2. Буквы

Итак, мы узнали всё о числах, а как же насчёт букв? слов? текста?

Группы букв в программе мы называем строками. (Вы можете считать, что напечатанные буквы нанизаны друг за другом на бечёвку, как флажки.) Чтобы было легче увидеть, какие части кода — строки, я буду выделять их **красным** цветом. Вот несколько строк:

```
'Привет.'
'Ruby - потрясающий.'
'5 - это моё любимое число... а какое ваше?'
'Снупи восклицает: "#%^?&*@!", когда он запинается.'
'     '
''
```

Как видите, строки могут содержать знаки пунктуации, цифры, символы и пробелы... не только одни буквы. А в последней строке совсем ничего нет; такую строку называют пустой строкой.

Мы использовали команду **puts**, чтобы напечатать числа; давайте опробуем её с несколькими строками:

```
puts 'Привет, мир!'
puts ''
puts 'До свидания.'
```

```
Привет, мир!

До свидания.
```

Всё отлично отработало. А теперь попробуйте вывести несколько своих строк.

Строковая арифметика

Точно так же, как вы выполняли арифметические действия с числами, вы можете выполнять арифметические действия со строками! Ну, почти так же... Во всяком случае, вы можете складывать строки. Давайте попробуем сложить две строки и посмотреть, что нам покажет **puts**.

```
puts 'Я люблю' + 'яблочный пирог.'
```

```
Я люблюяблочный пирог.
```

Опаньки! Я забыл поместить пробел между **'Я люблю'** и **'яблочный пирог.'**. Пробелы обычно не имеют значения, но они имеют зачение внутри строк. (Правду говорят: "компьютеры делают не то, что вы *хотите*, чтобы они делали, а только то, что вы *велите* им делать".) Давайте попробуем снова:

```
puts 'Я люблю ' + 'яблочный пирог.'
puts 'Я люблю' + ' яблочный пирог.'
```

```
Я люблю яблочный пирог.
Я люблю яблочный пирог.
```

(Как видите, неважно, к какой из строк добавлять пробел.)

Итак, вы можете складывать строки, но вы также можете умножать их! (На число, во всяком случае.) Посмотрите:

```
puts 'миг ' * 4
```

```
мигаю глазами
```

(Шучу, конечно... на самом деле, она выводит вот что:)

```
миг миг миг миг
```

Если подумать, это имеет безупречный смысл. В конце концов, `7*3` действительно означает `7+7+7`, поэтому `'му'*3` означает в точности `'му'+'му'+'му'`.

`12` или `'12'`

Прежде, чем мы двинемся дальше, мы должны убедиться, что понимаем различие между *числами* и *цифрами*. `12` это число, а `'12'` это — строка из двух цифр.

Давайте немного поиграем с этим:

```
puts  12  +  12
puts '12' + '12'
puts '12  +  12'
```

```
24
1212
12  +  12
```

А как насчёт этого:

```
puts  2  *  5
puts '2' *  5
puts '2  *  5'
```

```
10
22222
2  *  5
```

Эти примеры были довольно простыми. Однако, если вы не будете очень осторожными, смешивая в своей программе строки и числа, у вас могут возникнуть...

Сложности

К этому моменту вы может быть попробовали некоторые вещи, которые *не желают* работать. Если ещё нет, то вот несколько из них:

```
puts '12' + 12
puts '2' * '5'
```

```
#<TypeError: can't convert Fixnum into String>
```

[#<Ошибка типа: невозможно преобразовать Целое к Строке> — *Прим. перев.*] Мммдааа... сообщение об ошибке. Дело в том, что вы действительно не можете прибавить число к строке или умножить строку на другую строку. В этом не больше смысла, чем вот в таких выражениях:

```
puts 'Бетти' + 12
puts 'Фред' * 'Джон'
```

Вот ещё чего следует остерегаться: вы можете написать в программе **'поросёнок'*5**, поскольку это просто обозначает **5** экземпляров строки **'поросёнок'**, соединённых вместе. Однако, вы *не можете* написать **5*'поросёнок'**, поскольку это обзначает **'поросёнок'** экземпляров числа **5**, что просто-напросто глупо.

И, наконец: а что, если я хочу, чтобы программа напечатала 'Ты шикарная!'? Можно попробовать так:

```
puts ''Ты шикарная!''
```

Ну так вот, *это* не будет работать; я даже не буду пробовать выполнить это. Компьютер подумал, что мы закончили строку. [После второго апострофа. — *Прим. перев.*] (Вот почему хорошо иметь текстовый редактор, который расцвечивает синтаксис для вас.) Так как же дать понять компьютеру, что мы желаем остаться внури строки? Нам нужно экранировать символ апострофа вот так:

```
puts '\'Ты шикарная!\''
```

```
'Ты шикарная!'
```

Обратная косая черта — это символ экранирования (escape character). Другими словами, если в строке стоит обратная черта и другой символ [которые образуют так называемую "escape-последовательность". — *Прим. перев.*], то они оба иногда трансформируются в один новый символ. Но две вещи, которые обратная черта всё-таки экранирует, это апостроф и сама обратная черта. (Если хорошенько подумать, то символы экранирования должны всегда экранировать себя.) Ещё несколько примеров, я думаю, будут здесь к месту:

```
puts '\'Ты шикарная!\''
puts 'обратная черта в конце строки:  \\'
puts 'вверх\\вниз'
puts 'вверх\вниз'
```

```
'Ты шикарная!'
обратная черта в конце строки:  \
вверх\вниз
вверх\вниз
```

Поскольку обратная черта *не* экранирует **'в'**, но *экранирует* себя, две последних строки идентичны. Они не выглядят одинаковыми в коде программы, но в вашем компьютере они действительно одинаковы.

Если у вас есть другие вопросы, просто продолжайте читать дальше! В конце концов, я не могу отвечать на каждый вопрос на *этой* странице.