Хранение функций в модулях

Одно из преимуществ функций заключается в том, что они отделяют блоки кода от основной программы. Если для функций были выбраны содержательные имена, ваша программа будет намного проще читаться. Можно пойти еще дальше и сохранить функции в отдельном файле, называемом *модулем*, а затем *импортировать* модуль в свою программу. Команда `import` сообщает Python, что код модуля должен быть доступен в текущем выполняемом программном файле.

Хранение функций в отдельных файлах позволяет скрыть второстепенные детали кода и сосредоточиться на логике более высокого уровня. Кроме того, функции можно использовать во множестве разных программ. Функции, хранящиеся в отдельных файлах, можно передать другим программистам без распространения полного кода программы. А умение импортировать функции позволит вам использовать библиотеки функций, написанные другими программистами.

Существует несколько способов импортирования модулей; все они кратко рассматриваются ниже.

Импортирование всего модуля

Чтобы заняться импортированием функций, сначала необходимо создать модуль. *Модуль* представляет собой файл с расширением .py, содержащий код, который вы хотите импортировать в свою программу. Давайте создадим модуль с функцией `make_pizza()`. Для этого из файла pizza.py следует удалить все, кроме функции `make_pizza()`:

pizza.py

```
def make_pizza(size, *toppings):
    """Выводит описание пиццы."""
    print(f"\nMaking a {size}-inch pizza with the following toppings:")
    for topping in toppings:
        print(f"- {topping}")
```

Теперь создайте отдельный файл с именем making_pizzas.py в одном каталоге с pizza.py. Файл импортирует только что созданный модуль, а затем дважды вызывает `make_pizza()`:

making_pizzas.py

```
  import pizza

❶ pizza.make_pizza(16, 'pepperoni')
  pizza.make_pizza(12, 'mushrooms', 'green peppers', 'extra cheese')
```

В процессе обработки этого файла строка `import pizza` говорит Python открыть файл pizza.py и скопировать все функции из него в программу. Вы не видите, как происходит копирование, потому что Python копирует код незаметно для пользователя во время выполнения программы. Вам необходимо знать одно: что любая функция, определенная в pizza.py, будет доступна в making_pizzas.py.

Чтобы вызвать функцию из импортированного модуля, укажите имя модуля (`pizza`), точку и имя функции (`make_pizza()`), как показано в строке ❶. Код выдает тот же результат, что и исходная программа, в которой модуль не импортировался:

```
Making a 16-inch pizza with the following toppings:
- pepperoni

Making a 12-inch pizza with the following toppings:
- mushrooms
- green peppers
- extra cheese
```

Первый способ импортирования, при котором записывается команда `import` с именем модуля, открывает доступ программе ко всем функциям из модуля. Если вы используете эту разновидность команды `import` для импортирования всего модуля *имя_модуля*.py, то каждая функция модуля будет доступна в следующем синтаксисе:

```
имя_модуля.имя_функции()
```

Импортирование конкретных функций

Также возможно импортировать конкретную функцию из модуля. Общий синтаксис выглядит так:

```
from имя_модуля import имя_функции
```

Вы можете импортировать любое количество функций из модуля, разделив их имена запятыми:

```
from имя_модуля import функция_0, функция_1, функция_2
```

Если ограничиться импортированием только той функции, которую вы намереваетесь использовать, пример making_pizzas.py будет выглядеть так:

```
from pizza import make_pizza

make_pizza(16, 'pepperoni')
make_pizza(12, 'mushrooms', 'green peppers', 'extra cheese')
```

При таком синтаксисе использовать точечную запись при вызове функции не обязательно. Так как функция `make_pizza()` явно импортируется в команде `import`, при использовании ее можно вызывать прямо по имени.

Назначение псевдонима для функции

Если имя импортируемой функции может конфликтовать с именем существующей функции или функция имеет слишком длинное имя, его можно заменить коротким уникальным *псевдонимом* (alias) — альтернативным именем для функции. Псевдоним назначается функции при импортировании.

В следующем примере функции `make_pizza()` назначается псевдоним `mp()`, для чего при импортировании используется конструкция `make_pizza as mp`. Ключевое слово `as` переименовывает функцию, используя указанный псевдоним:

```
from pizza import make_pizza as mp

mp(16, 'pepperoni')
mp(12, 'mushrooms', 'green peppers', 'extra cheese')
```

Команда `import` в этом примере назначает функции `make_pizza()` псевдоним `mp()` для этой программы. Каждый раз, когда потребуется вызвать `make_pizza()`, достаточно включить вызов `mp()` — Python выполнит код `make_pizza()` без конфликтов с другой функцией `make_pizza()`, которую вы могли включить в этот файл программы.

Общий синтаксис назначения псевдонима выглядит так:

```
from имя_модуля import имя_функции as псевдоним
```

Назначение псевдонима для модуля

Псевдоним также можно назначить для всего модуля. Назначение короткого имени для модуля — скажем, `p` для `pizza` — позволит вам быстрее вызывать функции модуля. Вызов `p.make_pizza()` получается более компактным, чем `pizza.make_pizza()`:

```
import pizza as p

p.make_pizza(16, 'pepperoni')
p.make_pizza(12, 'mushrooms', 'green peppers', 'extra cheese')
```

Модулю `pizza` в команде `import` назначается псевдоним `p`, но все функции модуля сохраняют свои исходные имена. Вызов функций в записи `p.make_pizza()` не только компактнее `pizza.make_pizza()`; он также отвлекает внимание от имени модуля и помогает сосредоточиться на содержательных именах функций. Эти имена функций, четко показывающие, что делает каждая функция, важнее для удобочитаемости вашего кода, чем использование полного имени модуля.

Общий синтаксис выглядит так:

```
import имя_модуля as псевдоним
```

Импортирование всех функций модуля

Также можно приказать Python импортировать каждую функцию в модуле; для этого используется оператор *:

```
from pizza import *

make_pizza(16, 'pepperoni')
make_pizza(12, 'mushrooms', 'green peppers', 'extra cheese')
```

Звездочка в команде `import` приказывает Python скопировать каждую функцию из модуля `pizza` в файл программы. После импортирования всех функций вы сможете

вызывать каждую функцию по имени без точечной записи. Тем не менее лучше не использовать этот способ с большими модулями, написанными другими разработчиками; если модуль содержит функцию, имя которой совпадает с существующим именем из вашего проекта, возможны неожиданные результаты. Python обнаруживает несколько функций или переменных с одинаковыми именами, и вместо импортирования всех функций по отдельности происходит замена этих функций.

В таких ситуациях лучше всего импортировать только нужную функцию или функции или же импортировать весь модуль с последующим применением точечной записи. При этом создается чистый код, легко читаемый и понятный. Я включил этот раздел только для того, чтобы вы понимали команды `import` вроде следующей, когда вы встретите их в чужом коде:

```
from имя_модуля import *
```

Стилевое оформление функций

В стилевом оформлении функций необходимо учитывать некоторые подробности. Функции должны иметь содержательные имена, состоящие из букв нижнего регистра и символов подчеркивания. Содержательные имена помогают вам и другим разработчикам понять, что же делает ваш код. Эти соглашения следует соблюдать и в именах модулей.

Каждая функция должна быть снабжена комментарием, который кратко поясняет, что же делает эта функция. Комментарий должен следовать сразу же за определением функции в формате строк документации. Если функция хорошо документирована, другие разработчики смогут использовать ее, прочитав только описание. Конечно, для этого они должны доверять тому, что код работает в соответствии с описанием, но если знать имя функции, то, какие аргументы ей нужны и какое значение она возвращает, они смогут использовать ее в своих программах.

Если для параметра задается значение по умолчанию, слева и справа от знака равенства не должно быть пробелов:

```
def имя_функции(параметр_0, параметр_1='значение_по_умолчанию')
```

Те же соглашения должны применяться для именованных аргументов в вызовах функций:

```
имя_функции(значение_0, параметр_1='значение')
```

Документ PEP 8 (https://www.python.org/dev/peps/pep-0008/) рекомендует ограничить длину строк кода 79 символами, чтобы строки были полностью видны в окне редактора нормального размера. Если из-за параметров длина определения функции превышает 79 символов, нажмите Enter после открывающей круглой скобки в строке определения. В следующей строке дважды нажмите Tab, чтобы отделить список аргументов от тела функции, которое должно быть снабжено отступом только на один уровень.

Многие редакторы автоматически выравнивают дополнительные строки параметров по отступам, установленным в первой строке:

```
def имя_функции(
        параметр_0, параметр_1, параметр_2,
        параметр_3, параметр_4, параметр_5):
    тело функции...
```

Если программа или модуль состоят из нескольких функций, эти функции можно разделить двумя пустыми строками. Так вам будет проще увидеть, где кончается одна функция и начинается другая.

Все команды `import` следует записывать в начале файла. У этого правила есть только одно исключение: файл может начинаться с комментариев, описывающих программу в целом.