Основи мови Python

Синтаксис мови

Синтаксис мови Python, як і сама мова, доволі простий. Можна виділити наступні твердження.

Кінець рядка є кінцем інструкції (прикінцеві символи не потрібні).

Основна інструкція та вкладені інструкції (вкладений блок інструкцій) записуються відповідно до одного шаблону: основна інструкція завершується двокрапкою, на наступних рядках розміщуємо вкладені інструкції з однаковим відступом на початку рядків по відношенню до основної інструкції. Наприклад:

Основна інструкція:

Вкладений блок інструкцій

Тобто вкладені інструкції об'єднуються в блоки за величиною відступів. Відступ може бути будь-яким, головне, щоб в межах одного вкладеного блоку відступ був однаковим. Не варто забувати про читабельність коду: відступ в 1 пропуск читатиметься погано. Рекомендується використовувати **4 пропуски**. В більшості редакторів коду на такий відступ налаштовано клавішу **Tab**.

Можна бачити, що синтаксис оформлення основної інструкції та вкладеного блоку інструкцій істотно відрізняється від синтаксису більшості мов, в яких використовуються операторні дужки для виділення вкладеного блоку інструкції (наприклад, begin … end в Pascal або { … } в C та JS).

Розмір літер має значення, тобто великі й маленькі літери вважаються різними. Більшість службових слів (окрім: *False, None, True*) та вбудованих функцій пишуться маленькими літерами.

Коментарі призначені для вставки пояснень в текст програми. Інтерпретатор їх повністю ігнорує. Всередині коментаря може розміщуватися будь-який текст, у тому числі й інструкції, які виконувати не слід. Пам'ятайте, коментарі потрібні програмісту, а не інтерпретатору Python. Вставка коментарів до коду дозволить через деякий час шв[illegible] згадати призначення фрагмента коду.

У мові Python присутній тільки однорядковий коментар. Він починається з символу #. Однорядковий коментар може починатися не тільки з початку рядка, а й розташовуватися після інструкції. Якщо символ коментаря розмістити перед інструкцією, то вона також не буде виконана.

Змінні та ідентифікатори

Ідентифікатор – це ім’я, яке використовується для ідентифікації змінної, функції, класу, модуля або іншого об’єкта. При виборі ідентифікаторів необхідно дотримуватися таких правил:

Першим символом ідентифікатора може бути буква з латинського алфавіту (у верхньому або нижньому регістрі) або символ підкреслення «_».

Інша частина ідентифікатора (всі символи, крім першого) може складатися з букв (у верхньому або нижньому регістрі), символу підкреслення «_» або цифр (0-9).

Імена ідентифікаторів чутливі до регістру. Наприклад, *myname* і *myName* - це два різні ідентифікатори (зверніть увагу на регістр літер "n" та "N").

Ідентифікатор не може співпадати з ключовими (зарезервованими) словами інтерпретатора Python.

При написанні програм досить часто необхідно зберігати різні дані та мати можливість маніпулювати цими даними. Ось тут якраз і знадобляться змінні. Слово «змінні» говорить саме за себе - їхнє значення може змінюватися, а значить, можна зберігати в змінних усе що завгодно. ***Змінна в мові Python*** – це просто посилання на ділянку пам'яті комп'ютера, в якій зберігаються деякі дані (посилання на деякий об’єкт). Для того, щоб задати імена змінним, використовуються ідентифікатори. У змінних можна зберігати значення різних типів.

Типи даних

Мова Python належить до мов з неявною строгою динамічною типізацією.

Неявна типізація означає, що при оголошенні змінної її тип не вказується.

Для мов з динамічною типізацією тип змінної визначається безпосередньо при виконанні програми. Окрім того, можна зазначити:

будь-яка змінна є посиланням;

типом змінної є те, на що вона посилається;

тип змінної може довільно змінюватися по ходу виконання коду, коли змінна починає посилатися на інший об’єкт.

Строга типізація (сильна типізація, або *strong typing*) не дає можливості проводити операції у виразах з даними різних несумісних типів.

У Python типи даних можна розділити на вбудовані в інтерпретатор (built-in) і невбудовані, які можна використовувати при імпортуванні відповідних модулів. До основних вбудованих типів належать:

1. ***None*** (невизначене значення змінної)

2. Логічний тип (Boolean Type)

a. ***bool***

3. Числа (Numeric Type)

a. ***int*** – ціле число

b. ***float*** – число з плаваючою точкою (дійсне число)

c. ***complex*** – комплексне число

4. Послідовності (Sequence Type)

a. ***list*** – список

b. ***tuple*** – кортеж

c. ***range*** – діапазон

5. Рядки (Text Sequence Type)

a. ***str***

6. Бінарні послідовності (Binary Sequence Types)

a. ***bytes*** – байти

b. ***bytearray*** – масиви байт 23

c. ***memoryview*** – спеціальні об'єкти для доступу до внутрішніх даних об'єкта через protocol buffer

7. Множини (Set Types)

a. ***set*** – множина

b. ***frozenset*** – незмінювана множина

8. Словники (Mapping Types)

a. ***dict*** – словник

Для перетворення типів, як правило, використовують функції, ідентичні до назви типів: `int()`, `str()`, `float()`, `list()` і т.д.

Ініціалізація змінних

Враховуючи неявну типізацію мови Python при оголошенні змінної, їй повинно бути надане значення (вона має бути ***ініціалізована***). Щоб оголосити та ініціалізувати змінну, необхідно написати її ім'я, потім поставити оператор присвоєння (знак рівності `=`) і вказати значення, з яким дана змінна буде створена. Наприклад: `z = 5`.

Також при ініціалізації змінних можливі наступні вирази:

```
x, y, z = 2, 3, 4

a = b = c = 2
```

У першому випадку змінним `x, y, z` будуть надані значення відповідно `2`, `3`, `4`. В другому випадку змінним `a, b, c` буде надане значення `2`.

Для обміну значеннями двох змінних можна використати

```
x, y = y, x
```

Введення та виведення даних

Для введення даних у Python призначена функція `input()`. Вона призупиняє виконання програми і чекає, доки користувач введе деякий текст. Отримавши дані, Python збе їх у змінній, щоб вам було зручніше працювати з ними. Функція має наступний фор

```
a = input('Вкажіть своє ім'я')
```

де `a` - змінна, *`input()`* - функція, *'Вкажіть своє ім'я'* - підказка, яка точно повідомить користувачеві, яку інформацію ви від нього хочете отримати.

При використанні функції *`input()`* Python інтерпретує всі дані, введені користувачем як *рядок*.

Вивести результати роботи програми можна за допомогою функції `print()`. Функція ***`print()`*** перетворює об'єкт на рядок і посилає її в стандартний вивід.

Функція має наступний формат:

```
print([<Об’єкти>][, sep=' '][, end='\n'])`
```

де ***sep*** (символ між елементами виводу, основний — пробіл) та ***end*** (символ в кінці виведення, основний — перехід на нову стрічку `’*\n*’)`. Символи можна задати свої.

Операції над числами

+	Додавання
-	Віднімання
*	Множення
/	Ділення
//	Знаходження цілої частини від цілочисельного ділення
%	Знаходження залишку від цілочисельного ділення
**	Піднесення до степеня

Також згадаємо деякі найбільш використовувані функції

abs(x)	повертає абсолютне значення (модуль) числа
round(number [, ndigits])	повертає число number, округлене до ndigits знаків після десяткової точки