Елена Крылова

Python для детей, которые пока не программируют

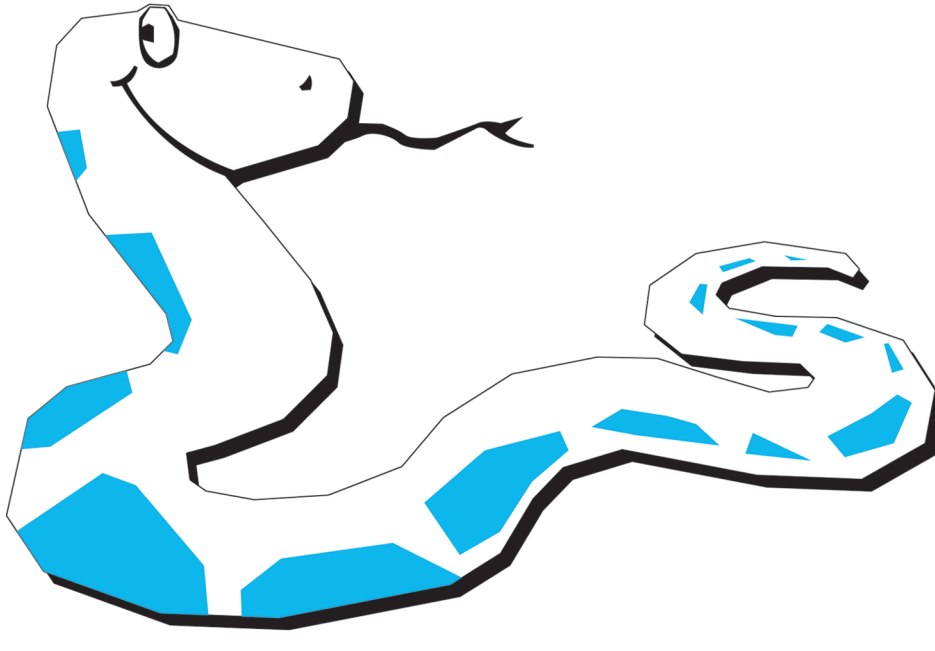
В задачах, экспериментах, проектах, диалогах, играх и сновидениях

+

Материалы на www.bhv.ru

Елена Крылова

Python для детей, которые пока не программируют



Санкт-Петербург
«БХВ-Петербург»
2023

УДК 004.43-053.2
ББК 32.973.26-018.1
К85

Крылова Е. Г.

К85 Python для детей, которые пока не программируют. — СПб.: БХВ-Петербург, 2023. — 208 с.: ил.

ISBN 978-5-9775-1182-7

Назначение книги — помочь ребёнку 10–13 лет сделать первые шаги в программировании, используя популярный язык Python, и получить удовольствие от этого процесса. Книга даст базовые навыки создания программ, поможет подготовиться к экзамену в IT-класс хорошей школы, станет первой ступенькой на пути к профессии программиста.

В каждой главе читатель-школьник сталкивается с проблемой, экспериментирует, выслушивает мнения экспертов, решает задачи и выполняет проекты, как простые, доступные каждому, так и повышенной трудности. Сюжеты задач и проектов реалистичные или фантастические, но всегда занимательные. На страницах встречаются неожиданные персонажи с собственным взглядом на программирование — всё это превращает овладение азами Python в увлекательную игру.

В книге есть ответы и подсказки к задачам и тестам, а в электронном архиве, доступном на сайте издательства, — рабочие материалы, тексты программ, наборы тестовых значений.

УДК 004.43-053.2
ББК 32.973.26-018.1

Группа подготовки издания:

Руководитель проекта	*Евгений Рыбаков*
Зав. редакцией	*Людмила Гауль*
Иллюстрации	*Елены Крыловой*
Компьютерная верстка	*Людмилы Гауль*
Дизайн обложки	*Зои Канторович*

Подписано в печать 04.07.22.
Формат 70×100$^{1}/_{16}$. Печать офсетная. Усл. печ. л. 16,77.
Тираж 1500 экз. Заказ №
«БХВ-Петербург», 191036, Санкт-Петербург, Гончарная ул., 20.

Отпечатано в типографии филиала
АО «ТАТМЕДИА» «ПИК «Идел-Пресс».
420066, Россия, г. Казань, ул. Декабристов, 2.
e-mail: idelpress@mail.ru

ISBN 978-5-9775-1182-7

Содержание

Введение для родителей

Зачем школьнику эта книга

Назначение книги — помочь ребёнку 10–13 лет сделать первые шаги в программировании и получить удовольствие от этого процесса.

Почему не «научить программировать», «освоить Python», «подготовиться к олимпиадам»? Потому что всё это будет вторым, третьим, n-м шагами на пути программиста. А для начала нужно на этот путь встать и проверить, подходит ли он тебе. В 4–6 классах школы нужно как можно больше всего попробовать, чтобы к 7–9 классу найти то, что нравится и хорошо получается и углубиться в него, а в 10–11 прагматично идти к будущей профессии.

Почему Python и работа с числовыми и строковыми данными, а не визуальный язык и визуальное программирование? Потому что в основе профессии «программист» — прочные навыки алгоритмизации, а работа с данными на аскетичном структурном Питоне — самый быстрый путь к выработке этих навыков.

В основе книги — методика «программирования по аналогии», когда ребёнок сначала экспериментирует с текстами программ, модифицирует их, догадывается о работе операторов и языковых конструкций, и только потом обобщает приобретённый опыт. За рубежом такой стиль обучения используется давно, а в последние годы стал широко применяться и у нас с разной степенью успешности.

Книга сделана по мотивам курса для учеников 5–7 классов, построенного именно на этой методике и проведённого онлайн (что лишало преподавателя кнута и вынуждало рассчитывать только на пряники). Курс удался: интересно и весело было всем, и примерно пятая часть учеников продолжила программировать самостоятельно после его завершения, — это примерно соответствует доле людей, которые могут быть счастливы, программируя[1]. А остальные, попробовав программирование, убедились, что оно интересное и нестрашное, и побежали примерять на себя другие виды деятельности.

---

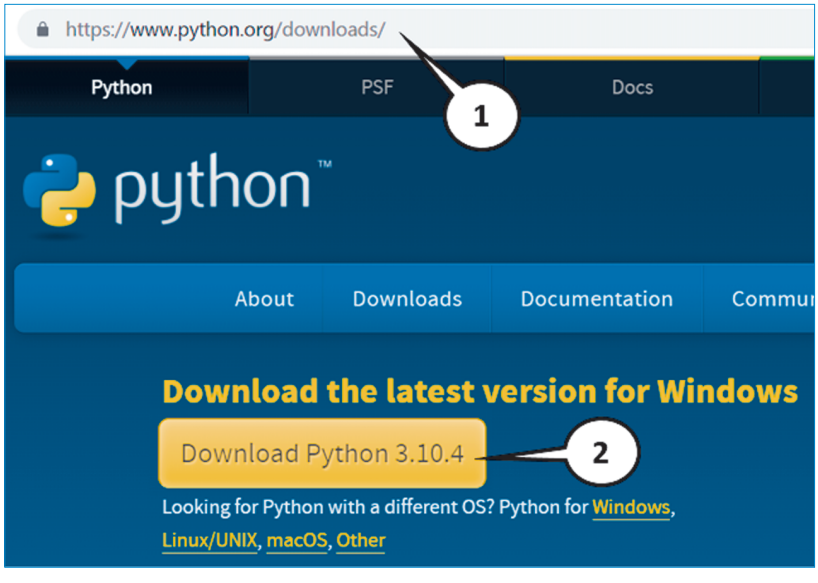
[1] Абсолютно ненаучные данные из жизненного опыта автора, более 20 лет обучающего программированию детей и взрослых.

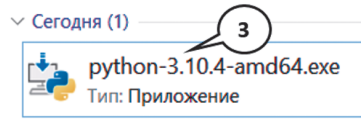
Читателю предлагаются задания — задачи и проекты — разных уровней (те, что со звёздочкой, требуют нестандартных идей), с привлекательными для детей сюжетами. К сложным заданиям есть подсказки, к принципиальным — ответы. Полные тексты программ, в том числе версии реализации проектов, можно найти в электронном архиве, прилагаемом к книге, и на веб-странице книги на сайте издательства **www.bhv.ru**.

Для работы на Python, естественно, понадобятся язык и среда программирования. Всё это можно совершенно бесплатно и без особых технических проблем скачать с официального сайта — **https://www.python.org/downloads/**, выбрав версию, подходящую для вашей операционной системы. Хорошо, если ребёнок примет участие в процессе, однако вряд ли стоит полностью доверять ему это ответственное дело.

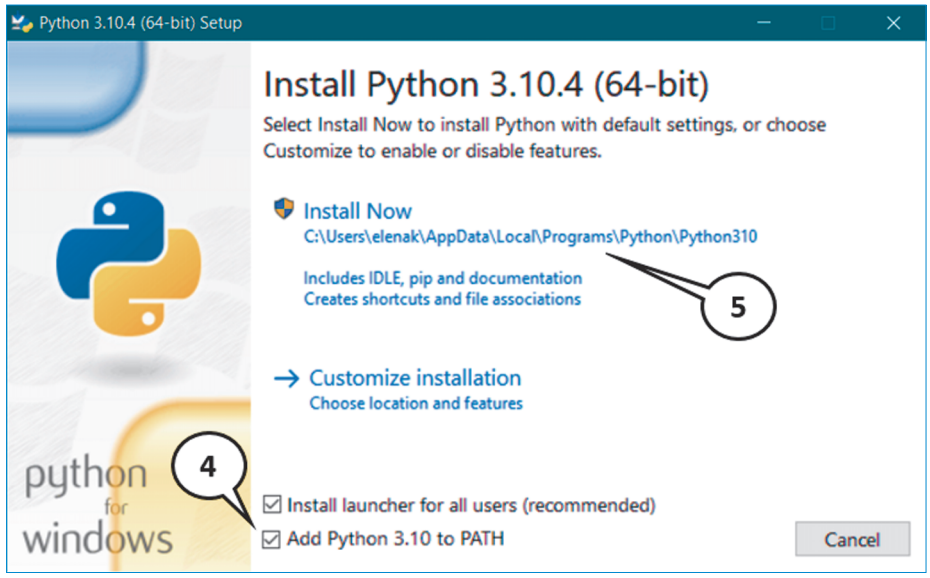
Как установить Python

1. Идём на официальный сайт Python — **https://www.python.org/downloads/**.
2. Нажимаем жёлтую кнопку (на ней могут быть другие цифры — пока книжку издадут, «питонщики» уже пяток новых версий выпустят).





3. Запускаем установщик Python.
4. Ставим галочки у предложений сделать Python доступным для всех пользователей и запомнить путь к папке Python.



5. Щелкаем по ссылке **Install Now**, запомнив, куда мы устанавливаем Python.
6. Дожидаемся сообщения, что установка завершилась успешно, и закрываем окно.

Если ваше чадо — уверенный пользователь компьютера, можно с этого момента оставить его с книгой наедине. Если вы в этом не уверены, то стоит, наверное, вместе найти и запустить Python, что-то понаколачивать на клавиатуре, убедиться, что ребёнок сумеет и переключить раскладку, и найти на клавиатуре кавычку и двоеточие. А можно, если интересно, и всю первую главу вместе пройти (если чадо не возражает). А можно и не только первую.

Введение для детей

Питон: зачем и как?

Питон (Python, «Пайтон») — язык программирования. Настоящий «взрослый» язык, один из самых популярных в мире. При этом немногословный и логичный. Местами забавный. И название у него прикольное: можно почувствовать себя заклинателем змей. И если уж начинать программировать, то на Питоне.

Осваивать его вы будете в отличной компании (на следующей странице познакомитесь), будет с кем поболтать и посмеяться, найдётся, к кому обратиться за подсказкой.

Только программируя, можно научиться программировать. Поэтому расслабленно читать книгу, лёжа на диване, не получится: придётся экспериментировать, вводить коды из книги и придумывать свои.

Главы книги не уроки: одну вы преодолеете за вечер, с другой проведёте пару недель, доводя до совершенства свой проект. Книга тем и хороша, что читать её можно медленно или быстро, пропускать то, что и так понятно, возвращаться к тому, что понравилось. Можно отвлечься на чай с печеньками или пролетающего за окном крокодила. Можно отодвинуть книгу в сторону и колотить по клавишам, реализуя только что пришедшую в голову идею... а потом вернуться к книге и увидеть: то, что заняло у вас два экрана, можно было запрограммировать в три строки.

Ваша первая книга по программированию немного научит вас языку Python, слегка познакомит с основами алгоритмизации, покажет, как придумывать свои проекты и реализовывать их. Но главное — позволит вам получить удовольствие от создания забавных программ или решения не совсем обычных задач. А дальше вы сами решите, хочется вам программировать ещё или нет.

Контрольный вопрос *(проверка умения читать)*: в каком абзаце упоминался крокодил и что он там делал?

Знакомьтесь: Чайник, Кофейник и другие

Чайник

Ваш ровесник, в меру любопытный, неглупый, но пока неопытный. Не боится задавать глупые вопросы и делать дурацкие ошибки.

Кофейник

Взрослый. Знает не всё, но очень много. Про программирование, про свойства чисел и законы логики, про повадки пресмыкающихся и привычки школьников... И у него масса интересных знакомых.

Пайтон

Вполне дружелюбное пресмыкающееся. Язык-тёзку знает в совершенстве, из-за этого слегка задаётся. Приползает и уползает, когда захочет.

Эксперты

С ними вы познакомитесь по ходу дела. Они в каждой главе разные.

Предупреждение: не надо удивляться тому, что все персонажи курса, включая хвостатых, крылатых и вообще на вид неодушевлённых, программируют (разговаривают, думают...) на Питоне. Жить в мире, где всё мыслит, значительно интереснее. А если чайники, коты, кукушки думают — почему бы им не думать на Питоне?

Элементы книги

Теория в диалогах — видна по картинкам с персонажами.

Эксперимент — наколотить текст программы, запустить, осмыслить то, что получилось.

Конспект — короткий справочник с описанием команд и другой полезной информацией.

Задача — проблема, для решения которой чаще всего нужно написать программу по заданным требованиям. Иногда задачи объединяются в серии по теме или применяемым приёмам.

Тест — вопросы, на которые нужно ответить, а потом сверить ответы. Не нравится — не делайте...

Проект — самое интересное: вы придумываете свою программу на предложенную тему и создаёте её. Оценить её могут родители, друзья, одноклассники.

Капелька информации — это тоже теория, но не про программирование, а про что-то другое. Например, кусочек математики, без знакомства с которым ваше удовольствие от общения с Питоном будет неполным.

Переход на новый уровень — доступ к проекту, с которым вы теперь справитесь, или ещё к чему-то интересному.

Глава 1

Очень приятно, Пайтон

Языки программирования

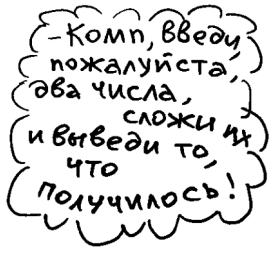
— Так... Ну, я готов! Где этот ваш Питон? Мне Миксер, который в 7 классе, рассказывал, что Питон — такой крутой язык программирования, на котором можно игры писать. Поехали!

— А почему для написания программ нужен какой-то особый язык — вы не задумывались? Почему нельзя объяснить компьютеру, что надо делать, по-человечески?

Ч: Ну, он, наверное, тупит... ой, простите! Компьютер — он как иностранец: знает свой язык, а наш не знает. И поэтому приходится говорить с ним на его родном языке.

К: Направление мысли правильное. Но в действительности язык программирования — вовсе не родной язык компьютера. Это такой язык, который понятен и компьютеру, и нам. Вы японский не знаете, верно? Предположим, вам очень надо что-то спросить у японца, не говорящего по-русски. Как вы это сделаете?

Ч: Спрошу на английском — скорее всего, он его знает. Я буду использовать только самые простые слова (я ж только их и знаю). И он, видимо, ответит мне так же.



К: Так вот, язык программирования — это такой компьютерный английский. Вот (слева) пример программы на языке человека.

А программа в памяти компьютера, которую он в состоянии выполнить, выглядит примерно так:

Ч: В смы-ы-ы-сле?

К: Компьютер — электронное устройство. Память его состоит из элементов с двумя состояниями: есть

сигнал — 1, нет сигнала — 0. Мозг компьютера, процессор, преобразует сигналы, пропуская их через сложные электронные схемы, и получает результаты. Но вы не пугайтесь — Вас никто не заставляет разговаривать с компьютером, вводя с клавиатуры нолики и единички. Вот так выглядит, к примеру, программа на Питоне. А рядом — программа на Small Basic.

```
a = int(input())
b = int(input())
print(a+b)
```

```
a = TextWindow.ReadNumber()
b = TextWindow.ReadNumber()
TextWindow.WriteLine(a+b)
```

Ч: Не-не, лучше Питон, там буковок меньше! Зачем вообще другие языки?

Вопрос для размышления: действительно, почему есть много языков программирования, а не какой-то один, лучший? Ответ Кофейника — в Ответах. ОТВЕТ>>>

Зачем и как придуман Python

— Лучше Питон? Уверены? Давайте спросим эксперта. Гвидо ван Россум — «тот, кто придумал Пайтон». Гвидо, не могли бы Вы рассказать, чем Питон отличается от других языков?

— Привет! А у вас картинка устарела, я сейчас без бороды!

Про Python? Когда мы его придумывали, хотелось сделать язык без лишних знаков препинания, расставляемых по сложным правилам, и чтобы вообще текст программы был коротким и при этом понятным. И вроде получилось.

Почему он «питон»? Ну, вообще-то это не в честь змеи, а в честь одного телешоу... впрочем, неважно. Так вышло, что на Питоне можно писать программы в смешном таком стиле — вся программа вписывается в одну длинную строку с множеством скобок. Так что змеистое название языку подошло.

Сейчас Python — один из самых популярных языков программирования. И к нему можно подключать множество библиотек — таких дополнений к языку, заточенных под конкретные виды задач.

Да, самое-то важное! Python — свободная программа, его можно совершенно бесплатно поставить на свой комп.

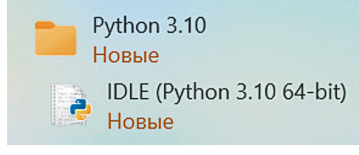
Ч: А игры на нём делать можно?

— Да конечно! Всё можно — хоть игрушку, хоть сайт, хоть искусственный интеллект... Скачивайте и начинайте работать! Это просто и весело. Удачи!

Питон интерактивный и файловый

— Э-э-э... а где взять этот самый Python и как на нём программировать?

— Посмотрите в главном меню — есть там вот такая штучка?



Если нет — попросите родителей открыть своё, взрослое «Введение» и вместе с вами установить Python по инструкции. Должна ж быть от родителей какая-то польза!

ЭКСПЕРИМЕНТ: интерактивная арифметика — проба пера

IDLE (Python 3.10 64-bit)

Приложение

Запустите IDLE Python. Как? Ну, например, через главное меню. А можно воспользоваться поиском по слову Python или IDLE.

Вот эта штука и есть IDLE (читается «айдл») — программа для разработки программ. Не заморачивайтесь многословным выведенным текстом. Вводить мы будем после знаков **>>>** — это сигнал о том, что Python готов к работе и ждёт нашей команды.

Введите **>>> 5+3** и нажмите клавишу <Enter>. Верно сосчитал Python?

А если посложнее? И с множеством разных действий? К примеру, вот это: `>>> 5*8 + 77/11`

Теперь вы знаете, как в Питоне обозначают умножение и деление (с плюсом и минусом и без того всё понятно). А давайте-ка проверим, как Python отнесётся к большим числам: введём

```
>>> 999999999*999999999*999999999
```

Верно? Не уверены? А вы проверьте на калькуляторе! Чего-о-о?

– Слабак ваш калькулятор! Я могу правильно работать с числами любой длины. Правда, иногда приходится несколько секунд подумать... ну и результат долго выводится.

Продолжим. Введите **`a = 13`**.

Ничего не произошло. А теперь введите просто **`a`**. Вывелось **`13`**. Мы выполнили сейчас очень важное действие: записали число 13 в переменную с именем «a». И теперь можем его использовать в наших вычислениях.

Проверим: вводите примеры, в которых используется наша переменная — например, **`a*(a+1)`**. Убедитесь, что в переменной **`a`** по-прежнему 13. Кстати, вы заметили, что теперь в тексте не картинки, а просто то, что вы должны вводить? От остального текста отличается другим шрифтом — тем же, что в IDLE.

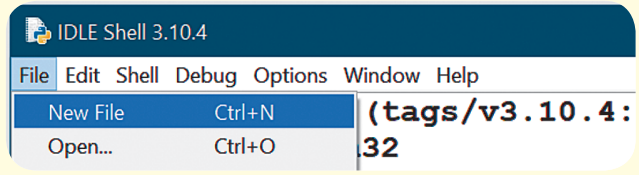
А теперь введём **`a=a*a`**. И выведем **`a`**. Произошло ещё одно важное событие — мы изменили содержимое переменной, присвоив ей значение вычисленного выражения.

— Вау! Я могу писать программы!

— То, что мы делали — это ещё не программа, а так, наброски к ней. Программа — это файл, который хранится в памяти. Его можно использовать многократно, выполняя для разных данных. Чтобы перейти в режим работы с файлом, надо его создать. Потом наполнить содержимым, сохранить и исполнить. При этом не факт, что программа сразу окажется правильной...

ЭКСПЕРИМЕНТ: создаём файл с программой

1. Откройте меню **File** и выберите **New File** (это же действие можно выполнить, одновременно нажав клавиши <Ctrl>+<N>)[1].

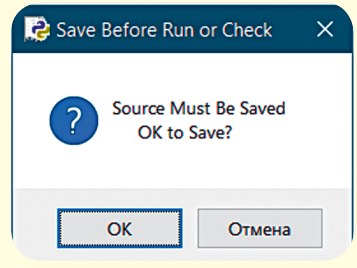


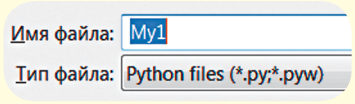
2. Введите текст программы. К примеру, вот такой:

```
print("Привет! Ты кто?")
u = input()
print("Круто! Мне впервые встречается", u + "!")
```

Да, сло-о-о-жно-о-о — приходится переключаться с русского на английский и искать всякие знаки препинания. А зато цвет сам собой меняется!

3. Готово? Запускаем программу, нажимая клавишу <F5> или же выбирая в меню **Run** команду **Run Module**. Ой, вылезло окошко! Это вам напоминают, что файл должен быть сохранён. Жмите **OK** и сочиняйте имя для своей первой программы. И стоит обратить внимание, на какой диск и в какую папку вы всё это сохраняете. Если ничего не менять, файлы будут там же, где сам Python — для начала сойдёт.





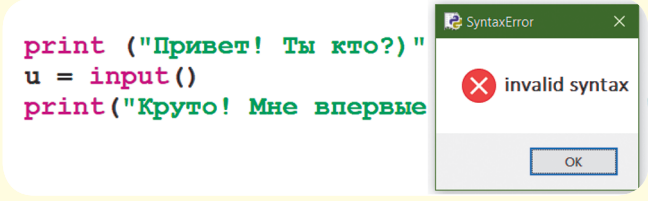
4. Если вы всё ввели правильно, программа задаст вопрос, вы введёте ответ... получится что-то вроде этого:

[1] Для непродвинутых пользователей: <Ctrl>+<N> — это нажать клавишу <Ctrl> и, не отпуская её, другим пальцем (той же или другой руки, ноги и т. п.) нажать <N>. А не нажимать последовательно <C>, <T>, <R>, <L>, <+>...

```
Привет! Ты кто?
Конь в пальто
Круто! Мне впервые встречается Конь в пальто!
>>>
```

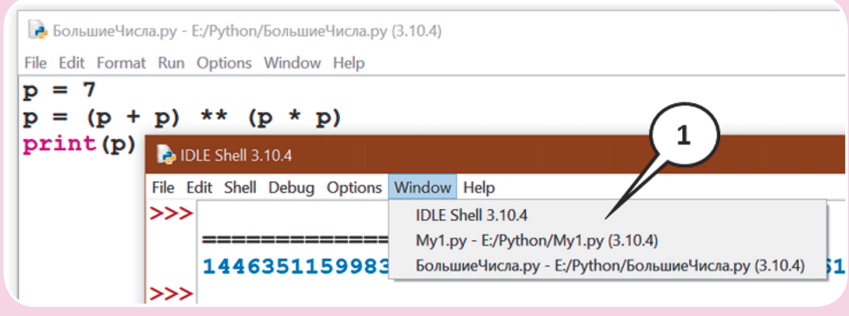
5. А вот если вы что-то ввели неправильно (например, поменяли местами «)» и «"» в первой строке) — IDLE скажет, что вы неправы. Причём почувствует она эту ошибку только на следующей строке.

   Исправляйте ошибку и возвращайтесь к пункту 3:



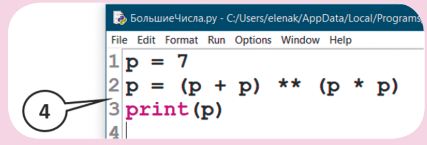
КОНСПЕКТ: полезные возможности IDLE Python

1. Окно выполнения (то, где мы работаем в интерактивном режиме и видим результат выполнения программы) одно. Но при этом может быть открыто несколько файлов. Переключение между окнами — через меню **Windows**.

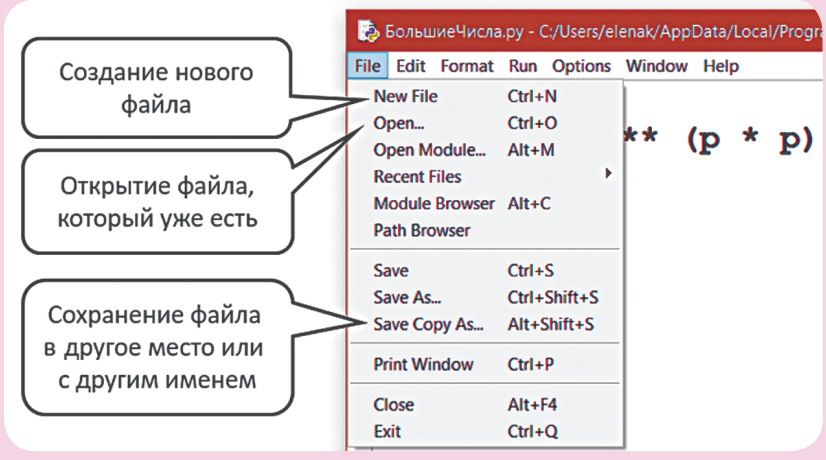


2. Можно изменить размеры окна файла и окна выполнения и расположить их на экране рядом.

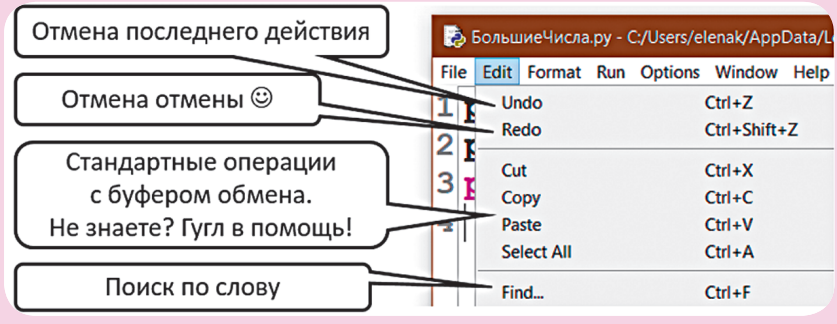
3. Хорошо бы сразу настроить IDLE так, чтобы вам в ней было комфортно. В меню **Options** щёлкните по **Configure IDLE** и выберите подходящий размер шрифта (те, кто уже испортил зрение, могут ещё и жирность использовать).
4. Полезная штука — номера строк. Их можно включить в том же меню **Options** — **Show Line Numbers**.



5. Операции с файлами — естественно, в меню **File**.



6. Масса полезных возможностей в меню **Edit**. Для начала нам хватит того, что сверху. И хорошо бы выучить клавиатурные комбинации.



Практикум

ЭКСПЕРИМЕНТ: приключения трёхзначного числа

1. Создаём новый файл и вводим текст программы (красненькие комментарии вводить не надо, они вам потребуются для осмысления).

```
s = input("Введите трёхзначное число: ")
s = s + s     # склеили то, что ввели, с самим собой
n = int(s)    # сделали из строки число
n = n // 7    # разделили число на 7
n = n // 11   # потом разделили на 11
n = n // 13   # ну и на 13 разделили
print("Результат:", n)
```

2. Сохранив программу и убедившись, что в ней нет ошибок, запускаем её несколько раз. Сколько? Столько, чтобы заметить некоторую странную закономерность.

Почему так получается — вот мы курочим число, делим его на всякие гадкие числа, а оно делится да ещё и даёт в итоге интересный результат? ПОДСКАЗКА>>> ОТВЕТ>>>

ЗАДАЧА: от роста к весу – программируем по аналогии

Имеется текст программы, которая просит пользователя ввести рост в сантиметрах, и переводит его в метры и в миллиметры. Превратите её в программу, которая получает от пользователя его вес в килограммах, и выводит его в пудах (Гугл в помощь! Или бабушка!) и в граммах.

```
rost = int(input("Введите ваш рост в см: "))
m = rost / 100
mm = rost * 10
print("Ваш рост в метрах =", m)
print("Ваш рост в мм =", mm)
```

Если вам ну очень трудно вводить текст программы, можете скачать его из архива рабочих материалов книги (файл *1-1.py* в папке 1). Но вообще-то это для слабаков.

ПРОЕКТ: псевдографический конструктор

Вот текст программы. А справа — результат её работы.

```
a0 = "..."
a1 = "..#"
a2 = ".#."
a3 = ".##"
a4 = "#.."
a5 = "#.#"
a6 = "##."
a7 = "###"
print(a4+3*a7+a1)
print(a6+3*a5+a3)
print(a7*5)
print(a6+3*a5+a3)
print(a4+3*a7+a1)
```

```
#..#########..#
##.#.##.##.#.##
###############
##.#.##.##.#.##
#..#########..#
```

А вот ещё несколько примеров изображений, построенных из этого конструктора.

```
#######....##..##.....######...#.......#.#######..#....#....#####....#....#
...#.....###.##.###...#.....#..##.....##....#.....#....#...##...##...##...#
...#....##........##..#.....#...##..##......#.....######..##.....##..#.#..#
...#.....##.......##...######......###.......#.....#....#..##.....##..#.#.#
...#.......##..##.....#...........#........#.....#....#...##...##...#...##
#######.....##.......#...........#........#.....#....#....#####....#....#
>>>
```

```
.........###.........
.......#.....#.......
......#...#...#......
......#.......#......
.........###.........
..........#..........
..........#..##......
..........#.##.......
..........#..........
..........#..........
..........#..........
..........#..........
```

```
...#.......#.....
...##.....##.....
...###...###.....
...#########.....
...##.###.##.....
...####.####.....
...#########.....
......###........
...#########.....#
...#########...##.
...#########...##.
...#########...##.
...#########.##...
..############..
```

Вам предлагается соорудить из псевдографического[1] конструктора — строк a0–a7 — свои изображения. Какие? Любые. Лишь бы они вам нравились. Если понравятся кому-то ещё — совсем здорово будет.

Лентяи могут текст конструктора не вводить, а скопировать (файл *1-2.py* в папке 1).

Вопрос для размышления: как связаны номера строк-картинок и их содержание? ПОДСКАЗКА>>> ОТВЕТ>>>

[1] Псевдографика — построение изображений из символов.

МИКРОТЕСТ: что мы знаем о Питоне

Вы уже кое-что знаете о Python. И кое-какое мнение о нём у вас уже есть. Давайте вы отметите те утверждения, с которыми согласны, а потом сверим ваше мнение с мнением автора.

А. Python придумали в Южной Америке, потому он и Питон.

Б. Python — язык не новый, он давно стал популярным и оброс полезными дополнениями.

В. Со строками и с числами Python работает по-разному: например, знак «+» строки склеивает, а числа складывает.

Г. Максимальный размер числа в Python — 64 цифры.

Д. IDLE — среда для разработки программ на Python.

Е. Можно выполнять команды Python по одной, не записывая в файл.

Ж. Файлы с текстом программы на Python имеют расширение .pi.

Прошли тест? Вы заслужили маленький бонус! Вот такой значок показывает, что теперь вам доступен проект в приложении: **«Вырасти своего Питона!»**

ПРИЛОЖЕНИЕ 2
Подсказки

Глава 1

ЭКСПЕРИМЕНТ: приключения трёхзначного числа

Попробуйте перемножить числа 7, 11 и 13.

ПРОЕКТ: псевдографический конструктор (вопрос для размышления).

Погуглите про двоичную систему счисления и запишите в ней числа от 0 до 7.

Глава 2

ЗАДАЧИ: повседневная арифметика

Плата за воду. Формула для расчёта такая:

(ПоказанияВТекущемМесяце – ПоказанияВПрошломМесяце) · Тариф.

Дурацкая подсказка, это вы и сами знаете? Вот вторая: чтобы вещественные числа выводились не с множеством цифр после точки, а всего с двумя, результат (переменную `plata`) можно вывести так:

```
print("Плата за воду, руб.:", round(plata, 2))
```

Квасим капусту. Количество любого ингредиента надо умножить на число, равное 1/10 от желаемого количества литров капусты. Непонятно? Проверим на числах: если мы хотим 20 литров, всего придётся брать в 2 раза больше: 20 : 10 = 2. А если хотим всего 2 литра, всего надо брать в 5 раз меньше, то есть 2 : 10 = 0.2.

Содержимое копилки. Посчитайте сумму в копейках, а потом с помощью операций `//100` и `%100` выделите из неё рубли и копейки.

Глава 4

ЗАДАЧА: программа по блок-схеме

На какую из блок-схем Бени похожа эта? Правильно, но ту, где `elif`. Структура программы будет та же, изменятся только условия.

ЗАДАЧА: уравнивание яблок

В программе будут три ветки: яблок поровну, у Анечки больше, у Боречки больше. А отдать тот, у кого больше, должен половину разности между количеством своих яблок и количеством яблок другого.

ЗАДАЧА: распознавание объектов

Объекты в условии задачи очень удачно расположены. Первым `if` отсечём эдакера с его нечётным количеством виляек. Затем пойдут `elif`, описывающие другие виды, тут лучше прислушаться к совету Пайтона и использовать двойные неравенства. Ну а нераспознанные существа — это `else`.

ПРОЕКТ: тестер

Для подсчёта верных ответов в начале программы подготовьте переменную, присвоив ей значение 0 (например, `c = 0`). А потом всякий раз, как ответ оказался верным, прибавляйте к ней 1: `c = c + 1`. Значение этой переменной в конце программы и будет результатом.

Глава 8

ЗАДАЧА: 10 чисел

Вот пример тестовых значений. При вводе чисел 11, 0, –7, –1.13, 4, –7, 8, 1, 9 результаты должны быть такие:

1. 99. 2. 2. 3. 24. 4. 2. 5. 4.

И идеи решения задач.

1. До цикла заводим накопитель суммы квадратов (начальное значение 0). В цикле после ввода числа проверяем, отрицательное ли оно; если да, прибавляем *r* накопителю его квадрат.
2. Обычный алгоритм подсчёта. Проверяемое условие — число больше нуля и его остаток от деления на 10 равен 1.
3. Давайте попробуем прямо внутри цикла по-особенному работать с первым и вторым числами (для этого придётся проверять значение переменной цикла. Пусть M1 — накопитель для самого большого числа, M2 — для второго по величине. Пер-

вое число записываем в M1. Второе записываем в M2, и если M2 > M1, поменяем их местами (M1, M2 = M2, M1). А с остальными числами так: если введённое число больше M1, запишем M1 в M2, а на его место новое число. Иначе если новое число больше M2, пишем его в M2.

4. Будем сохранять в одном накопителе наименьшее число, а во втором — количество равных ему. Каждый раз, как значение наименьшего числа будет меняться, количество значений, равных ему, будет становиться равным 1. Если же введённое число равно минимальному, увеличим на 1 второй накопитель.
5. Всё просто: потребуется переменная, в которую мы будем записывать очередное число, прежде чем вводить следующее. Все числа, начиная со второго, будем сравнивать с предыдущим, и если новое больше старого — увеличим счётчик.

Глава 9

ЗАДАЧА: распознавание объектов-2

В программе будет цикл с постусловием — **`while True`**. В цикле мы введём количество хапок, если оно равно нулю — прервём цикл. А ненулевые значения числа хапок проанализируем в многоветочном условном операторе.

Помимо распознавания нам придётся вести и подсчёт, точнее, несколько подсчётов. Нам понадобятся счётчики для каждого вида инопланетных существ — 4 штуки. В каждой ветке условного оператора будем выводить название существа и увеличивать на 1 соответствующий счётчик.

ЗАДАЧА: решение подбором и перебором

В задаче потребуется находить первую, вторую и третью цифры трёхзначного числа. Отделять цифры от числа в цикле (задача 4 из эксперимента «Что делает программа») не стоит — число небольшое, каждая цифра вычисляется в 1–2 действия:

```
z1 = n // 100
z2 = (n // 10) % 10
z3 = n % 10
```

ПРИЛОЖЕНИЕ 3

Ответы

Глава 1

Ответ Кофейника на контрольный вопрос о том, почему языков программирования много

В разных ситуациях нам важны разные свойства языка программирования. Например, иногда надо, чтобы программа работала очень быстро и занимала мало места в памяти. А иногда важнее, чтобы программа легко и быстро писалась, а сколько она там работает, одну миллисекунду или две, неважно. Некоторые языки ценят за то, что они отлично подходят для решения задач какой-то предметной области, а некоторые за то, что на них можно работать на чём угодно — на компьютере, планшете, смартфоне, часах... Ну не может один язык удовлетворять всем требованиям сразу! Поэтому языков программирования много.

ЭКСПЕРИМЕНТ: приключения трёхзначного числа

Склеив трёхзначное число с самим собой, мы тем самым умножили его на $1001 = 7 \cdot 11 \cdot 13$. А потом на эти же числа разделили, тем самым вернувшись к исходному числу.

ПРОЕКТ: псевдографический конструктор (вопрос для размышления)

Элементы конструктора можно получить, заменив в двоичном представлении чисел от 0 до 7 (короткие числа дополним слева нулями до трёх символов) цифру 0 на «.», а цифру 1 на «#».

МИКРОТЕСТ: что мы знаем о Питоне

Верны высказывания Б, В, Д, Е.

А. Питон придумали в Южной Америке, потому он и Питон.
Нет, его придумал голландец, живущий сейчас в Америке (Северной).

Г. Максимальный размер числа в Python — 64 цифры.
Неверно, размер числа в Python не ограничен.
Ж. Файлы с текстом программы на Питоне имеют расширение .pi.
Расширение файлов с программами на Python — .py.

Глава 2

ЭКСПЕРИМЕНТ: строки и числа

1. Результат не изменится — переменные можно называть и на русском. Но не нужно.
2. Красивее всего так: `str3 * int3 + str(int3) * (int3 + 2)`.
3. Строку можно преобразовать в число с помощью функции `int()`. Естественно, не любую: если Питон не сможет преобразовать строку в число, программа прервётся с сообщением об ошибке.
4. Допустимы и те, и другие кавычки.
5. Чёрное `str` — часть имени переменной, сиреневое — название функции, то есть слово, которое Питон считает своим. Если вы попытаетесь назвать переменную не `str3`, а просто `str` — получите сообщение о синтаксической ошибке.

Вопрос для размышления про то, чем недоволен Пайтон

Варианты, начинающиеся с «он», откинем сразу — Пайтон не ошибается, не подвержен суевериям и перепадам настроения. А вот случайное использование пользователем сходной по начертанию буквы кириллицы — одна из распространённых и трудноуловимых ошибок.

ЭКСПЕРИМЕНТ: как решать задачу, или Курить вредно!

Все предложенные изменения пойдут на пользу: и сокращение текста программы (А, Б), и совершенствование интерфейса (В, Г).

ЭКСПЕРИМЕНТ: шесть единиц

11**1111. В этом числе 1157 цифр (а в 111**111 всего 228).

МИКРОТЕСТ: что мы знаем о данных

Верны высказывания Б, В, Г, Д, Е (да, '99' действительно больше, чем '100', потому что это строки, а они сравниваются по первой цифре: у кого она больше — тот и больший).

ПРИЛОЖЕНИЕ 4

Состав электронного архива

В архив включены:

- тексты программ, приведённых в книге;
- версии текстов программ, являющихся решениями задач и реализациями проектов;
- тестовые наборы значений для олимпиадных задач.

Файлы архива распределены по папкам, соответствующим главам книги. В папке находятся файлы с текстами программ из книги (они нумеруются последовательно, например, файлы к первой главе называются 1-1.py и 1-2.py) и вложенная папка *Ответы*. Имена файлов с ответами соответствуют названиям задач и подзадач, они русскоязычные и вполне внятные, например, Квасим_капусту.py).

Тестовые значения к задачам главы 13 помещены во вложенную папку Тесты, пары тестовых значений к каждой задаче упакованы в ZIP-архив с названием, соответствующим названию задачи.

Таблица 1. *Состав электронного архива*

Папка/глава	Вид контента	Количество файлов
1	Тексты программ из книги	2
	Ответы к заданиям	1
2	Ответы к заданиям	3
3	Тексты программ из книги	2
	Ответы к заданиям	1
4	Тексты программ из книги	1
	Ответы к заданиям	5
5	Тексты программ из книги	2
	Ответы к заданиям	3

***Табл.1** (окончание)*

Папка/глава	Вид контента	Количество файлов
6	Тексты программ из книги	1
	Ответы к заданиям	3
7	Тексты программ из книги	1
	Ответы к заданиям	2
8	Тексты программ из книги	3
	Ответы к заданиям	4
9	Тексты программ из книги	3
	Ответы к заданиям	4
10	Тексты программ из книги	5
	Ответы к заданиям	2
11	Ответы к заданиям	7
12	Тексты программ из книги	1
	Ответы к заданиям	4
13	Тестовые пары к задачам главы	12
	Ответы к заданиям	12
Приложение	Ответы к заданиям	22

PYTHON ДЛЯ ДЕТЕЙ, КОТОРЫЕ ПОКА НЕ ПРОГРАММИРУЮТ

Назначение книги — помочь ребёнку 10–13 лет сделать первые шаги в программировании и получить удовольствие от этого процесса.

Почему не «научить программировать», «освоить Python», «подготовиться к олимпиадам»? Потому что всё это будет вторым, третьим, n-м шагами на пути программиста. А для начала нужно на этот путь встать и понять, подходит ли он.

В основе книги — проверенная на учениках разного возраста и уровня подготовки методика «программирования по аналогии», когда ребёнок сначала экспериментирует с текстами программ, модифицирует их, догадывается о работе операторов и языковых конструкций и только потом обобщает приобретённый опыт.

В каждой главе читатель-школьник сталкивается с проблемой, экспериментирует, выслушивает мнения экспертов, решает задачи и реализует проекты. В электронном архиве есть ответы, подсказки и другие необходимые материалы, в том числе почти сотня текстов программ.

«Так это что, читать придётся?» Да, читать. Тот, кто читать не умеет и не любит, хорошим программистом не станет. А вот из гуманитариев-книгочеев программисты вырастают прекрасные, проверено.

Читать будет не скучно: в книге множество разнообразных персонажей, есть с кем порешать задачи, обсудить проекты, поболтать и посмеяться.

Крылова Елена Геннадьевна — преподаватель Высшей инженерной школы Санкт-Петербургского Политехнического университета Петра Великого (Академия информатики для школьников), автор курсов для детей и взрослых, организатор олимпиад по информатике и программированию, интеллектуальных игр, квестов. Сфера профессиональных интересов — методики обучения программированию и алгоритмизации, адаптация образовательного контента к особенностям восприятия современных школьников.

Электронный архив с файлами к заданиям можно скачать по ссылке https://zip.bhv.ru/9785977511827.zip, а также со страницы книги на сайте bhv.ru

ISBN 978-5-9775-1182-7

9 785977 511827

191036, Санкт-Петербург,
Гончарная ул., 20
Тел.: (812) 717-10-50,
339-54-17, 339-54-28
E-mail: mail@bhv.ru
Internet: www.bhv.ru