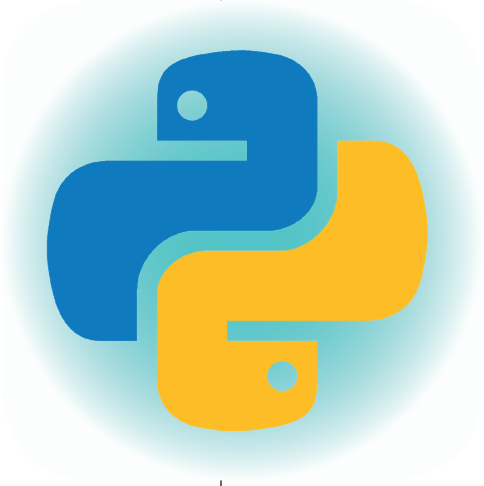
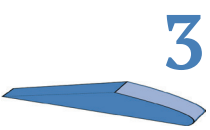
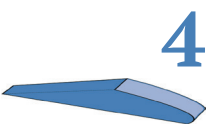
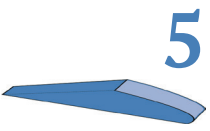
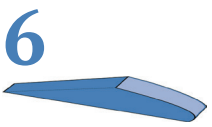


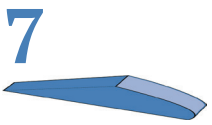
Содержание

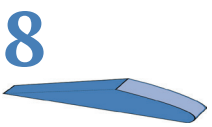


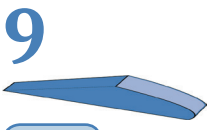


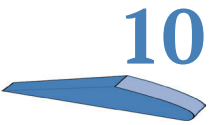


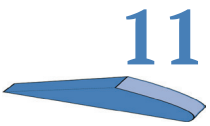


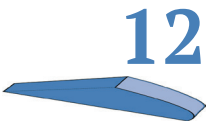


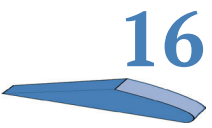


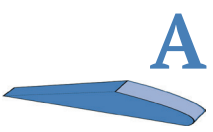


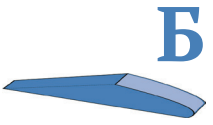


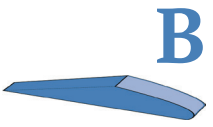


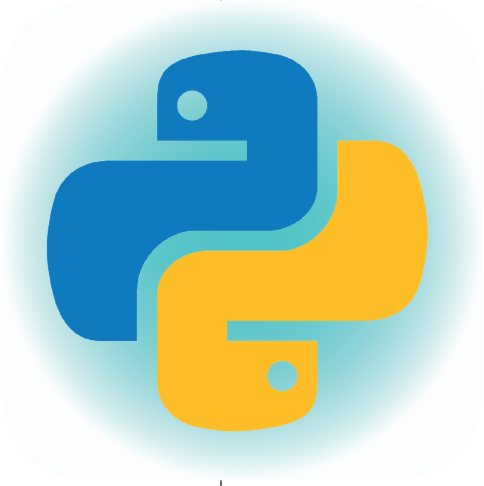












В своё время, для того чтобы изучить основы Python, я потратил очень много времени на поиск необходимой информации в интернете. Книг тогда было достаточно, но они были перегружены огромным количеством информации, которая только путала новичка.

Эта книга представляет собой руководство средней сложности для изучения языка программирования Python.

Она очень хорошо подойдёт для детей, изучающих Python с наставником, и для самостоятельного освоения данного языка, а также для составления методических пособий в организациях дополнительного образования.

Дмитрий Филиппов,
преподаватель курса по Python
в международной школе программирования
для детей «CODDY»

Введение

Python – это *язык программирования*, который был разработан в начале 1990-х годов. Название этого языка программирования произошло от имени английской комедийной труппы под названием «Монти Пайтон», которая в 70-е годы была довольно популярна и принимала участие в кинокомедиях, таких как «Жизнь Брайана».

Хотя Python во многом похож на другие языки программирования, он стал очень популярен, так как именно на этом языке довольно легко научиться программировать. Эта книга посвящена третьей версии языка программирования Python, на сегодня последней (и универсальной).

Что такое программирование?

Программирование – это когда ты записываешь то, что должен делать компьютер. Самое замечательное, что ты сам можешь решить, что конкретно ему нужно делать. Если ты запустишь свою программу, компьютер выполнит то, что ты только что придумал и записал.

Конечно, твой компьютер не сможет убрать твою комнату и не принесет тебе чашечку какао в постель. Но как только ты освоишь программирование, то сразу же сможешь позволить компьютеру, так сказать, танцевать под свою дудку.

Однако во время программирования часто бывает так, что компьютер не хочет выполнять то, что хочешь от него ты. Обычно такое поведение вызывает ошибка в программе. Проблема также может быть где-то еще в компьютере или

операционной системе. Проблема ошибок заключается в том, что им очень уж нравится прятаться, так что их поиск заставляет некоторых программистов попросту отчаиваться.

Я очень надеюсь, что тебе все-таки захочется научиться программировать. Тогда все, что тебе нужно, – это подходящая среда разработки, и ты уже готов к работе!

Что такое среда разработки?

Чтобы создать программу, сначала нужно что-то куда-то ввести. Это как письмо или рассказ, который ты пишешь. Текстовая программа для этого может быть очень простой, поскольку не нужно выбирать разные шрифты или вставлять картинки. Такие программы принято называть *текстовыми редакторами*.

Если программа набрана в виде текста в редакторе, то компьютер не сможет просто прочитать этот документ и выполнить программу. Текстовая версия (*код*) программы должна быть переведена так, чтобы компьютер понял, что от него требуется в конечном итоге. Но поскольку он говорит на совершенно ином языке, чем ты или твои друзья, то придется использовать специальный переводчик.

Ты программируешь на понятном тебе языке, а специальная программа переводит то, что ты написал, в команды, понятные для компьютера. Такую программу называют *компилятором* или *интерпретатором*.

Для языка Python существуют интерпретаторы для нескольких операционных систем. Твой компьютер может работать под управлением операционной системы Windows, Linux или macOS. Независимо от этого одна и та же программа, написанная на языке программирования Python, будет работать (возможно, с небольшими изменениями) на любом компьютере.

Наконец, программы должны быть проверены, пересмотрены, усовершенствованы, доработаны и собраны. А еще есть вспомогательные инструменты. Все это становится целой системой – *средой разработки*.

Почему Python?

К сожалению, ты не сможешь программировать так, как захочется. Язык программирования должен быть структурирован таким образом, чтобы как можно больше людей могли понять его по всему миру.

В мире большинство людей может, по крайней мере, прочитать несколько слов по-английски, и поэтому почти каждый язык программирования состоит из английских слов. Есть также проекты языков программирования на других языках, но в большинстве случаев команды звучат настолько непривычно, что ты, скорее всего, все же вернешься к английскому.

На самом деле не имеет значения, какой конкретно язык программирования ты используешь. Конечно, лучше всего тот, которому легко научиться. Как ты уже понял, в этой книге ты научишься создавать программы с помощью языка программирования Python, который сейчас очень популярен. (Если ты захочешь познакомиться с другими языками, я рекомендую обратить внимание на языки программирования C++ и Java.)

Стать хорошим программистом может быть очень трудно. Нередко человек теряет желание, потому что у него просто не работают написанные им программы. Программа делает нечто совершенно другое, а он сам не может найти ошибку и спрашивает себя: зачем я учусь программировать, если в мире уже и так достаточно программ?

Хорошие программисты всегда легко находят работу, а потребность в качественном программном обеспечении будет только расти. Программисты на языке Python, безусловно, востребованы в трудовой сфере. И, честно сказать, хорошие программисты действительно получают хорошую заработную плату. Это не просто твоя первая попытка, ведь, может быть, тебе действительно станет интересно узнать, как программировать на Python.

Среда разработки

Тебе не нужно переживать о том, где взять среду разработки на языке Python, так как это достаточно просто. Ты получишь ее бесплатно в составе пакета Python (называемого IDLE (Integrated Development and Learning Environment) – *интегрированная среда разработки и обучения на языке Python*). Мы будем пользоваться ею в этой книге постоянно.

Скачать полный пакет ты сможешь на сайте www.python.org.

Необязательно устанавливать самую последнюю версию. В этой книге мы будем работать с версией Python 3.1.

О чем эта книга?

В этой книге ты:

- получишь базовые знания о языке Python;

- узнаешь об объектно-ориентированном программировании;
- научишься работать с компонентами библиотеки `tkinter` (это строительные блоки, с помощью которых ты сэкономишь время в процессе программирования);
- узнаешь о возможностях Python при работе с графикой;
- обучишься основам работы с игровыми модулями `pygame`;
- узнаешь, как разработать собственную игру.

В приложениях содержится дополнительная информация и справочные сведения, которые помогут разобраться в настройках и способах устранения ошибок.

Как читать эту книгу

Эта книга состоит из большого количества текста с иллюстрациями. И я, конечно, старался написать ее так, чтобы она была понятной. Чтобы сделать информацию в ней еще более понятной для тебя, я добавил некоторые символы. Вот что они обозначают:

Практические шаги

- Если ты видишь такой символ, знай: тебе нужно выполнить задание в данном абзаце текста. Это приблизит тебя к новой цели в программировании.

В целом ты будешь быстрее усваивать материал, когда будешь вводить код программы или станешь изменять его самостоятельно. Но у тебя не всегда будет желание этим заниматься. Потому все примеры данной книги доступны для бесплатного скачивания по адресу dmkpress.com.

Скачав файлы примеров, ты сможешь найти проект по имени (например, ⇒ *project1.py*). Поэтому если ты не захочешь самостоятельно создавать проект, то всегда сможешь открыть соответствующий файл.

Задачи

В конце каждой главы ты найдешь несколько вопросов и задач. Эти упражнения не всегда просты, но они помогают освоить программирование. Решения задач можно найти в папке с файлами примеров. Ты сможешь просмотреть их в текстовом редакторе. Или распечатать на принтере и сложить стопкой рядом с компьютером.

Решение проблем

Случается, что ты не знаешь, как что-то сделать, или, возможно, просто что-то забыл. И тогда вся работа кажется невыполнимой и сложной. И ты спрашиваешь себя: что же мне делать? С помощью этого значка ты сможешь довольно легко найти решение или восстановить что-то в памяти. При необходимости также можешь заглянуть в последнюю часть этой книги, там расположено приложение Б, в котором приведены советы и другая справочная информация.

Важные примечания

Время от времени ты будешь встречать в книге такой восклицательный знак. Он указывает на текст с очень важной информацией.

Экспертное мнение

Если ты видишь такой значок, «Bay!», этот текст содержит дополнительную информацию по теме.

Что тебе нужно для этой книги

Среда разработки Python устанавливается в каталог на твой выбор, например в папку *C:\Python*. Там ты также сможешь разместить свои проекты Python, только чуть позже. Примеры программ в этой книге доступны для загрузки с главной страницы сайта издательства русского издания этой книги dmkpress.com.

Там же ты найдешь ответы на вопросы и решения задач (все они находятся в папке с примерами).

Операционная система

Большинство компьютеров сегодня работает под управлением операционной системы Windows. Для работы с примерами из этой книги тебе понадобится версия Windows 7 или 10. (Среда Python также доступна и для macOS и Linux.)

Носители информации

Тебе понадобится USB-накопитель (т. н. «флешка») или SD-карта, если ты захочешь сохранить свои программы на диск. На внешнем хранилище твоя работа всегда будет в безопас-

ности. При необходимости попроси своих родителей или преподавателя купить тебе такой носитель информации.

Примечания для преподавателей

Эта книга также может использоваться как учебный материал на уроках информатики в школе. Конечно, каждый учитель устанавливает свои собственные приоритеты в обучении детей программированию. Если вы уже используете другой учебник в своей работе, то сможете использовать это издание в качестве источника информации в дополнение к существующему учебнику. Эта книга начинает обучение «с нуля», так сказать, является прямым вхождением в язык программирования Python, без необходимости наличия каких-либо навыков программирования.

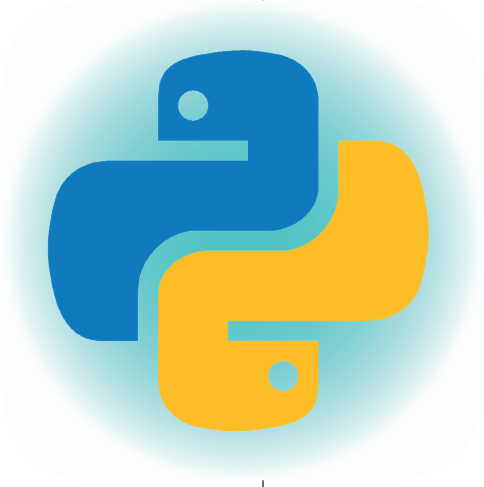
Важным направлением в этой книге является объектно-ориентированное программирование (ООП). Наиболее важные концепции (инкапсуляция, наследование и полиморфизм) обсуждаются в книге достаточно подробно. Еще одним направлением является игровое программирование. В проектах используются все основные элементы словаря Python, а также наиболее важные графические компоненты библиотеки `tkinter`. Вы встретите большое количество задач по программированию.

Хранение файлов проектов

На уроках информатики каждому ученику необходимо иметь собственный внешний носитель для хранения всех заданий и собственноручно написанных программ. Таким образом, жесткому диску школьного компьютера не придется накапливать горы лишнего «мусора». Кроме того, собственный носитель данных служит для их защиты: ученик не потеряет свои файлы и с удобством сможет ими управлять.

Постоянное сохранение прогресса

Отличная идея – сохранять файлы программы в процессе работы примерно каждые десять минут. Все знают, что компьютеры любят «выходить из строя» именно тогда, когда файл не был сохранен.



1 Твои первые шаги в программировании

В этой главе разговор пойдет о том, как после установки и запуска Python сделать свои первые шаги. Я расскажу тебе, как настроить рабочее окружение так, чтобы твоя первая, написанная на языке программирования Python программа появилась на свет.

В этой главе ты узнаешь:

- как запустить Python;
- как использовать инструкции для вывода и ввода данных;
- что такое переменные;
- что такое строковый тип данных;
- как использовать среду разработки IDLE;
- как создавать и сохранять программы;
- как завершать работу с Python.

Начало работы с Python

Прежде чем ты начнешь программировать, давай настроим Python.

Настройка предполагает запуск установленной программы. Подробные сведения об установке ищи в приложении А. Тебе понадобится помощь взрослых, если ты не знаешь, как работать с такой программой самостоятельно. Один из способов запуска Python такой:

- Открой папку, в которую ты установил Python, например *C:\Program Files\Python* или *C:\Python* (рис. 1.1).

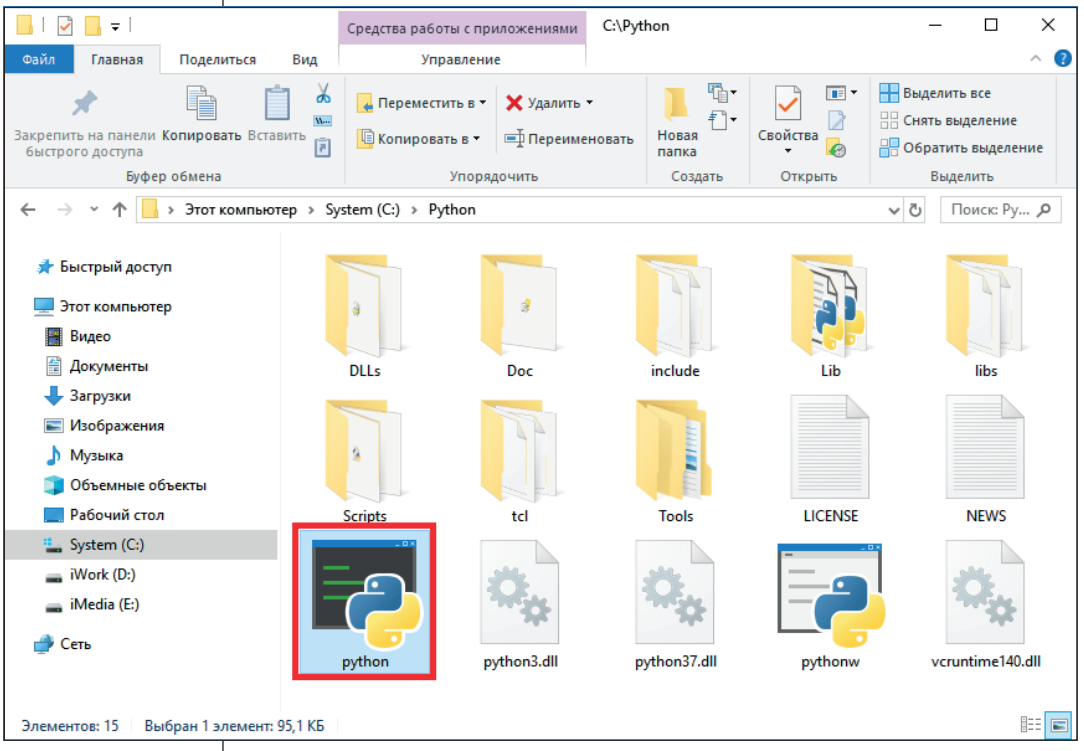


Рис. 1.1. *Содержимое папки с установленным Python*

- Найди среди значков файл с именем *python.exe*. Дважды щелкни мышью по значку с этим именем.

Для удобства ты можешь создать ярлык на рабочем столе:

- щелкни правой кнопкой мыши по значку *python.exe*;
- в контекстном меню выбери пункт **Отправить ⇒ Рабочий стол (Создать ярлык)** (Send To ⇒ Desktop (Create shortcut));
- не обязательно ставить стандартное имя для ярлыка на файл *python.exe*. Ты можешь использовать любое подходящее слово.

Теперь можешь дважды щелкнуть мышью по новому значку и запустить Python.

Что же произойдет после запуска? Появится окно оболочки командной строки, показанное на рис. 1.2.

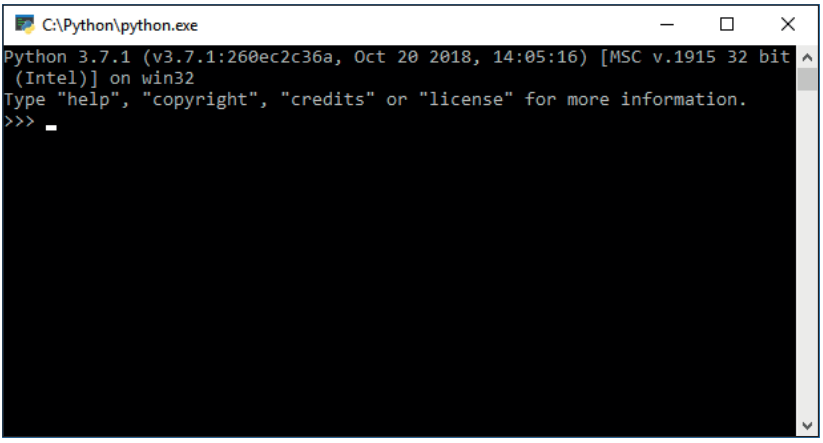


Рис. 1.2. *Запущенный Python*

Первая строка сообщает о текущей версии Python, а ниже указано несколько справочных команд и три угловые скобки (>>>).

Эти три символа называются «приглашением». Это действительно своего рода приглашение, потому что ты сможешь ввести команду после них (и тебе придется это сделать, если ты хочешь продолжить работу).

- Давай попробуем поработать с Python с помощью команды `help`. Введи это слово. Обрати внимание, что помимо самого слова тебе нужно набрать две круглые скобки следом.
- Введи команду `help()` и нажми клавишу **Enter**.

На экране появится много текста (рис. 1.3).

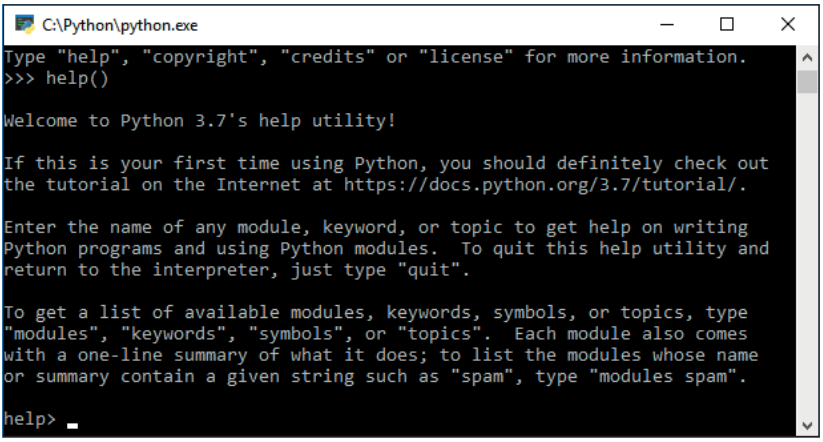


Рис. 1.3. *Результат выполнения команды* `help()`

Теперь в оболочке командной строки указано слово `help>`, которое служит подсказкой. Ты можешь указать следом слово (на английском языке), и если оно есть в словаре Python, ты увидишь небольшое объяснение.

- Чтобы вернуться к приглашению, введи команду `quit()` (рис. 1.4).

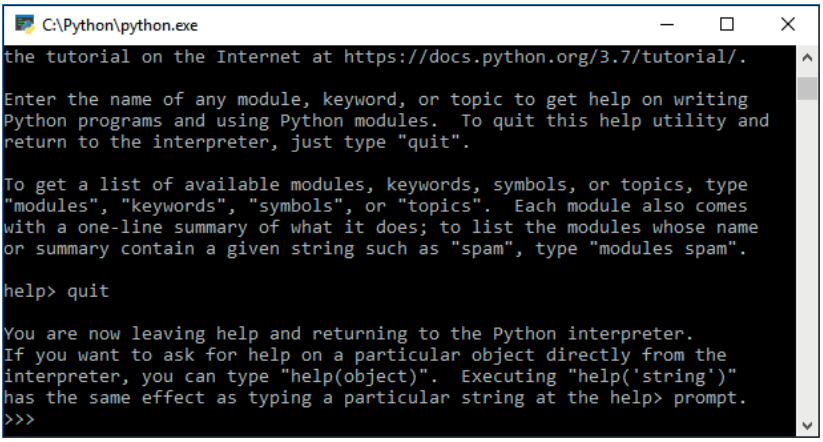


Рис. 1.4. *Ввод команды* `quit()`

И вот мы снова находимся в ***интерпретаторе*** языка Python.

Что же такое *интерпретатор* (другими словами, переводчик)? Во-первых, ты должен знать, что то, что ты вводишь в качестве команды, совершенно непонятно для компьютера. Как правило, он не может выполнить такую команду.

Интерпретатор *переводит* командную строку на *машинный* язык, который компьютер понимает, чтобы он мог выполнить данную тобой команду. Для программы, которая может состоять из многих-многих строк, интерпретатор переводит каждую строку по очереди и выполняет ее.

Кроме того, существуют *компиляторы*, которые переводят всю программу на машинный язык. Программа выполняется только после того, как скомпилирована и проверена, так что без ошибок может быть выполнена компьютером. В этой книге мы используем интерпретатор, но есть также и компиляторы для языка Python.

Числа и текст

Теперь попробуем что-нибудь ввести.

- Введи: `1+2+3`, а затем нажми клавишу **Enter** (рис. 1.5).

Рис. 1.5. *Выполнение операции сложения*

Результат решения этой маленькой математической задачи появился на экране.

- Выполни еще несколько задач, используя операции вычитания (-), умножения (*) и деления (/). Как калькулятор интерпретатор Python работает отлично, но мы же с тобой хотим гораздо большего, не так ли?
- Давай попробуем небольшое приветствие: напечатай `Привет` и нажми клавишу ↵ **Enter** (рис. 1.6).

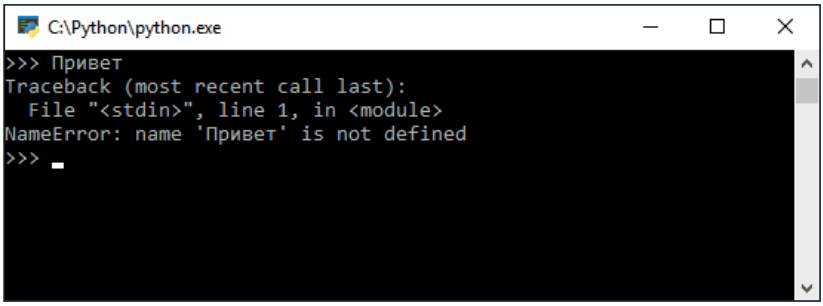


Рис. 1.6. *Ввод слова привел к ошибке*

Что случилось? Тут явно что-то не работает. А я хотел, чтобы компьютер сказал мне (т. е. написал) дружеское «Привет!».

- Введи строку `print("Привет")` и нажми клавишу **Enter** (рис. 1.7):

Рис. 1.7. *Отображение приветствия*

Работает! Здесь имя команды `print()` означает отображение текста, вывод на экран. В круглых скобках после слова `print` ты вводишь именно то, что хочешь отобразить на экране. Это называется *параметром*.

Конечно, такое же возможно и с числами (рис. 1.8):

Рис. 1.8. *Отображение чисел*

➢ Поэкспериментируй с командой `print()`.

Теперь задание будет немного сложнее. До сих пор мы вводили только одну команду. Но все программы, конечно, состоят из нескольких строк. Попробуем ввести небольшую программу:

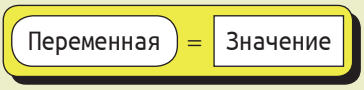
```
Text = "Привет"
print(Text)
```

➢ Введи эти две строки (рис. 1.9). Каков результат?

Рис. 1.9. *Небольшая программа-приветствие*

После первой строки ничего не выводится. Но, по-видимому, интерпретатор Python понимает, что означает `Text`. И он знает, какое значение в команде `print()` следует использовать в качестве параметра.

Чуть подробнее: `Text` – это так называемая *переменная*, которой *присваивается значение*, в нашем случае это слово "Привет". А знак равенства (=) называется *оператором присваивания*.



В Python переменные создаются при первом присвоении им значений. При присвоении левая часть всегда является именем переменной, а правая – значением, поэтому знак равенства можно изобразить как стрелку:

Переменная ← Значение

Переменные полезны, потому что они сохраняют данные, чтобы компьютер мог запомнить значение. Для сложных программ важно, чтобы содержимое переменной можно было использовать несколько раз. Далее я покажу несколько примеров.

Раз уж компьютер с нами так хорошо поздоровался, давай усложним нашу программу. Введи:

```
Text = "Привет, кто ты?"
print(Text)
Name = input()
print(Name)
```

При программировании на языке Python текст, подобный приветствию, всегда должен быть указан в кавычках. В примере я использую двойные кавычки (""), но также допускаются и одинарные (' ').

Поэтому следующие две строчки будут полностью эквивалентны:

```
Text = "Привет, кто ты?"
Text = 'Привет, кто ты?'
```

- Введи все эти строки по очереди. Обрати внимание, что в процессе тебе придется ввести свое имя.

Вот так программа выглядит в моем случае (рис. 1.10):

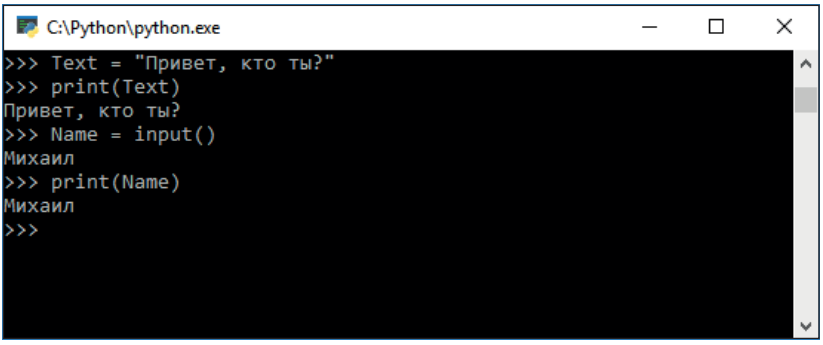


Рис. 1.10. *Программа с запросом имени*

Не так уж плохо! Заодно ты познакомился с новой командой `input()` – она означает ввод текста с клавиатуры. Отличная тренировка – напиши несколько вопросов `print()` и запроси ответы на них с помощью команды `input()` – и у тебя получится интересный чат с компьютером.

Но что-то здесь мне не очень нравится. Например, то, что приходится постоянно снова набирать строки кода про-

граммы. И не получается просто «прогуляться» по программе с помощью клавиш со стрелками. И также не получается щелкнуть в любом месте мышью и изменить там текст. Было бы лучше, если бы мы могли перемещаться в окне Python так же, как в текстовом редакторе.

Ужасно неудобно, если мы хотим создавать более крупные программные проекты! Для этого также должна быть возможность сохранить этот текст в виде файла. Одной этой программы, которая у нас есть, очевидно, недостаточно.

Рабочая среда IDLE

Поэтому нам нужен интерфейс, с помощью которого можно сохранить введенный текст и менять его при необходимости. Такая программа называется *редактором*. В дистрибутиве Python такой редактор присутствует, тебе просто нужно его найти. В меню **Пуск** (Start) операционной системы Windows ты найдешь ярлык **IDLE** (рис. 1.11).

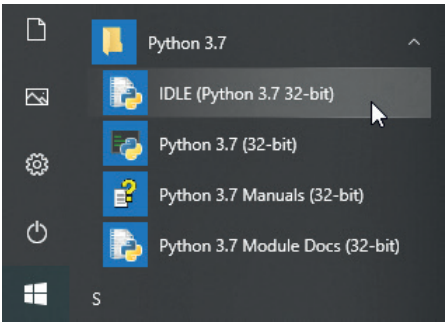
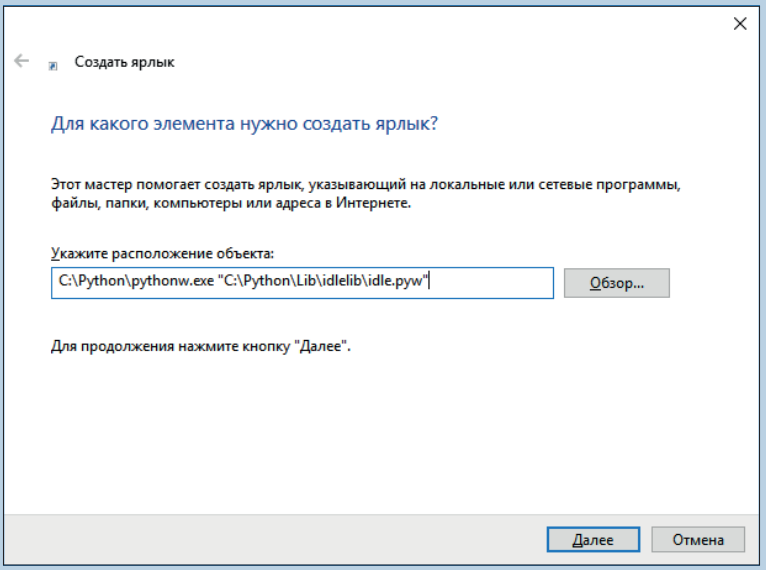


Рис. 1.11. *Выбор пункта* ***IDLE*** *в меню* ***Пуск***

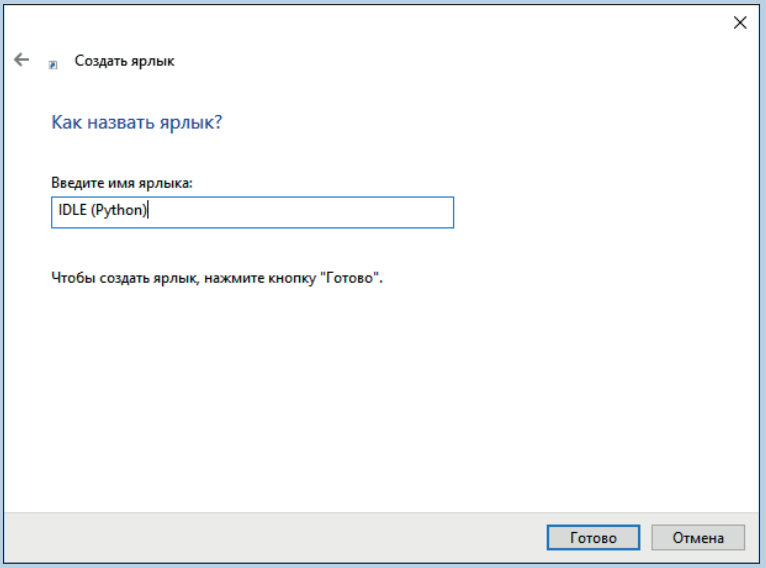
➢ Выбери значок с надписью **IDLE**, чтобы запустить эту программу.

Если у тебя нет этого значка в меню **Пуск** (Start), тогда создай ярлык на рабочем столе самостоятельно.

- ❖ Щелкни правой кнопкой мыши на пустом месте рабочего стола и выбери пункт **Создать** ⇒ **Ярлык** (New ⇒ Shortcut).
- ❖ В появившемся диалоговом окне укажи строку **C:\Python\pythonw.exe «C:\Python\Lib\idlelib\idle.pyw»**.



- Важно, чтобы вместо *C:\Python* ты указал ту папку, в которую установил Python. В противном случае ярлык не будет работать.
- Нажми кнопку **Далее** (Next).
- Присвой новому ярлыку на рабочем столе имя **IDLE (Python)**, как показано на рисунке ниже.



- Нажми кнопку **Готово** (Finish).

С этого момента ты сможешь запускать редактор Python прямо с рабочего стола, дважды щелкнув по ярлыку **IDLE (Python)**.

После запуска программы IDLE ты увидишь окно, показанное на рис. 1.12.

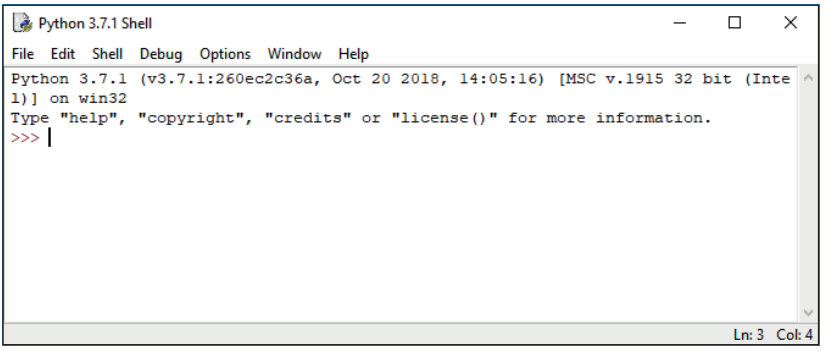


Рис. 1.12. *Интерфейс IDLE*

Сама программа напоминает и в то же время существенно отличается от предыдущего интерпретатора Python. Кроме того что вместо черного фона здесь мы будем работать в области белого цвета, тут также есть и строка меню. И теперь вся эта программа будет называться *оболочкой*.

IDLE – это аббревиатура названия Integrated Development and Learning Environment. То есть *интегрированная среда разработки и обучения*, которая необходима для программирования на языке Python.

Мы также можем вводить команды на языке Python в этой среде и получим те же результаты, что и в предыдущем окне (в «черном» интерфейсе Python) (рис. 1.13):

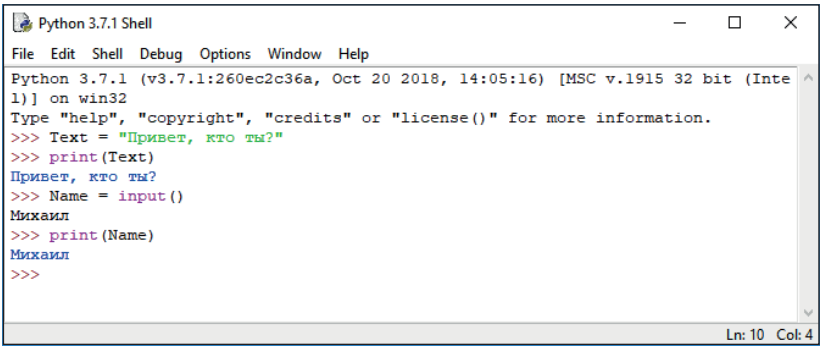


Рис. 1.13. *Ввод команд в IDLE*

Теперь среда разработки выглядит немного более красочной. Здесь команды наподобие `print()` и `input()`, а также вывод отображаются цветным шрифтом (это ведь удобно!).

- Попробуй сам, набрав строки примера в окне IDLE.

Работа с PY-файлами

Как же получается так, что всего лишь несколько строк становятся целой программой, которую можно загружать и выполнять снова и снова?

- Щелкни мышью по пункту **File** (Файл) в строке меню, а затем выбери пункт **New file** (Создать файл) (или нажми сочетание клавиш **Ctrl**+**N**) (рис. 1.14).

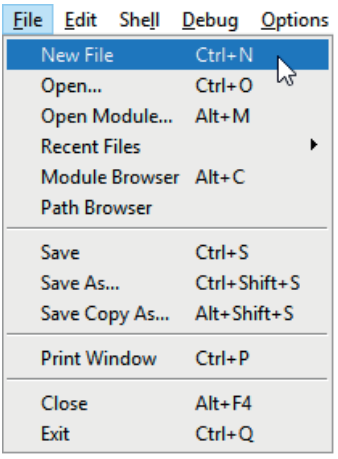


Рис. 1.14. *Создание файла*

Эта команда, только что выполненная тобой, создаст новый документ с именем *Untitled*. В этом окне тоже есть строка меню (рис. 1.15).

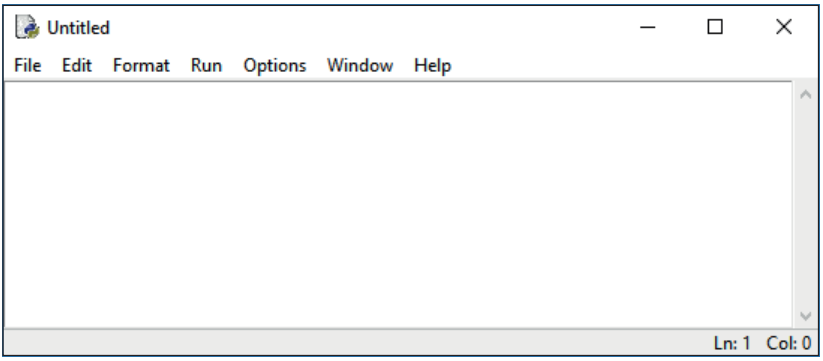


Рис. 1.15. *Новый файл*

Здесь мы сможем видеть нашу программу, не выполняя ни одной строки непосредственно в интерпретаторе Python.

- Введи следующие строки:

```
Text = "Привет! Кто ты?"
print(Text)
Name = input()
print(Name)
```

- Щелкни мышью по пункту **File** (Файл) в строке меню, а затем выбери пункт **Save as** (Сохранить как) (рис. 1.16).

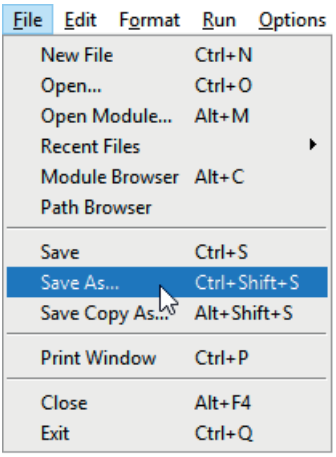


Рис. 1.16. *Команда сохранения файла*

- В диалоговом окне введи имя для этой программы, какое захочешь. Сама же программа автоматически добавит расширение *PY* как идентификатор программ для интерпретатора Python, если ты его сам не укажешь в имени (рис. 1.17). Нажми кнопку **Сохранить** (Save).

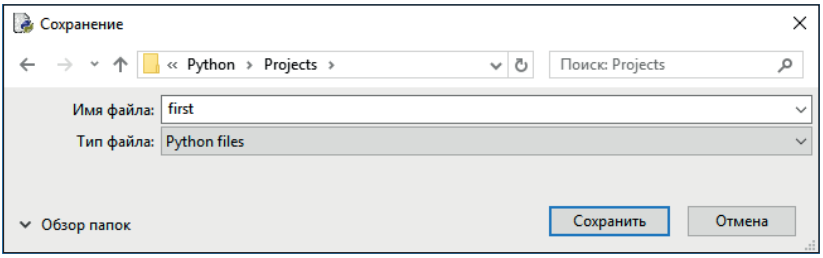


Рис. 1.17. *Сохранение файла*

Я создал подпапку под названием *Projects* в каталоге *Python* и сохранил в ней все свои проекты Python.

- Чтобы запустить программу, выбери команду меню **Run** ⇒ **Run Module** (Выполнение ⇒ Выполнить модуль). Или нажми клавишу **F5** (рис. 1.18).

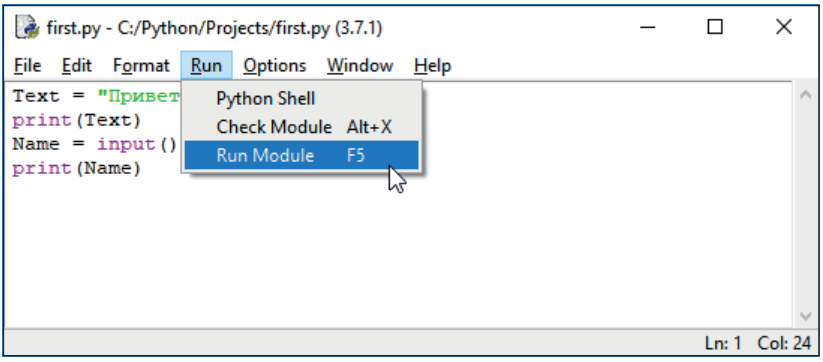


Рис. 1.18. *Запуск программы*

Ты вернешься к окну с приветствием «Привет! Кто ты?».

- Теперь введи свое имя и подтверди ввод с помощью клавиши **Enter**. Результат будет выглядеть так, как показано на рис. 1.19.

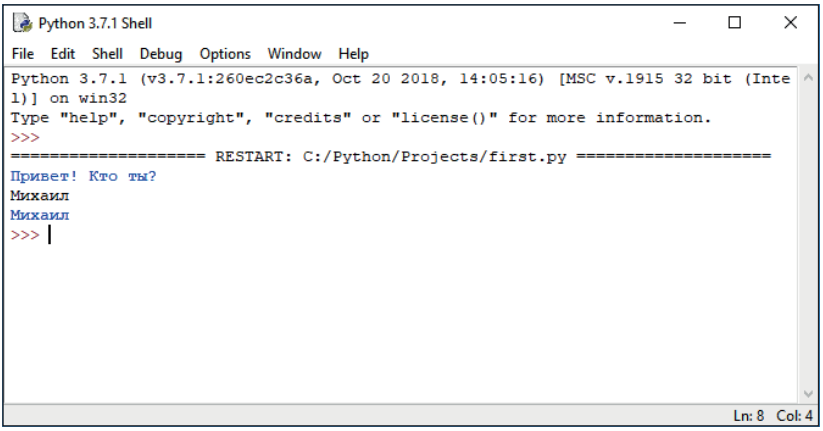


Рис. 1.19. *Запущенная программа*

Если сейчас ты закроешь это окно на своем компьютере, файл данной программы все равно будет сохранен. Тебе лишь нужно будет заново его открыть.

- Щелкни мышью по пункту **File** (Файл) в строке меню, а затем выбери пункт **Open** (Открыть) (или нажми сочетание клавиш **Ctrl**+**O**) (рис. 1.20).

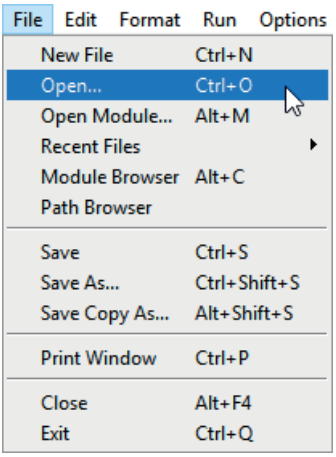


Рис. 1.20. *Команда открытия файла*

- В диалоговом окне выбери соответствующий файл (в папке *Python\Projects*) (рис. 1.21).

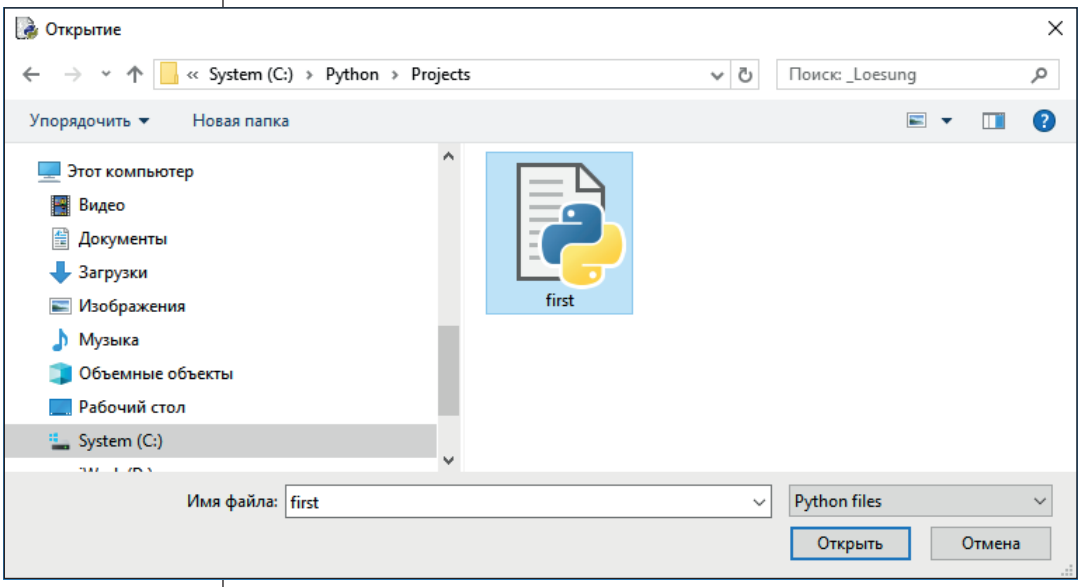


Рис. 1.21. *Открытие файла*

- Нажми кнопку **Открыть** (Open). Откроется окно редактора, и твоя первая программа будет готова к запуску.

В Python программу также называют *скриптом*.

Эксперименты с исходным кодом

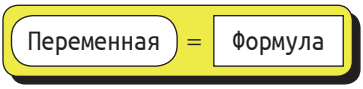
Теперь давай более подробно рассмотрим *исходный код*, так как именно так называется последовательность этих строк:

```
Text = "Привет! Кто ты?"
print(Text)
Name = input()
print(Name)
```

Здесь есть две переменные, `Text` и `Name`. В каждой из них сохраняется последовательность символов, называемая *строкой*. Кроме того, здесь используются две функции:

`print()`	Отвечает за вывод чисел и текста на экран
`input()`	Отвечает за ввод чисел и текста с помощью клавиатуры

Как видишь, функции всегда имеют круглые скобки, в которых что-то может быть написано, но они также могут быть и пустыми, в зависимости от функции и ее применения. И если ты посмотришь немного внимательнее на них, то наверняка заметишь, что `print()` выглядит как инструкция, а `input()` – как присваивание. Таким образом, значение присваивается переменной `Name` с помощью этой функции.



Я использую здесь термин «формула», потому что в правой части инструкции, помимо функций, можно использовать что-то вроде этого:

```
Amount = 1 + 2 * 3
Text = "Добрый вечер, " + Name
```

Обрати внимание, что символ плюс (+) играет двойную роль: позволяет складывать числа и конкатенировать (соединять) строки.

Теперь, когда ты знаешь, как написать несколько программных строк и сохранить их в файл, давай продолжим наш первый пример:

```
Text = "Привет! Кто ты?"
print(Text)
Name = input()
Text = "Так значит, ты - " + Name
print(Text)
print("А как у тебя дела?")
Answer = input()
print("Я рад, что у тебя дела " + Answer);
```

Как видишь, я не использую переменные на каждом шагу, на самом деле это необходимо, только если ты захочешь, чтобы компьютер что-то запомнил. В нашем примере это касается только данных, которые мы будем сами вводить. Соответственно, наша программа может выглядеть так:

```
print("Привет! Кто ты?")
Name = input()
print("Так значит, ты - " + Name)
print("А как у тебя дела?")
Answer = input()
print("Я рад, что у тебя дела " + Answer);
```

- Введи предыдущий исходный код и выбери команду меню **Run** ⇒ **Run Module** (Выполнение ⇒ Выполнить модуль) (или нажми клавишу **F5**). Затем измени код второй версией. Запусти программу снова.

Каждый раз, когда ты вносишь изменения, тебе будет предложено сохранить исходный код перед запуском программы (рис. 1.22):

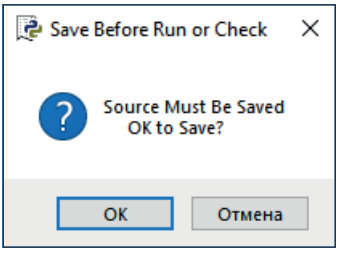


Рис. 1.22. *Диалоговое окно с предупреждением*

- Нажми кнопку **OK**.

Вот как будет выглядеть последняя версия нашей программы в окне интерпретатора Python (рис. 1.23):

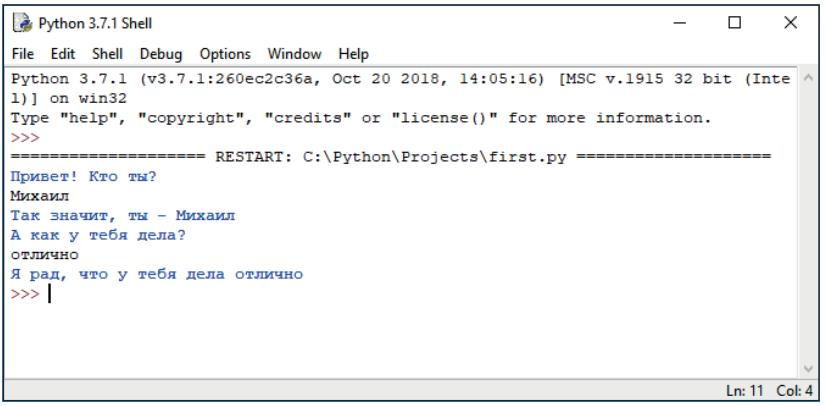
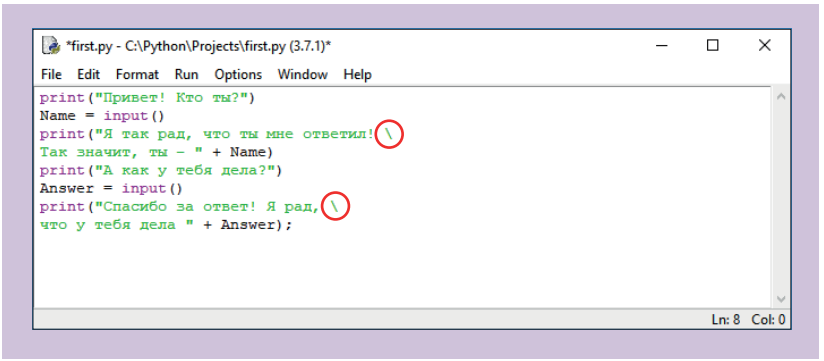


Рис. 1.23. *Результат выполнения программы*

Как видишь, интерпретатор Python читает код по строкам: в каждой строке есть оператор (даже знак равенства – это оператор присваивания). Поэтому ты не можешь просто перенести текст на две строки. Если это все же необходимо, в конце строки нужно указывать так называемый «обратный слеш» (\).



Выход из Python

Чтобы выйти из среды разработки Python, все открытые ранее окна должны быть, естественно, закрыты:

- Это можно сделать либо с помощью команды меню **File** ⇒ **Exit** (Файл ⇒ Выход), либо ты можешь щелкнуть по маленькому крестику (×) в правом верхнем углу каждого открытого окна. Или использовать сочетание клавиш **Ctrl**+**Q**.

Если ты ранее что-то изменил в коде и не сохранил файл, то увидишь окно, показанное на рис. 1.24.

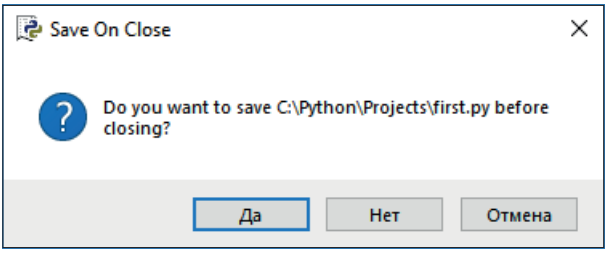


Рис. 1.24. *Предупреждение о необходимости сохранения программы*

- В зависимости от ситуации нажимай нужную кнопку. (Нажав кнопку **Отмена** (Cancel), ты вернешься обратно в окно редактора.)

Подведение итогов

С нашим первым проектом на языке программирования Python у тебя уже получилось кое-чего достичь. Давай-ка посмотрим, что ты узнал в этой главе. Прежде всего существует несколько команд, связанных с программой IDLE:

Запуск IDLE	Дважды щелкни мышью по ярлыку IDLE (Python)
Создать файл	Выбери команду меню **File** ⇒ **New File** (Файл ⇒ Создать файл)
Сохранить файл	Выбери команду меню **File** ⇒ **Save** (Файл ⇒ Сохранить)
Сохранить файл как...	Выбери команду меню **File** ⇒ **Save as** (Файл ⇒ Сохранить как)
Открыть файл	Выбери команду меню **File** ⇒ **Open** (Файл ⇒ Открыть)
Выполнить программу	Выбери команду меню **Run** ⇒ **Run Module** (Выполнение ⇒ Выполнить модуль)
Вызвать справочную систему	Выполни команду `help()` (используй команду `quit` для выхода)
Завершить работу Python	Выбери команду меню **File** ⇒ **Exit** (Файл ⇒ Выход)

А еще ты узнал кое-что из словаря языка Python:

`print()`	Функция, отвечающая за вывод чисел и текста на экран
`input()`	Функция, отвечающая за ввод чисел и текста с помощью клавиатуры
`=`	Оператор присваивания
`+`	Оператор сложения/конкатенации
`\`	Перенос строк (обратный слеш)

Несколько контрольных вопросов...

1. Какие функции отвечают за ввод и вывод информации?
2. Какая разница между компилятором и интерпретатором?
3. Инструкция присвоения – это то же самое, что и уравнение?

...а задач пока нет!