Язык программирования Python

04 Переменные

Литерал

Литерал — это фиксированное значение в коде.

Рассмотренные в предыдущей теме выражения состояли из литералов целочисленного и вещественного типа.

Числа 1, 100, –5, 34.0, 1.0009 в коде — это числовые литералы.

Введенное число помещается в ячейку памяти компьютера, но в какую? Как обращаться к этим данным?

Справиться с этой проблемой нам помогают переменные.

Переменная

С помощью переменных мы можем обратиться к ячейкам памяти, где лежат наши данные.

Связать между собой данные и имя переменной позволяет присваивание.
Оператор присваивания (=) используется для присвоения значения переменной.

```
>>> a = 2
>>> b = 3
>>> a * b
6
```

Имя переменной, стоящее слева от знака присваивания, связывается со значением, стоящим справа.

Правила именования переменных

- имя переменной может состоять только из цифр, букв и нижнего подчеркивания;

```
side_a, my_super_variable, i, j, n12345, _b678 # годится
:%var, $#length, ?!yes # не годится
```

- имя переменной не может начинаться с цифры.

```
2my_var, 34number # не годится
```

Называйте переменные так, чтобы другие программисты, взглянув на ваш код, могли разобраться, что и куда.

Если имя переменной состоит из нескольких слов, то разделяйте их нижним подчеркиванием.

Оператор присваивания

Знак оператора присваивания — равно (=), но это не операция равенства.

Когда переменной присваивается значение, она становится определенной.

При использовании неопределенной переменной, интерпретатор выдаст ошибку о том, что имя не определено.

```
>>> a = 1
>>> b = 2
>>> a
1
>>> b
2
>>> c
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    c
NameError: name 'c' is not defined
>>>
```

Знак операции равенства — два подряд идущих знака равенства (==) без пробела между ними.

Нижнее подчеркивание

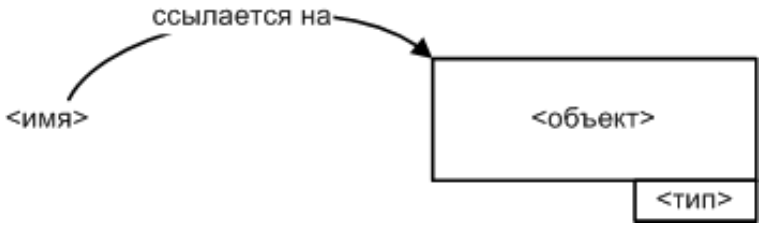
В интерактивном режиме результат вычисления последнего выражения присваивается переменной _. Это облегчает использование интерпретатора как калькулятора.

Однако, присваивать значение этой переменной «руками» не стоит, так как она «только для чтения». При присвоении своего значения переменной _ вы создадите другую переменную с тем же именем, но скрывающую встроенную _ (в вашу переменную значение последнего выражения записываться не будет).

Переменная

Предположим, у нас есть 2 переменных a и b.

```
>>> a = 1
>>> b = 2
```



Выведем a.

```
>>> a
1
```

Переменные можно присваивать друг другу, переменная a будет ссылаться на то же объект, что и b.

```
>>> a = b
>>> a
2
```

Присваивание

Существует обычное и множественное присваивания.

Обычное присваивание мы уже рассматривали, выглядит оно следующим образом:

```
>>> a = 1 # a ссылается на объект типа int со значением 1
>>> b = 2 # b ссылается на объект типа int со значением 2
>>> a = b
```

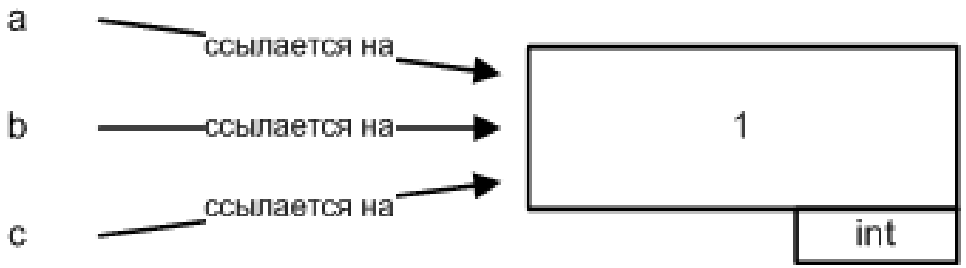
В последней строке переменная a после присваивания начинает ссылаться на тот же объект, что и b.

Присваивание

Также имеется возможность присвоить одно значение группе переменных.

```
>>> a = b = c = 1
```

Перменные a, b и c ссылаются на один и тот же объект типа int со значением 1



Множественное присваивание

Множественное присваивание реализовано следующим образом:

```
>>> a, b, c = 1, 2, 3 # a = 1, b = 2, c = 3
>>> a
1
>>> b
2
>>> c
3
```

Перестановка значений переменных

Существует классическая задача о перестановке значений 2 переменных: дано 2 переменных и необходимо, чтобы переменные обменялись значениями.

- Без использования множественного присваивания решение бы выглядело так:

```
>>> tmp = a # вводится дополнительная переменная tmp
>>> a = b
>>> b = tmp
```

(tmp — temporary — временный)

- С использованием множественного присваивания:

```
>>> a, b = b, a
```

Составные операторы присваивания

```
>>> a = 1
>>> a += 1 # равносильно a = a + 1
>>> a
2

>>> a = 1
>>> a -= 1 # равносильно a = a - 1
>>> a
0
```

А также *=, /=, //=, %= и так далее.

Сборщик мусора

```
>>> a = 1 # a ссылается на объект типа int со значением 1
>>> b = 2 # b ссылается на объект типа int со значением 2
>>> a = b
```

В последней строке имя a перестает ссылаться на 1 и так как на этот объект больше не ссылается ни одна переменная, то он удаляется сборщиком мусора.

>>> a = b = c = 1 # a, b и c ссылаются на один объект типа int со значением 1

Сборщик мусора удалит этот объект в случае если все 3 переменные перестанут ссылаться на него.

Пример

- Задача
  Необходимо найти площадь прямоугольника со сторонами 4 и 6.

- Решение
  Свяжем 4 и 6 с именами side_a (сторона a) и side_b (сторона b) соответственно.

```
>>> side_a = 4
>>> side_b = 6
```

  Найдем площадь (area).

```
>>> area = side_a * side_b
>>> area
24
```

Итоги

Мы познакомились с такими понятиями, как литерал, переменная, присваивание, оператор присваивания, множественное присваивание, определение переменной, составные операторы присваивания, сборщик мусора,

а также попрактиковались в работе с интерпретатором в интерактивном режиме.