8.3. Глобальні та локальні змінні

Кожна змінна має свою область видимості. Область видимості всіх змінних обмежена блоком, в якому вони оголошені, починаючи з точки оголошення. Тому при введенні у програму функції виникає поділ змінних і відповідних їм даних на глобальні і локальні.

8.3.1. Глобальні змінні

Всередині функції можуть бути використані змінні, оголошені в основній програмі. Наприклад, маємо програму:

```
def f():
    print(Z)
Z = 1
f()
```

В результаті виконання такої програми буде виведене число 1.

Під час запуску програми функція `f()` буде інтерпретуватися лише після її виклику. Проте до виклику функції `f()` змінній `Z` присвоюється значення 1. Отже, коли дійде черга до виконання функції, змінна `Z` вже буде ініціалізована, а отже за оператором `print(Z)` буде виведене її значення.

Змінні, оголошені за межами функції, але доступні всередині функції, називаються глобальними.

8.3.2.Локальні змінні

Якщо в тілі функції ініціалізувати деяку змінну, то її областю видимості буде тіло цієї функції. Отже, використання такої змінної за межами функції недопустиме. Наприклад, маємо програму:

```
def f():
    Z = 1
f()
print(Z)
```

В результаті виконання такої програми отримаємо виняток: `NameError: name 'Z' is not defined`.

Змінні, оголошені всередині функції або вказані у списку формальних параметрів функції, називаються локальними. За межами функції її локальні змінні недоступні.

Обмежена доступність локальних змінних надає можливість використовувати змінні з одним і тим же ім'ям в різних функціях.

8.3.3. Зв'язок однойменних локальних і глобальних змінних

Під час написання програми може виникнути випадок, коли програма матиме локальні та глобальні змінні з однаковими іменами. В такому випадку може виникнути питання: що ж буде відбуватися з локальними та глобальними змінними під час зміни їх значень?

Наприклад, маємо програму:

```
def func():
    Z=2
    print('Z всередині функції =', Z)
Z=5
func()
print('Z за межами функції=', Z)
```

Після її виконання отримаємо:

```
Z всередині функції= 2
Z за межами функції= 5
```

Незважаючи на те, що значення змінної `Z` змінилося всередині функції, за межами функції воно залишилося незміненим. Це пов’язане з тим, що при виконанні функції `func()` було ініціалізовано локальну змінну `Z`, а отже областю видимості даної змінної стало лише тіло функції. Надане значення локальній змінній `Z` ніяк не вплинуло на значення глобальної змінної `Z`. Така реалізація забезпечує "захист" глобальних змінних від випадкової зміни в тілі функції.

Якщо всередині функції модифікується значення деякої змінної, то вона стає локальною, і її модифікація не призводить до зміни глобальної змінної з таким же ім'ям. Більш формально можна це сформулювати так: інтерпретатор Python вважає змінну локальною для даної функції, якщо в тілі функції є хоча б одна інструкція, що модифікує значення цієї змінної.

Інструкцією, що модифікує значення змінної, може бути оператор присвоєння «=», або використання змінної в якості параметра циклу for.

Зміна значень глобальних змінних у функції

Інколи виникає необхідність написання функцій, в середині яких зміна значень глобальних змінних є необхідною. В такому випадку для забезпечення можливості зміни значення глобальної змінної в середині функції необхідно оголосити цю змінну в тілі функції за допомогою ключового слова `global`:

```
def func():
    global Z
    print('Z в середині функції (до зміни) =', Z)
    Z=2
    print('Z в середині функції (після зміни) =', Z)
Z=5
func()
print('Z за межами функції =', Z)
```

Після виконання отримаємо:

```
Z в середині функції (до зміни) = 5
Z в середині функції (після зміни) = 2
Z за межами функції = 2
```

Проте, краще не виконувати зміну значень глобальних змінних всередині функції. Якщо за функцією необхідно змінити значення змінної, краще повернути це значення функцією, і в головній програмі виконати операцію зміни. Якщо слідувати цьому правилу, то функції виходять незалежними від коду головної програми, і їх можна легко копіювати з однієї програми в іншу.

8.3.4. Нелокальні змінні

Окрім локальних та глобальних змінних є ще один тип змінних, так звана «нелокальна» (nonlocal) змінна. Нелокальна змінна є чимось середнім між локальною та глобальною. Такі змінні зустрічаються в функціях, що визначені в середині інших функцій. Для оголошення нелокальної змінної використовується службове слово `nonlocal`.

```
def func():
    Z=2
    print('Z в середині функції (до зміни) =', Z)
    def func1():
        nonlocal Z
        Z=3
    func1()
    print('Z в середині функції (після зміни) =', Z)
Z=5
func()
print('Z за межами функції =', Z)
```

Опис нелокальної змінної `nonlocal Z` всередині функції `func1()` означає, що областю видимості цієї змінної є не тіло функції `func1()`, а тіло функції `func()`.

Після виконання отримаємо:

```
Z в середині функції (до зміни) = 2
Z в середині функції (після зміни) = 3
Z за межами функції = 5
```

Якщо ж в тілі функції `func1()` буде відсутній оператор `nonlocal Z`, то матимемо: глобальну змінну `Z`, локальну змінну `Z` для функції `func()` та локальну змінну `Z` для функції `func1()`, тобто три окремі незалежні змінні.

8.4. Правила локалізації

Програма на Python має модульну структуру і може складатися з ряду вкладених один в одного блоків (функцій). Головна програма – це найбільший блок. Змінні, описані в головній програмі, є глобальними і можуть використовуватись у всіх вкладених блоках. Змінні, описані в вкладеному блоці, є локальними і недоступні у зовнішніх блоках.

Для правильного використання змінних слід дотримуватись таких правил щодо їх ідентифікаторів:

- Кожний ідентифікатор має бути ініціалізований до його використання.
- Областю дії ідентифікатора є блок, в якому його ініціалізовано.