Работа с переменными

Создание переменных

Роман Булгаков / Skillbox

Переменная в Python — это ссылка на объект, а не сам объект. Объект (например, строка или число) создаётся и хранится в памяти компьютера, а переменная лишь ссылается на место в компьютере, где хранятся эти данные.

Оператор ввода input

`input` — функция, которая позволяет запрашивать ввод из консоли.

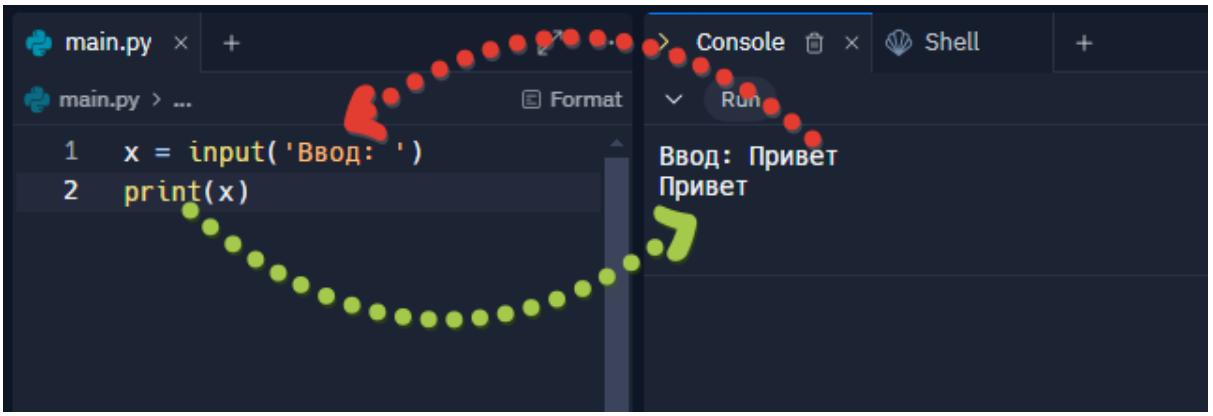
- Если мы напишем:

  ```
  x = input('Ввод: ')
  ```

  и выполним код, то в консоли появится слово «Ввод» и будет ожидаться ввод данных.

- Если мы введём данные и нажмём Enter — этот ввод сперва превратится в строку, а затем будет записан в переменную, которую мы использовали.

Пример:



То, что мы ввели, вернулось результатом выполнения функции и было записано в переменную `x`.

После этого мы с помощью `print` вывели переменную `x`, чтобы убедиться, что всё сработало, как задумано. Советуем выполнять такие проверки.

Обратите внимание

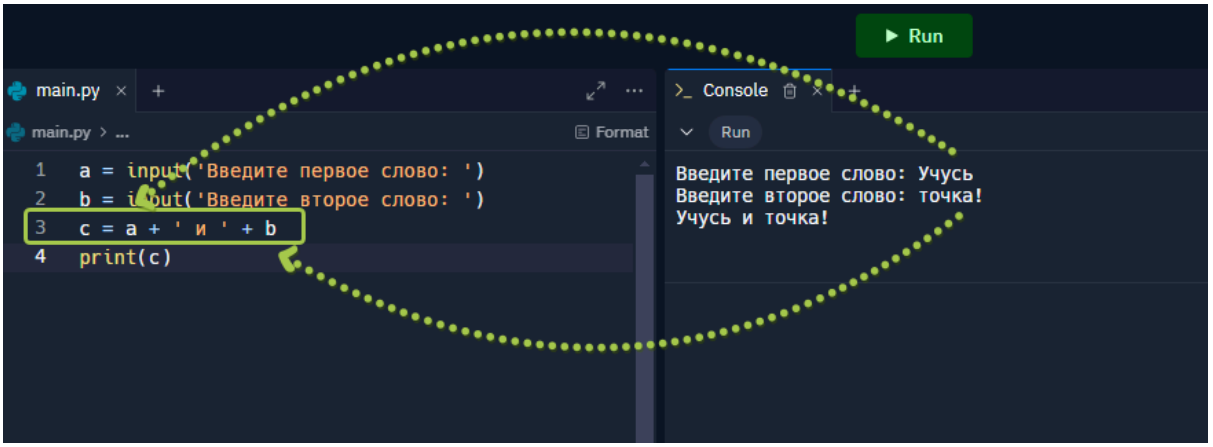
1. Мы передали в функцию `input` параметр-строку `'Ввод: '`. Эта строка служит приглашением к вводу и подсказывает пользователю, что от него ожидается. Например, это может быть запрос логина и пароля или даты рождения (вводы могут быть разными). Важно писать понятные сообщения, чтобы пользователь знал, что ему нужно ввести.

2. Функция `input` (в отличие от `print`) принимает только одну строку. В `print` вы можете передать множество объектов, и эта функция сама объединит их в единую, добавив в неё свой стандартный разделитель-пробел между элементами. Функция `input` так не умеет.

Конкатенация строк

Конкатенация — это «склеивание» строк с помощью оператора `+`. Этот приём часто используется для подстановки каких-либо изменяемых значений.

Пример:



В этом примере мы:

1) запросили два ввода;
2) записали их в переменные `a` и `b`;
3) составили из этих переменных и строки `' и '` новую строку;
4) вывели строку `c` в консоль с помощью `print`.

Не забывайте, что пробел — такой же символ, как и все остальные. Если он вам нужен — добавляйте его, как показано в примере выше, иначе ваши строки склеятся без разделителя, а сообщение превратится в «кашу» из букв.

Названия для переменных (нейминг)

Следуйте этим правилам, чтобы ваш код соответствовал мировым стандартам стиля кода:

1. Используйте в названиях переменных только латинские символы.
2. Не используйте специальные символы.
3. Не используйте пробелы, вместо них используйте нижнее подчёркивание.
4. Старайтесь придумывать такие названия, которые будут отражать содержимое переменных.

Например, если вы запрашиваете у пользователя возраст — используйте название `age`:

```python
age = input('Введите свой возраст: ')
```

Так будет ясно, что переменная относится к возрасту.

Множественное присваивание

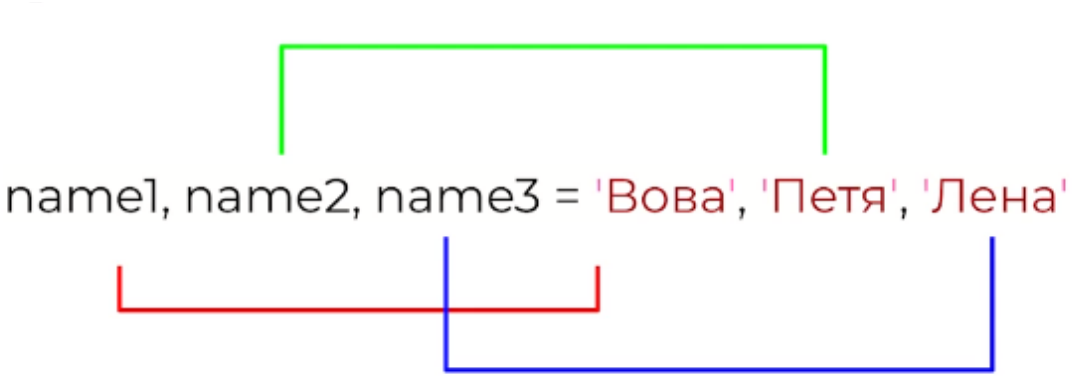
Если нужно присвоить несколько схожих объектов разным переменным, используйте множественное присваивание:

```python
a, b = 'Привет', 'Мир'
```

Слева через запятую перечисляются переменные, а справа через запятую — объекты в том же порядке. Это эквивалентно записи:

```python
a = 'Привет'
b = 'Мир'
```

От порядка перечисления зависит, в какую переменную попадёт объект.



Роман Булгаков / Skillbox

Не злоупотребляйте этим приёмом. Помните, что длинные строки снижают читаемость кода. Кроме того, если элементы будут разного типа, будет сложно воспринимать такое присваивание.

Например:

```python
first_car, important_documents, surnames = 'Toyota',
('doc1.txt', 'doc_2.xls', 'doc-3.docx'), ['Петрова', 'Иванов',
'Смирнов']
```

Такой код читать сложнее, чем:

```python
first_car = 'Toyota'
important_documents = ('doc1.txt', 'doc_2.xls', 'doc-3.docx')
surnames = ['Петрова', 'Иванов', 'Смирнов']
```

Как не допустить ошибку

1. Не пишите название переменной в кавычках, если хотите использовать объект, на который она ссылается.

```python
x = 5
print('x')  # Выведет строку 'x'.
print(x)    # Выведет число 5.
```

Если вы используете `print('x')` — Python не поймёт, что вы хотите вывести число 5. Программа подумает, что вы хотите вывести строку `'x'` и выведет её.

`print(x)` — правильный способ обращения к переменной.

2. Python — регистрозависимый язык, поэтому большие и маленькие буквы воспринимаются им как совершенно разные символы:

```python
x = 5
print(X)  # Приведёт к ошибке, потому что x и X — разные
переменные.
```