```

В этом примере, мы просим Python запомнить значение 6 и используем метку *x*, по которой сможем восстановить значение позже. Мы убедились, что Python фактически вспоминает значение, используя *print*. Затем мы просим Python взять *x*, умножить его на 7 и полученный результат сохранить в переменной *y*. После этого мы просим Python вывести на экран значение, хранящееся в переменной *y*.

Хотя мы вводим команды в Python на разных строках, они рассматриваются в качестве упорядоченной последовательности инструкций, мы можем обратиться к данным, созданным ранее.

Сущность *интерпретатора* заключается в возможности интерактивного общения, как было показано выше. Компилятор нуждается в передаче всей программы в файл, после этого запускается процесс трансляции с языка высокого уровня в машинный язык и только после этого компилятор помещает результат машинного языка в файл, который позже будет исполняться.

Если у вас ОС Windows, то часто программа, содержащая исполняемые машинные инструкции имеет расширение «.exe» или «.dll», «исполняемый файл» (executable) или «динамически загружаемая библиотека» (dynamically loadable library») соответственно. В Linux и Macintosh нет расширения, который бы однозначно отмечал файл как исполняемый.

Предположим, что вы проводите исследование сообщений в социальной сети Facebook, вам интересны наиболее часто встречающиеся слова в последовательности сообщений. Вы можете вывести на экран поток сообщений и корпеть над поиском наиболее распространенных слов, но это займет много времени и может привести к ошибке. Вы могли бы быть достаточно умны, чтобы написать программу на Python, быстро решающую задачу, и освободить себе выходные для более интересных занятий.

Для примера посмотрим на следующий текст о клоунах и автомобилях. Посмотрите на текст и выделите наиболее часто встречающиеся слова и количество раз, сколько они встречаются.

```
the clown ran after the car and the car ran into the tent  
and the tent fell down on the clown and the car
```

Представим, что вы выполняете это задание, просматривая миллионы строк текста. Честно говоря, это проще сделать с помощью программы на языке Python, которая подсчитывает количество слов в тексте.

Я написал программу, которая ищет наиболее часто повторяющиеся слова в тексте. Теперь вы можете воспользоваться этой программой, сэкономя себе немного времени:

```
name = raw_input('Enter file:')  
handle = open(name, 'r')  
text = handle.read()  
words = text.split()  
counts = dict()  
  
for word in words:  
    counts[word] = counts.get(word,0) + 1  
  
bigcount = None  
bigword = None  
for word,count in counts.items():  
    if bigcount is None or count > bigcount:  
        bigword = word
```

```
bigcount = count
```

```
print bigword, bigcount
```

Исходный текст программы доступен в архиве:
<http://pycode.ru/files/python/words.zip>

Вам даже не надо знать Python, чтобы воспользоваться этой программой. Вся необходимую информацию по внутренней работе программы вы получите в главе 10 данного пособия.

Это хороший пример того, как интерпретатор Python и язык программирования Python выступают посредниками между вами (конечными пользователями) и мной (разработчиком). Python является способом обмена полезными инструкциями (например, программами) на общем языке, который доступен всем, кто установил Python на свои компьютеры. Таким образом, мы общаемся не с Python, а между собой посредством Python.

1.9. Построение частей программ

В следующих нескольких главах, мы узнаем чуть больше о словаре, структуре фраз и отступов. Мы узнаем о мощных возможностях Python и как совместить эти возможности, чтобы создавать полезные программы.

Есть несколько важных шаблонов, которые используются при построении программы. Эти конструкции относятся не только к Python, они являются частью любого языка программирования, начиная от машинного языка и заканчивая языками высокого уровня.

Входные данные (input): получение данных из внешнего мира. Это может быть чтение данных из файла или из других источников, например, микрофона или GPS. В наших первых программах, входные данные будут задаваться пользователем с клавиатуры.

Выходные данные (output): отображение результатов работы программы на экране, сохранение их в файл или, возможно, воспроизведение в виде музыки или голоса.

Последовательное исполнение (sequential execution): инструкции выполняются в том порядке, в котором они встречаются в скрипте.

Условное выполнение (conditional execution): проверка определенных условий и выполнение/пропуск последовательности инструкций.

Повторное выполнение (repeated execution): выполнять некоторый набор команд несколько раз, обычно с некоторыми изменениями.

Повторное использование (reuse): один раз написать набор инструкций, присвоить ему имя и затем повторно использовать этот набор во всей программе.

1.12. Словарь

Ошибка (bug): ошибка в программе.

Центральный процессор (central processing unit): сердце компьютера, он исполняет программы, которые мы пишем.

Компиляция (compile): преобразование программы, написанной на языке высокого уровня, в низкоуровневый язык и ее подготовка для последующего выполнения.

Язык высокого уровня (high-level language): язык программирования, подобный Python, который разрабатывался, чтобы быть понятным для чтения и написания человеком.

Интерактивный режим (interactive mode): способ использования интерпретатора Python в режиме ввода команд и выражений.

Интерпретация (interpret): построчное исполнение программы на языке высокого уровня.

Язык программирования низкого уровня (low-level language): язык программирования, который разрабатывался, чтобы быть простым для исполнения компьютером; он называется «машинным кодом» или «языком ассемблера».

Машинный код (machine code): низкоуровневый язык программирования для программного обеспечения, который напрямую выполняется CPU.

Оперативная память (main memory): хранит программы и данные, теряет всю информацию, когда прекращается питание компьютера.

Разбор (parse): исследование программы и анализ синтаксиса.

Переносимость (portability): свойство программы, которое позволяет ей выполняться на более, чем одном виде компьютера.

Оператор печати (print statement): инструкция, которая отображает значение на экране.

Решение задачи (problem solving): процесс формулировки задачи, поиска решения и реализации.

Программа (program): набор инструкций, предназначенных для обработки на компьютере.

Приглашение (prompt): когда программа отображает сообщение и ожидает ввода пользовательских данных.

Вторичная память (secondary memory): хранит программы, данные и сохраняет информацию после выключения компьютера из сети питания. Примеры вторичной памяти: диски, флеш-память.

Семантика (semantics): смысл программы.

Семантическая ошибка (semantic error): ошибка в программе, когда происходит то, что программист не планировал.

Исходный текст (source code): программа на языке высокого уровня (high-level language).

1.13. Упражнения

1. Перечислите основные элементы компьютера.
2. В чем отличия интерпретатора и компилятора.
3. Перечислите шаблоны, которые используются при построении программы.