Урок 1.

**Введение. Инструменты ввода и вывода.**

короткая линия

# План урока

1. Основные понятия
2. Язык Python
3. Вывод данных
4. Переменные
5. Ввод данных

# Основные понятия

Добро пожаловать на курс “Введение в Python”! На этом курсе мы разберем основные принципы и инструменты мощного и универсального языка Python. Прежде чем мы приступим к такому увлекательному занятию как программирование, стоит начать с некоторых основных теоретических понятий.

Компьютер понимает понятие “программа” как последовательный список команд. Для того чтобы облегчить написание программ и делегирование их компьютеру, люди придумали языки программирования. Названия некоторых из них вы, возможно, уже слышали: Pascal, Delphi, Java, C/C++ и т. д. Со времени создания первых программируемых машин человечество придумало более восьми тысяч языков программирования, и каждый год их число увеличивается. Небольшую часть их них вы можете просмотреть по ссылке:

<https://bit.ly/1bX9U05>

Также существует способ просмотреть наиболее популярные на текущий момент языки программирования с помощью индекса TIOBE, который был выведен из степени обсуждаемости языки в сети:

<https://www.tiobe.com/tiobe-index/>

Как мы видим, рассматриваемый нами язык Python на данный момент находится на 3-й строчке рейтинга, а в 2007 и 2010 годах и вовсе возглавлял рейтинг.

# Язык Python

Настало время рассмотреть причины, по которым Python представляет особый интерес для программистов по всему миру и для нас, в частности. Итак, Python — отличный язык, потому что:

* он обладает лаконичным и понятным синтаксисом
* он широко применим; основные области применения: веб-разработка, машинное обучение, работа с данными, математические и прочие научные вычисления
* он экономит время программиста, позволяя не затрачивать много времени на написание кода
* он довольно прост для обучения (как говорится, имеет низкий порог вхождения) и позволяет сходу погрузиться в мир программирования.
* он популярен. Это скорее вывод из остальных преимуществ, но данный факт позволяет быстро найти нужную книгу или туториал по любой теме, которую вы хотите изучить.

Справедливо будет сказать и о его минусах. Основной недостаток, который стоит иметь в виду — это то, что Python проигрывает в скорости таким мощным языкам, как Java или C++ — за краткость кода приходится платить производительностью.

Так как же писать программы на Python? Собственно сами команды можно писать в любом текстовом редакторе (например, в Блокноте), но чаще всего отдают предпочтение интегрированным средам разработки (IDE). Мы, например, будем использовать PyCharm:

<https://www.jetbrains.com/pycharm/download/#section=windows> (необходимо выбрать Community версию).

Однако, написав программу, мы не сможем ее запустить без интерпретатора, который создан специально чтобы обрабатывать и анализировать написанные вами команды и передавать их компьютеру. Интерпретатор можно скачать с официального сайта (на момент написания урока последняя версия интерпретатора — 3.6.5):

<https://www.python.org/downloads/>

# Вывод данных

После того как обе программы установлены — интерпретатор и среда разработки — мы можем начать программировать.

Попробуем написать нашу первую программу. Создадим новый файл в PyCharm и напишем там

print(**"Hello world"**)

Все, на этом написание программы закончено. Вы только что написали программу, которую ежедневно пишут тысячи начинающих программистов по всему миру.

Если вы знаете синтаксис языка Pascal, вы можете заметить, что написанная программа сильно отличается от типичной программы на Паскале: она короче и проста для понимания, а также в конце строки отсутствует символ точки с запятой.

Нетрудно догадаться, что делает данный код: вызывается команда print(), назначение которой — вывод строки на экран. print() принимает в круглые скобки любую строку, заключенную в двойные или одинарные кавычки (например, команда print(**'Hello world'**) делает абсолютно то же самое).

Чтобы запустить программу в PyCharm, нажмите правой кнопкой мыши и выберите опцию Run, после чего вы увидите вывод программы в консоли.

Попробуйте передать на вход другую строку, в том числе на русском языке. Попробуйте не ставить закрывающую скобку или не ставить кавычки. Что произойдет?

После того как вы научились корректно выводить любые строки, вы можете сдать свое первое задание по программированию:

**Задача “Привет, мир! 2.0”.**

Команда print() также может принимать на вход несколько аргументов, в таком случае строки будут выведены через пробел. Например, так:

print("Hello", "world")

# Переменные

Теперь допустим, что нам было дано задание напечатать строку “I am learning Python” 5 раз. Первое решение, которое может прийти в голову, — напечатать команду print(), которую мы выучили ранее, скопировать ее и вставить в документ еще 4 раза. Однако это не лучший подход. Предположим, что заказчик принял нашу работу, но прислал нам правки: изменить строку на “I know Python!”. В таком случае нам придется изменять строку в каждой инструкции print(), что займет некоторое время, особенно если печатать нужно намного больше, чем 5 раз.

В таком случае пригодятся переменные. Наиболее часто используемая аналогия для объяснения переменных — это некая коробочка, в которой может хранится одно значение.



На картинке изображена коробка с надписью Message и содержимым “Hello”. Применяя данную аналогию к языку Python, мы получим переменную message, хранящую строковое значение “Hello”. Создание переменной выглядит так:

message = **"Hello"**

Слева от знака “равно” указывается имя переменной, а справа — ее значение. Существуют определенные правила для имен переменных — их можно посмотреть по ссылке:

<http://pythonlearn.ru/python-dlya-nachinayushhix/imena-dlya-peremennyx-v-python/>

Также по ссылке указано очень важное правило, на которое стоит обратить внимание: переменные нужно называть так, чтобы ее имя отражало суть содержащегося в ней значения. Это универсальное правило, применяющееся во всех языках программирования.

Итак, чтобы исполнить волю заказчика, поместим в переменную text значение “I know Python!”:

|  |
| --- |
| text = **"I know Python!"** |

Теперь мы можем определить переменную лишь однажды и передавать ее в качестве параметра для print():

|  |
| --- |
| text = **"I know Python!"**  print(text)  print(text)  print(text)  print(text)  print(text) |

Как видно, передавая переменную, нам не нужно использовать кавычки.

Наш заказчик наконец доволен, но и это не лучший способ выполнить задание. Если бы от нас потребовали написать ту же строчку 100 раз, было бы неудобно и долго вставлять один и тот же print() вручную. Время программиста очень ценно, поэтому мы можем выполнить это задание в одну строчку, в лучших традициях языка Python:

|  |
| --- |
| print(**"I know Python!\n"** \* 100) |

Этот пример содержит концепты, которые будут изучаться только в следующих уроках, он нужен лишь для того, чтобы показать, что зачастую самое эффективное решение занимает мало места и не требует сотен строчек однообразного кода. Несмотря на это, вы можете самостоятельно ознакомиться с тем, как арифметические операторы могут применяться к строкам в Python, и при желании поэкспериментировать с этим.

**Задача “Мощь переменных”**

# 

# Ввод данных

Предположим, что заказчику понравилась наша работа со строками, и он решил назначить нам новое задание. Он хочет, чтобы программа просила ввести имя пользователя с клавиатуры, а затем приветствовала его.

Это легко сделать с помощью команды input(), что буквально можно перевести как “ввод”. Когда программа доходит до этой инструкции, она останавливается и ожидает ввод с клавиатуры. Чтобы сохранить написанное для дальнейшего использования, мы можем использовать переменные.

text = input()

Теперь любой текст, который мы введем с клавиатуры, будет храниться в переменной text, после того как мы нажмем Enter.

Попробуйте самостоятельно завершить задание для заказчика:

**Задача “Обращение по имени”**

**Домашнее задание:**

**Творческая задача “Анкета”**

**Задача “Любимая книга”**

**Задача “Гадание по имени”**