3

ПЕРЕМЕННЫЕ И ОПЕРАТОРЫ

Итак, мы закончили с вводным материалом, давайте приступим к реальным задачам. В этой главе мы изучим переменные и операторы. Вы узнаете, что такое переменные, как их называть и объявлять, а также об общих операциях, которые можно с ними выполнять. Готовы?

Поехали!

3.1. ЧТО ТАКОЕ ПЕРЕМЕННЫЕ?

Переменные — это именованные объекты данных, которые хранятся и используются в наших программах. Допустим, ваша программа должна хранить возраст пользователя. Для этого мы можем назвать эти данные `userAge` и присвоить значение переменной `userAge`, используя следующую инструкцию:

```
userAge = 0
```

После того как вы определите переменную `userAge`, программа выделит некоторую область дискового пространства компьютера для хранения этих данных. Затем можно получить доступ к этим данным и изменить их, сославшись на них по имени `userAge`. Каждый раз, когда вы объ-

являете новую переменную, нужно присвоить ей начальное значение. В этом примере мы присвоили ей значение `0`. Значение всегда можно изменить позднее.

Также можно определить несколько переменных за раз. Для этого напишите

```
userAge, userName = 30, 'Peter'
```

Это будет равнозначно

```
userAge = 30
userName = 'Peter'
```

3.2. ИМЕНОВАНИЕ ПЕРЕМЕННЫХ

Имя переменной в Python может содержать только буквы (a–z, A–B), числа или символы подчеркивания (_). Первый символ не может быть числом. Таким образом, можно называть свои переменные `userName`, `user_name` или `userName2`, но не `2userName`.

Кроме того, есть некоторые зарезервированные слова, которые нельзя использовать в качестве имени переменной, так как они уже имеют заранее заданные значения в Python. Это `print`, `input`, `if`, `while` и т. д. Подробнее о них поговорим в следующих главах.

Наконец, имена переменных чувствительны к регистру. `username` не то же самое, что `userName`.

При именовании переменных в Python существуют два соглашения. Можно придерживаться верблюжьего ре-

гистра или использовать символы подчеркивания. Верблюжий регистр — это способ написания составных слов в смешанном регистре (например, `thisIsAVariableName`). Мы будем использовать верблюжий регистр в оставшейся части книги. Другой распространенной практикой является использование символа подчеркивания (_) для разделения слов. Таким образом, переменные можно назвать так: `this_is_a_variable_name`.

3.3. ОПЕРАТОР ПРИСВАИВАНИЯ

Обратите внимание: символ = в инструкции `userAge = 0` имеет другое значение, чем математический символ «равно». В программировании = называется оператором присваивания. Это означает, что справа от = присваивается значение переменной слева. Чтобы было легче понять инструкцию `userAge = 0`, можно думать о ней как о `userAge <- 0`.

В программировании выражения `x = y` и `y = x` означают совершенное разные вещи.

Удивлены? Приведу пример.

Напишите следующий код в редакторе IDLE и сохраните его.

```
x = 5
y = 10
x = y
print ("x = ", x)
print ("y = ", y)
```

Запустите программу. Вы увидите такой результат:

```
x = 10
y = 10
```

Хотя x имеет начальное значение 5 (как было объявлено в первой строке), третья строка x = y присваивает значение y переменной x (вспоминаем: x <- y), следовательно, значение x меняется на 10, в то время как значение y остается неизменным.

Модифицируем программу, изменив ТОЛЬКО ОДИН оператор: поменяйте третью строку с x = y на y = x. С точки зрения математики x = y и y = x — это одно и то же. Однако в программировании все не так.

Запустите программу. В результате получится:

```
x = 5
y = 5
```

Здесь значение x остается равным 5, а вот значение y изменяется на 5. Оператор y = x присваивает значение x значению y.

Значение y становится равным 5, а x остается неизменным.

3.4. ОСНОВНЫЕ ОПЕРАТОРЫ

Помимо присвоения переменной начального значения можно выполнять обычные математические операции

с переменными. Основные операторы в Python включают в себя сложение (+), вычитание (-), умножение (*), деление (/), целочисленное деление (//), остаток от деления (%) и возведение в степень (**).

Примеры:

Предположим, `x = 5, y = 2`

Сложение:

```
x + y = 7
```

Вычитание:

```
x - y = 3
```

Умножение:

```
x*y = 10
```

Деление:

```
x/y = 2.5
```

Целочисленное деление:

```
x//y = 2 (округляет результат до ближайшего целого
числа)
```

Остаток от деления:

```
x%y = 1 (дает остаток от деления 5 на 2)
```

Возведение в степень:

```
x**y = 25 (5 в степени 2)
```

3.5. ДОПОЛНИТЕЛЬНЫЕ ОПЕРАТОРЫ ПРИСВАИВАНИЯ

Помимо основного оператора `=` в Python (как и в большинстве языков программирования) есть еще несколько дополнительных операторов присваивания. К ним относятся такие операторы, как `+ =`, `- =` и `* =`.

Предположим, есть переменная `x` с начальным значением `10`. Если требуется увеличить `x` на 2, можно написать:

```
x = x + 2
```

Программа *сначала оценит выражение справа* (`x + 2`) и назначит ответ слева. Таким образом, в итоге приведенное выше выражение становится `x = 12`.

Вместо того чтобы писать `x = x + 2`, можно написать `x + = 2`, чтобы выразить то же значение. Обозначение `+ =` на самом деле является сокращением, которое объединяет знак присваивания с оператором сложения. Следовательно, `x + = 2` означает `x = x + 2`.

Точно так же, если требуется вычитание, можно написать `x = x - 2` или `x - = 2`. Это работает для всех 7 операторов, упомянутых в разделе выше.