PEP 8. Ввод-вывод. Переменные

Аннотация

На этом уроке мы продолжим практиковаться во вводе и выводе строк. Не будем забывать о правилах оформления программ PEP 8. А также познакомимся с переменными.

1.Повторение

На прошлом уроке мы научились запускать простую программу, которая выводит на экран сообщение, а также вводить данные с клавиатуры и запоминать их в переменной. Затем их можно использовать в программе. Сегодня мы познакомимся с правилами именования переменных.

Вспомним:

Если нужно, чтобы программа что-то печатала на экране (и это увидел пользователь) — `print`. Если нужно, чтобы пользователь что-то напечатал с клавиатуры (чтобы программа могла использовать эти данные) — `input()`.

2.Переменные

Команду `name = input()` можно считать так: «Подожди, пока пользователь введет какую-то строку и помести введенную строку в переменную `name`».

Попробуем разобраться, что это значит. Переменные имеют имя и значение.

Имя переменной

Имя переменной должно отражать ее назначение и может состоять из латинских букв, цифр и символа подчеркивания.

Имя не может начинаться с цифры.

PEP 8

Для именования переменных принято использовать стиль

lower_case_with_underscores (слова из маленьких букв с подчеркиваниями).

Избегайте использовать такие символы, которые могут не однозначно трактоваться в различных шрифтах: это буква O (большая и маленькая) и цифра 0, буква I (большая и маленькая) и цифра 1. Нельзя использовать в качестве имени переменной и ключевые слова, которые существуют в языке.

В вышеописанном примере переменная содержит в себе имя пользователя, поэтому мы назвали ее `name` (имя). Обратите внимание: слово `name` не подсвечено никаким цветом — в Python это слово ничего не обозначает. Оно что-то значит только в этой программе и только потому, что мы употребили оператор присваивания. При этом интерпретатору совершенно неважно, что значит слово `name` в английском языке, и мы с тем же успехом могли использовать любое другое имя: например, `user` («пользователь») или просто `n`, или даже `hello`. За имена переменных отвечает программист, то есть вы.

Соблюдайте правило: если в переменной хранится приветствие, пусть она так и называется, если имя — пусть она и называется соответственно.

Значение переменной

Значение переменной — то, что сохраняет в себе переменная.

Знак «=» обозначает команду под названием «оператор присваивания». Оператор присваивания присваивает значение, которое находится справа от знака равно, переменной, которая находится слева от знака равно.

В нашем случае это то, что поместил в нее пользователь командой `input()`. Это текстовое значение — строка. То есть переменная сохраняет в себе строковое значение. Говорят, что переменная строкового типа.

PEP 8

Всегда окружайте оператор присваивания одним пробелом с каждой стороны:

Правильно:

```
bird = "Тук-тук"
```

~~**Неправильно:**~~

```
bird="Тук-тук"
```

Еще пример:

```
print('Какая твоя любимая еда?')

meal = input()

print('Да.', meal, '- это вкусно.')
```

Обратите внимание: интерпретатор ждет, что пользователь что-то введет с клавиатуры ровно столько раз, сколько команд `input()` встречается в программе. Каждый `input()` завершается нажатием Enter на клавиатуре.

```
print('Как тебя зовут?')

name = input()

print('Привет,', name)

print('А какая твоя любимая еда?')

meal = input()

print('Да.', meal, '- это вкусно.')
```

Мы задали значение переменной. И что же, оно никогда не меняется? Конечно, в двух разных программах могут быть переменные с одинаковыми названиями, но разными значениями. Но могут ли в пределах одной программы под одним именем быть разные значения?

Да! Оператор присваивания сообщает переменной то или иное значение независимо от того, была ли эта переменная введена раньше. Вы можете менять значение переменной, записав еще один оператор присваивания. Если у нас имеется переменная, мы можем делать с ее значением все что угодно — например, присвоить другой переменной:

```
hello = 'Здравствуйте.'

hello2 = hello

print(hello2)
```

Итак, если вы хотите, чтобы у вас была переменная с каким-то именем и каким-то значением, нужно написать на отдельной строчке:

```
<имя переменной> = <значение переменной>
```

Как только эта команда выполнится, в программе появится указанная переменная с таким значением.

Помните: команды выполняются последовательно, в том же порядке, в котором они написаны.

А теперь **смоделируем небольшой взлом программы**.

Загрузите файл guessing_game.py. Это тоже программа на Python. Для начала просто запустите ее (двойным кликом). Как видите, это игра-угадайка. Шанс угадать невелик, у кого получится — тот везучий. Как бы облегчить выигрыш?

Теперь откройте эту программу в редакторе. Это делается не двойным кликом, а кликом правой кнопкой и выбором пункта **Edit with Wing IDE** (или **Редактировать с помощью Wing IDE**). Смысл некоторых строчек этой программы вы уже знаете. О других скоро узнаете или сможете догадаться.

Сейчас мы научимся жульничать в игре путем изменения этой программы. Конечно, можно было бы заменить всю программу единственной строчкой — поздравлением с победой. Но мы будем считать, что менять можно только одну строчку — ту, которая сейчас пуста. Мы можем вписать туда любую команду.

Мы знаем, что к тому моменту, когда выполнение программы доходит до пустой строчки, в переменной под названием `planet` лежит название загаданной планеты. Сделайте так, чтобы в этот момент оно выводилось прямо на экран, — тогда игроку останется лишь повторить название для победы (это потребуется вам в задаче «Взлом планетной угадайки»).

3.Трассировка

Задача (для разбора). Предположим, у нас есть программа, которая входит в интерфейс сайта «Госуслуги» и служит для смены имени. Как будет работать эта программа, что она выведет при каком-либо пользовательском вводе?

```
print('Введите фамилию:')
surname = input()
print('Введите имя:')
name = input()
print(name, surname)
print('Введите новое имя:')
new_name = input()
print(name, surname)
```

```
print(new_name, name)
name = new_name
print(new_name, name)
print(name, surname)
```

Давайте немного изменим программу и посмотрим, что теперь получится.

```
print('В в е д и т е  ф а м и л и ю:')
surname = input()
print('В в е д и т е  и м я:')
name = input()
print(name, surname)
print('В в е д и т е  н о в о е  и м я:')
new_name = input()
old_name = name
name = new_name
print(new_name, old_name)
print(name, surname)
```

Если вы написали программу и не уверены в правильности ее написания, разберите, что она делает. Если не уверены, что все сделали правильно, — запустите и проверьте свои рассуждения.

Комментарии

Для удобства можно использовать комментарии, которые позволяют программисту делать для себя пометки в коде или делать часть кода не выполнимой, не видимой для интерпретатора.

Если вы начнете строчку со знака решетки #, интерпретатор Python будет игнорировать всю эту строчку. Программа будет выполняться так, как будто строчки нет. Такая строчка называется **комментарием**.

Комментарии нужны в двух случаях:

Когда нужно добавить в программу какую-то пометку для человека, который будет читать эту программу (например, см. третью строчку guessing_game).

Когда нужно убрать какую-то строчку кода, но удалять ее не хочется (например, потом ее, возможно, понадобится вернуть). Это называется «закомментировать» строчку.

PEP 8

«Встрочные» комментарии находятся в той же строке, что и

инструкция. Они должны **отделяться по крайней мере двумя пробелами** от инструкции и начинаться с символа # и одного пробела.

Комментарии в строке с кодом не нужны и только отвлекают от чтения, если они объясняют очевидное.

Правильно:

```
x = x + 1  # компенсация границы
```

~~Неправильно:~~

```
x = x + 1  # увеличение на единицу
```

Если нужно закомментировать сразу несколько строчек подряд, не делайте это вручную. Выделите эти строчки и выберите **Source → Toggle block comment** (там же можно убрать комментирование).

Заметьте, что в конце guessing_game на отдельной строчке стоит знакомая команда `input()`. Зачем она нужна?

Ответ: когда запускаешь программу двойным кликом, окно с программой закрывается сразу, как только программа заканчивает работу. Если программа что-то выводит на экран в конце работы, пользователь этого не увидит. А так программа ждет, пока пользователь нажмет клавишу Enter. Конечно, он может что-то ввести, но это неважно — мы все равно никак не используем этот ввод. Попробуйте закомментировать последнюю строчку guessing_game и запустить программу из проводника двойным кликом.

Заметьте: если запустить программу, у которой в конце `input()`, из редактора Wing IDE, в конце ее работы придется лишний раз нажать Enter, хотя в этом и нет необходимости.

Попробуйте запустить программу hello или любую другую из своих старых программ двойным кликом — сначала в исходном виде, потом с `input()` в конце.