

Переменные и типы данных

№ урока: 2 Курс: Python Starter

Средства обучения: PyCharm

Обзор, цель и назначение урока

После завершения урока обучающиеся будут иметь представление о переменных и константах, арифметических и логических операциях, смогут форматировать и выводить информацию на экран.

Изучив материал данного занятия, учащийся сможет:

- Хранить значения в переменных и работать с ними
- Вычислять в своих программах значения арифметических и логических выражений
- Пользоваться форматированным выводом

Содержание урока

1. Переменные и константы
2. Арифметические операции
3. Операции сравнения
4. Логические операции
5. Форматирование и вывод строк на экран

Резюме

Компьютеры и его программы работают с данными. Приложение Instagram загружает данные о новых подписчиках, компьютеры Facebook обрабатывают запросы пользователей и отдают нужные данные в ответ. Даже этот документ содержит данные, которые компьютер обработал, чтобы отобразить на экране.

Данные — это важная часть любой программы. Их можно хранить в постоянной памяти (ПЗУ) или в оперативной (ОЗУ). В постоянной памяти хранятся фотографии, музыка, системные файлы и прочее. В оперативной памяти хранятся данные, что программам необходимы вовремя их работы: браузеру необходимо хранить страницу, а Skype хранит сообщения.

Как правило, постоянной памяти намного больше, чем ОЗУ. При этом скорость работы ОЗУ намного выше скорости ПЗУ. Именно из-за скорости программам выгодно хранить данные в ОЗУ, чтобы быстро при необходимости их достать, обработать и отобразить.

Обратите внимание, что, когда программа завершается, она теряет доступ к данным, которые она хранила в ОЗУ. Поэтому перед завершением программы, важные данные необходимо сохранить в файл на ПЗУ.

Переменная — это область памяти ОЗУ, которой программист дал имя. В дальнейшем в этой области памяти можно сохранять значения. В одной программе могут быть десятки тысяч переменных для разных данных.

Типизация

В Python любые данные характеризуются своими типами. **Тип данных** определяет множество допустимых значений, которые может принимать переменная данного типа и множество допустимых операций, применимых к этому типу.

Для человека понятно, что автомобиль предназначен для езды и на нём не полетишь в космос (пока что).

Для компьютера не все так очевидно. Ему явно надо сказать, что *“это число”*, а значит с этим число можно делать математические операции. *“А это строка”* - значит можем соединить строки, но никак не их квадрат.

То есть, тип определяет возможные значения и их смысл, операции, а также способы хранения значений типа.

Операция назначения типа **называется типизацией**. Назначение и проверка типов может осуществляться заранее (статическая типизация), непосредственно при использовании (динамическая типизация) или совмещать оба метода. Типы могут назначаться «раз и навсегда» (сильная типизация) или позволять себя изменять (слабая типизация).

Статическая типизация — приём, при котором переменная связывается с типом в момент объявления и тип не может быть изменён позже. Примеры статически типизированных языков — C#, Java, C++, Pascal, Haskell, Scala.

Динамическая типизация — приём, при котором переменная связывается с типом в момент присваивания значения, а не в момент объявления переменной. Таким образом, в различных участках программы одна и та же переменная может принимать значения разных типов. Примеры языков с динамической типизацией — Smalltalk, Python, Ruby, PHP, Perl, JavaScript, Lisp, Erlang.

По одной из классификаций, языки программирования неформально делятся на сильно (строго) и слабо типизированные (англ. strongly and weakly typed), то есть обладающие сильной или слабой системой типов. Эти термины не являются однозначно трактуемыми, и чаще всего используются для указания на достоинства и недостатки конкретного языка. Существуют более конкретные понятия, которые и приводят к называнию тех или иных систем типов «сильными» или «слабыми».

Система типов называется сильной, если она исключает возможность возникновения не проконтролированных ошибок времени выполнения, связанных с согласованием типов.

Примеры языков с сильной типизацией — Haskell, Scala, Java. Примеры языков со слабой типизацией — C, JavaScript, PHP.

Python – язык с сильной динамической типизацией.

Константа в программировании — способ адресации данных, изменение которых рассматриваемой программой не предполагается или запрещается. Использование именованных констант — приём, повышающий надёжность и безошибочность программ, позволяя избегать использования «магических чисел».

В Python нет отдельного механизма для объявления констант. Принято называть их идентификаторами, написанными заглавными буквами (MY_CONSTANT), тогда как переменные в Python объявляются с именами из строчных букв (my_variable). Это не более, чем соглашение, между программистами.

Идентификатор – это имя, по которому можно обращаться к какому-либо объекту. Он может состоять из больших и маленьких букв, цифр, знаков подчёркивания, не должен начинаться с цифры и не может совпадать с зарезервированными ключевыми словами: False, class, finally, is, return, None, continue, for, lambda, try, True, def, from, nonlocal, while, and, del, global, not, with, as, elif, if, or, yield, assert, else, import, pass, break, except, in, raise.

Обратите внимание, Python чувствителен к регистру: identifier и Identifier — это два разных имени.

Типы данных

- **None** – специальное значение типа NoneType, которое используется, чтобы указать отсутствие значения.
- **int** – тип данных, который используется для целых чисел. Примеры: 3, 0, -1, 256123462346523235252 (десятичная система счисления), 0b11101 (двоичная система счисления), 0o730 (восьмеричная система счисления), 0x7de (шестнадцатеричная система счисления).
- **bool** – логические значения. Есть два допустимых значения: True (истина) и False (ложь).
- **float** – действительные числа двойной точности. Примеры: 0.3215235, 125256.73, -2.3e8 (экспоненциальная форма записи).
- **complex** – комплексные числа. Примеры: 2 + 3j, 0.7 - 0.1j
- **str** – строки.

Кроме вышеперечисленных типов данных, есть ещё огромное количество других типов данных, которые будут рассмотрены в курсах Python Essential и Python Advanced, а также имеется

возможность создавать свои типы данных. Любая сущность в Python является объектом с определённым типом.

Переменные в Python – это ссылки, имена, которые привязываются к объектам. Можно привязать несколько разных имён к одному объекту, и тогда при изменении объекта по одному имени, все эти изменения будут доступны и по остальным.

Объекты бывают изменяемые (mutable) и неизменяемые (immutable). Объекты рассмотренных выше типов являются неизменяемыми.

Операции с числами:

Код	Значение
$x + y$	сумма
$x - y$	разница
$x * y$	произведение
x / y	частное
$x // y$	операция целочисленного деления
$x \% y$	остаток от деления
$-x$	число, противоположное x
$+x$	x
$\text{abs}(x)$	модуль числа x
$\text{int}(x)$	преобразовать x в целое число
$\text{float}(x)$	преобразовать x в комплексное число
$\text{complex}(re, im)$	создать комплексное число $re + im*i$
$\text{c.conjugate}()$	число, сопряжённое комплексному числу c
$x ** y, \text{pow}(x, y)$	x в степени y
$\text{round}(x), \text{round}(x, n)$	округлить действительное число x (до n цифр после запятой, если n указано)

У арифметических операций есть приоритеты (как и в математике). Чтобы изменить порядок вычислений, следует использовать круглые скобки.

Арифметические операции, первым операндом которых является та же переменная, что и результат, можно записывать в сокращённой форме, например, $x += y$ вместо $x = x + y$.

Если импортировать модуль `math`:

```
import math
```

то можно также использовать ряд его функций и констант. Среди них:

Код	Значение
$\text{math.trunc}(x)$	отбросить дробную часть действительного числа x , возвращает целое число
$\text{math.floor}(x)$	наибольшее целое число (целое в математическом смысле, а не как тип данных), которое не превосходит данное вещественное число
$\text{math.ceil}(x)$	наименьшее целое число, большее или равное данному вещественному
$\text{math.pi}, \text{math.e}$	константы π , e
$\text{math.sin}, \text{math.cos}$	математические функции
и т.д.	

Полный список математических функций можно посмотреть в документации или набрав в консоли интерпретатора:

```
import math  
dir(math)
```

Логические операции:

Код	Значение
x or y	если x – ложь, то y, иначе x
x and y	если x – ложь, то x, иначе y
not x	если x – ложь, то True, иначе False

Операции сравнения:

Код	Значение
x < y	x строго меньше y
x > y	x строго больше y
x <= y	x меньше или равен y
x >= y	x больше или равен y
x == y	x равен y
x != y	x не равен y
x is y	x и y – это один и тот же объект
x is not y	x и y не являются одним и тем же объектом в памяти

Можно также использовать двойные сравнения, например: `-2 <= x < 3`, `a < b < c`. Некоторые операции со строками:

Код	Значение
s1 + s2	конкатенация (объединение) строк
s % x, s % (x1, x2, ..., xn)	форматирование строки в стиле C, s – форматная строка, x1...xn – значения, https://docs.python.org/3/library/stdtypes.html#printf-style-string-formatting
s.format(args)	форматирование строки в стиле C#, https://docs.python.org/3/library/stdtypes.html#str.format

Для вывода значений на экран служит функция print:

`print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)`

где: * objects – это объекты, которые необходимо вывести (символ * в объявлении функции значит, что при вызове функции они все передаются как отдельные параметры функции, а интерпретатор Python объединяет их в список и передаёт функции как один аргумент) * sep – разделитель, который вставляется между выводом отдельных объектов (по умолчанию пробел) * end – строка, которую необходимо вывести после всех объектов (по умолчанию – символ новой строки) * file – файл, в который необходимо вывести данные. В рамках этого курса мы не будем использовать этот параметр, так как файлы рассматриваются в курсе Python Essential * flush – нужно ли сразу после вывода сбросить содержимое буфера в файл. Если вы выводите информацию на одной строке через продолжительные промежутки времени, и она не появляется на экране, пока вы не выведете символ новой строки, добавьте параметр `flush=True`

Ни один из этих параметров не является обязательным.

Для ввода данных с клавиатуры можно использовать функцию input: `input(prompt)` или `input()`

Если параметр prompt задан, то он выводится, как поясняющий текст, приглашение ко вводу.

Функция приостанавливает выполнение программы, пока пользователь не введёт строку текста, считывает её и возвращает.

Обратите внимание, что она возвращает именно строку, то есть значение типа str, поэтому если необходимо ввести число, то нужно воспользоваться одной из функций, которые были рассмотрены выше, чтобы сконструировать число из его текстового представления.

Закрепление материала

- Что такое переменная?
- Что такое тип данных?
- В чём разница между статической и динамической типизацией?

- В чём разница между слабой и сильной (строгой) типизацией?
- Охарактеризуйте типизацию в языке Python.
- Что такое идентификатор?
- Какие основные числовые типы данных есть в Python?
- Что такое None?
- Какие два способа форматирования строк есть в Python?
- Какие аргументы может принимать функция print?
- Каким будет результат выражения `3 != 4 and not ("test" != "test" or "Python" == "Python")`?

Дополнительное задание

Задание

Напишите программу, которая запрашивает у пользователя радиус круга и выводит его площадь. Формула площади круга $S = \pi r^2$.

Самостоятельная деятельность учащегося

Задание 1

Напишите программу, которая спрашивает у пользователя два слова и выводит их разделёнными запятой.

Задание 2

Напишите программу, которая запрашивает три целых числа a , b и x и выводит `True`, если x лежит между a и b , иначе – `False`.

Задание 3

Напишите программу, которая решает квадратное уравнение $ax^2 + bx + c = 0$ по формулам $x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$. Значения a , b и c вводятся с клавиатуры. Для извлечения корня используйте оператор возведения в степень, а не функцию `math.sqrt`, чтобы получить комплексные числа в случае, если подкоренное выражение отрицательно.

Рекомендуемые ресурсы

Документация по Python о стандартных типах данных и операциях с ними

<https://docs.python.org/3/reference/datamodel.html#the-standard-type-hierarchy>
<https://docs.python.org/3/library/stdtypes.html>

Документация Python о операциях над строками и их форматировании

<https://docs.python.org/3/library/stdtypes.html#text-sequence-type-str>
<https://docs.python.org/3/library/stdtypes.html#printf-style-string-formatting>
<https://docs.python.org/3/library/stdtypes.html#str.format>
<https://docs.python.org/3/library/string.html#formatstrings>

Статьи в Википедии о ключевых понятиях, рассмотренных на этом уроке

[https://ru.wikipedia.org/wiki/Переменная_\(программирование\)](https://ru.wikipedia.org/wiki/Переменная_(программирование))
[https://ru.wikipedia.org/wiki/Константа_\(программирование\)](https://ru.wikipedia.org/wiki/Константа_(программирование))
https://ru.wikipedia.org/wiki/Тип_данных
https://ru.wikipedia.org/wiki/Статическая_типизация
https://ru.wikipedia.org/wiki/Динамическая_типизация
https://ru.wikipedia.org/wiki/Сильная_и_слабая_типизация
https://ru.wikipedia.org/wiki/Целое_число
https://ru.wikipedia.org/wiki/Вещественное_число
https://ru.wikipedia.org/wiki/Комплексное_число
https://ru.wikipedia.org/wiki/Логический_тип