Когда вы придумываете имя для переменной, функции или класса, вы руководствуетесь примерно одинаковыми для всех программистов принципами. Мы предлагаем вам относиться к имени как к небольшому комментарию. Вы можете сообщить довольно много информации, выбрав хорошее имя.

ОСНОВНАЯ ИДЕЯ

Помещайте в имена полезную информацию.

Большое количество имен, встречающихся в программе, ни о чем не говорят (например, `tmp`). Даже слова, применение которых обычно оправданно (такие как `size` или `get`), не содержат много информации. Из этой главы вы узнаете, как подобрать более подходящие имена.

Данная глава состоит из шести отдельных тем:

- выбор конкретных слов;
- избегание общих имен (или умение использовать их к месту);
- использование конкретных имен вместо абстрактных;
- добавление дополнительной информации с использованием суффикса или префикса;
- определение длины имени;
- форматирование имени для того, чтобы сообщить дополнительную информацию.

Выбираем конкретные слова

Одним из важных этапов добавления информации в имена является выбор конкретных слов, а также избегание пустых и неинформативных.

Например, слово `get` в следующем примере совершенно неконкретное:

```
def GetPage(url):
    ...
```

Слово `get` (получать) не скажет вам многого. Откуда получает информацию этот метод: из локальной кэш-памяти, из базы данных или из Интернета? Если верным является последний вариант, то было бы правильнее использовать имя `FetchPage()` или `DownloadPage()`.

Рассмотрим также пример класса `BinaryTree`:

```
class BinaryTree {
    int Size();
    ...
};
```

Что возвращает метод `Size()`? Высоту бинарного дерева, количество узлов или объем используемой памяти?

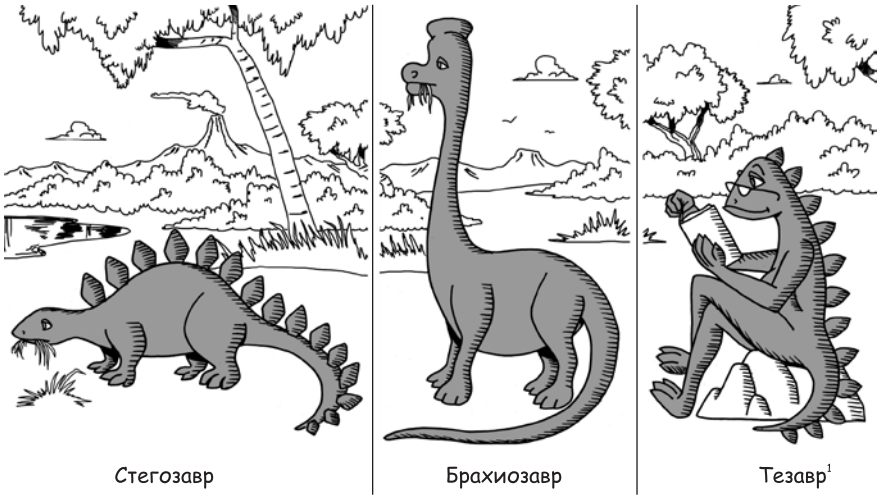
Проблема в том, что имя метода `Size()` не сообщает нам достаточной информации. Более конкретными именами в данном случае были бы `Height()`, `NumNodes()` или `MemoryBytes()`.

В качестве еще одного примера рассмотрим вариант реализации класса `Thread`:

```
class Thread {
    void Stop();
    ...
};
```

Имя `Stop()` вполне подходит, но, в зависимости от того, что именно делает этот метод, можно выбрать и более конкретное имя. Например, вместо него можно использовать имя `Kill()` (убить), если результат действия этой операции нельзя откатить. Или можно применить имя `Pause()` (приостановить), если у вас есть возможность продолжить его работу (`Resume()`).

Находим более «яркие» слова. Не бойтесь воспользоваться тезаурусом или попросить друга помочь вам придумать более подходящее имя. Английский язык богат различными синонимами, вы можете выбрать имя из огромного множества слов.



В табл. 2.1 рассмотрим «яркие» версии слов, которые вы повседневно используете. Они могут пригодиться в определенных ситуациях.

Таблица 2.1

Слово	Альтернативы
send (посылать)	deliver (доставлять), dispatch (отсылать), announce (извещать), distribute (распространять), route (направлять)
find (искать)	search (искать), extract (извлекать), locate (обнаруживать), recover (восстанавливать)
start (начинать)	launch (запускать), create (создавать), begin (начинать), open (открывать)
make (создавать)	create (создавать), set up (устанавливать), build (строить), generate (генерировать), compose (составлять), add (добавлять), new (создавать)

[1] Здесь имеет место игра слов, обыгрываются названия видов динозавров и слова «тезаурус» (вид словарей). — *Примеч. пер.*

Однако не слишком фантазируйте. В языке PHP есть функция для работы со строками, которая называется `explode()` (взорвать). Это довольно яркое имя, которое рисует прекрасную картину того, как объект разлетается на кусочки. Но чем эта функция отличается от `split()` (разделить)? (Эти функции используются для разных целей, но как определить это по названию?)

ОСНОВНАЯ ИДЕЯ

Лучше быть ясным и четким, чем крутым и оригинальным.

Избегаем общих имен, например таких, как tmp и retval

Имена вроде `tmp`, `retval` и `foo` можно сравнить с отговорками вроде «никак не могу придумать имя». Вместо использования подобного бесполезного имени **придумайте такое имя, которое описывает назначение или содержание объекта**.

В качестве примера приведем функцию на языке JavaScript, которая использует параметр `retval`:

```
var euclidian_norm = function (v) {
    var retval = 0.0;
    for (var i = 0; i < v.lenght; i += 1)
        retval += v[i] * v[i];
    return Math.sqrt(retval);
};
```

Очень хочется воспользоваться именем `retval`, если вы не можете придумать более подходящее имя для возвращаемого значения. Но `retval` не содержит никакой другой информации, кроме как о том, что это возвращаемое значение (а это в любом случае очевидно).

Более подходящее имя описывает назначение переменной или значение, которое она содержит. В данном случае в переменную записывается сумма квадратов значений массива `v`. Поэтому лучше назвать ее `sum_squares` (сумма квадратов). Такое имя говорит само за себя и может помочь определить ошибку.

Например, представьте, что внутри цикла вы случайно написали:

```
retval += v[i];
```

Ошибка будет гораздо более очевидной, если переменная будет называться `sum_squares`:

```
sum_squares += v[i]; // Почему мы суммируем значения,
                     // а не их квадраты? Ошибка!
```

СОВЕТ

Имя переменной retval недостаточно информативное. Используйте вместо него имя, которое описывает значение переменной.

Однако иногда общие имена довольно информативны. Рассмотрим, в каких ситуациях целесообразно их использовать.

tmp

Представим классический вариант обмена двух переменных:

```
if (right < left) {
    tmp = right;
    right = left;
    left = tmp;
}
```

В подобных случаях имя tmp отлично подходит. Единственная цель, с которой оно создавалось, — временное хранилище значения. Продолжительность ее жизни равна всего нескольким строкам. Имя tmp имеет особое значение для читателя — у переменной с таким именем нет другого предназначения, кроме как функции временного хранилища. Она не передается в другие функции, не меняет свое значение и не используется несколько раз.

А в следующем случае имя tmp использовано только из-за того, что программист поленился:

```
String tmp = user.name();
tmp += " " + user.phone_number();
tmp += "" + user.email();
...
template.set("user_info", tmp);
```

Хотя жизненный цикл этой переменной очень краток, она важна не только потому, что применяется как временное хранилище. Вместо имени tmp лучше использовать более содержательное название вроде user_info.

В следующем примере мы могли использовать имя tmp, но применили его как *часть* имени:

```
tmp_file = tempfile.NamedTemporaryFile()
...
SaveData(tmp_file, ...)
```

Обратите внимание на то, что мы назвали переменную tmp_file, а не просто tmp, поскольку она является объектом файла. Представьте, что мы назвали ее tmp:

```
SaveData(tmp, ...)
```

Если взглянуть только на эту строку кода, не совсем понятно, чем является tmp: файлом, именем файла или данными, которые мы записываем в файл.

СОВЕТ

Имя tmp должно использоваться только в тех случаях, когда вы объявляете переменную с небольшой продолжительностью жизни и эта информация о ней является самой важной.

Итераторы циклов

Имена вроде `i`, `j`, `iter` и `it` чаще всего используются в качестве индексов и итераторов цикла. Даже несмотря на то, что их имена абстрактны, чаще всего они рассматриваются именно как итераторы. (Фактически использование этих имен для какой-нибудь другой цели может кого-нибудь запутать — не делайте так!)

Однако иногда можно выбрать более полезные, чем `i`, `j` и `k`, имена для итераторов. Например, с помощью следующих циклов определяется, какие клубы посещают пользователи:

```
for (int i = 0; i < clubs.size(); i++)
    for (int j = 0; j < clubs[i].members.size(); j++)
        for (int k = 0; k < users.size(); k++)
            if (clubs[i].members[k] == users[j])
                cout << "user[" << j << "] is in club[" << i << "]" << endl;
```

В конструкции `if` массивы `members[]` и `users[]` используют неправильные индексы. Такие ошибки очень трудно распознать, поскольку в отдельности эта строка кода выглядит вполне правильной:

```
if (clubs[i].members[k] == users[j])
```

В данном случае может быть целесообразно применять более подробные имена. Вместо того чтобы называть индексы циклов (`i`, `j`, `k`), можно выбрать следующий вариант — `club_i`, `members_i`, `users_i` или более краткий — `ci`, `mi`, `ui`. Такой подход поможет вам обнаруживать некоторые ошибки:

```
if (clubs[ci].members[ui] == users[mi]) # Ошибка! Первые буквы имен
                                        # не совпадают.
```

Если же индексы использованы правильно, первые буквы их имен будут соответствовать первым буквам названий массивов:

```
if (clubs[ci].members[mi] == users[ui]) # Все в порядке.
                                        # Первые буквы совпадают
```

Вердикт по общим именам

Как вы могли заметить, применение общих имен иногда может быть полезным.

СОВЕТ

Если вы собираетесь использовать общее имя, например tmp, it или retval, у вас на это должна быть веская причина.

В большинстве случаев общие имена используются из-за обыкновенной лени. И это можно понять: когда ничего не приходит на ум, гораздо проще придумать какое-нибудь бессмысленное имя, как, например, `foo`, и писать программу дальше. Но если вы заведете привычку тратить несколько секунд на то, чтобы придумать хорошее имя, то вскоре обнаружите, что у вас это получается все лучше и лучше.