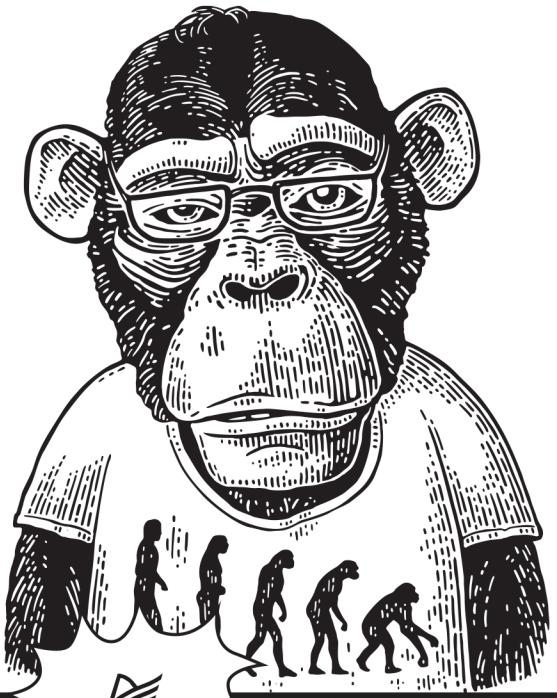


# РУТНОН, НАПРИМЕР

НИКОЛА  
ЛЕЙСИ



# PYTHON BY EXAMPLE

Learning to Program in 150 Challenges

NICHOLA LACEY

*Nichola Wilkin Ltd*

НИКОЛА ЛЕЙСИ

# РУТНОН, НАПРИМЕР



Санкт-Петербург · Москва · Минск

2021

Никола Лейси

# Python, например

Серия «Библиотека программиста»

Перевел с английского  
Заведующая редакцией  
Руководитель проекта  
Ведущий редактор  
Литературный редактор  
Художественный редактор  
Корректоры

*Е. Матвеев  
Ю. Сергиенко  
А. Питиримов  
А. Юринова  
Е. Шубина  
В. Мостипан  
Н. Сидорова, Г. Шкатова*

ББК 32.973.2-018.1

УДК 004.43

**Лейси Никола**

Л92 Python, например. – СПб.: Питер, 2021. – 208 с.: ил. – (Серия «Библиотека программиста»).  
ISBN 978-5-4461-1826-7

Это Python, например! Познакомьтесь с самым быстрорастущим языком программирования на сегодняшний день. Легкое и увлекательное руководство поможет шаг за шагом прокачать навыки разработки. Никаких архитектур компьютера, теорий программирования и прочей абракадабры – больше практики! В книге 150 задач, которые плавно перенесут читателя от изучения основ языка к решению более сложных вещей. Руководство подойдет всем, у кого голова идет кругом от технического жаргона и пространных объяснений – автор уверен, что учить можно и без этого.

**16+** (В соответствии с Федеральным законом от 29 декабря 2010 г. № 436-ФЗ.)

ISBN 978-1108716833 англ.  
ISBN 978-5-4461-1826-7

© Nichola Lacey 2019  
© Перевод на русский язык ООО Издательство «Питер», 2021  
© Издание на русском языке, оформление ООО Издательство «Питер», 2021  
© Серия «Библиотека программиста», 2021

Права на издание получены по соглашению с Cambridge University Press. Все права защищены. Никакая часть данной книги не может быть воспроизведена в какой бы то ни было форме без письменного разрешения владельцев авторских прав.

Информация, содержащаяся в данной книге, получена из источников, рассматриваемых издательством как надежные. Тем не менее, имея в виду возможные человеческие или технические ошибки, издательство не может гарантировать абсолютную точность и полноту приводимых сведений и не несет ответственности за возможные ошибки, связанные с использованием книги. Издательство не несет ответственности за доступность материалов, ссылки на которые вы можете найти в этой книге. На момент подготовки книги к изданию все ссылки на интернет-ресурсы были действующими.

Изготовлено в России. Изготовитель: ООО «Прогресс книга». Место нахождения и фактический адрес: 194044, Россия, г. Санкт-Петербург, Б. Сампсониевский пр., д. 29А, пом. 52. Тел.: +78127037373.

Дата изготовления: 06.2021. Наименование: книжная продукция.  
Срок годности: не ограничен.

Импортер в Беларусь: ООО «ПИТЕР М», 220020, РБ, г. Минск, ул. Тимирязева, д. 121/3, к. 214, тел./факс: 208 80 01.

Налоговая льгота – общероссийский классификатор продукции ОК 034-2014, 58.11.12 – Книги печатные профессиональные, технические и научные.

Подписано в печать 27.05.21. Формат 84x108/16. Бумага офсетная. Усл. п. л. 21,840. Тираж 1000. Заказ 0000.



# Оглавление

Введение . . . . .	7
Как загрузить Python. . . . .	10
Подсказки . . . . .	12
<b>ЧАСТЬ I. ИЗУЧАЕМ PYTHON . . . . .</b>	<b>15</b>
Основы . . . . .	16
Инструкция if . . . . .	22
Строки . . . . .	29
Математические операции . . . . .	36
Цикл for . . . . .	40
Цикл while . . . . .	45
Случайные числа . . . . .	50
Черепашья графика . . . . .	56
Кортежи, списки и словари . . . . .	63
Снова о работе со строками. . . . .	72
Числовые массивы . . . . .	77
Двумерные списки и словари. . . . .	83
Чтение и запись текстовых файлов. . . . .	89
Чтение и запись файлов .csv . . . . .	94
Подпрограммы . . . . .	103
Графический интерфейс Tkinter . . . . .	113
Подробнее о Tkinter . . . . .	126
SQLite . . . . .	137
<b>ЧАСТЬ II. УЧЕБНЫЕ ПРОЕКТЫ . . . . .</b>	<b>151</b>
Введение в часть II . . . . .	152
Задача 146. Код сдвига . . . . .	153
Задача 147. Mastermind . . . . .	156
Задача 148. Пароли . . . . .	160
Задача 149. Таблица умножения . . . . .	165
Задача 150. Картичная галерея. . . . .	168
Что дальше? . . . . .	175
Глоссарий . . . . .	176



Python – стремительно развивающийся язык программирования современности. В этом увлекательном и необычном руководстве материал разбивается на доступные пошаговые фрагменты, а теория объясняется кратко и понятно. Вместо того чтобы обрушивать на читателя лавину технического жаргона, вгоняющего в ступор, эта книга предлагает ему поработать над 150 практическими задачами. Создавая программы для решения этих задач, читатель быстро перейдет от азов к уверенному использованию сопрограмм, графического пользовательского интерфейса и к работе с внешними файлами – текстовыми, в формате .csv и базами данных SQL. Книга идеально подойдет каждому, кто хочет освоить программирование на Python. В частности студенты, изучающие computer science, и преподаватели, которые хотят более уверенно овладеть Python, найдут в книге подборку готовых задач для использования на учебных курсах.

**Никола Лейси** (Nichola Lacey) – директор компании Nichola Wilkin Ltd., является авторитетным разработчиком учебных материалов, поставила тысячи курсов в образовательные учреждения по всему миру. Никола – один из самых популярных авторов TES (образовательный ресурс <https://www.tes.com/international>), а ее материалы набирают высокий рейтинг просмотров и сотни тысяч загрузок. Она была программистом до перехода в сферу корпоративного обучения и последующей подготовки на должность преподавателя, а после повышения до уровня руководителя компьютерного направления в частной школе для мальчиков приобрела уникальные навыки из области программирования и практики преподавания.

# Введение



Вам когда-нибудь доводилось брать новый учебник по программированию только для того, чтобы почувствовать, как у вас голова начинает идти кругом, а глаза лезут на лоб в попытках разобраться в пространных объяснениях? Если так, то эта книга написана для вас.

Я была на вашем месте, пытаясь изучать программирование и полагаясь только на традиционные руководства. По собственному печальному опыту знаю, насколько быстро текст начинает плыть, а мозг отключаться. Всего несколько страниц — и я начинаю бездумно глотать слова, уже не понимая их смысл. В итоге неизбежно сдаюсь и после всего произошедшего чувствую себя утопающей, которая жадно хватает воздух после погружения в технический жаргон.

Я ненавидела необходимость вчитываться в бессмысленную чушь и затем сталкиваться с короткой программой. Автор точно указывал, что нужно ввести, а потом на 20 страницах объяснял, что я только что сделала, и описывал 101 способ запуска программы. Я ненавидела отсутствие свободы опробовать что-то самостоятельно и то, что каждая глава с теоретическим материалом завершалась одним-двумя упражнениями.

Я знала, что должен быть способ получше, и, к счастью, он и правда есть. Я написала эту книгу, вы сейчас ее читаете... В общем, вам повезло. Она отличается от других тем, что учит программировать на языке Python с помощью практических примеров, а не мозгодробительных объяснений.



Многие программисты предпочитают учиться через эксперименты. Они просматривают чужой код и разбираются в том, какой способ лучше подходит для конкретной ситуации. Эта книга — пример практического подхода к изучению программирования. После минимальных объяснений читателю предоставляется набор упражнений для создания программ. Вы можете исследовать язык программирования, экспериментировать с ним и просматривать решения, чтобы научиться мыслить как программист. В книге нет глав, которые бы назывались «Архитектура компьютера», «Теория программирования» и содержали бы прочую абракадабру, на которую так часто тратят время другие авторы. Я не стану грузить вас теорией или оглушать самоуверенными рассуждениями, которые начисто отбивают всякую охоту к изучению программирования.

Надеюсь, вам не терпится засесть за создание программ и решение практических задач и наслаждаться чувством достижения, которое вы получаете, гордо глядя на свои строки кода и зная, что вы создали нечто работающее. Это здорово — ваш энтузиазм достоин похвалы. Мой поклон тем, кто уже сидит за компьютером, разминает пальцы и готовится взяться за дело. Если вы относитесь к числу этих людей, если вы уже запустили Python и вам не терпится взяться за дело, тогда вперед. Увидимся в первой главе, которая называется «Основы» (с. 16).

Если вы еще сомневаетесь, расскажу кое-что, прежде чем вы сделаете решительный шаг.



## Как использовать эту книгу

Книга начинается с очень простых программ и постепенно переходит к более сложным. Если у вас еще нет опыта программирования или вы никогда не сталкивались с Python, начните с главы «Основы» и продолжайте читать дальше.

Если же вы знакомы с программированием на Python и уверенно разбираетесь в основах, владеете теорией и логикой программирования, то можете просто переходить к произвольным частям книги для получения интересующих вас деталей.

Книга делится на две части.



### Часть I

В каждой главе части I сначала излагаются базовые правила программирования с задачами, которые вы должны выполнить. Здесь вы найдете:

- **простое объяснение** со ссылками, которые пригодятся начинающим программистам Python;
- **примеры кода** с коротким пояснением, которое может стать основой для последующего решения задач;
- **список задач** постепенно нарастающей сложности. На решение каждой задачи у вас должно уходить от двух до двадцати минут; впрочем, более сложные задачи ближе к концу части I потребуют больше времени, так как арсенал ваших приемов программирования заметно расширится. Однако даже если вы не управитесь за это время, не паникуйте. Пока вам удается решать задачи без слишком частого копирования из предлагаемого решения, все идет нормально;

- код с **возможным решением** каждой задачи. Часто у задачи существует несколько возможных решений, но я привожу только один вариант. Вы можете обращаться к приведенному решению, если у вас возникнут проблемы с конкретным аспектом кода.

## Часть II

В части II будут предложены более серьезные задачи, для решения которых вам потребуются навыки программирования, изученные в части I. Они помогут укрепить и свести воедино те приемы, которые вы осваивали. В этом разделе вы не получите тех подсказок и примеров кода, которые приводятся в части I, поэтому на решение каждой задачи потребуется больше времени. После каждой задачи приводится один возможный вариант ответа – он пригодится вам, если вы зайдете в тупик. Однако нельзя исключить, что вы найдете другое решение, которое работает ничуть не хуже.



## Для кого написана эта книга?

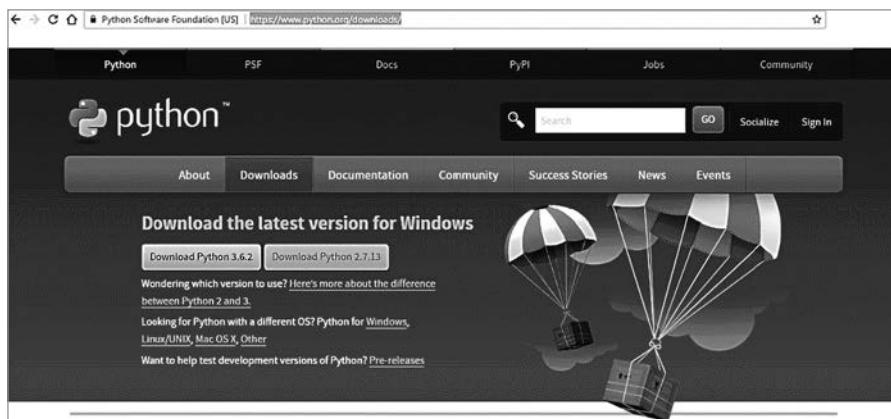


Эта книга подойдет всем, кто хочет научиться программировать на Python. Она также пригодится преподавателям и студентам, которым нужны помощь и готовые примеры для отработки приемов программирования и закрепления навыков. Также книга может использоваться для построения ресурсной базы проекта по программированию, чтобы ученики могли получить дополнительную поддержку или хотя бы краткую подсказку по синтаксису при написании программ.

# Как загрузить Python

Python можно бесплатно загрузить с официального веб-сайта Python:

[www.python.org/downloads/](http://www.python.org/downloads/)



Выберите последнюю версию (в приведенном примере нажмите кнопку Download Python 3.6.2), чтобы запустить программу установки.

На ваш компьютер загружается исполняемый (.exe) файл. При запуске этой программы на экране появляется окно установки, которое выглядит примерно так, как показано ниже.



Выберите вариант **Install Now**. Программа начнет устанавливать Python в вашей системе.



# Запуск Python

Чтобы запустить Python в системе Windows, щелкните по значку Windows или по кнопке меню «Пуск», после чего выберите пункт IDLE (номер версии Python), как показано ниже.



# Подсказки

## Расположение файлов

В системе Windows папка установки Python обычно находится на диске C:\, и ей присваивается имя Python36 (или нечто похожее). Файлы будут автоматически сохраняться в той же папке, если только вы сознательно не выберете другую.



## Комментарии



Комментарии – исключительно полезный инструмент программиста. Комментарии служат двум целям:

- объясняют, как работает программа;
- временно блокируют работу отдельных частей программы, чтобы вы могли запустить и протестировать другие части программы.

Суть первой (и основной) цели – это помочь другим программистам разобраться в том, как работает программа, чтобы они могли понять логику кода, если его потребуется обновить в будущем. Кроме того, комментарии напомнят вам, почему вы написали те или иные строки кода.

```
print("This is a simple program")
print() #Выдает пустую строку для удобства чтения
name = input("Please input your name: ") #Запрашивает ввод имени
print("Hello", name) #Соединяет слово "Hello" и введенное имя
```

В этом примере комментарии добавляются в конце последних трех строк. Они выделены красным цветом и начинаются с символа #.

В реальных условиях вы не будете сопровождать комментариями строки, содержащие очевидный код. Добавляйте комментарии только там, где они необходимы.

Так как Python знает, что все символы после # игнорируются, программисты стали добавлять символ # в начало строк для временной блокировки некоторых строк кода. Это позволяет уделить больше внимания другим частям кода.

```
#print("This is a simple program")
print()
name = input("Please input your name: ")
print("Hello", name)
```

В этом примере символ `#` был добавлен к первой строке программы, чтобы временно запретить ее выполнение. Чтобы вернуть ее в порядок выполнения, просто удалите `#` – код снова будет выполняться.



Я не снабжаю программы комментариями, чтобы вы внимательно читали код и лучше разбирались в нем. Ведь именно так вы сможете научиться программировать по-настоящему! Но если вы будете писать программы в своих курсовых работах, обязательно включайте комментарии, чтобы пояснить смысл кода преподавателю.



## Форматирование кода Python

Во многих версиях Python IDLE можно быстро добавлять комментарии и расставлять отступы с помощью команд меню. Если вам понадобится заблокировать фрагмент кода с помощью комментариев, просто выделите нужные строки, откройте меню Format и выберите команду Comment Out Region. Аналогично поступайте, если для фрагмента нужно изменить величину отступа (позднее я объясню, для чего нужны отступы): это тоже легко делается через меню.



File	Edit	Format	Run	Options	Windows	Help
		<b>Indent Region</b> <b>Ctrl+]</b> <b>Dedent Region</b> <b>Ctrl+[</b> <b>Comment Out Region</b> <b>Alt:+3</b> <b>Uncomment Region</b> <b>Alt:+4</b>				

Что ж, со всеми техническими мелочами мы разобрались. А теперь сделайте глубокий вдох – и за дело!



## От издательства

Ваши замечания, предложения, вопросы отправляйте по адресу [comp@piter.com](mailto:comp@piter.com) (издательство «Питер», компьютерная редакция).

Мы будем рады узнать ваше мнение!

На веб-сайте издательства [www.piter.com](http://www.piter.com) вы найдете подробную информацию о наших книгах.

# Часть 1

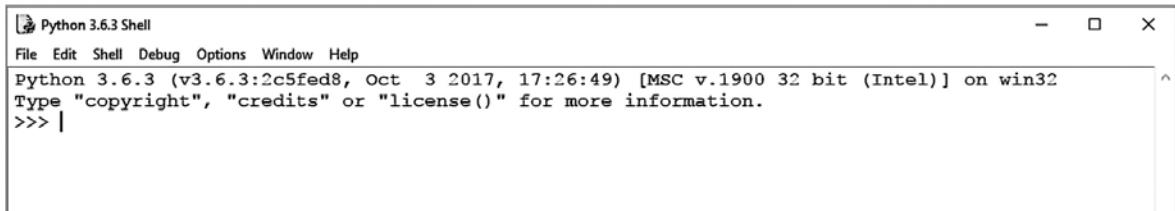
Изучаем Python



# Основы

## Объяснение

Это окно **командной оболочки** – первый экран, который вы видите при запуске Python.



Теоретически код Python можно писать прямо в оболочке, но как только вы нажимаете **[Return]** в конце строки кода, она будет немедленно выполнена. В принципе Python в таком режиме можно использовать как несложный калькулятор; например, можно ввести в приглашении **3\*5**, и Python выведет ответ **15** в следующей строке. Тем не менее такой стиль ввода не подойдет для более сложных программ.

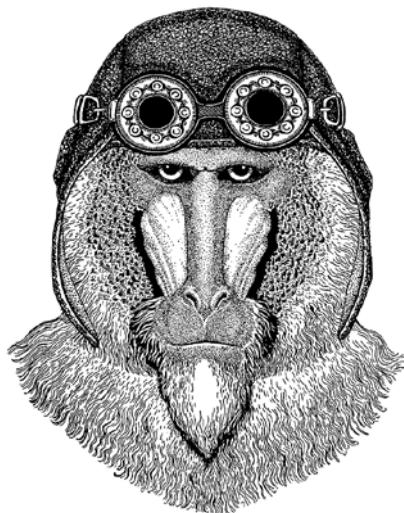


Намного лучше открыть новое окно, ввести в нем весь код, а затем сохранить и запустить его.



Чтобы создать новое окно, в котором вы будете писать код, откройте меню **File** и выберите команду **New**. После того как код будет введен в новом окне, вы можете сохранить его и выполнить сразу всю программу. Код будет выполнен в окне командной оболочки.

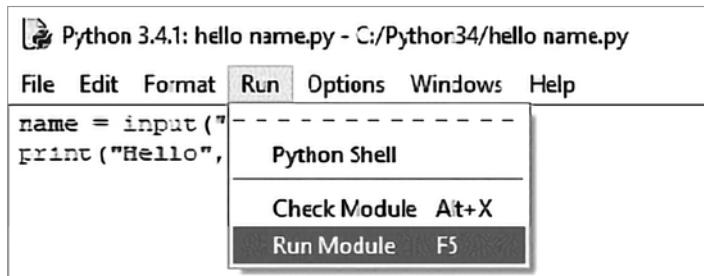
Возможен и другой вариант: программу Python можно написать в любом текстовом редакторе, а потом сохранить файл с расширением **.py** (без этого программа работать не будет). Такие программы можно запускать из командной строки, при этом необходимо указать полный путь к каталогу и имя файла.



## Запуск программы

Каждый раз, когда вы запускаете выполнение кода, программу необходимо сохранить. Это необходимо на тот случай, если с последнего запуска в него были внесены какие-либо изменения.

В этой версии Python для запуска программы можно открыть вкладку меню Run и выбрать команду Run Module. Также можно нажать клавишу F5. Если программа сохраняется впервые, Python предложит ввести имя файла и сохранить файл до того, как программа начнет выполняться.



## О чём важно помнить при написании программ

Python **различает регистр символов**, поэтому очень важно использовать правильный регистр; в противном случае ваш код работать не будет.

Текстовые значения должны заключаться в кавычки (""), но с числами этого делать не нужно.



При выборе имен **переменных** (то есть ячеек памяти, в которых вы хотите хранить данные) нельзя использовать специальные слова (**print**, **input** и т. д.), иначе ваш код работать не будет.

При сохранении файлов **не используйте** в качестве их имен зарезервированные слова, уже используемые Python (**print**, **input** и т. д.) В противном случае программа работать не будет, и вам придется переименовать файл.

Чтобы отредактировать программу, которая была сохранена и закрыта, щелкните правой кнопкой мыши по файлу и выберите команду **Edit in IDLE**. Если сделать двойной щелчок на файле, IDLE попробует его запустить, вместо того чтобы открыть для редактирования.

# Примеры кода

```
num1 = 93
```

Присваивает значение **переменной**. Если переменная еще не существует, она будет создана. Переменная представляет собой ячейку памяти для хранения значения (в данном случае переменная называется «`num1`», и в ней хранится значение 93.) Значение, хранящееся в переменной, может изменяться во время выполнения программы. Переменной можно присвоить практически любое имя (кроме зарезервированных слов Python — `print`, `save` и т. д.); это имя должно начинаться с буквы (но не с цифры или знака) и не может содержать пробелов.



```
answer = num1 + num2
```

Складывает `num1` и `num2` и сохраняет результат в переменной с именем `answer`.

```
answer = num1 - num2
```

Вычитает `num2` из `num1` и сохраняет результат в переменной с именем `answer`.

```
answer = num1 * num2
```

Умножает `num1` на `num2` и сохраняет результат в переменной с именем `answer`.

```
answer = num1 / num2
```

Делит `num1` на `num2` и сохраняет результат в переменной с именем `answer`.

```
answer = num1 // num2
```

Выполняет целочисленное деление (например, `9//4 = 2`) и сохраняет результат в переменной с именем `answer`.

```
print ("This is a message")
```

Выводит сообщение, заключенное в скобки. Так как выводимое значение содержит текст, оно заключается в кавычки, которые не будут отображаться при выводе. Если же вы хотите вывести числовое значение или содержимое переменной, кавычки не понадобятся.

```
print ("First line\nSecond line")
```

Последовательность "`\n`" обозначает разрыв строки.

```
print ("The answer is", answer)
```

Выводит сообщение "The answer is" и значение переменной `answer`.

```
textValue = input("Enter a text value: ")
```

Выводит сообщение "Enter a text value: " и сохраняет значение, введенное пользователем, в переменной с именем `textValue`. Пробел после двоеточия нужен для того, чтобы вводимые пользователем данные отделялись от сообщения. Без этого все символы будут сцеплены в одну длинную строку, а это некрасиво.

```
numValue = int(input("Enter a number: "))
```

Выводит сообщение "Enter a number: " и сохраняет введенное значение в виде целого числа в переменной `numValue`. Целые числа можно использовать в вычислениях, а переменные, хранимые в текстовом виде, — нельзя.



# Задачи

**001**

Предложите пользователю ввести его имя и выведите приветственное сообщение.

**Hello [имя].**

**002**

Предложите пользователю ввести его имя и фамилию, после чего выведите приветственное сообщение.

**Hello [имя] [фамилия].**

**003**

Напишите код, который выводит вопрос: «What do you call a bear with no teeth?», а в следующей строке выводит ответ: «A gummy bear!» Попробуйте обойтись одной строкой кода.

**004**

Предложите пользователю ввести два числа. Сложите эти числа и выведите результат в виде

**The total is [результат].**

**005**

Предложите пользователю ввести три числа. Сложите первые два числа, затем умножьте сумму на третье число. Выведите результат в виде

**The answer is [результат].**

**006**

Спросите, сколько кусков пиццы было у пользователя и сколько кусков он съел. Вычислите, сколько кусков пиццы у него осталось, и выведите результат в форме, удобной для пользователя.

**007**

Предложите пользователю ввести его имя и возраст. Увеличьте возраст на 1 и выведите сообщение:

**[имя] next birthday you will be [новый возраст].**

**008**

Предложите пользователю ввести общую сумму счета, а затем запросите общее количество участников обеда. Разделите сумму счета на количество участников и выведите сумму, которую должен заплатить каждый участник.

**009**

Напишите программу, которая предлагает ввести промежуток времени в днях, а потом выводит количество часов, минут и секунд в этом промежутке.

**010**

В одном килограмме 2,204 фунта. Предложите пользователю ввести вес в килограммах и переведите его в фунты.

**011**

Предложите пользователю ввести число больше 100, а затем число меньше 10. Сообщите, сколько раз меньшее число помещается в большем, в удобном формате.

Продолжайте,  
у вас  
отлично  
получается.



# Ответы

**001**

```
firstname = input("Please enter your first name: ")
print ("Hello ", firstname)
```

**002**

```
firstname = input("Please enter your first name: ")
surname = input("Please enter your surname: ")
print ("Hello ", firstname, surname)
```

**003**

```
print("What do you call a bear with no teeth?\nA gummy bear!")
```

**004**

```
num1 = int(input("Please enter your first number: "))
num2 = int(input("Please enter your second number: "))
answer = num1 + num2
print("The answer is ", answer)
```

**005**

```
num1 = int(input("Please enter your first number: "))
num2 = int(input("Please enter your second number: "))
num3 = int(input("Please enter your third number: "))
answer = (num1 + num2) * num3
print("The answer is ", answer)
```

**006**

```
startNum = int(input("Enter the number of slices of pizza you started with: "))
endNum = int(input("How many slices have you eaten? "))
slicesLeft = startNum - endNum
print("You have ", slicesLeft, " slices remaining")
```

**007**

```
name = input("What is your name? ")
age = int(input("How old are you? "))
newAge = age + 1
print(name, " next birthday you will be ", newAge)
```

**008**

```
bill = int(input("What is the total cost of the bill? "))
people = int(input("How many people are there? "))
each = bill / people
print("Each person should pay £", each)
```

**009**

```
days = int(input("Enter the number of days: "))
hours = days * 24
minutes = hours * 60
seconds = minutes * 60
print("In ", days, " days there are...")
print(hours, " hours")
print(minutes, " minutes")
print(seconds, " seconds")
```

**010**

```
kilo = int(input("Enter the number of kilos: "))
pound = kilo * 2.204
print("That is ", pound, " pounds")
```

**011**

```
larger = int(input("Enter a number of 100: "))
smaller = int(input("Enter a number under 10: "))
answer = larger // smaller
print(smaller, " goes into ", larger, answer, " times")
```

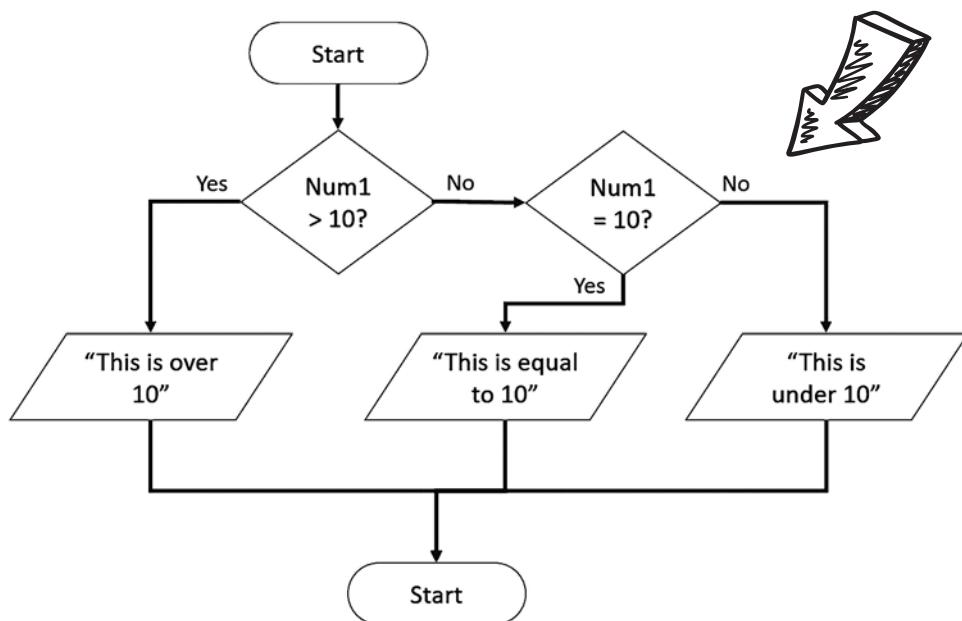


Удалось ли вам  
справиться с задачами?  
Не забывайте: те навыки,  
которыми вы овладеете  
сейчас, пригодятся  
вам потом.

# Инструкция if

## Объяснение

**Инструкция if** позволяет вашей программе принимать решения и изменять тот путь, по которому происходит передача управления в вашей программе.



А вот как инструкция **if** с этой блок-схемы будет выглядеть на языке Python:

```

if num1 > 10:
    print ("This is over 10")
elif num1 == 10:
    print ("This is equal to 10")
else:
    print ("This is under 10")
  
```



# Отступы в коде

Отступы играют очень важную роль в Python: они наглядно отделяют строки, зависящие от других, как видно из примера на предыдущей странице. Для создания отступов в тексте можно использовать клавишу Tab или нажать «пробел» четыре раза. Отступы удаляются клавишей Backspace.

Первая строка инструкции `if` проверяет условие. Если условие выполняется (то есть первое условие истинно), то выполняются строки кода, расположенные непосредственно под ним. Если оно не выполняется (то есть первое условие ложно), будет проверено второе условие, если оно есть, и т. д. Ниже приведены примеры разных операторов сравнения и логических операторов, которые могут использоваться в условиях инструкции `if`.

## Операторы сравнения

Оператор	Описание
<code>==</code>	Равно
<code>!=</code>	Не равно
<code>&gt;</code>	Больше
<code>&lt;</code>	Меньше
<code>&gt;=</code>	Больше или равно
<code>&lt;=</code>	Меньше или равно

## Логические операторы

Оператор	Описание
<code>and</code>	Должны выполняться оба условия
<code>or</code>	Должно выполняться хотя бы одно из условий



## Примеры кода

Внимание: в следующих примерах **num** – переменная, которая была введена пользователем и сохранена как целое число.



```
if num > 10:
    print("This is over 10")
else:
    print("This is not over 10")
```

Если значение **num** больше 10, выводится сообщение «This is over 10»; в противном случае выводится сообщение «This is under 10».

```
if num > 10:
    print("This is over 10")
elif num == 10:
    print("This is equal to 10")
else:
    print("This is under 10")
```

Если значение **num** больше 10, будет выведено сообщение «This is over 10»; в противном случае проверяется следующее условие. Если значение **num** равно 10, выводится сообщение «This is equal to 10». Если же ни одно из первых двух условий не выполняется, выводится сообщение «This is under 10».



```
if num >= 10:
    if num <= 20:
        print("This is between 10 and 20")
    else:
        print("This is over 20")
else:
    print("This is under 10")
```

Если значение **num** больше или равно 10, выполняется другая команда **if**, которая проверяет, что значение **num** меньше или равно 20. В этом случае выводится сообщение «This is between 10 and 20». Если же значение **num** не меньше или равно 20, выводится сообщение «This is over 20». Если **num** не больше 10, выводится сообщение «This is under 10».



```
text = str.lower(text)
```

Текст преобразуется к нижнему регистру. Так как Python различает регистр символов, введенный текст обычно преобразуется пользователем в нижний, чтобы его было удобнее проверять.

```
num = int(input("Enter a number between 10 and 20: "))
if num >= 10 and num <= 20:
    print("Thank you")
else:
    print("Out of range")
```

Оператор **and** используется для проверки нескольких условий в инструкции **if**. Для вывода сообщения "**Thank you**" должны выполняться оба условия.

```
num = int(input("Enter an EVEN number between 1 and 5: "))
if num == 2 or num == 4:
    print("Thank you")
else:
    print("Incorrect")
```

Оператор **or** используется для проверки нескольких условий в инструкции **if**. Чтобы было выведено сообщение "**Thank you**", достаточно, если будет выполняться всего одно условие.



# Задачи

**012**

Предложите пользователю ввести два числа. Если первое число больше второго, сначала выведите второе число, а потом первое. В противном случае выведите сначала первое число, а потом второе.

**013**

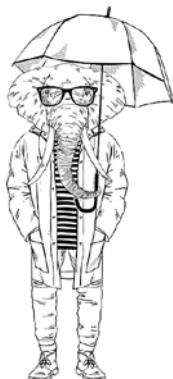
Предложите пользователю ввести число, меньшее 20. Если введенное число больше или равно 20, выведите сообщение «Too high»; в противном случае выведите сообщение «Thank you».

**014**

Предложите пользователю ввести число от 10 до 20 (включительно). Если будет введено число из этого диапазона, выведите сообщение «Thank you»; в противном случае выведите сообщение «Incorrect answer».

**015**

Предложите пользователю ввести любимый цвет. Если он введет «red», «RED» или «Red», выведите сообщение «I like red too». В противном случае выведите сообщение «I don't like [colour], I prefer red».

**016**

Спросите пользователя, идет ли дождь. Преобразуйте его ответ к нижнему регистру. Если пользователь ответит «yes», спросите, ветрено ли на улице. Если пользователь ответит «yes» и на второй вопрос, выведите сообщение «It is too windy for an umbrella»; в противном случае выведите сообщение «Take an umbrella». Если же пользователь не дал положительного ответа на первый вопрос, выведите сообщение «Enjoy your day».

**017**

Предложите пользователю ввести его возраст. Если введенное значение равно 18 и более, выведите сообщение «You can vote»; если 17 — сообщение «You can learn to drive»; если 16 — сообщение «You can buy a lottery ticket». Если значение меньше 16, выведите сообщение «You can go Trick-or-Treating».

**018**

Предложите пользователю ввести число. Если оно меньше 10, выведите сообщение «Too low»; если число лежит в диапазоне от 10 до 20 — сообщение «Correct». В остальных случаях выведите сообщение «Too high».

**019**

Предложите пользователю ввести значение 1, 2 или 3. Если введено значение 1, выведите сообщение «Thank you»; если 2 — сообщение «Well done»; если 3 — сообщение «Correct». Если введено любое другое значение, выведите сообщение «Error message».

# Ответы

## 012

```
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))
if num1 > num2:
    print(num2, num1)
else:
    print(num1, num2)
```

## 013

```
num = int(input("Enter a value less than 20: "))
if num >= 20:
    print("Too high")
else:
    print("Thank you")
```

## 014

```
num = int(input("Enter a value between 10 and 20: "))
if num >= 10 and num <= 20:
    print("Thank you")
else:
    print("Incorrect answer")
```

## 015

```
colour = input("Type in your favorite colour: ")
if colour == "red" or colour == "RED" or colour == "Red":
    print("I like red too")
else:
    print("I don't like that colour, I prefer red")
```

## 016

```
raining = input("Is it raining? ")
raining = str.lower(raining)
if raining == "yes":
    windy = input("Is it windy? ")
    windy = str.lower(windy)
    if windy == "yes":
        print("It is too windy for an umbrella")
    else:
        print("Take an umbrella")
else:
    print("Enjoy your day")
```

**017**

```
age = int(input("Enter a number: "))
if age >= 18:
    print("You can vote")
elif age == 17:
    print("You can learn to drive")
elif age == 16:
    print("You can buy a lottery ticket")
else:
    print("You can go Trick-or-Treating")
```

**018**

```
num = int(input("Enter a number: "))
if num < 10:
    print("Too low")
elif num >= 10 and num <= 20:
    print("Correct")
else:
    print("Too high")
```

**019**

```
num = input("Enter 1, 2 or 3: ")
if num == "1":
    print("Thank you")
elif num == "2":
    print("Well done")
elif num == "3":
    print("Correct")
else:
    print("Error message")
```

# Строки

## Объяснение

**Строка** – термин для обозначения текста. Чтобы определить блок кода как строку, необходимо заключить его в двойные кавычки ("") или в одинарные кавычки (''). Неважно, какой из способов вы выберете, главное – действовать последовательно.

При включении некоторых символов в строки необходимо действовать особенно осторожно. Прежде всего речь идет о таких:

" " \

Дело в том, что эти символы имеют особый смысл в Python, и при использовании их в строках может возникнуть путаница.

Если вы хотите использовать один из этих символов в строке, поставьте перед ним символ \; тогда Python будет знать, что специальный смысл этого символа нужно игнорировать, и будет рассматривать его как обычный текст, который нужно вывести на экран.

Символ	Как записывать для включения в строку Python
"	\"
'	\'
\	\\



## Строки и числа как переменные

Если вы определяете переменную как строку (даже если в ней хранятся только представления чисел в текстовом виде), такую переменную нельзя будет позднее использовать в математических вычислениях. Для этого нужно преобразовать строку в число перед использованием.

```
num = input("Enter a number: ")
total = num + 10
print(total)
```

В этом примере я запрашиваю число, но не определяю его как числовое значение, и при попытке выполнить программу произойдет ошибка:

```
Enter a number: 45
Traceback (most recent call last):
  File "C:/Python34/CHALLENGES/String/example.py", line 2, in <module>
    total = num + 10
TypeError: Can't convert 'int' object to string implicitly
>>>
```

Хотя это сообщение об ошибке выглядит устрашающее, оно просто объясняет, что строка **total = num + 10** не работает, потому что переменная **num** содержит строковые данные.



Проблему можно решить двумя способами. Переменная либо определяется как число при исходном создании переменной:

```
num = int(input("Enter a number: "))
```

либо преобразуется в число уже после создания:

```
num = int(num)
```



То же самое может происходить со строками.

```
name = input("Enter a name: ")
num = int(input("Enter a number: "))
ID = name + num
print(ID)
```

В этой программе пользователю предлагается ввести имя и номер. Предполагается, что эти данные будут соединены и образуют идентификатор; при использовании со строками знак `+` выполняет операцию **конкатенации**. При выполнении этого кода вы получите почти такое же сообщение об ошибке, как и прежде:

```
Enter a name: Bob
Enter a number: 23
Traceback (most recent call last):
  File "C:/Python34/CHALLENGES/String/example.py", line 3, in <module>
    ID = name + num
TypeError: Can't convert 'int' object to string implicitly
>>>
```

Чтобы обойти эту проблему, либо изначально не определяйте переменную как число, либо преобразуйте ее в строку позднее:

```
num = str(num)
```



## Внутренние разрывы строк

Если вы хотите определить строку, которая состоит из нескольких внутренних (логических) строк, либо используйте символ новой строки (`\n`), либо заключите всю строку в тройные кавычки (этот способ сохраняет форматирование текста).

```
address="""123 Long Lane
Oldtown
AB1 23CD"""
print(address)
```



## Примеры кода

Примечание: в следующих примерах слова `word`, `phrase`, `name`, `firstname` и `surname` являются именами переменных.

```
len(word)
```

Определяет длину переменной `word`.

```
print(word.capitalize())
```

Выводит содержимое переменной так, чтобы каждое слово начиналось с буквы верхнего регистра, а все остальные буквы были в нижнем регистре.

```
name = firstname+surname
```

Соединяет `firstname` и `surname`, не разделяя их пробелами. Эта операция называется конкатенацией.

```
text = " This is some text. "
print(text.strip(" "))
```

Удаляет лишние символы (в данном случае пробелы) в начале и в конце строки.

```
print ("Hello world"[7:10])
```

Каждому символу (в том числе и пробелам) соответствует индекс, представляющий его позицию в строке. В Python нумерация индексов начинается с 0, а не с 1.

0	1	2	3	4	5	6	7	8	9	10
H	e	l	l	o		w	o	r	l	d

Таким образом, в данном примере будут выведены значения символов в позициях 7, 8 и 9, то есть «ол».

```
word.upper()
```

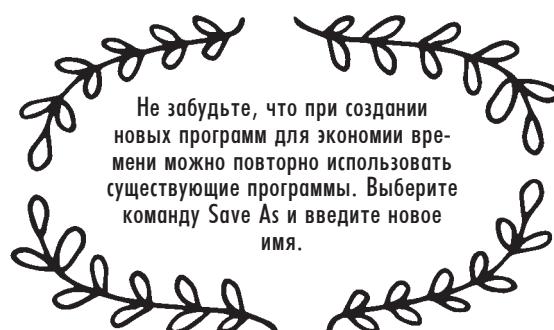
Преобразует строку к верхнему регистру.

```
word.lower()
```

Преобразует строку к нижнему регистру.

```
phrase.title()
```

Изменяет `phrase` так, чтобы каждое слово начиналось с буквы верхнего регистра, а все остальные буквы были в нижнем регистре (так называемый «титульный регистр»).



# Задачи

**020**

Предложите пользователю ввести имя. Выведите длину имени.

**023**

Предложите пользователю ввести первую строку какого-нибудь стихотворения, выведите длину строки. Запросите начальную и конечную позицию и выведите только эту часть строки (не забудьте, что в Python нумерация индексов начинается с 0, а не с 1).

**021****021**

Предложите пользователю ввести сначала имя, а затем фамилию. Соедините их, разделив пробелом, после чего выведите полное имя и его длину.

**022**

Предложите пользователю ввести имя и фамилию в нижнем регистре. Преобразуйте строки к титульному регистру и соедините их. Выведите полученный результат.

**024**

Предложите пользователю ввести имя. Выведите его в верхнем регистре.

**025**

Предложите пользователю ввести имя. Если длина имени меньше 5 символов, предложите ввести фамилию, соедините их (без пробела) и выведите полное имя в верхнем регистре. Если длина имени составляет 5 и более символов, выведите имя в нижнем регистре.

Не забывайте, что вы всегда можете сделать шаг назад и припомнить то, что вы узнали ранее. А вы узнали уже достаточно много.

**026**

В шифре «поросячья латынь» начальная согласная буква слова перемещается в конец слова, и к ней добавляется суффикс «ay». Если слово начинается с гласной, к нему просто добавляется суффикс «way». Например, pig превращается в igray, banana — в ananabay, aardvark — в aardvarkway. Напишите программу, которая предлагает пользователю ввести слово и преобразует его в «поросячью латынь». Проследите за тем, чтобы новое слово выводилось в нижнем регистре.



# Ответы

**020**

```
name = input("Enter your first name: ")
length = len(name)
print(length)
```

**021**

```
firstname = input("Enter your first name: ")
surname = input("Enter your surname: ")
name = firstname + " " + surname
length = len(name)
print(name)
print(length)
```

**022**

```
firstname = input("Enter your first name in lowercase: ")
surname = input("Enter your surname in lowercase: ")
firstname = firstname.title()
surname = surname.title()
name = firstname + " " + surname
print(name)
```

**023**

```
phrase = input("Enter the first line of a nursery rhyme: ")
length = len(phrase)
print("This has ", length, " letters in it")
start = int(input("Enter a starting number: "))
end = int(input("Enter an end number: "))
part = (phrase[start:end])
print(part)
```

**024**

```
word = input("Enter a word: ")
word = word.upper()
print(word)
```

**025**

```
name = input("Enter your first name: ")
if len(name) < 5:
    surname = input("Enter your surname: ")
    name = name+surname
    print(name.upper())
else:
    print(name.lower())
```

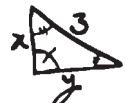
**026**

```
word = input("Please enter a word: ")
first = word[0]
length = len(word)
rest = word[1:length]
if first != "a" and first != "e" and first != "i" and first != "o" and first != "u":
    newword = rest + first + "ay"
else:
    newword = word + "way"
print(newword.lower())
```

# Математические операции

## Объяснение

В Python при обработке данных можно применять различные математические функции, однако они доступны только в том случае, когда данные интерпретируются как целое число или как число с плавающей точкой (то есть число с дробной частью). Если данные хранятся в виде строки, даже если эта строка содержит только цифровые символы, Python не сможет выполнять с ней вычисления (подробности приведены на с. 29).



## Примеры кода

Примечание: чтобы использовать некоторые математические функции (`math.sqrt(num)` и `math.pi`), необходимо импортировать библиотеку `math` в самом начале программы. Для этого достаточно ввести команду `import math` в первой строке программы.

```
**
```

Выполняет возведение в степень (например, 102 записывается в виде `10**2`).

```
math.sqrt(num)
```

Вычисляет квадратный корень из числа. Чтобы использовать эту функцию, необходимо включить строку `import math` в начало программы.

```
num=float(input("Enter number: "))
```

Позволяет использовать в вычислениях числа с плавающей точкой (то есть числа, имеющие как целую, так и дробную часть).

```
math.pi
```

Предоставляет значение числа «пи» ( $\pi$ ) с точностью до 15 знаков. Чтобы использовать эту функцию, необходимо включить строку `import math` в начало программы.

```
x // y
```

Выполняет целочисленное деление (например, выражение `15//2` дает результат 7).

```
x % y
```

Вычисляет остаток (например, выражение `15%2` дает результат 1).

```
print(round(num, 2))
```

Выводит число, округленное до двух знаков в дробной части.



# Задачи

**027**

Предложите пользователю ввести число с большим количеством знаков в дробной части. Умножьте это число на 2 и выведите ответ.

**030**

Выведите число «пи» ( $\pi$ ) с точностью до 5 знаков.



**032**

Предложите пользователю ввести радиус и высоту цилиндра. Вычислите его объем (площадь круга \* высота) и выведите его с точностью до трех знаков.



**028**

Измените программу из задачи 027 так, чтобы она выводила результат с точностью до двух знаков в дробной части.

**029**

Предложите пользователю ввести целое число больше 500. Вычислите квадратный корень из этого числа и выведите его с точностью до двух знаков в дробной части.

**031**

Предложите пользователю ввести радиус круга (расстояние от центра до внешней границы.) Вычислите площадь круга ( $\pi * \text{радиус}^2$ ).

**033**

Предложите пользователю ввести два числа. Используйте целочисленное деление, чтобы разделить первое число на второе; вычислите остаток и выведите ответ в виде, удобном для пользователя (например, если пользователь ввел 7 и 2, выведите строку вида «если разделить 7 на 2, получится 3 с остатком 1»).

**034**

Выведите следующее сообщение:

- 1) Square
- 2) Triangle

Enter a number:

Если пользователь вводит 1, программа запрашивает длину стороны квадрата и выводит его площадь. Если пользователь вводит 2, программа запрашивает длину стороны и высоту треугольника, проведенную к этой стороне, после чего выводит его площадь. Если пользователь вводит что-то другое, программа должна выдать подходящее сообщение об ошибке.

Вы начинаете мыслить как программист.



# Ответы

**027**

```
num = float(input("Enter a number with lots of decimal places: "))
print(num*2)
```

**028**

```
num = float(input("Enter a number with lots of decimal places: "))
answer = num*2
print(answer)
print (round(answer, 2))
```

**029**

```
import math
num = int(input("Enter a number over 500: "))
answer = math.sqrt(num)
print(round(answer, 2))
```

**030**

```
import math
print(round(math.pi,5))
```

**031**

```
import math
radius = int(input("Enter the radius of the circle: "))
area = math.pi * (radius**2)
print(area)
```

**032**

```
import math
radius = int(input("Enter the radius of the circle: "))
depth = int(input("Enter depth: "))
area = math.pi * (radius**2)
volume = area * depth
print(round(volume,3))
```

**033**

```
num1 = int(input("Enter a number: "))
num2 = int(input("Enter another number: "))
ans1 = num1 // num2
ans2 = num1 % num2
print(num1, " divided by ", num2, " is ", ans1, " with ", ans2, " remaining.")
```

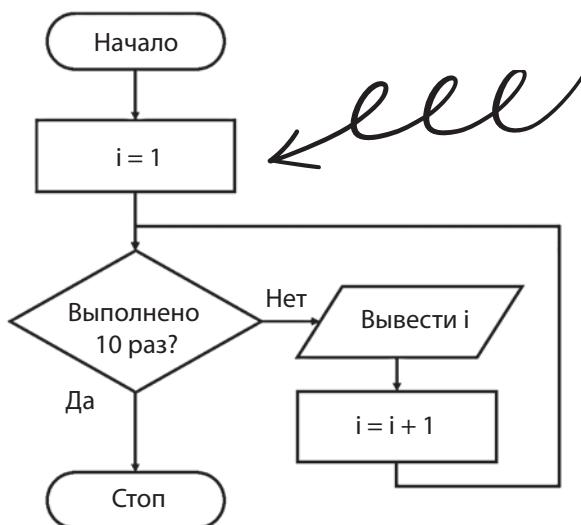
**034**

```
print("1) Square")
print("2) Triangle")
print()
menuselection = int(input("Enter a number: "))
if menuselection == 1:
    side = int(input("Enter the length of one side: "))
    area = side * side
    print("The area of your chosen shape is ", area)
elif menuselection == 2:
    base = int(input("Enter the length of the base: "))
    height = int(input("Enter the height of the triangle: "))
    area = (base * height) / 2
    print("The area of your chosen shape is ", area)
else:
    print("Incorrect option selected")
```

# Циклы for

## Объяснение

Цикл **for** позволяет выполнить фрагмент кода заданное количество раз. Иногда он называется **счетным циклом**, потому что количество выполнений цикла известно до его начала.



В данном случае цикл начинается с 1 и повторяется (выводит значение **i**), пока не достигнет 10, после чего останавливается. В коде Python цикл выглядит так:

```
for i in range(1,10):
    print(i)
```

В этом примере будет выведена последовательность чисел 1, 2, 3, 4, 5, 6, 7, 8 и 9.

**Добавившись до 10, цикл останавливается, поэтому 10 в выводе отсутствует.**

*Не забудьте снабдить отступами код в теле цикла for.*



## Примеры кода

Функция `range` часто используется в циклах `for`. При ее вызове указываются начало и конец диапазона. Функция также может включать приращение переменной цикла (например, 1, 5 или любое другое значение на ваше усмотрение).

```
for i in range(1, 10):  
    print(i)
```

В этом цикле переменная с именем `i` используется для управления количеством повторений цикла. Сначала `i` присваивается 1 (начальное значение функции `range`). При каждом повторении цикла переменная `i` увеличивается на 1 и выводится ее текущее значение. Цикл повторяется, пока переменная не достигнет 10 (как указано при вызове функции `range`), после чего цикл останавливается. Таким образом, цикл не будет выполнен в десятый раз, а вывод будет выглядеть так:

1, 2, 3, 4, 5, 6, 7, 8, 9.



```
for i in range(1, 10, 2):  
    print(i)
```

Функция `range` включает третье значение, которое определяет приращение `i` при каждом проходе цикла (в данном случае 2). В этом случае будут выведены следующие числа: 1, 3, 5, 7, 9.

```
for i in range(10, 1, -3):  
    print(i)
```

В этом диапазоне переменная `i` будет каждый раз уменьшаться на 3. Результат: 10, 7, 4.



Циклы – эффективный инструмент программирования, который очень часто используется в более сложных программах.

```
for i in word:  
    print(i)
```

Каждый символ в строке с именем `word` будет выведен в отдельной строке.



# Задачи

**035**

Предложите пользователю ввести имя. Выведите имя три раза.

**038**

Измените программу из упражнения 37 так, чтобы она также запрашивала число. Выведите имя (по одной букве в каждой строке) и повторите вывод равное введенному числу количества раз.

**041**

Предложите пользователю ввести имя и число. Если число меньше 10, программа должна вывести имя заданное количество раз; в противном случае она выводит сообщение «Too high» три раза.


**036**

Измените программу из упражнения 35 так, чтобы она предлагала пользователю ввести имя и число, а затем выводила имя заданное количество раз.

**037**

Предложите пользователю ввести имя. Выведите каждую букву имени в отдельной строке.

**039**

Предложите пользователю ввести число от 1 до 12. Выведите таблицу умножения для этого числа.

**040**

Предложите пользователю ввести число до 50. Проведите обратный отсчет от 50 до введенного числа. Проследите за тем, чтобы введенное число было включено в вывод.

**042**

Присвойте переменной с именем **total** значение 0. Предложите пользователю ввести пять чисел, и после каждого ввода спрашивайте, хочет ли он включить это число в суммирование. Если ответ будет положительным, прибавьте введенное число к **total**. Если же ответ будет отрицательным, число к **total** не прибавляется. После ввода всех пяти чисел выведите значение **total**.

**043**

Спросите у пользователя, в каком направлении он хочет вести отсчет (в прямом или обратном). Если выбран прямой отсчет, запросите число и проведите отсчет от 1 до введенного числа. Если выбран обратный отсчет, запросите число меньше 20, а затем проведите обратный отсчет от 20 до заданного числа. Если введено что-то другое, выведите сообщение «I don't understand».

**044**

Спросите у пользователя, скольких людей он хочет пригласить на вечеринку. Если будет введено число меньше 10, запросите имена и после каждого имени выведите строку «[имя] has been invited». Если введенное число больше или равно 10, выведите сообщение «Too many people».

# Ответы

## 035

```
name = input("Type in your name: ")  
for i in range(0, 3):  
    print(name)
```

## 036

```
name = input("Type in your name: ")  
number = int(input("Enter a number: "))  
for i in range(0, number):  
    print(name)
```

## 037

```
name = input("Enter your name: ")  
for i in name:  
    print(i)
```

## 038

```
num = int(input("Enter a number: "))  
name = input("Enter your name: ")  
for x in range(0, num):  
    for i in name:  
        print(i)
```

## 039

```
num = int(input("Enter a number between 1 and 12: "))  
for i in range(1, 13):  
    answer = i * num  
    print(i, "x", num, "=", answer)
```

## 040

```
num = int(input("Enter a number below 50: "))  
for i in range(50, num-1, -1):  
    print(i)
```

## 041

```
name = input("Enter your name: ")  
num = int(input("Enter a number: "))  
if num < 10:  
    for i in range(0, num):  
        print(name)  
else:  
    for i in range(0, 3):  
        print("Too high")
```

**042**

```
total = 0
for i in range(0, 5):
    num = int(input("Enter a number: "))
    ans = input("Do you want this number included? (y/n) ")
    if ans == "y":
        total = total + num
print(total)
```

**043**

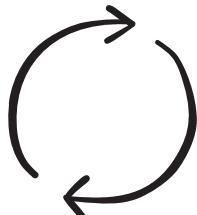
```
direction = input("Do you want to count up and down? (u/d) ")
if direction == "u":
    num = int(input("What is the top number? "))
    for i in range(1, num + 1):
        print(i)
elif direction == "d":
    num = int(input("Enter a number below 20: "))
    for i in range(20, num + 1, -1):
        print(i)
else:
    print("I don't understand")
```

**044**

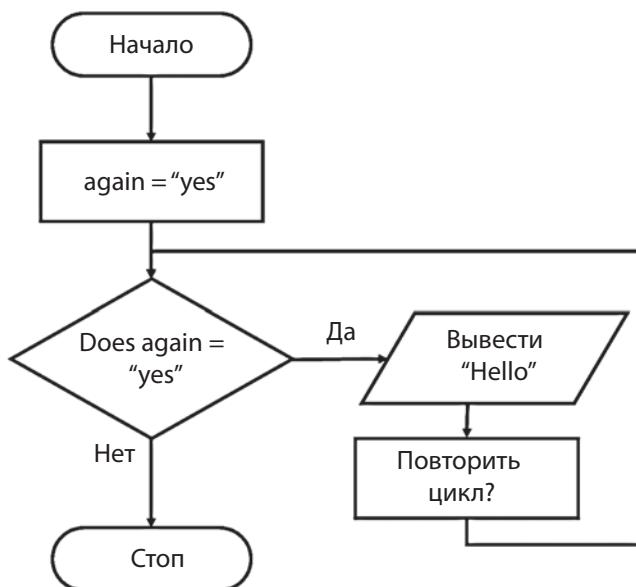
```
num = int(input("How many friends do you want to invite to the party? "))
if num < 10:
    for i in range(0, num):
        name = input("Enter a name: ")
        print(name, " has been invited")
else:
    print("Too many people")
```

# Цикл while

## Объяснение



**Цикл while** позволяет выполнить код неизвестное количество раз, пока выполняется некоторое условие. Цикл может быть выполнен 100 раз, единожды или вообще ни разу. В цикле **while** условие проверяется до выполнения кода; это означает, что если при первой проверке условие не выполняется, цикл будет полностью пропущен. А значит, вы должны проследить за тем, чтобы перед запуском цикла были заданы правильные условия.



В Python код для этой блок-схемы выглядит так:

```
again = "yes"
while again == "yes"
    print ("Hello")
    again=input("Do you want to loop again? ")
```

Код будет повторяться до тех пор, пока пользователь не введет любой другой ответ, кроме «yes».

## Пример кода



```
total = 0
while total < 100:
    num = int(input("Enter a number: "))
    total = total + num
print("The total is ", total)
```

Эта программа создает переменную с именем **total** и сохраняет в ней значение 0. Она предлагает пользователю ввести число и прибавляет его к **total**. Цикл продолжает выполняться, пока значение **total** остается ниже 100. Как только **total** достигнет 100 и более, цикл перестанет выполнять и будет выведено значение **total**.



### Операторы сравнения

Оператор	Описание
<b>==</b>	Равно
<b>!=</b>	Не равно
<b>&gt;</b>	Больше
<b>&lt;</b>	Меньше
<b>&gt;=</b>	Больше или равно
<b>&lt;=</b>	Меньше или равно

### Логические операторы

Оператор	Описание
<b>and</b>	Должны выполняться оба условия
<b>or</b>	Должно выполняться хотя бы одно из условий

**Не забывайте: текстовые значения должны заключаться в кавычки, а числовые значения – нет.**



# Задачи

**045**

Присвойте **total** значение 0. Пока значение **total** равно 50 или менее, предложите пользователю ввести число. Прибавьте это число к **total** и выведите сообщение «The total is... [total]». Цикл должен остановиться, когда значение **total** превысит 50.

**047**

Предложите пользователю ввести сначала одно число, а затем другое. Сложите два числа и спросите, хочет ли он прибавить еще одно. Если он введет «у», предложите ввести еще одно число; это продолжается до тех пор, пока пользователь не введет ответ «у». После того как цикл остановится, выведите сумму.

**049**

Создайте переменную с именем **comprnum** и присвойте ей значение 50. Предложите пользователю ввести число. Пока предположение не совпадает со значением **comprnum**, сообщите, больше оно или меньше **comprnum**, и предложите ввести другое число. Если введенное значение совпадет с **comprnum**, выведите сообщение «Well done, you took [попытки] attempts».



**046**

Предложите пользователю ввести число. Продолжайте запрашивать число, пока введенное число не будет больше 5, после чего выведите сообщение «The last number you entered was a [число]» и остановите программу.

**048**

Предложите пользователю ввести имя человека, которого пользователь хочет пригласить на вечеринку. После этого выведите сообщение «[имя] has been invited» и увеличьте счетчик на 1. Спросите, хочет ли пользователь пригласить кого-то еще. Продолжайте запрашивать имена, пока пользователь не ответит отрицательно, и выведите количество приглашенных.



**050**

Предложите пользователю ввести число от 10 до 20. Если введенное значение меньше 10, выведите сообщение «Too low» и предложите повторить попытку. Если введенное значение больше 20, выведите сообщение «Too high» и предложите повторить попытку. Повторяйте до тех пор, пока не будет введено значение из диапазона от 10 до 20, после чего выведите сообщение «Thank you».

**051**

Выведите строки «There are [счетчик] green bottles hanging on the wall, [счетчик] green bottles hanging on the wall, and if 1 green bottle should accidentally fall». Затем выведите вопрос: «how many green bottles will be hanging on the wall?». Если пользователь ответит правильно, выведите сообщение «There will be [счетчик] green bottles hanging on the wall». Если пользователь ответит неправильно, выведите сообщение «No, try again», пока не будет дан правильный ответ. Когда счетчик уменьшится до 0, выведите сообщение «There are no more green bottles hanging on the wall».

# Ответы

## 045

```
total = 0
while total <= 50:
    num = int(input("Enter a number: "))
    total = total + num
    print("The total is...", total)
```

## 046

```
num = 0
while num <= 5:
    num = int(input("Enter a number: "))
print("The last number you entered was a ", num)
```

## 047

```
num1 = int(input("Enter a number: "))
total = num1
again = "y"
while again == "y":
    num2 = int(input("Enter another number: "))
    total = total + num2
    again = input("Do you want to add another number? (y/n) ")
print("The total is ", total)
```

## 048

```
again = "y"
count = 0
while again == "y":
    name = input("Enter a name of somebody you want to invite to the party: ")
    print(name, " has been invited")
    count = count + 1
    again = input("Do you want to invite somebody else? (y/n) ")
print("You have ", count, " people coming to your party")
```

## 049

```
compnum = 50
guess = int(input("Can you guess the number I am thinking of? "))
count = 1
while guess != compnum:
    if guess < compnum:
        print("Too low")
    else:
        print("Too high")
    count = count + 1
    guess = int(input("Have another guess: "))
print("Well done, you took ", count, " attempts")
```

**050**

```
num = int(input("Enter a number between 10 and 20: "))
while num < 10 or num > 20:
    if num < 10:
        print("Too low")
    else:
        print("Too high")
    num = int(input("Try again: "))
print("Thank you")
```

**051**

```
num = 10
while num > 0:
    print("There are ", num, " green bottles hanging on the wall.")
    print( num, " green bottles hanging on the wall.")
    print("And if 1 green bottle should accidentally fall,")
    num = num - 1
    answer = int(input("How many green bottles will be hanging on the wall? "))
    if answer == num:
        print("There will be ", num, " green bottles hanging on the wall.")
    else:
        while answer != num:
            answer = int(input("No, try again: "))
print("There are no more green bottles hanging on the wall.")
```

# Случайные числа

## Объяснение

Python может генерировать **случайные** значения. На самом деле эти значения не являются полностью случайными – такая задача не по силам компьютеру; вместо этого используется невероятно сложный алгоритм, результаты которого практически невозможно точно спрогнозировать, так что фактически функция ведет себя как генератор случайных чисел.

Нас интересуют две разновидности случайных значений:

- случайные числа из заданного диапазона;
- случайный выбор из диапазона введенных элементов.

Чтобы реализовать любую из этих разновидностей, необходимо импортировать библиотеку `random`. Для этого достаточно ввести команду `import random` в первой строке программы.



## Примеры кода

```
import random
```

Команда должна находиться в начале программы, иначе функции **random** работать не будут.

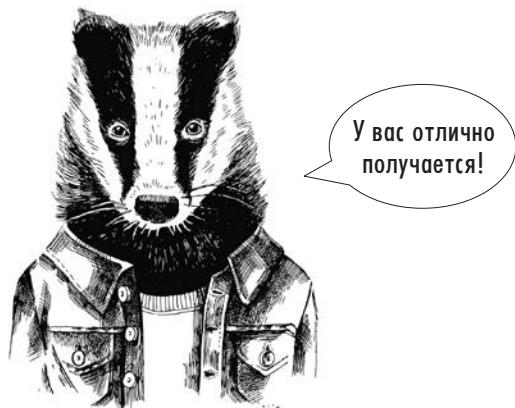
```
num = random.random()
```

Выбирает случайное число с плавающей точкой в диапазоне от 0 до 1 и сохраняет его в переменной с именем **num**. Если потребуется получить большее число, умножьте его на масштабный коэффициент, как показано ниже:

```
import random
num = random.random()
num = num * 100
print(num)
```

```
num = random.randint(0, 9)
```

Выбирает случайное целое число в диапазоне от 0 до 9 (включительно).



```
num1 = random.randint(0, 1000)
num2 = random.randint(0, 1000)
newrand = num1 / num2
print(newrand)
```

Создает случайное число с плавающей точкой, для чего генерирует два случайных целых числа в больших диапазонах (в данном случае от 0 до 1000) и делит одно на другое.

```
num = random.randrange(0, 100, 5)
```

Выбирает случайное число в диапазоне от 0 до 100 (включительно) с шагом 5 — иначе говоря, выбираются только числа из ряда 0, 5, 10, 15, 20 и т. д.

```
colour = random.choice(["red", "black", "green"])
```

Выбирает случайное значение из вариантов «red», «black» или «green» и сохраняет его в переменной **colour**. Не забывайте: текстовые значения должны заключаться в кавычки, а числовые значения — нет.



# Задачи

**052**

Вывести случайное число в диапазоне от 1 до 100 включительно.

**053**

Вывести случайное название фрукта из списка, содержащего пять названий.

**054**

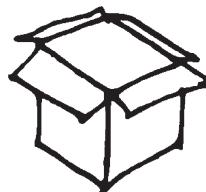
Случайным образом выберите «орел» или «решку» ("h" или "t"). Предложите пользователю угадать ваш выбор. Если ваш выбор совпадает со случайно выбранным значением, выведите сообщение «You win»; в противном случае выведите сообщение «Bad luck». В конце сообщите пользователю, какое значение было загадано — «орел» или «решка».

**055**

Выберите случайное число в диапазоне от 1 до 5. Предложите пользователю выбрать число. Если он угадал, выведите сообщение «Well done»; в противном случае сообщите, что его число больше или меньше вашего, и предложите выбрать другое число. Если со второго раза пользователь угадал, выведите сообщение «Correct», а если нет — сообщение «You lose».

**056**

Выберите случайное целое число в диапазоне от 1 до 10. Предложите пользователю ввести число и проверьте, совпадает ли оно с загаданным. Продолжайте запрашивать числа до тех пор, пока пользователь не введет случайно выбранное число.

**058**

Напишите математическую игру, в которой пользователь должен ответить на пять вопросов. Каждый вопрос строится из двух случайно сгенерированных целых чисел (например, [num1] + [num2]). Предложите пользователю ввести ответ. Если пользователь ввел правильный ответ, добавьте одно очко в его пользу. В конце игры сообщите пользователю количество правильных ответов.

**057**

Измените программу 056 так, чтобы перед повторным предположением она сообщала пользователю, является ли его предположение больше или меньше загаданного числа.

**059**

Выведите названия пяти цветов, случайным образом выберите один и предложите сделать то же пользователю. Если пользователь выберет тот же цвет, который выбрала программа, выведите сообщение «Well done»; в противном случае выведите ответ, в котором скрывается намек на правильный цвет. Предложите пользователю повторить попытку; если пользователь и на этот раз не угадает, снова выведите ту же подсказку и предложите выбрать цвет (и так далее, пока пользователь не выдаст правильный ответ).

# Ответы

## 052

```
import random
num = random.randint(1, 100)
print(num)
```

## 053

```
import random
fruit = random.choice(["apple", "orange", "grape", "banana", "strawberry"])
print(fruit)
```

## 054

```
import random
coin = random.choice(["h", "t"])
guess = input("Enter (h)eads or (t)ails: ")
if guess == coin:
    print("You win")
else:
    print("Bad luck")
if coin == "h":
    print("It was heads")
else:
    print("It was tails")
```

## 055

```
import random
num = random.randint(1, 5)
guess = int(input("Enter a number: "))
if guess == num:
    print("Well done")
elif guess > num:
    print("Too high")
    guess = int(input("Guess again: "))
    if guess == num:
        print("Correct")
    else:
        print("You lose")
elif guess < num:
    print("Too low")
    guess = int(input("Guess again: "))
    if guess == num:
        print("Correct")
    else:
        print("You lose")
```

**056**

```
import random
num = random.randint(1, 10)
correct = False
while correct == False:
    guess = int(input("Enter a number: "))
    if guess == num:
        correct = True
```

**057**

```
import random
num = random.randint(1, 10)
correct = False
while correct == False:
    guess = int(input("Enter a number: "))
    if guess == num:
        correct = True
    elif guess > num:
        print("Too high")
    else:
        print("Too low")
```

**058**

```
import random
score = 0
for i in range(1, 6):
    num1 = random.randint(1, 50)
    num2 = random.randint(1, 50)
    correct = num1 + num2
    print(num1, "+", num2, "= ")
    answer = int(input("Your answer: "))
    print()
    if answer == correct:
        score = score + 1
print("You scored ", score, " out of 5")
```

**059**

```
import random

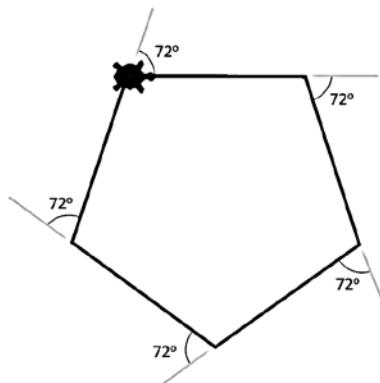
colour = random.choice(["red", "blue", "green", "white", "pink"])
print("Select from red, blue, green, white or pink")
tryagain = True
while tryagain == True:
    theirchoice = input("Enter a colour: ")
    theirchoice = theirchoice.lower()
    if colour == theirchoice:
        print("Well done")
        tryagain = False
    else:
        if colour == "red":
            print("I bet you are seeing RED right now!")
        elif colour == "blue":
            print("Don't feel BLUE.")
        elif colour == "green":
            print("I bet you are GREEN with envy right now.")
        elif colour == "white":
            print("Are you WHITE as a sheet, as you didn't guess correctly?")
        elif colour == "pink":
            print("Some of you are not feeling in the PINK, as you got it wrong!")
```

# Черепашья графика

## Объяснение

В Python можно рисовать с использованием *черепахи* (*turtle*). Используя различные команды и циклы, можно строить сложные изображения. Вот как это работает.

Черепаха перемещается по определяемому вами пути, оставляя за собой след. В то время когда вы управляете черепахой, на изображении за ней остается видимый след. Чтобы нарисовать пятиугольник, изображенный ниже, следует ввести следующий код.



```
import turtle

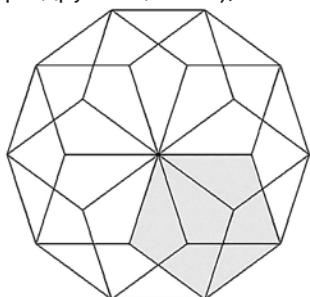
turtle.shape("turtle")

for i in range(0, 5):
    turtle.forward(100)
    turtle.right(72)

turtle.exitonclick()
```



Объединяя эти простые фигуры и используя **вложенные** циклы (то есть циклы внутри других циклов), можно очень легко создавать красивые узоры.



```
import turtle

for i in range(0, 10):
    turtle.right(36)
    for i in range(0, 5):
        turtle.forward(100)
        turtle.right(72)

turtle.exitonclick()
```

В этом узоре один пятиугольник рисуется 10 раз с поворотом на 36 градусов вокруг центральной точки. **Внимание:** на иллюстрации один из пятиугольников выделен цветом, чтобы вам было проще разглядеть его в узоре, но вообще он выделяться не должен.

# Примеры кода

**import turtle**

Эта строка включается в начало вашей программы, чтобы импортировать библиотеку **turtle** в Python. Импортирование позволит вам использовать функции для работы с черепахой.

**scr.bgcolor("yellow")**

Назначает экрану желтый цвет фона. По умолчанию используется белый цвет фона, если только вы не измените его.

**turtle.pensize(3)**

Устанавливает размер пера черепахи (толщину рисуемой линии) со значением 3. По умолчанию он равен 1, пока вы не измените его.

**turtle.forward(50)**

Перемещает черепаху вперед на 50 шагов.

**turtle.hideturtle()**

Скрывает черепаху, чтобы она не была видна на экране.

**turtle.showturtle()**

Отображает черепаху на экране. По умолчанию черепаха отображается.

**turtle.color("black", "red")**

Определяет цвета заполнения фигур. В данном примере рисуется фигура с черным контуром и красной заливкой. Команда должна выполняться до рисования фигуры.

**scr = turtle.Screen()**

Определяет окно с помощью названия «scr». Это позволяет использовать сокращенную запись **scr** вместо того, чтобы каждый раз обращаться к окну, используя полное имя.



**turtle.penup()**

Отключает перо, чтобы при перемещении за черепахой не оставался видимый след.

**turtle.left(120)**

Поворачивает черепаху на 120° налево (против часовой стрелки).



**turtle.pendown()**

Включает перо, чтобы при перемещении черепахи оставался видимый след. По умолчанию перо включено.

**turtle.right(90)**

Поворачивает черепаху на 90° вправо (по часовой стрелке).

**turtle.shape("turtle")**

Изменяет внешний вид черепахи так, чтобы она выглядела как черепаха. По умолчанию черепаха изображается в виде маленькой стрелки.

**turtle.begin\_fill()**

Выполняется перед кодом, рисующим фигуру, чтобы нарисованная фигура была автоматически заполнена.

**turtle.end\_fill()**

Выполняется после кода, рисующего фигуру, чтобы отключить автоматическое заполнение фигур.

**turtle.exitonclick()**

Когда пользователь щелкает на окне черепахи, оно автоматически закрывается.

## Задачи

060

Нарисуйте квадрат.

061

Нарисуйте треугольник.

062

Нарисуйте круг.



063

Нарисуйте в один ряд три квадрата, разделенных промежутками. Заполните их тремя различными цветами.

064

Нарисуйте пятиконечную звезду.

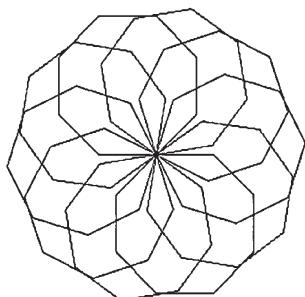


066

Нарисуйте восьмиугольник, все стороны которого окрашены в разные цвета (случайно выбираемые из списка шести возможных цветов).

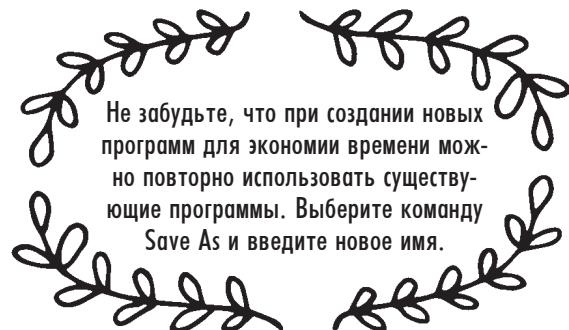
067

Нарисуйте следующий узор:



068

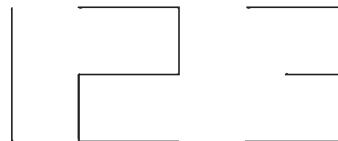
Нарисуйте узор, который меняется при каждом запуске программы. Используйте функцию `random` для выбора количества линий, длины каждой линии и каждого угла поворота.



Не забудьте, что при создании новых программ для экономии времени можно повторно использовать существующие программы. Выберите команду Save As и введите новое имя.

065

Нарисуйте цифры, изображенные ниже, начиная от нижней точки цифры 1.



066



# Ответы

**060**

```
import turtle

for i in range(0, 4):
    turtle.forward(100)
    turtle.right(90)

turtle.exitonclick()
```

**061**

```
import turtle

for i in range(0, 3):
    turtle.forward(100)
    turtle.left(120)

turtle.exitonclick()
```

**062**

```
import turtle

for i in range(0, 360):
    turtle.forward(1)
    turtle.right(1)

turtle.exitonclick()
```

**063**

```
import turtle

turtle.color("black", "red")
turtle.begin_fill()
for i in range(0, 4):
    turtle.forward(70)
    turtle.right(90)
turtle.penup()
turtle.end_fill()
turtle.forward(100)

turtle.pendown()
turtle.color("black", "yellow")
turtle.begin_fill()
for i in range(0, 4):
    turtle.forward(70)
    turtle.right(90)
turtle.penup()
turtle.end_fill()
turtle.forward(100)

turtle.pendown()
turtle.color("black", "green")
turtle.begin_fill()
for i in range(0, 4):
    turtle.forward(70)
    turtle.right(90)
turtle.end_fill()

turtle.exitonclick()
```

**064**

```
import turtle

for i in range(0, 5):
    turtle.forward(100)
    turtle.right(144)

turtle.exitonclick()
```

**065**

```
import turtle

turtle.left(90)
turtle.forward(100)
turtle.right(90)
turtle.penup()
turtle.forward(50)
turtle.pendown()
turtle.forward(75)
turtle.right(90)
turtle.forward(50)
turtle.right(90)
turtle.forward(75)
turtle.left(90)
turtle.forward(50)
turtle.left(90)
turtle.forward(75)
turtle.penup()
turtle.forward(50)
turtle.pendown()
turtle.forward(75)
turtle.left(90)
turtle.forward(50)
turtle.left(90)
turtle.forward(45)
turtle.left(180)
turtle.forward(45)
turtle.left(90)
turtle.forward(50)
turtle.left(90)
turtle.forward(75)

turtle.hideturtle()

turtle.exitonclick()
```

**066**

```
import turtle
import random

turtle.pensize(3)

for i in range(0, 8):
    turtle.color(random.choice( ["red", "blue", "yellow", "green", "pink", "orange"]))
    turtle.forward(50)
    turtle.right(45)

turtle.exitonclick()
```

**067**

```
import turtle
import random

for x in range(0,10):
    for i in range(0,8):
        turtle.forward(50)
        turtle.right(45)
    turtle.right(36)

turtle.hideturtle()

turtle.exitonclick()
```

**068**

```
import turtle
import random

lines = random.randint(5, 20)

for x in range(0,lines):
    length = random.randint(25, 100)
    rotate = random.randint(1, 365)
    turtle.forward(length)
    turtle.right(rotate)

turtle.exitonclick()
```

# Кортежи, списки и словари

## Объяснение

Пока что мы использовали переменные, в которых могло храниться только одно значение. Выполняя строку `random.choice(["red", "blue", "green"])`, вы выбираете один случайный элемент из списка возможных вариантов. Однако этот пример показывает, что одно значение может содержать несколько элементов данных (в данном случае это набор цветов).

Есть несколько вариантов сохранения наборов данных в одном значении. Три простейших варианта:

- кортежи;
- списки;
- словари.

### Кортежи

После того как **кортеж** будет определен, вы уже не сможете изменить его содержимое. Это означает, что при написании программы необходимо указать, какие данные хранятся в кортеже, и они останутся неизменными во время выполнения программы. Кортежи обычно используются для команд меню, которые не будут изменяться во время выполнения.



### Списки

Содержимое **списка** может изменяться во время выполнения программы, поэтому списки стали одним из самых распространенных способов хранения наборов данных под одним именем переменной в Python. Данные в списке не обязаны относиться к одному типу. Например, в одном списке могут храниться как строки, так и целые числа; однако позднее это может создать путаницу при обработке списка, поэтому поступать так не рекомендуется.

 **Примечание:** в других языках программирования для обозначения переменной, содержащей набор данных, часто используется термин «**массив**». В Python также существует тип данных под именем «массив», но он используется только для хранения чисел. Числовые массивы Python рассматриваются на с. 77.

## Словари

Содержимое **словаря** тоже может изменяться во время выполнения программы. Каждому значению присваивается индекс или ключ, по которому можно идентифицировать каждый элемент данных. Этот индекс не изменяется при добавлении или удалении других строк данных – в отличие от списков, в которых позиция элемента может измениться (что приведет к изменению его индекса).



# Примеры кода

```
fruit_tuple = ("apple", "banana", "strawberry", "orange")
```

Создает переменную с именем **fruit\_tuple**, в которой хранятся названия четырех фруктов. Круглые скобки определяют группу как кортеж; следовательно, содержимое этого набора данных не может изменяться во время выполнения программы.

```
print(fruit_tuple.index("strawberry"))
```

Выводит индекс (то есть числовой ключ) элемента **"strawberry"**. В данном примере будет возвращено число 2, так как в Python нумерация элементов начинается с 0, а не с 1.



```
print(fruit_tuple[2])
```

Выводит значение элемента с индексом 2 из кортежа **fruit\_tuple** (в данном случае **"strawberry"**).

```
del names_list[1]
```

Удаляет элемент 1 из списка **names\_list**. Не забудьте, что нумерация начинается с 0, а не с 1. В данном случае из списка будет удалено значение **"Tim"**.

```
names_list.sort()
```

Сортирует список **name\_list** по алфавиту и сохраняет его в новом порядке. Не работает, если в списке хранятся данные разных типов (например, если один список содержит как строки, так и числовые данные).



```
names_list = ["John", "Tim", "Sam"]
```

Создает список имен и сохраняет его в переменной **names\_list**. Квадратные скобки определяют эту группу данных как список; следовательно, ее содержимое может быть изменено во время выполнения программы.

```
names_list.append(input("Add a name: "))
```

Предлагает пользователю ввести имя и добавляет его в конец списка **names\_list**.

```
print(sorted(names_list))
```

Выводит содержимое **names\_list** в алфавитном порядке, но не изменяет исходного списка, который по-прежнему хранится в исходном порядке. Не работает, если в списке хранятся данные разных типов (например, если один список содержит как строки, так и числовые данные).

```
colours = {1:"red", 2:"blue", 3:"green"}
```

Создает словарь с именем **colours**, в котором каждому элементу присваивается индекс, указанный вами. Первый элемент в каждом блоке содержит индекс, за которым следуют двоеточие и строка с названием цвета.

```
colours[2] = "yellow"
```

Изменяет данные, хранящиеся в позиции с индексом **[2]** словаря **colours**. В данном случае значение **"blue"** будет заменено на **"yellow"**.



Так как списки являются одной из самых распространенных структур данных, мы приведем дополнительные примеры только для списков.

```
x = [154, 634, 892, 345, 341, 43]
```

В этом примере создается список, содержащий числа. Обратите внимание: так как он содержит только числовые данные, кавычки не нужны.

```
print(x[1:4])
```

Выводит данные в позициях 1, 2 и 3 (в данном случае 634, 892 и 345). Не забудьте, что в Python нумерация начинается с 0.

```
num = int(input("Enter number: "))
if num in x:
    print(num, " is in the list")
else:
    print("Not in the list")
```

Предлагает пользователю ввести число, проверяет, присутствует ли число в списке, и выводит соответствующее сообщение.

**x.insert(2, 420)**

Вставляет число 420 в позицию 2 и сдвигает все последующие элементы, чтобы освободить место. Вставка приводит к изменению индексов элементов списка.

**x.remove(892)**

Удаляет элемент из списка. Это может быть полезно, если индекс элемента неизвестен. Если значение находится в списке в нескольких экземплярах, удаляется только первый экземпляр.

**x.append(993)**

Добавляет число 993 в конец списка.

**print(len(x))**

Выводит длину списка (то есть количество элементов в списке).

```
for i in x:
    print(i)
```

Использует элементы списка в цикле **for**; может быть полезно, если вы хотите вывести элементы списка в отдельных строках.



# Задачи

**069**

Создайте кортеж с названиями пяти стран. Выведите все содержимое кортежа. Предложите пользователю ввести название одной из этих стран и выведите индекс (то есть позицию в списке) этого элемента кортежа.

**070**

Доработайте программу 069 так, чтобы она предлагала пользователю ввести число и выводила название страны, находящейся в заданной позиции.

**071**

Создайте список с названиями двух видов спорта. Предложите пользователю ввести свой любимый вид спорта и добавьте его в конец списка. Отсортируйте список и выведите его.

**072**

Создайте список с названиями шести школьных предметов. Спросите у пользователя, какие из этих предметов ему не нравятся. Удалите выбранные предметы из списка и выведите его повторно.

**073**

Предложите пользователю ввести названия четырех любимых блюд и сохраните их в словаре с числовыми индексами, начиная с 1. Выведите содержимое словаря с указанием индексов и элементов. Спросите пользователя, какой элемент он хочет исключить, и удалите его из списка. Отсортируйте оставшиеся данные и выведите содержимое словаря.

**074**

Введите список из десяти цветов. Предложите пользователю ввести начальное число в диапазоне от 0 до 4 и конечное число в диапазоне от 5 до 9. Выведите список цветов из интервала, заданного начальным и конечным числом.

**075**

Создайте список из четырех трехзначных чисел. Выведите содержимое списка, при этом каждый элемент должен выводиться на отдельной строке. Предложите пользователю ввести число из трех цифр. Если введенное число совпадает с одним из чисел в списке, выведите позицию этого числа; в противном случае выведите сообщение «That is not in the list».

**076**

Предложите пользователю ввести имена трех людей, которых он хочет пригласить на вечеринку, и сохраните их в списке. После того как будут введены все три имени, спросите, хочет ли пользователь добавить еще одно имя. Если ответ будет положительным, предложите ему добавлять имена, пока не получите ответ «no». После ответа «no» выведите количество людей, приглашенных на вечеринку.

**077**

Измените программу 076, чтобы после ввода списка имён программа выводила полный список. Предложите пользователю ввести одно из имён в списке и выведите позицию имени в списке. Спросите, хочет ли пользователь, чтобы этот человек присутствовал на вечеринке. Если пользователь ответит «*по*», удалите элемент из списка и снова выведите список.

**078**

Создайте список с названиями четырех телевизионных передач и выведите их на отдельных строках. Предложите пользователю ввести название еще одной передачи и позицию, на которой она должна быть вставлена в список. Снова выведите список, в котором все пять передач находятся на новых позициях.

**079**

Создайте пустой список с именем **nums**. Предложите пользователю последовательно вводить числа. После ввода каждого числа добавьте его в конец списка **nums** и выведите список. После того как пользователь введет три числа, спросите, хочет ли он оставить последнее введенное число в списке. Если пользователь ответит «*по*», удалите последний элемент из списка. Выведите список.



# Ответы

**069**

```
country_tuple = ("France", "England", "Spain", "Germany", "Australia")
print(country_tuple)
print()
country = input("Please enter one of the countries from above: ")
print(country, " has index number ", country_tuple.index(country))
```

**070**

```
country_tuple = ("France", "England", "Spain", "Germany", "Australia")
print(country_tuple)
print()
country = input("Please enter one of the countries from above: ")
print(country, " has index number ", country_tuple.index(country))
print()
num = int(input("Enter a number between 0 and 4: "))
print(country_tuple[num])
```

**071**

```
sports_list = ["tennis", "football"]
sports_list.append(input("What is your favorite sport? "))
sports_list.sort()
print(sports_list)
```

**072**

```
subject_list = ["maths", "english", "computing", "history", "science", "spanish"]
print(subject_list)
dislike = input("Which of these subjects do you dislike? ")
getrid = subject_list.index(dislike)
del subject_list[getrid]
print(subject_list)
```

**073**

```
food_dictionary = {}
food1 = input("Enter a food you like: ")
food_dictionary[1] = food1
food2 = input("Enter another food you like: ")
food_dictionary[2] = food2
food3 = input("Enter a third food you like: ")
food_dictionary[3] = food3
food4 = input("Enter one last food you like: ")
food_dictionary[4] = food4
print(food_dictionary)
dislike = int(input("Which of these do you want to get rid of? "))
del food_dictionary[dislike]
print(sorted(food_dictionary.values()))
```

**074**

```
colours = ["red", "blue", "green", "black", "white", "pink", "grey", "purple", "yellow",
"brown"]
start = int(input("Enter a starting number (0-4): "))
end = int(input("Enter an end number (5-9): "))
print(colours(start:end))
```

**075**

```
nums = [123, 345, 234, 765]
for i in nums:
    print(i)
selection = int(input("Enter a number from the list: "))
if selection in nums:
    print(selection, " is in position ", nums.index(selection))
else:
    print("That is not in the list")
```

**076**

```
name1 = input("Enter a name of somebody you want to invite to your party: ")
name2 = input("Enter another name: ")
name3 = input("Enter a third name: ")
party = [name1, name2, name3]
another = input("Do you want to invite another (y/n): ")
while another == "y":
    newname = party.append(input("Enter another name: "))
    another = input("Do you want to invite another (y/n): ")
print("You have ", len(party), " people coming to your party")
```

**077**

```
name1 = input("Enter a name of somebody you want to invite to your party: ")
name2 = input("Enter another name: ")
name3 = input("Enter a third name: ")
party = [name1, name2, name3]
another = input("Do you want to invite another (y/n): ")
while another == "y":
    newname = party.append(input("Enter another name: "))
    another = input("Do you want to invite another (y/n): ")
print("You have ", len(party), " people coming to your party")
print(party)
selection = input("Enter one of the names: ")
print(selection, " is in position ", party.index(selection), " on the list ")
stillcome = input("Do you still want them to come? (y/n): ")
if stillcome == "n":
    party.remove(selection)
print(party)
```

**078**

```
tv = ["Task master", "Top Gear", "The Big Bang Theory", "How I met Your Mother"]
for i in tv:
    print(i)
print()
newtv = input("Enter another TV show: ")
position = int(input("Enter a number between 0 and 3: "))
tv.insert(position, newtv)
for i in tv:
    print (i)
```

**079**

```
nums = []
count = 0
while count < 3:
    num = int(input("Enter a number: "))
    nums.append(num)
    print(nums)
    count = count + 1
lastnum = input("Do you want the last number saved (y/n): ")
if lastnum == "n":
    nums.remove(num)
print(nums)
```

# Снова о работе со строками

## Объяснение

**Строка** – термин для обозначения группы символов, с которыми *не нужно выполнять вычисления*. Например, "Hello" или "7B" – примеры строк.

Следующая команда присваивает переменной **name** значение "**Simon**".

```
name = "Simon"
```

Строчку "**Simon**" можно рассматривать как последовательность отдельных символов, в которой каждый символ идентифицируется с его индексом.

Индекс	0	1	2	3	4
Значение	<b>S</b>	<b>i</b>	<b>m</b>	<b>o</b>	<b>n</b>

Обратите внимание: индексирование символов в строках начинается с 0, а не с 1 (как и в списках.) Если строка содержит пробел, то этот пробел (а также все знаки препинания в строке) тоже рассматривается как символ.

Индекс	0	1	2	3	4	5	6	7	8	9	10	11
Значение	<b>H</b>	<b>e</b>	<b>l</b>	<b>l</b>	<b>o</b>		<b>W</b>	<b>o</b>	<b>r</b>	<b>l</b>	<b>d</b>	<b>!</b>

Вам уже знакома работа со списками, поэтому со строками проблем быть не должно, ведь они используют те же методы, что и списки. Тем не менее ниже я включила дополнительный код, который может оказаться полезным.

# Примеры кода

**Примечание:** в следующих примерах "msg" – имя переменной, содержащей строку.

```
if msg.isupper():
    print("Uppercase")
else:
    print("This is not in uppercase")
```

Если строка состоит из букв верхнего регистра, выводится сообщение «Uppercase», в противном случае выводится сообщение «This is not in uppercase».

**msg.islower()**

Может использоваться вместо функции **isupper()** для проверки того, что переменная состоит из букв нижнего регистра.



```
msg="Hello"
for letter in msg:
    print(letter,end="*")
```

Выводит сообщение, при этом после каждого символа выводится символ \*. В данном примере будет выведена следующая строка: **H\*e\*1\*l\*o\***

Помните: вы всегда можете вернуться к предыдущим программам и припомнить то, что выучили ранее.



# Задачи

**080**

Предложите пользователю ввести свое имя, а затем выведите длину имени. Запросите фамилию и выведите длину фамилии. Соедините имя с фамилией, разделив их пробелом, и выведите результат. Наконец, выведите длину полного имени (включая пробел).

**084**

Предложите пользователю ввести его почтовый индекс. Выведите первые две буквы слова в верхнем регистре<sup>1</sup>.

**086**

Предложите пользователю ввести пароль, а затем предложите ввести его повторно. Если два пароля совпадут, выведите сообщение «Thank you». Если буквы введены правильно, но различаются регистром, выведите сообщение «They must be in the same case»; в противном случае выводится сообщение «Incorrect».

**087**

Предложите пользователю ввести слово, а затем выведите буквы слова в обратном порядке в разных строках. Например, если пользователь ввел строку «Hello», результат должен выглядеть так:

Enter a word: Hello

```
o
1
l
e
H
>>>
```

**081**

Предложите пользователю ввести его любимый школьный предмет. Выведите его так, чтобы после каждой буквы следовал дефис — например, S-p-a-n-i-s-h-.

**082**

Выведите строку из своего любимого стихотворения и предложите пользователю ввести начальную и конечную позицию. Выведите символы, находящиеся между ними.

**083**

Предложите пользователю ввести слово в верхнем регистре. Если не все буквы слова будут указаны в верхнем регистре, попросите ввести слово заново. Повторяйте попытки, пока пользователь не введет сообщение в верхнем регистре.



**085**

Предложите пользователю ввести имя, а затем сообщите, сколько в нем гласных букв.



<sup>1</sup> В англоязычных странах почтовые индексы зачастую начинаются с латинских букв; вы можете найти любой индекс этого типа для выполнения данной задачи. — Примеч. ред.

# Ответы

**080**

```
fname = input("Enter your first name: ")
print("That has ", len(fname), " characters in it")
sname = input("Enter your surname: ")
print("That has ", len(sname), " characters in it")
name = fname + " " + sname
print("Your full name is", name)
print("That has ", len(name), " characters in it")
```

**081**

```
subject = input("Enter your favorite school subject: ")
for letter in subject:
    print(letter, end = "-")
```

**082**

```
poem = "Oh, I wish I'd looked after me teeth,"
print(poem)
start = int(input("Enter a starting number: "))
end = int(input("Enter an end number: "))
print(poem[start:end])
```

**083**

```
msg = input("Enter a message in uppercase: ")
tryagain = False
while tryagain == False:
    if msg.isupper():
        print("Thank you")
        tryagain = True
    else:
        print("Try again")
    msg = input("Enter a message in uppercase: ")
```

**084**

```
postcode = input("Enter your postcode: ")
start = postcode[0:2]
print(start.upper())
```

**085**

```
name = input("Enter your name: ")
count = 0
name = name.lower()
for x in name:
    if x == "a" or x == "e" or x == "i" or x == "o" or x == "u":
        count = count + 1
print("Vowels = ", count)
```

**086**

```
pswd1 = input("Enter a password: ")
pswd1 = input("Enter it again: ")
if pswd1 == pswd2:
    print("Thank you")
elif pswd1.lower() == pswd2.lower():
    print("They must be in the same case")
else:
    print("Incorrect")
```

**087**

```
word = input("Enter a word: ")
length = len(word)
num = 1
for x in word:
    position = length - num
    letter = word[position]
    print(letter)
    num = num + 1
```

# Числовые массивы

## Объяснение

Ранее в книге уже рассматривались списки (с. 63). В списках могут храниться данные разных типов, включая строки и числа. **Массивы** Python похожи на списки, но используются **только для хранения чисел**. В массиве все данные **должны относиться к одному типу данных** из перечисленных в следующей таблице.



Код типа	Общепринятое название	Описание	Размер в байтах
'i'	Целое число	Целое число в диапазоне от -32768 до 32767	2
'l'	Длинное целое число	Целое число в диапазоне от -2 147 483 648 до 2 147 483 648	4
'f'	Число с плавающей точкой	Числа в диапазоне от -1038 до 1038 с дробной частью (то есть число может содержать до 38 знаков, включая десятичную точку в любой позиции, и может быть как отрицательным, так и положительным)	4
'd'	Число двойной точности	Числа в диапазоне от -10 308 до 10 308 с дробной частью	8

При создании массива необходимо определить тип содержащихся в нем данных. Этот тип не может быть изменен во время работы программы. Следовательно, если вы определяете массив с типом 'i' (что позволяет использовать в нем целые числа в диапазоне от -32 768 до 32 767), позже вы не сможете добавить десятичную точку в число, потому что это приведет к выдаче сообщения об ошибке и аварийному завершению программы.



**Примечание:** в других языках программирования термин «массив» обычно используется для хранения данных любого типа, но в массивах Python могут храниться только числа, тогда как списки подходят для хранения любых типов данных. Если вы хотите создать переменную для хранения нескольких строк, в Python для этого придется создать список вместо массива.



# Примеры кода

```
from array import *
```

Эта строка должна находиться в самом начале программы, чтобы Python мог использовать библиотеку массивов.



```
nums = array ('i',[45,324,654,45,264])
print(nums)
```

Создает массив с именем **nums**. Массив использует целочисленный тип данных и состоит из пяти элементов. При выводе будет получен следующий результат:

```
array('i', [45, 324, 654, 45, 264])
```

```
for x in nums:
    print(x)
```

Выводит массив, при этом каждый элемент выводится в отдельной строке.

```
newValue = int(input("Enter number: "))
nums.append(newValue)
```

Предлагает пользователю ввести новое число, которое добавляется в конец существующего массива.

```
nums.reverse()
```

Переставляет элементы массива в обратном порядке.

```
nums = sorted(nums)
```

Сортирует массив по возрастанию.

```
nums.pop()
```

Удаляет последний элемент из массива.



```
newArray = array('i',[])
more = int(input("How many items: "))
for y in range(0,more):
    newValue=int(input("Enter num: "))
    newArray.append(newValue)
nums.extend(newArray)
```

Создает пустой массив с именем **newArray**, использующий целочисленный тип данных. Пользователю предлагается ввести количество элементов, после чего соответствующее количество элементов добавляется в **newArray**. После того как все элементы будут добавлены, содержимое массивов **newArray** и **nums** объединяется.

```
getRid = int(input("Enter item index: "))
nums.remove(getRid)
```

Предлагает пользователю ввести элемент, который удаляется из массива, после чего удаляет первый элемент массива, совпадающий с введенным значением.

```
print(nums.count(45))
```

Показывает, сколько раз значение **45** встречается в массиве.

# Задачи

**088**

Предложите пользователю ввести пять целых чисел и сохраните их в массиве. Отсортируйте список и выведите его содержимое в обратном порядке.

**089**

Создайте массив для хранения целых чисел. Сгенерируйте пять случайных чисел и сохраните их в массиве. Выведите массив (каждый элемент должен выводиться в отдельной строке).

**090**

Предложите пользователю вводить целые числа. Если пользователь вводит число от 10 до 20, сохраните его в массиве; в противном случае выведите сообщение «Outside the range». После того как пять чисел будут успешно добавлены в массив, выведите сообщение «Thank you» и выведите массив, каждый элемент которого находился бы на отдельной строке.

**091**

Создайте два массива: один будет содержать три числа, введенных пользователем, а другой — пять случайных чисел. Объедините эти два массива в один большой. Отсортируйте и выведите его, при этом каждое число должно выводиться в отдельной строке.

Не останавливайтесь!



**092**

Создайте массив, содержащий пять чисел (два из которых должны повторяться). Выведите весь массив. Предложите пользователю ввести одно из чисел массива, после чего выведите сообщение, в котором указано, сколько раз число встречается в этом массиве.

**093**

Предложите пользователю ввести пять чисел. Отсортируйте их и выведите для пользователя. Предложите выбрать одно из чисел. Удалите выбранное число из исходного массива и сохраните его в новом.



**094**

Выведите массив из пяти чисел. Предложите пользователю выбрать одно из них. После того как число будет выбрано, выведите его позицию в массиве. Если пользователь введет значение, отсутствующее в массиве, предложите ему выбрать снова, пока не будет выбрано допустимое значение.

**095**

Создайте массив из пяти чисел от 10 до 100, каждое из которых содержит два знака в дробной части. Предложите пользователю ввести целое число от 2 до 5. Если пользователь введет значение, выходящее за границы диапазона, выведите сообщение об ошибке и предложите выбрать снова, пока не будет введено допустимое значение. Разделите каждое из чисел в массиве на число, введенное пользователем, и выведите ответы с точностью до двух знаков.

# Ответы

## 088

```
from array import *

nums = array('i', [])

for i in range(0, 5):
    num = int(input("Enter a number: "))
    nums.append(num)

nums = sorted(nums)
nums.reverse()

print(nums)
```

## 089

```
from array import *
import random

nums = array('i', [])

for i in range (0, 5):
    num = random.randint(1, 100)
    nums.append(num)

for i in nums:
    print(i)
```

## 090

```
from array import *

nums = array('i', [])

while len(nums) < 5:
    num = int(input("Enter a number between 10 and 20: "))
    if num >= 10 and num <= 20:
        nums.append(num)
    else:
        print("Outside the range")

for i in nums:
    print(i)
```

**091**

```
from array import *

nums = array('i', [5, 7, 9, 2, 9])

for i in nums:
    print(i)

num = int(input("Enter a number: "))

if nums.count(num) == 1:
    print(num, "is in the list once")
else:
    print(num, "is in the list ", nums.count(num),"times")
```

**092**

```
from array import *
import random

num1 = array('i', [])
num2 = array('i', [])

for i in range(0, 3):
    num = int(input("Enter a number: "))
    num1.append(num)

for i in range(0, 5):
    num = random.randint(1, 100)
    num2.append(num)

num1.extend(num2)

num1 = sorted(num1)

for i in num1:
    print(i)
```

**093**

```
from array import *
nums = array('i', [])
for i in range(0, 5):
    num = int(input("Enter a number: "))
    nums.append(num)
nums = sorted(nums)
for i in nums:
    print(i)
num = int(input("Select a number from the array: "))
if num in nums:
    nums.remove(num)
    num2 = array('i', [])
    num2.append(num)
    print(nums)
    print(num2)
else:
    print("That is not a value in the array")
```

**094**

```
from array import *
nums = array('i', [4, 6, 8, 2, 5])
for i in nums:
    print(i)
num = int(input("Select one of the numbers: "))
tryagain = True
while tryagain == True:
    if num in nums:
        print("This is in position ", nums.index(num))
        tryagain = False
    else:
        print("Not in array")
        num = int(input("Select one of the numbers: "))
```

**095**

```
from array import *
import math
num1 = array('f',[34.75, 27.23, 99.58, 45.26, 28.65])
tryagain = True
while tryagain == True:
    num = int(input("Enter a number between 2 and 5: "))
    if num < 2 or num > 5:
        print("Incorrect value, try again.")
    else:
        tryagain = False
for i in range(0, 5):
    ans = num1[i] / num
    print(round(ans, 2))
```

# Двумерные списки и словари

## Объяснение

С технической точки зрения в Python возможно создать двумерный массив, но так как массивы Python ограничиваются хранением чисел, а большинство программистов Python чувствует себя более уверенно при работе со списками, двумерные массивы используются редко, а **двумерные списки** встречаются гораздо чаще.



Представьте ужасную ситуацию: вы работаете учителем. Понимаю, это не для слабонервных! Также представьте, что у вас есть четыре ученика, и вы преподаете им три разных предмета. Вы как сознательный учитель ведете учет баллов этих учеников по всем предметам. На бумаге подобная таблица может выглядеть так:

	Математика	Английский	Французский
Сьюзен	45	37	54
Питер	62	58	59
Марк	49	47	60
Энди	78	83	62



Двумерные списки работают аналогичным образом:

	0	1	2
0	45	37	54
1	62	58	59
2	49	47	60
3	78	83	62

В Python двумерный список записывается так:

```
grades = [[45, 37, 54], [62, 58, 59], [49, 47, 60], [78, 83, 62]]
```

Если вы не хотите использовать стандартные числовые индексы столбцов Python, можно работать со словарем:

```
grades = [{"Ma":45, "En":37, "Fr":54}, {"Ma":62, "En":58, "Fr":59},
           {"Ma":49, "En":47, "Fr":60}]
print(grades[0]["En"])
```

Программа выводит значение 37 (оценка для ученика с индексом 0 по предмету "En") и упрощает понимание данных.

Можно пойти еще дальше и добавить символические индексы для строк:

```
grades = {"Susan":{"Ma":45, "En":37, "Fr":54}, "Peter":{"Ma":62, "En":58,
              "Fr":59}}
print(grades["Peter"]["En"])
```

Команда выводит значение 58 (оценка для ученика "Peter" по предмету "En").



# Примеры кода

```
simple_array = [[2, 5, 8], [3, 7, 4], [1, 6, 9]]
```

Создает двумерный список (изображенный справа) с использованием стандартных индексов Python для строк и столбцов.

	0	1	2
0	2	5	8
1	3	7	4
2	1	6	9

```
print(simple_array)
```

Выводит все данные в виде двумерного списка.



```
print(simple_array [1])
```

Выводит данные из строки 1 — в данном случае [3, 7, 4].

```
simple_array [2] [1] = 5
```

Присваивает элементу, находящемуся в строке 2 и столбце 1, значение 5.

```
print(simple_array [1] [2])
```

Выводит данные из строки 1 и столбца 2 — в данном случае 4.

```
simple_array[1].append(3)
```

Добавляет значение 3 в конец данных строки 1, так что в данном случае строка принимает вид [3, 7, 4, 3].

	x	y	z
A	54	82	91
B	75	29	80



```
data_set = {"A": {"x": 54, "y": 82, "z": 91}, "B": {"x": 75, "y": 29, "z": 80}}
```

Создает двумерный список с использованием пользовательских меток для строк и столбцов (см. выше).

```
print(data_set ["A"])
```

Выводит информацию из набора данных "A".

```
print(data_set ["B"] ["y"])
```

Выводит значение элемента из строки "B" и столбца "y".

```
for i in data_set:
    print(data_set [i] ["y"])
```

Выводит элемент из столбца "y" каждой строки.

```
data_set ["B"] ["y"] = 53
```

Присваивает элементу из строки "B" и столбца "y" значение 53.

```
Grades [name] = {"Maths": mscore, "English": escore}
```

Добавляет еще одну строку данных в двумерный словарь. В этом случае **name** становится индексом строки, а **"Maths"** и **"English"** — индексами столбцов.

```
for name in grades:
    print((name), grades [name] ["English"])
```

Выводит только значение **name** и оценку по предмету **"English"** для каждого ученика.

```
del list [getRid]
```

Удаляет выбранный элемент.

# Задачи

**096**

Создайте следующий набор данных в виде простого двумерного списка со стандартными индексами Python:

	0	1	2
0	2	5	8
1	3	7	4
2	1	6	9
3	4	2	0

**097**

Используя двумерный список из задачи 096, предложите пользователю выбрать строку и столбец и выведите выбранное значение.

**098**

Используя двумерный список из задачи 096, предложите пользователю выбрать строку и выведите только ее. Предложите ввести новое значение, добавьте его в конец строки, после чего снова выведите измененную строку.

**099**

Измените программу из задачи 098. Предложите пользователю выбрать строку и выведите только ее. Предложите выбрать столбец из выведенной строки и выведите только хранящееся там значение. Спросите, хочет ли пользователь изменить его. Если ответ будет положительным, предложите ввести новое значение и измените данные. Наконец, снова выведите измененную строку.

**100**

Создайте следующий набор данных, представляющий объемы продаж по регионам, в виде двумерного словаря:

	N	S	E	W
John	3056	8463	8441	2694
Tom	4832	6786	4737	3612
Anna	5239	4802	5820	1859
Fiona	3904	3645	8821	2451

**101**

Используя программу из задачи 100, запросите у пользователя имя и регион. Выведите соответствующие данные. Запросите у пользователя имя и регион того значения, которое он хочет изменить, и позвольте скорректировать объем продаж. Выведите объемы продаж по всем регионам для имени, выбранного пользователем.

**102**

Предложите пользователю ввести имя, возраст и размер обуви для четырех человек. Запросите имя одного из них в списке и выведите значения его возраста и размера обуви.

**103**

Измените программу 102, чтобы она выводила имя и возраст для всех людей в списке, но не их размер обуви.

**104**

После получения имени, возраста и размера обуви для четырех человек запросите у пользователя имя человека для удаления из списка. Удалите эту строку и выведите остальные данные с разбивкой по строкам.



# Ответы

**096**

```
list = [[2, 5, 8], [3, 7, 4], [1, 6, 9], [4, 2, 0]]
```

**097**

```
list = [[2, 5, 8], [3, 7, 4], [1, 6, 9], [4, 2, 0]]
row = int(input("Select a row: "))
col = int(input("Select a column: "))
print(list[row][col])
```

**098**

```
list = [[2, 5, 8], [3, 7, 4], [1, 6, 9], [4, 2, 0]]
row = int(input("Select a row: "))
print(list[row])
newvalue = int(input("Select a new number: "))
list[row].append(newvalue)
print(list[row])
```

**099**

```
list = [[2, 5, 8], [3, 7, 4], [1, 6, 9], [4, 2, 0]]
row = int(input("Select a row: "))
print(list[row])
col = int(input("Select a column: "))
print(list[row][col])
change = input("Do you want to change the value? (y/n)")
if change == "y":
    newvalue = int(input("Enter new value: "))
    list[row][col] = newvalue
print(list[row])
```

**100**

Данные разбиты на строки для удобства чтения. Такая разбивка возможна при условии, что разрывы строк располагаются в естественных позициях и заключаются в фигурные скобки.

```
sales = {"John": {"N": 3056, "S": 8463, "E": 8441, "W": 2694},
         "Tom": {"N": 4832, "S": 6786, "E": 4737, "W": 3612},
         "Anne": {"N": 5239, "S": 4802, "E": 5820, "W": 1859},
         "Fiona": {"N": 3904, "S": 3645, "E": 8821, "W": 2451}}
```

**101**

```

sales = {"John": {"N": 3056, "S": 8463, "E": 8441, "W": 2694},
         "Tom": {"N": 4832, "S": 6786, "E": 4737, "W": 3612},
         "Anne": {"N": 5239, "S": 4802, "E": 5820, "W": 1859},
         "Fiona": {"N": 3904, "S": 3645, "E": 8821, "W": 2451}}
person = input("Enter sales person's name: ")
region = input("Select region: ")
print(sales[person][region])
newdata = int(input("Enter new data: "))
sales[person][region] = newdata
print(sales[person])

```

**102**

```

list = []
for i in range(0, 4):
    name = input("Enter name: ")
    age = int(input("Enter age: "))
    shoe = int(input("Enter shoe size: "))
    list[name] = {"Age": age, "Shoe size": shoe}

ask = input("Enter a name: ")
print(list[ask])

```

**103**

```

list = []
for i in range(0, 4):
    name = input("Enter name: ")
    age = int(input("Enter age: "))
    shoe = int(input("Enter shoe size: "))
    list[name] = {"Age": age, "Shoe size": shoe}

for name in list:
    print((name), list[name]["Age"])

```

**104**

```

list = []
for i in range(0, 4):
    name = input("Enter name: ")
    age = int(input("Enter age: "))
    shoe = int(input("Enter shoe size: "))
    list[name] = {"Age": age, "Shoe size": shoe}

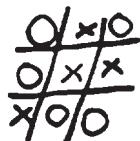
getrid = input("Who do you want to remove from the list? ")
del list[getrid]

for name in list:
    print((name), list[name]["Age"], list[name]["Shoe size"])

```

# Чтение и запись текстовых файлов

## Объяснение



Теперь вы знаете, как определить список, внести в него изменения и добавить новые данные. Все это, конечно, хорошо, но если при следующем запуске программа вернется к исходным данным, а все ваши изменения будут потеряны, то пользы от такой программы немного. Следовательно, иногда возникает необходимость хранить данные вне программы (вместе со всеми изменениями, вносимыми в них в программе).



Запись и чтение из внешних файлов проще всего изучать на примере **текстовых** файлов.

При открытии внешнего файла необходимо указать, как этот файл будет использоваться в программе. Поддерживаются следующие режимы:

Режим	Описание
w	<b>Режим записи:</b> используется для создания нового файла. Существующий файл с заданным именем стирается, и вместо него создается новый файл
r	<b>Режим чтения:</b> используется в том случае, если существующий файл используется только для чтения, а запись в него не выполняется
a	<b>Режим присоединения:</b> используется для добавления новых данных в конец файла

Текстовые файлы могут использоваться только для записи, чтения и присоединения данных. Природа текстовых файлов такова, что данные, записанные в файл, не так просто удалить или изменить; приходится перезаписывать весь файл или создавать новый для хранения обновленных данных. Если вы хотите иметь возможность изменять отдельные элементы после того, как файл будет создан, лучше воспользоваться файлом с расширением .csv (с. 94) или базой данных SQL (с. 137).



## Примеры кода

```
file = open("Countries.txt", "w")
file.write("Italy\n")
file.write("Germany\n")
file.write("Spain\n")
file.close()
```

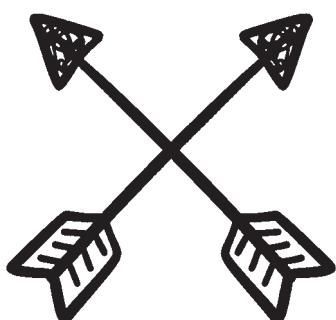
Создает файл с именем **Countries.txt**. Если файл с таким именем уже существует, то он заменяется новым пустым файлом. В него добавляются три строки данных (**\n** вставляет разрыв строки после каждой записи). Затем файл закрывается, чтобы изменения в текстовом файле были сохранены.

```
file = open("Countries.txt", "r")
print(file.read())
```

Открывает файл **Countries.txt** в режиме чтения и выводит все содержимое файла.

```
file = open("Countries.txt", "a")
file.write("France\n")
file.close()
```

Открывает файл **Countries.txt** в режиме присоединения, добавляет новую строку и закрывает файл. Если строка **file.close()** отсутствует, то изменения не будут сохранены в текстовом файле.



# Задачи

**105**

Создайте новый файл с именем **Numbers.txt**. Добавьте в него пять чисел, которые хранятся в одной строке и разделяются только запятыми. После запуска программы найдите папку, в которой располагается ваша программа; убедитесь в том, что файл был создан. В системе Windows для просмотра содержимого нового текстового файла проще всего воспользоваться «Блокнотом».

**110**

С помощью созданного ранее файла **Names.txt** выведите список имен в Python. Попросите пользователя ввести одно из имен, а затем сохраните все, кроме выбранного в новом файле, под названием **Names2.txt**.

**106**

Создайте новый файл с именем **Names.txt**. Добавьте в него пять имен, отображающихся на разных строках. После запуска программы найдите папку, в которой располагается ваша программа; убедитесь в том, что файл был создан.

**107**

Откройте файл **Names.txt** и выведите данные из кода Python.

**108**

Откройте файл **Names.txt**. Предложите пользователю ввести новое имя. Добавьте его в конец файла и выведите все содержимое файла.

**109**

Выведите следующее меню:

- 1) Create a new file
  - 2) Display the file
  - 3) Add a new item to the file
- Make a selection 1, 2 or 3:

Предложите пользователю выбрать один из вариантов. Если пользователь введет что-либо, кроме 1, 2 и 3, программа должна вывести соответствующее сообщение об ошибке.

Если пользователь выберет 1, предложите ему ввести название школьного предмета и сохраните его в новом файле с именем **Subject.txt**. Существующий файл с таким именем должен быть заменен новым файлом.

Если пользователь выберет 2, выводится содержимое файла **Subject.txt**.

Если пользователь выберет 3, предложите пользователю ввести новый предмет, сохраните его в файле, а затем выведите все его содержимое.

Запустите программу несколько раз, чтобы протестировать разные команды.

Отличная работа! Сохранение данных во внешних файлах – очень важный навык для программиста.

## Ответы

**105**

```
file = open("Numbers.txt", "w")
file.write("4, ")
file.write("6, ")
file.write("10, ")
file.write("8, ")
file.write("5, ")
file.close()
```

**106**

```
file = open("Names.txt", "w")
file.write("Bob\n")
file.write("Tom\n")
file.write("Gemma\n")
file.write("Sarah\n")
file.write("Timothy\n")
file.close()
```

**107**

```
file = open("Names.txt", "r")
print(file.read())
file.close()
```

**108**

```
file = open("Names.txt", "a")
newname = input("Enter a new name: ")
file.write(newname + "\n")
file.close()

file = open("Names.txt", "r")
print (file.read())
file.close()
```

**109**

```
print("1) Create a new file")
print("2) Display the file")
print("3) Add a new item to the file")
selection = int(input("Make a selection 1, 2 or 3: "))
if selection == 1:
    subject = input("Enter a school subject: ")
    file = open("Subject.txt", "w")
    file.write(subject + "\n")
    file.close()
elif selection == 2:
    file = open("Subject.txt", "r")
    print(file.read())
elif selection == 3:
    file = open("Subject.txt", "a")
    subject = input("Enter a school subject: ")
    file.write(subject + "\n")
    file.close()
    file = open("Subject.txt", "r")
    print(file.read())
else:
    print("Invalid option")
```

**110**

```
file = open("Names.txt", "r")
print(file.read())
file.close()

file.open("Names.txt", "r")
selectedname = input("Enter a name from the list: ")
selectedname = selectedname + "\n"
for row in file:
    if row != selectedname:
        file = open("Names2.txt", "a")
        newrecord = row
        file.write(newrecord)
        file.close()
file.close()
```

# Чтение и запись файлов .csv

## Объяснение

Сокращение **CSV** означает **Comma Separated Values** (то есть «значения, разделенные запятыми»); это формат, обычно связанный с импортированием и экспортацией данных из электронных таблиц и баз данных. Он предоставляет больше возможностей для управления данными по сравнению с простыми текстовыми файлами, так как каждая строка делится на легко определяемые столбцы. Ниже приведен пример данных, которые может понадобиться сохранить.

Имя	Возраст	Знак зодиака
Brian	73	Taurus
Sandra	48	Virgo
Zoe	25	Scorpio
Keith	43	Leo



В файле .csv эти данные хранятся в следующем виде:



Brian, 73, Taurus  
 Sandra, 48, Virgo  
 Zoe, 25, Scorpio  
 Keith, 43, Leo

Тем не менее кому-то будет проще представить себе эти данные разделенными на столбцы и строки, для идентификации которых используются числовые индексы.

	0	1	2
0	Brian	73	Taurus
1	Sandra	48	Virgo
2	Zoe	25	Scorpio
3	Keith	43	Leo

При открытии файла .csv необходимо указать, как этот файл будет использоваться в программе. Доступны следующие варианты:

Режим	Описание
w	Создает новый файл для записи данных. Если файл уже существует, то он стирается и вместо него создается новый
x	Создает новый файл для записи данных. Если файл уже существует, то вместо перезаписи в программе происходит фатальный сбой
r	Открывает файл для чтения, никакие изменения при этом вноситься не могут
a	Открывает файл для записи, при этом данные присоединяются в конец файла



## Примеры кода

```
import csv
```

Эта строка должна находиться в самом начале программы, чтобы Python мог использовать библиотеку поддержки CSV.

```
file = open ("Stars.csv", "w")
newRecord = "Brian, 73, Taurus\n"
file.write(str(newRecord))
file.close()
```

Создает новый файл с именем **Stars.csv**, при этом существующий файл с таким именем будет уничтожен. В файл добавляется новая запись, после чего он закрывается с сохранением внесенных изменений.

```
file = open ("Stars.csv", "a")
name = input("Enter name: ")
age = input("Enter age: ")
star = input("Enter star sign: ")
newRecord = name + ", " + age + ", " + star + "\n"
file.write(str(newRecord))
file.close()
```

Открывает файл **Stars.csv**, предлагает пользователю ввести имя, возраст и знак зодиака и присоединяет запись в конец файла.



```
file = open("Stars.csv", "r")
for row in file:
    print(row)
```

Открывает файл **Stars.csv** в режиме чтения и последовательно выводит его строки.

```
file = open("Stars.csv", "r")
reader = csv.reader(file)
rows = list(reader)
print(rows [1])
```

Открывает файл **Stars.csv** и выводит только строку 1 (не забудьте, что в Python нумерация индексов начинается с 0).

```
file = open ("Stars.csv", "r")
search = input("Enter the data you are searching for: ")
reader = csv.reader(file)
for row in file:
    if search in str(row):
        print(row)
```

Предлагает пользователю ввести искомые данные. После этого выводятся все строки, в которых эти данные где-либо присутствуют.



```
import csv
file = list(csv.reader(open("Stars.csv")))
tmp = []
for row in file:
    tmp.append(row)
```

Файл .csv нельзя изменять, к нему можно только добавлять новые данные. Если вам потребуется изменить существующее содержимое, запишите его во временный список. Этот блок кода читает исходный файл .CSV и записывает его в список **tmp**. После этого с ним можно работать и изменять как список (с. 63).

```
file = open("NewStars.csv", "w")
x = 0
for row in tmp:
    newRec = tmp[x][0] + ", " + tmp[x][1] + ", " + tmp[x][2] + "\n"
    file.write(newRec)
    x = x + 1
file.close()
```

Записывает данные из списка в новый файл .CSV с именем NewStars.csv.



# Задачи

**111**

Создайте файл .csv с данными, приведенными в следующей таблице. Назовите его Books.csv.

	Книга	Автор	Год выпуска
0	To Kill a Mockingbird	Harper Lee	1960
1	A Brief History of Time	Stephen Hawking	1988
2	The Great Gatsby	F. Scott Fitzgerald	1922
3	The Man Who Mistook His Wife for a Hat	Oliver Sacks	1985
4	Pride and Prejudice	Jan Austen	1813

**112**

Используя файл Books.csv из программы 111, предложите пользователю ввести новую запись и добавьте ее в конец файла. Выведите каждую строку файла .csv в отдельной строке.

**113**

Используя файл Books.csv, спросите пользователя, сколько записей он хочет добавить в список, и предоставьте ему такую возможность. После того как данные будут добавлены, запросите автора и выведите все книги указанного автора из списка. Если в списке нет ни одной книги этого автора, выведите соответствующее сообщение.

**114**

Используя файл Books.csv, предложите пользователю ввести начальный и конечный год. Выведите все книги, выпущенные в заданном промежутке времени.

**115**

Используя файл Books.csv, выведите данные с нумерацией строк.



**116**

Импортируйте данные из файла Books.csv в список. Выведите список, предложите пользователю выбрать, какую строку он хочет исключить, и удалите ее. Спросите пользователя, какие данные он хочет изменить, и предоставьте ему соответствующую возможность. Запишите данные обратно в файл .csv с заменой существующих.

**117**

Создайте простую математическую игру, которая запрашивает у пользователя имя, а затем генерирует два случайных вопроса. Сохраните имя, введенные вопросы, ответы пользователя и итоговый счет в файле .csv. При каждом запуске программа должна добавлять информацию в файл .csv без перезаписи существующих данных.

# Ответы

111

```
import csv

file = open("Books.csv", "w")
newrecord = "To Kill a Mockingbird, Harper Lee, 1960\n"
file.write(str(newrecord))
newrecord = "A Brief History of Time, Stephen Hawking, 1988\n"
file.write(str(newrecord))
newrecord = "The Great Gatsby, F. Scott Fitzgerald, 1922\n"
file.write(str(newrecord))
newrecord = "The Man Who Mistook His Wife for a Hat, Oliver Sacks, 1985\n"
file.write(str(newrecord))
newrecord = "Pride and Prejudice, Jane Austen, 1813\n"
file.write(str(newrecord))
file.close()
```

112

```
import csv

file = open("Books.csv", "a")
title = input("Enter a title: ")
author = input("Enter author: ")
year = input("Enter the year it was released: ")
newrecord = title + "," + author + ", " + year + "\n"
file.write(str(newrecord))
file.close()

file = open("Books.csv", "r")
for row in file:
    print(row)
file.close()
```

**113**

```
import csv

num = int(input("How many books do you want to add to the list? "))
file = open("Books.csv", "a")
for x in range(0, num):
    title = input("Enter a title: ")
    author = input("Enter author: ")
    year = input("Enter the year it was released: ")
    newrecord = title + ", " + author + ", " + year + "\n"
    file.write(str(newrecord))
file.close()

searchauthor = input("Enter an authors name to search for: ")

file = open("Books.csv", "r")
count = 0
for row in file:
    if searchauthor in str(row):
        print(row)
        count = count + 1
if count == 0:
    print ("There are no books by that author in this list.")
file.close()
```

**114**

```
import csv

start = int(input("Enter a starting year: "))
end = int(input("Enter an end year: "))

file = list(csv.reader(open("Books.csv")))
tmp = []
for row in file:
    tmp.append(row)

x = 0
for row in tmp:
    if int(tmp[x][2]) >= start and int(tmp[x][2]) <=end:
        print(tmp[x])
    x = x + 1
```

**115**

```
import csv

file = open("Books.csv", "r")
x = 0
for row in file:
    display = "Row: " + str(x) + " - " + row
    print(display)
    x = x + 1
```

**116**

```
import csv

file = list(csv.reader(open("Books.csv")))
Booklist = []
for row in file:
    Booklist.append(row)

x = 0
for row in Booklist:
    display = x, Booklist[x]
    print(display)
    x = x + 1
getrid = int(input("Enter a row number to delete: "))
del Booklist[getrid]

x = 0
for row in Booklist:
    display = x, Booklist[x]
    print(display)
    x = x + 1
alter = int(input("Enter a row number to alter: "))
x = 0
for row in Booklist[alter]:
    display = x, Booklist[alter][x]
    print(display)
    x = x + 1
part = int(input("Which part do you want to change? "))
newdata = input("Enter new data: ")
Booklist[alter][part] = newdata
print(Booklist[alter])

file = open("Books.csv", "w")
x = 0
for row in Booklist:
    newrecord = Booklist[x][0] + ", " + Booklist[x][1] + ", " + Booklist[x][2] + "\n"
    file.write(newrecord)
    x = x+1
file.close()
```

**117**

```
import csv
import random

score = 0
name = input("What is your name? ")
q1_num1 = random.randint(10, 50)
q1_num2 = random.randint(10, 50)
question1 = str(q1_num1) + " + " + str(q1_num2) + " = "
ans1 = int(input(question1))
realans1 = q1_num1 + q1_num2
if ans1 == realans1:
    score = score + 1
q2_num1 = random.randint(10, 50)
q2_num2 = random.randint(10, 50)
question2 = str(q2_num1) + " + " + str(q2_num2) + " = "
ans2 = int(input(question2))
realans2 = q2_num1 + q2_num2
if ans2 == realans2:
    score = score + 1

file = open("QuizScore.csv", "a")
newrecord = name+","+question1+","+str(ans1)+","+question2+", "+str(ans2)+str(score)+"\n"
file.write(str(newrecord))

file.close()
```

# Подпрограммы

## Объяснение

**Подпрограммы** представляют собой блоки кода, которые решают определенные задачи и могут вызываться в любой момент во время работы программы для выполнения этого кода.

### Преимущества:

- Написанный вами блок кода может повторно использоваться в разных местах во время работы программы.
- Программа становится более понятной, так как код делится на блоки.

### Определение подпрограмм и передача переменных между подпрограммами

Ниже приведена простая программа, которую можно было бы написать и без подпрограмм. Здесь подпрограммы используются исключительно в демонстрационных целях:

```
def get_name():
    user_name = input("Enter your name: ")
    return user_name

def print_Msg(user_name):
    print("Hello", user_name)

def main():
    user_name = get_name()
    print_Msg(user_name)

main()
```

Программа состоит из трех подпрограмм, имеющих названия `get_name()`, `print_Msg()` и `main()`.

Подпрограмма `get_name()` предлагает пользователю ввести имя, а затем возвращает значение переменной `user_name`, чтобы оно могло использоваться в других подпрограммах. Это очень важно: если не вернуть значение из подпрограммы, то значения любых переменных, созданных или измененных в этой подпрограмме, нельзя будет использовать в других точках программы.



Подпрограмма `print_Msg()` выводит сообщение «Hello» и добавляет к нему имя пользователя. Переменная `user_name` заключена в скобки, так как текущее значение переменной импортируется в подпрограмму для последующего использования.

Подпрограмма `main()` получает значение `user_name` от подпрограммы `get_name()`, так как оно было возвращено подпрограммой `get_name()`. Затем переменная `user_name` используется в подпрограмме `print_Msg()`.

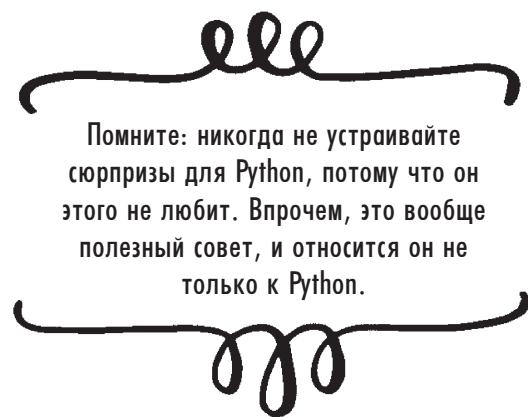
Последняя строка `main()` – это сама программа. Все, что она делает, – это запуск подпрограммы `main()`.

Разумеется, нет никакой необходимости так хитроумно запускать очень простую программу, но здесь этот пример приводится только для того, чтобы продемонстрировать структуру подпрограмм и механизм передачи и использования переменных в подпрограммах.



**Будьте внимательны:** Python не любит сюрпризы, так что если вы намереваетесь использовать подпрограмму в программе, Python должен заранее прочитать строку `def имя_подпрограммы()`, чтобы знать, где искать подпрограмму. Если вы попытаетесь вызвать подпрограмму до того, как Python получит информацию о ней, он впадет в панику и программа закроется.

Вызываемая подпрограмма должна быть записана **выше** той части кода, которая используется для ее вызова. Python читает программы сверху вниз и **выполняет** первую строку, которая не имеет отступа и не начинается со слова `def`. В приведенной выше программе это будет строка `main()`.



## Примеры кода

Все следующие примеры являются частью одной программы, поэтому они будут следовать в том порядке, в котором приводятся ниже.

```
def get_data():
    user_name = input("Enter your name: ")
    user_age = int(input("Enter your age: "))
    data_tuple = (user_name, user_age)
    return data_tuple
```

Определяет подпрограмму с именем **get\_data()**, которая запрашивает у пользователя его имя и возраст. Так как мы хотим передать сразу несколько наборов данных из одной части основной программы другим, их необходимо как-то сгруппировать. Команда **return** может содержать только одно значение; именно поэтому мы объединяем переменные **user\_name** и **user\_age** в кортеж (с. 63) с именем **data\_tuple**.

```
def message(user_name, user_age):
    if user_age <= 10:
        print("Hi ", user_name)
    else:
        print("Hello ", user_name)
```

Определяет подпрограмму с именем **message()**, которая использует две ранее определенные переменные (**user\_name** и **user\_age**).

```
def main():
    user_name, user_age = get_data()
    message(user_name, user_age)
```

Определяет подпрограмму **main()**, которая получает две переменные от подпрограммы **get\_data()**. Имена переменных должны следовать в том порядке, в котором они определяются в кортеже. Затем вызывается подпрограмма **message()** для выполнения задачи с этими двумя переменными.

**main()**

Запускает подпрограмму **main()**.



# Задачи

**118**

Определите подпрограмму, которая предлагает пользователю ввести число и сохраняет его в переменной **num**. Определите другую подпрограмму, которая использует значение **num** и проводит отсчет от 1 до этого числа.



**119**

Определите подпрограмму, которая предлагает пользователю выбрать большое и маленькое число, а затем генерирует случайное число из этого диапазона и сохраняет его в переменной с именем **comp\_num**.

**120**

Отобразите для пользователя следующее меню:

- 1) **Addition**
  - 2) **Subtraction**
- Enter 1 or 2:**

Если пользователь выбирает 1, запускается подпрограмма, генерирующая два случайных числа из диапазона между 5 и 20. Предложите пользователю сложить их. Рассчитайте правильный ответ и выведите его для пользователя вместе с его ответом.

Если он выбирает 2, должна запускаться подпрограмма, генерирующая случайное число между 25 и 50, а затем еще одно между 1 и 25. Попросите пользователя вычесть второе из первого: так ему не придется беспокоиться об отрицательных значениях. Выведите правильный ответ вместе с ответом пользователя.

Создайте еще одну подпрограмму, которая будет проверять совпадение ответа пользователя с правильным ответом. Если ответы совпали, выведите сообщение «Correct»; в противном случае выведите «Incorrect, the answer is» и правильный ответ.

Если пользователь ввел некорректное значение в самом первом меню, выведите соответствующее сообщение.

Определите другую подпрограмму, которая выводит сообщение «I am thinking of a number...», после чего предлагает пользователю угадать загаданное число.

Определите третью подпрограмму, которая проверяет, совпадает ли **comp\_num** с предположением пользователя. Если совпадает, то подпрограмма выводит сообщение «Correct, you win»; в противном случае цикл продолжается, а подпрограмма сообщает, больше или меньше их предположение загаданного числа, и предлагает сделать новую попытку до тех пор, пока пользователь его не угадает.

**121**

Напишите программу, которая помогает пользователю легко управлять списком имен. Программа должна выводить меню, дающее возможность добавлять, изменять и удалять имена из списка, а также отображать их все. Кроме того, в меню должна присутствовать команда для завершения работы программы. Если пользователь выбрал несуществующую команду, программа выводит соответствующее сообщение. После того как пользователь выбрал команду добавления, изменения или удаления имени или просмотра всех имен, меню должно выводиться снова без необходимости перезапуска программы. Программа должна быть по возможности простой и удобной в использовании.

**122**

Создайте следующее меню:

- 1) Add to file
- 2) View all records
- 3) Quit program

Enter the number of your selection:

Если пользователь выбрал вариант 1, данные должны добавляться в файл **Salaries.csv**, содержащий имена и зарплаты. Если пользователь выбрал вариант 2, программа выводит все записи из файла **Salaries.csv**. Если пользователь выбрал вариант 3, программа завершается. Если выбран несуществующий вариант, выводится сообщение об ошибке. Пользователь снова и снова возвращается к меню, пока не будет выбран вариант 3.

Меню упрощает работу с программой.

**123**

В языке Python невозможно напрямую удалить запись из файла **.csv**. Вместо этого приходится сохранять файл во временным списке, вносить в него изменения, а затем заменять исходный файл временным списком.

Измените предыдущую программу, чтобы она предоставляла такую возможность. Меню должно выглядеть так:

- 1) Add to file
- 2) View all records
- 3) Delete a record
- 4) Quit program

Enter the number of your selection:



## Ответы

118

```
def ask_value():
    num = int(input("Enter a number: "))
    return num

def count(num):
    n = 1
    while n <= num
        print(n)
        n = n + 1

def main():
    num = ask_value()
    count(num)

main()
```

119

```
import random

def pick_num():
    low = int(input("Enter the bottom of the range: "))
    high = int(input("Enter the top of the range: "))
    comp_num = random.randint(low, high)
    return comp_num

def first_guess():
    print("I am thinking of a number...")
    guess = int(input("What am I thinking of: "))
    return guess

def check_answer(comp_num, guess):
    try_again = True
    while try_again == True:
        if comp_num == guess:
            print("Correct, you win.")
            try_again = False
        elif comp_num > guess:
            guess = int(input("Too low, try again: "))
        else:
            guess = int(input("Too high, try again: "))

def main():
    comp_num = pick_num()
    guess = first_guess()
    check_answer(comp_num, guess)

main()
```

**120**

```
import random

def addition():
    num1 = random.randint(5, 20)
    num2 = random.randint(5, 20)
    print(num1, " + ", num2, " = ")
    user_answer = int(input("Your answer: "))
    actual_answer = num1 + num2
    answers = (user_answer, actual_answer)
    return answers

def subtraction():
    num3 = random.randint(25, 50)
    num4 = random.randint(1, 25)
    print(num3, " - ", num4, " = ")
    user_answer = int(input("Your answer: "))
    actual_answer = num3 - num4
    answers = (user_answer, actual_answer)
    return answers

def check_answer(user_answer, actual_answer):
    if user_answer == actual_answer:
        print("Correct")
    else:
        print("Incorrect, the answer is ", actual_answer)

def main():
    print("1) Addition")
    print("2) Subtraction")
    selection = int(input("Enter 1 or 2: "))
    if selection == 1:
        user_answer, actual_answer = addition()
        check_answer(user_answer, actual_answer)
    elif selection == 2:
        user_answer, actual_answer = subtraction()
        check_answer(user_answer, actual_answer)
    else:
        print("Incorrect selection")

main()
```

121

```
def add_name():
    name = input("Enter a new name: ")
    names.append(name)
    return names

def change_name():
    num = 0
    for x in names:
        print(num, x)
        num = num + 1
    select_num = int(input("Enter the number of the name you want to change: "))
    name = input("Enter new name: ")
    names[select_num] = name
    return names

def delete_name():
    num = 0
    for x in names:
        print(num, x)
        num = num + 1
    select_num = int(input("Enter the number of the name you want to delete: "))
    del names[select_num]
    return names

def view_names():
    for x in names:
        print(x)
    print()

def main():
    again = "y"
    while again == "y":
        print("1) Add a name")
        print("2) Change a name")
        print("3) Delete a name")
        print("4) View names")
        print("5) Quit")
        selection = int(input("What do you want to do? "))
        if selection == 1:
            names = add_name()
        elif selection == 2:
            names = change_name()
        elif selection == 3:
            names = delete_name()
        elif selection == 4:
            names = view_names()
        elif selection == 5:
            again = "n"
        else:
            print("Incorrect option: ")
    data = (names, again)

names = []
main()
```

**122**

```
import csv

def addtofile():
    file = open("Salaries.csv", "a")
    name = input("Enter name: ")
    salary = int(input("Enter salary: "))
    newrecord = name + ", " + str(salary) + "\n"
    file.write(str(newrecord))
    file.close()

def viewrecords():
    file = open("Salaries.csv", "r")
    for row in file:
        print(row)
    file.close()

tryagain = True
while tryagain == True:
    print("1) Add to file")
    print("2) View all records")
    print("3) Quit")
    print()
    selection = input("Enter the number of your selection: ")
    if selection == "1":
        addtofile()
    elif selection == "2":
        viewrecords()
    elif selection == "3":
        tryagain = False
    else:
        print("Incorrect option")
```

**123**

```
import csv

def addtofile():
    file = open("Salaries.csv", "a")
    name = input("Enter name: ")
    salary = int(input("Enter salary: "))
    newrecord = name + ", " + str(salary) + "\n"
    file.write(str(newrecord))
    file.close()

def viewrecords():
    file = open("Salaries.csv", "r")
    for row in file:
        print(row)
    file.close()

def deleterecord():
    file = open("Salaries.csv", "r")
    x = 0
    tmplist = []
    for row in file:
        tmplist.append(row)
    file.close()
    for row in tmplist:
        print(x, row)
        x = x + 1
    rowtodelete = int(input("Enter the row number to delete: "))
    del tmplist[rowtodelete]
    file = open("Salaries.csv", "w")
    for row in tmplist:
        file.write(row)
    file.close()

tryagain = True
while tryagain == True:
    print("1) Add to file")
    print("2) View all records")
    print("3) Delete a record")
    print()
    selection = input("Enter the number of your selection: ")
    if selection == "1":
        addtofile()
    elif selection == "2":
        viewrecords()
    elif selection == "3":
        deleterecord()
    elif selection == "4":
        tryagain = False
    else:
        print("Incorrect option")
```

# Графический интерфейс Tkinter

~~0/10  
0/x/x  
x/0/0~~

## Объяснение

**Графический интерфейс** (GUI, Graphic User Interface) упрощает использование программы. Он позволяет программисту создавать диалоговые окна, текстовые поля и кнопки, благодаря которым взаимодействие с программой становится более удобным для пользователя. **Tkinter** – библиотека Python, которая предоставляет функциональные возможности для создания графических интерфейсов.

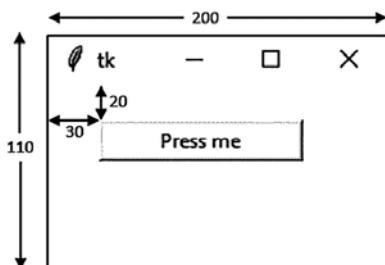
Взгляните на следующий код и особенно на единицы измерения в строках `window.geometry` и `button.place`.

```
from tkinter import *

def Call():
    msg = Label(window, text = "You pressed the button")
    msg.place(x = 30, y = 50)
    button["bg"] = "blue"
    button["fg"] = "white"

window = Tk()
window.geometry("200x110")
button = Button(text = "Press me", command = Call)
button.place(x = 30, y = 20, width = 120, height = 25)
window.mainloop()
```

А вот как выглядит окно, которое создается этим кодом:



Строка `geometry` определяет размер окна, а строка `place` – позицию конкретного элемента в этом окне.

При нажатии кнопки вызывается подпрограмма **Call**, а внешний вид окна изменяется:



# Примеры кода

```
from tkinter import *
```

Эта строка должна находиться в самом начале программы для импортирования библиотеки Tkinter.

```
window = Tk()  
window.title("Window Title")  
window.geometry("450x100")
```

Создает окно, которое будет использоваться для вывода (переменная **window**), добавляет заголовок и определяет размер окна.



```
label = Label(text = "Enter number: ")
```

Добавляет на экран текстовый элемент, содержащий заданное сообщение.

```
entry_box = Entry (text = 0)
```

Создает пустое текстовое поле. Текстовые поля используются для ввода или вывода данных.

```
output_box = Message(text = 0)
```

Создает область для вывода данных.

```
output_box ["bg"] = "red"
```

Задает цвет фона для объекта.

```
output_box ["fg"] = "white"
```

Задает цвет шрифта для объекта.

```
output_box ["relief"] = "sunken"
```

Задает стиль объемного обрамления элемента: отсутствие рамки (**flat**), эффект рельефа (**raised**), углубления (**sunken**), эффект углубленной рамки (**groove**) или рельефной рамки (**ridge**).

```
list_box = Listbox()
```

Создает раскрывающийся список, который может содержать только строки.



```
entry_box ["justify"] = "center"
```

Задает режим выравнивания текста в текстовом поле, но не работает для областей вывода данных.

```
button1 = Button(text = "Click here", command = click)
```

Создает кнопку для запуска подпрограммы **click**.

```
label.place(x = 50, y = 20, width = 100, height = 25)
```

Задает позицию, в которой объект будет располагаться в окне. Если позиция не задана, то объект не отобразится в окне.

```
entry_box.delete(0, END)
```

Удаляет содержимое отдельного элемента или всего списка.

```
num = entry_box.get()
```

Сохраняет содержимое поля ввода текста и сохраняет его в переменной **num**. Не работает для областей вывода данных.

```
answer = output_txt["text"]
```

Получает содержимое области вывода данных и сохраняет его в переменной **answer**. Не работает для полей текстового ввода.

```
output_txt["text"] = total
```

Изменяет содержимое области вывода данных для отображения значения переменной **total**.

```
window.mainloop()
```

Чтобы ваша программа работала, она должна завершаться этой командой.



С каждой задачей, которую  
вы решаете, растет ваша  
квалификация программиста.



# Задачи

**124**

Создайте окно, которое предлагает пользователю ввести имя. Когда пользователь нажимает кнопку, в окне должно выводиться сообщение «Hello [имя]» с изменением цвета фона и цвета шрифта текстовой области.

**127**

Создайте окно, которое предлагает пользователю ввести имя в текстовом поле. Когда пользователь нажимает кнопку, имя добавляется в конец списка, выводимого на экране. Создайте еще одну кнопку для очистки списка.

**129**

Создайте окно, которое предлагает пользователю ввести число в текстовом поле. При нажатии кнопки используйте конструкцию `variable.isdigit()` для проверки того, является ли число целым. Если проверка дает положительный результат, оно добавляется в список; в противном случае содержимое текстового поля стирается. Создайте дополнительную кнопку для очистки списка.

**131**

Создайте программу, которая предоставляет пользователю возможность создать новый файл `.CSV`. Программа предлагает ввести имя и возраст, а затем добавляет введенные данные в конец только что созданного файла.

**125**

Напишите программу, моделирующую бросок шестигранного кубика в настольной игре. Когда пользователь нажимает кнопку, на экране должно выводиться случайное целое число от 1 до 6 (включительно).

**126**

Напишите программу, которая предлагает пользователю ввести число в текстовом поле. Когда пользователь нажимает кнопку, это число прибавляется к накапливаемой сумме и выводится в другом поле. Ввод может повторяться сколько угодно раз на усмотрение пользователя, при этом вводимые данные будут прибавляться к сумме. В окне должна присутствовать еще одна кнопка, которая обнуляет накапливаемую сумму и стирает содержимое исходного поля, чтобы пользователь мог начать ввод заново.

**128**

1 километр = 0,6214 мили, а 1 миля = 1,6093 километра. Напишите программу, которая преобразует километры в мили и наоборот.

**130**

Измените программу 129, добавив третью кнопку для сохранения списка в файле `.CSV`. Команда `tmp_list = num_list.get(0,END)` может использоваться для сохранения содержимого списка в кортеже с именем `tmp_list`.

**132**

Используя файл `.CSV`, созданный для последней задачи, напишите программу, которая предлагает пользователю добавлять записи с именами и возрастом разных людей. Создайте кнопку для вывода содержимого файла `.CSV` посредством импортирования его данных в список.

## Ответы

**124**

```
from tkinter import *

def click():
    name = textbox1.get()
    message = str("Hello " + name)
    textbox2["bg"] = "yellow"
    textbox2["fg"] = "blue"
    textbox2["text"] = message

window = Tk()
window.geometry("500x200")

label1 = Label(text = "Enter your name: ")
label1.place(x = 30, y = 20)

textbox1 = Entry(text = "")
textbox1.place(x = 150, y = 20, width = 200, height = 25)
textbox1["justify"] = "center"
textbox1.focus()

button1 = Button(text = "Press me", command = click)
button1.place(x = 30, y = 50, width = 120, height = 25)

textbox2 = Message(text = "")
textbox2.place(x = 150, y = 50, width = 200, height = 25)
textbox2["bg"] = "white"
textbox2["fg"] = "black"

window.mainloop()
```

**125**

```
from tkinter import *
import random

def click():
    num = random.randint(1, 6)
    answer["text"] = num

window = Tk()
window.title("Roll a dice")
window.geometry("100x120")

button1 = Button(text = "Roll", command = click)
button1.place(x = 30, y = 30, width = 50, height = 25)

answer = Message(text = "")
answer.place(x = 40, y = 70, width = 30, height = 25)

window.mainloop()
```

**126**

```
from tkinter import *

def add_on():
    num = enter_txt.get()
    num = int(num)
    answer = output_txt["text"]
    answer = int(answer)
    total = num + answer
    output_txt["text"] = total

def reset():
    total = 0
    output_txt["text"] = 0
    enter_txt.delete(0, END)
    enter_txt.focus()

total = 0
num = 0

window = Tk()
window.title("Adding Together")
window.geometry("450x100")

enter_lbl = Label(text = "Enter a number:")
enter_lbl.place(x = 50, y = 20, width = 100, height = 25)

enter_txt = Entry(text = 0)
enter_txt.place(x = 150, y = 20, width = 100, height = 25)
enter_txt["justify"] = "center"
enter_txt.focus()

add_btn = Button(text = "Add", command = add_on)
add_btn.place(x = 300, y = 20, width = 50, height = 25)

output_lbl = Label(text = "Answer = ")
output_lbl.place(x = 50, y = 50, width = 100, height = 25)

output_txt = Message(text = 0)
output_txt.place(x = 150, y = 50, width = 100, height = 25)
output_txt["bg"] = "white"
output_txt["relief"] = "sunken"

clear_btn = Button(text = "Clear", command = reset)
clear_btn.place(x = 300, y = 50, width = 50, height = 25)

window.mainloop()
```

**127**

```
from tkinter import *

def add_name():
    name = name_box.get()
    name_list.insert(END, name)
    name_box.delete(0, END)
    name_box.focus()

def clear_list():
    name_list.delete(0, END)
    name_box.focus()

window = Tk()
window.title("Names list")
window.geometry("400x200")

label1 = Label(text = "Enter a name: ")
label1.place(x = 20, y = 20, width = 100, height = 25)

name_box = Entry(text = 0)
name_box.place(x = 120, y = 20, width = 100, height = 25)
name_box.focus()

button1 = Button(text = "Add to list", command = add_name)
button1.place(x = 250, y = 20, width = 100, height = 25)

name_list = Listbox()
name_list.place(x = 120, y = 50, width = 100, height = 100)

button2 = Button(text = "Clear list", command = clear_list)
button2.place(x = 250, y = 50, width = 100, height = 25)

window.mainloop()
```

**128**

```
def convert2():
    km = textbox1.get()
    km = int(km)
    message = km * 0.6214
    textbox2.delete(0, END)
    textbox2.insert(END, message)
    textbox2.insert(END, " miles")

window = Tk()
window.title("Distance")
window.geometry("260x200")

label1 = Label(text = "Enter the value you want to convert: ")
label1.place(x = 30, y = 20)

textbox1 = Entry(text = "")
textbox1.place(x = 30, y = 50, width = 200, height = 25)
textbox1["justify"] = "center"
textbox1.focus()

convert1 = Button(text = "Convert miles to km", command = convert1)
convert1.place(x = 30, y = 80, width = 200, height = 25)

convert2 = Button(text = "Convert km to miles", command = convert2)
convert2.place(x = 30, y = 110, width = 200, height = 25)

textbox2 = Entry(text = "")
textbox2.place(x = 30, y = 140, width = 200, height = 25)
textbox2["justify"] = "center"

window.mainloop()
```

**129**

```
from tkinter import *

def add_number():
    num = num_box.get()
    if num.isdigit():
        num_list.insert(END, num)
        num_box.delete(0, END)
        num_box.focus()
    else:
        num_box.delete(0, END)
        num_box.focus()

def clear_list():
    num_list.delete(0, END)
    num_box.focus()

window = Tk()
window.title("Number list")
window.geometry("400x200")

label1 = Label(text = "Enter a number: ")
label1.place(x = 20, y = 20, width = 100, height = 25)

num_box = Entry(text = 0)
num_box.place(x = 120, y = 20, width = 100, height = 25)
num_box.focus()

button1 = Button(text = "Add to list", command = add_number)
button1.place(x = 250, y = 20, width = 100, height = 25)

num_list = Listbox()
num_list.place(x = 120, y = 50, width = 100, height = 100)

button2 = Button(text = "Clear list", command = clear_list)
button2.place(x = 250, y = 50, width = 100, height = 25)

window.mainloop()
```

**130**

```
from tkinter import *
import csv

def add_number():
    num = num_box.get()
    if num.isdigit():
        num_list.insert(END, num)
        num_box.delete(0, END)
        num_box.focus()
    else:
        num_box.delete(0, END)
        num_box.focus()

def clear_list():
    num_list.delete(0, END)
    num_box.focus()

def save_list():
    file = open("numbers.csv", "w")
    tmp_list = num_list.get(0, END)
    item = 0
    for x in tmp_list:
        newrecord = tmp_list[item] + "\n"
        file.write(str(newrecord))
        item = item + 1
    file.close()

window = Tk()
window.title("Number list")
window.geometry("400x200")

label1 = Label(text = "Enter a number: ")
label1.place(x = 20, y = 20, width = 100, height = 25)

num_box = Entry(text = 0)
num_box.place(x = 120, y = 20, width = 100, height = 25)
num_box.focus()

button1 = Button(text = "Add to list", command = add_number)
button1.place(x = 250, y = 20, width = 100, height = 25)

num_list = Listbox()
num_list.place(x = 120, y = 50, width = 100, height = 100)

button2 = Button(text = "Clear list", command = clear_list)
button2.place(x = 250, y = 50, width = 100, height = 25)

button3 = Button(text = "Save list", command = save_list)
button3.place(x = 250, y = 80, width = 100, height = 25)

window.mainloop()
```

**131**

```
from tkinter import *
import csv

def create_new():
    file = open("ages.csv", "w")
    file.close()

def save_list():
    file = open("ages.csv", "a")
    name = name_box.get()
    age = age_box.get()
    newrecord = name + ", " + age + "\n"
    file.write(str(newrecord))
    file.close()
    name_box.delete(0, END)
    age_box.delete(0, END)
    name_box.focus()

window = Tk()
window.title("People list")
window.geometry("400x100")

label1 = Label(text = "Enter a name: ")
label1.place(x = 20, y = 20, width = 100, height = 25)

name_box = Entry(text = "")
name_box.place(x = 120, y = 20, width = 100, height = 25)
name_box["justify"] = "left"
name_box.focus()

label2 = Label(text = "Enter their age: ")
label2.place(x = 20, y = 50, width = 100, height = 25)

age_box = Entry(text = "")
age_box.place(x = 120, y = 50, width = 100, height = 25)
age_box["justify"] = "left"

button1 = Button(text = "Create new file", command = create_new)
button1.place(x = 250, y = 20, width = 100, height = 25)

button2 = Button(text = "Add to file", command = save_list)
button2.place(x = 250, y = 50, width = 100, height = 25)

window.mainloop()
```

**132**

```
from tkinter import *
import csv

def save_list():
    file = open("ages.csv", "a")
    name = name_box.get()
    age = age_box.get()
    newrecord = name + ", " + age + "\n"
    file.write(str(newrecord))
    file.close()
    name_box.delete(0, END)
    age_box.delete(0, END)
    name_box.focus()

def read_list():
    name_list.delete(0, END)
    file = list(csv.reader(open("ages.csv")))
    tmp = []
    for row in file:
        tmp.append(row)
    x = 0
    for i in tmp:
        data = tmp[x]
        name_list.insert(END, data)
        x = x + 1

window = Tk()
window.title("People list")
window.geometry("400x200")

label1 = Label(text = "Enter a name: ")
label1.place(x = 20, y = 20, width = 100, height = 25)

name_box = Entry(text = "")
name_box.place(x = 120, y = 20, width = 100, height = 25)
name_box["justify"]="left"
name_box.focus()

label2 = Label(text = "Enter their age: ")
label2.place(x = 20, y = 50, width = 100, height = 25)

age_box = Entry(text = "")
age_box.place(x = 120, y = 50, width = 100, height = 25)
age_box["justify"]="left"

button1 = Button(text = "Add to file", command = save_new)
button1.place(x = 250, y = 20, width = 100, height = 25)

button2 = Button(text = "Read list", command = read_list)
button2.place(x = 250, y = 50, width = 100, height = 25)

label3 = Label(text = "Saved names: ")
label3.place(x = 20, y = 80, width = 100, height = 25)

name_list = Listbox()
name_list.place(x = 120, y = 80, width = 230, height = 100)

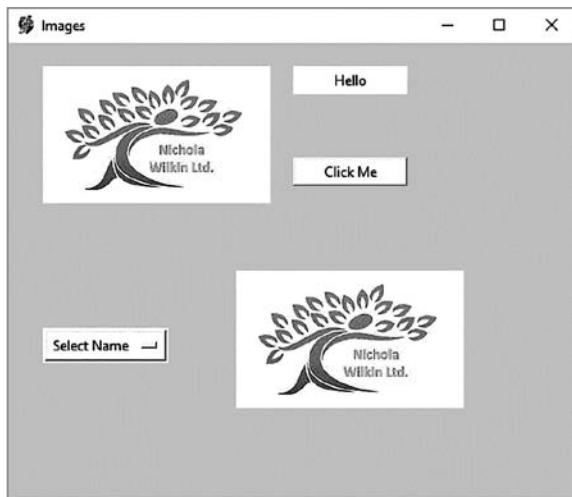
window.mainloop()
```

# Подробнее о Tkinter

## Объяснение



В этом разделе рассматривается создание **графических интерфейсов** с расширенными возможностями. Материал строится на базе того, что вы узнали в предыдущем разделе.



На этом экране мы:

- изменили значок в полосе заголовка;
- изменили цвет фона основного окна;
- добавили статическое изображение логотипа в левую верхнюю часть;
- создали надпись, на которой в настоящее время выводится текст «Hello»;
- добавили кнопку Click Me;
- добавили раскрывающийся список Select Name с тремя именами: Bob, Sue и Tim;
- добавили в нижней части окна второе изображение, которое будет заменяться фотографией человека, выбранного из списка, когда пользователь нажимает кнопку Click Me.

Ваши программы становятся  
все более впечатляющими.



Весь функционал этого окна может быть реализован с помощью кода, который рассматривался в предыдущем разделе, а также того, который будет рассмотрен ниже.

При использовании графики в программе изображения проще всего хранить в одной папке с программой. Если они хранятся в другом месте, то вам придется задавать местонахождение файла следующим образом:

```
logo = PhotoImage(file="c:\\Python34\\images\\logo.gif")
```

Если же изображение хранится в одной папке с программой, достаточно указать только имя файла:

```
logo = PhotoImage(file="logo.gif")
```

**Обратите внимание:** для изображений в Tkinter могут использоваться только типы файлов GIF или PGM/PPM, так как другие типы не поддерживаются. Прежде чем браться за создание программы, убедитесь в том, что изображения хранятся в подходящем формате и с подходящим именем — это упростит вам жизнь.



## Примеры кода



```
window.wm_iconbitmap("MyIcon.ico")
```

Изменяет значок в заголовке окна.

```
window.configure(background = "light green")
```

Изменяет цвет фона окна (в данном случае на светло-зеленый).

```
logo = PhotoImage(file = "logo.gif")
logoimage = Label(image = logo)
logoimage.place(x = 30, y = 20, width = 200, height = 120)
```

Выводит изображение в виджете **Label**. Это изображение не изменяется во время выполнения программы.



```
photo = PhotoImage(file = "logo.gif")
photobox = Label(window, image = photo)
photobox.image = photo
photobox.place(x = 30, y = 20, width = 200, height = 120)
```

Этот блок похож на приведенный выше, но так как мы хотим, чтобы изображение могло изменяться при обновлении данных, необходимо добавить команду **photobox.image = photo** — с ней изображение может обновляться.

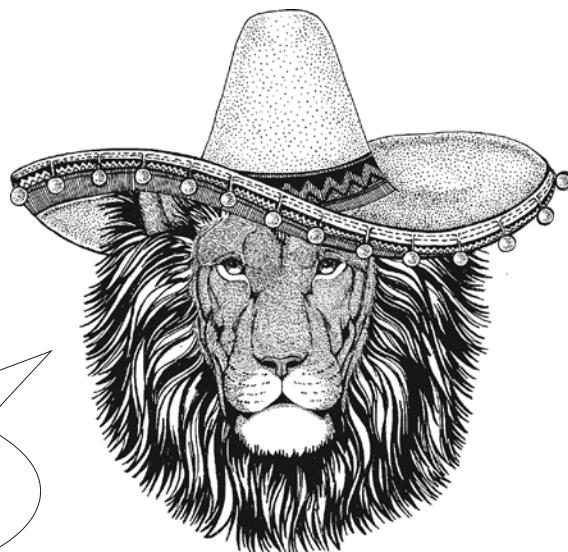
```
selectName = StringVar(window)
selectName.set("Select Name")
namesList = OptionMenu(window, selectName, "Bob", "Sue", "Tim")
namesList.place(x = 30, y = 250)
```

Создает переменную **selectName**, с которой изначально связывается строка «Select Name». Затем создается раскрывающийся список, в котором отображается значение из переменной **selectName**, и добавляются значения этого списка: **Bob**, **Sue** и **Tim**.



```
def clicked():
    sel = selectName.get()
    mesg = "Hello " + sel
    mlabel["text"] = mesg
    if sel == "Bob":
        photo = PhotoImage(file = "Bob.gif")
        photobox.image = photo
    elif sel == "Sue":
        photo = PhotoImage(file = "Sue.gif")
        photobox.image = photo
    elif sel == "Tim":
        photo = PhotoImage(file = "Tim.gif")
        photobox.image = photo
    else:
        photo = PhotoImage(file = "logo.gif")
        photobox.image = photo
    photobox["image"] = photo
    photobox.update()
```

В этом примере при нажатии кнопки выполняется подпрограмма **clicked**. Она получает значение из переменной **selectName** и создает сообщение, которое будет отображаться в надписи. Затем подпрограмма проверяет, какой вариант был выбран, и заменяет текущее изображение другим, соответствующим выбранному — оно выводится при помощи переменной **photo**. Если в списке не выбрано ни одно имя, то отображается логотип.

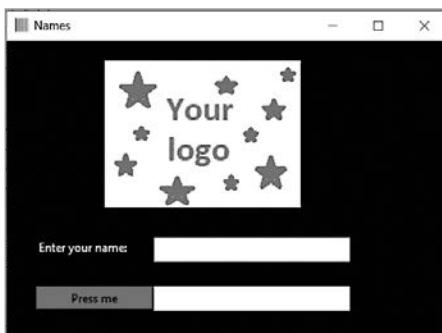


И не забывайте:  
если вы не знаете, что делать,  
вернитесь к более ранним  
программам — возможно,  
они вам помогут.

# Задачи

**133**

Создайте собственный значок для полосы заголовка, состоящий из нескольких вертикальных разноцветных полос. Создайте логотип с размерами 200×150 в Paint или другом графическом редакторе. Создайте следующее окно, используя собственный значок и логотип.



Когда пользователь введет свое имя и нажмет кнопку **Press Me**, во втором поле должно быть выведено приветствие «Hello [имя]»

**136**

Напишите программу, которая предлагает пользователю ввести имя, а затем выбрать пол человека из раскрывающегося списка. При нажатии кнопки разделенные запятыми имя и пол добавляются в список.



**134**

Напишите новую программу, которая генерирует два случайных целых числа в диапазоне от 10 до 50. Программа предлагает пользователю сложить числа и ввести ответ. Если пользователь дает правильный ответ, выведите подходящее изображение (например, галочку); если ответ будет ошибочным, выведите другое подходящее изображение (например, крестик). Чтобы получить другой вопрос, пользователь должен нажать кнопку **Next**.

**135**

Напишите простую программу, которая выводит раскрывающийся список, содержащий названия нескольких цветов, и кнопку **Click Me**. Когда пользователь выбирает цвет из списка и нажимает кнопку, цвет фона окна заменяется выбранным цветом. Чтобы задача была более интересной, попробуйте обойтись без использования инструкции **if**.

**137**

Измените программу 136, чтобы новые значения имени и пола, добавляемые в список, также записывались в текстовый файл. Добавьте еще одну кнопку, которая будет выводить весь текстовый файл в главном окне командной оболочки Python.

**138**

Сохраните несколько изображений в папке, в которой находится ваша программа; присвойте им имена **1.gif**, **2.gif**, **3.gif** и т. д. Убедитесь в том, что все файлы хранятся в формате GIF. Выберите один из них в окне и предложите пользователю ввести число. Введенное число используется для выбора правильного имени файла и вывода изображения.

# Ответы

**133**

```
from tkinter import *

def click():
    name = textbox1.get()
    message = str("Hello " + name)
    textbox2["text"] = message

window = Tk()
window.title("Names")
window.geometry("450x350")
window.wm_iconbitmap("stripes.ico")
window.configure(background = "black")

logo = PhotoImage(file = "Mylogo.gif")
logoimage = Label(image = logo)
logoimage.place(x = 100, y = 20, width = 200, height = 150)

label1 = Label(text = "Enter your name: ")
label1.place(x = 30, y = 200)
label1["bg"] = "black"
label1["fg"] = "white"

textbox1 = Entry(text = "")
textbox1.place(x = 150, y = 200, width = 200, height = 25)
textbox1["justify"] = "center"
textbox1.focus()

button1 = Button(text = "Press me", command = click)
button1.place(x = 30, y = 250, width = 120, height = 25)
button1["bg"] = "yellow"

textbox2 = Message(text = "")
textbox2.place(x = 150, y = 250, width = 200, height = 25)
textbox2["bg"] = "white"
textbox2["fg"] = "black"

window.mainloop()
```

**134**

```
from tkinter import *
import random

def checkans():
    theirans = ansbox.get()
    theirans = int(theirans)
    num1 = num1box["text"]
    num1 = int(num1)
    num2 = num2box["text"]
    num2 = int(num2)
    ans = num1 + num2
    if theirans == ans:
        img = PhotoImage(file = "correct.gif")
        imgbx.image = img
    else:
        img = PhotoImage(file = "wrong.gif")
        imgbx.image = img
    imgbx["image"] = img
    imgbx.update()

def nextquestion():
    ansbox.delete(0, END)
    num1 = random.randint(10, 50)
    num1box["text"] = num1
    num2 = random.randint(10, 50)
    num2box["text"] = num2
    img = PhotoImage(file = "")
    imgbx.image = img
    imgbx["image"] = img
    imgbx.update()

window = Tk()
window.title("Addition")
window.geometry("250x300")

num1box = Label(text = "0")
num1box.place(x = 50, y = 30, width = 25, height = 25)
addsymbl = Message(text = "+")
addsymbl.place(x = 75, y = 30, width = 25, height = 25)
num2box = Label(text = "0")
num2box.place(x = 100, y = 30, width = 25, height = 25)
eqlsymbl = Message(text = "=")
eqlsymbl.place(x = 125, y = 30, width = 25, height = 25)
```

Продолжение на следующей странице

```
ansbox = Entry(text = "")  
ansbox.place(x = 150, y = 30, width = 25, height = 25)  
ansbox["justify"] = "center"  
ansbox.focus()  
checkbtn = Button(text = "Check", command = checkans)  
checkbtn.place(x = 50, y = 60, width = 75, height = 25)  
nextbtn = Button(text = "Next", command = nextquestion)  
nextbtn.place(x = 130, y = 60, width = 75, height = 25)  
img = PhotoImage(file = "")  
imgbx = Label(image = img)  
imgbx.image = img  
imgbox.place(x = 25, y = 100, width = 200, height = 150)  
  
nextquestion()  
  
window.mainloop()
```

### 135

```
from tkinter import *  
  
def clicked():  
    sel = selectcolour.get()  
    window.configure(background = sel)  
  
window = Tk()  
window.title("background")  
window.geometry("200x200")  
  
selectcolour = StringVar(window)  
selectcolour.set("Grey")  
  
colourlist = OptionMenu(window, selectcolour, "Grey", "Red", "Blue", "Green", "Yellow")  
colourlist.place(x = 50, y = 30)  
  
clickme = Button(text = "Click Me", command = clicked)  
clickme.place(x = 50, y = 150, width = 60, height = 30)  
  
mainloop()
```

**136**

```
from tkinter import *

def add_to_list():
    name = namebox.get()
    namebox.delete(0, END)
    genderselection = gender.get()
    gender.set("M/F")
    newdata = name + ", " + genderselection + "\n"
    name_list.insert(END, newdata)
    namebox.focus()

window = Tk()
window.title("People list")
window.geometry("400x400")

namelbl = Label(text = "Enter their name: ")
namelbl.place(x = 50, y = 50, width = 100, height = 25)
namebox = Entry(text = "")
namebox.place(x = 150, y = 50, width = 150, height = 25)
namebox.focus()

genderlbl = Label(text = "Select Gender")
genderlbl.place(x = 50, y = 100, width = 100, height = 25)
gender = StringVar(window)
gender.set("M/F")
gendermenu = OptionMenu(window, gender, "M", "F")
gendermenu.place(x = 150, y = 100)

name_list = Listbox()
name_list.place(x = 150, y = 150, width = 150, height = 100)

addbtn = Button(text = "Add to List", command = add_to_list)
addbtn.place(x = 50, y = 300, width = 100, height = 25)

window.mainloop()
```

**137**

```
from tkinter import *

def add_to_list():
    name = namebox.get()
    namebox.delete(0, END)
    genderselection = gender.get()
    gender.set("M/F")
    newdata = name + ", " + genderselection + "\n"
    name_list.insert(END, newdata)
    namebox.focus()
    file = open("names.txt", "a")
    file.write(newdata)
    file.close()

def print_list():
    file = open("names.txt", "r")
    print(file.read())

window = Tk()
window.title("People list")
window.geometry("400x400")

namelbl = Label(text = "Enter their name:")
namelbl.place(x = 50, y = 50, width = 100, height = 25)
namebox = Entry(text = "")
namebox.place(x = 150, y = 50, width = 150, height = 25)
namebox.focus()

genderlbl = Label(text = "Select Gender")
genderlbl.place(x = 50, y = 100, width = 100, height = 25)
gender = StringVar(window)
gender.set("M/F")
gendermenu = OptionMenu(window, gender, "M", "F")
gendermenu.place(x = 150, y = 100)

name_list = Listbox()
name_list.place(x = 150, y = 150, width = 150, height = 100)

addbtn = Button(text = "Add to List", command = add_to_list)
addbtn.place(x = 50, y = 300, width = 100, height = 25)

printlst = Button(text = "Print List", command = print_list)
printlst.place(x = 175, y = 300, width = 100, height = 25)

window.mainloop()
```

**138**

```
from tkinter import *

def clicked():
    num = selection.get()
    artref = num + ".gif"
    photo = PhotoImage(file = artref)
    photobox.image = photo
    photobox["image"] = photo
    photobox.update()

window = Tk()
window.title("Art")
window.geometry("400x350")

art = PhotoImage(file = "1.gif")
photobox = Label(window, image = art)
photobox.image = art
photobox.place(x = 100, y = 20, width = 200, height = 150)

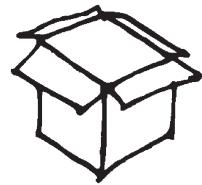
label = Label(text = "Select Art Number: ")
label.place(x = 50, y = 200, width = 100, height = 25)

selection = Entry(text = "")
selection.place(x = 200, y = 200, width = 100, height = 25)
selection.focus()

button = Button(text = "See Art", command = clicked)
button.place(x = 150, y = 250, width = 100, height = 25)

window.mainloop()
```

# SQLite



## Объяснение

**SQL** (Structured Query Language, то есть «язык структурированных запросов») – основной язык, используемый пакетами больших баз данных. **SQLite** – бесплатный программный продукт, который может использоваться в качестве базы данных SQL. Новейшую версию продукта можно загрузить с сайта [www.sqlite.org](http://www.sqlite.org).

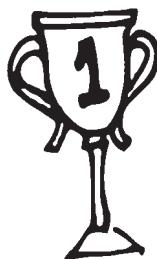
The screenshot shows the official SQLite download page. At the top is the SQLite logo. Below it is a navigation bar with links: Home, About, Documentation, Download, License, Support, and Purchase. A main text block states: "SQLite is a self-contained, high-reliability, embedded, full-featured, public-domain, SQL database engine. SQLite is the most used database engine in the world." It includes a "More Info" link. Below this is a section titled "Latest Release: Version 3.19.3 (2017-06-08)." It contains links for "Download" and "Prior Releases".

На странице загрузки необходимо выбрать один из вариантов Precompiled Binaries для Mac OS или Windows, включающий оболочку командной строки (command-line shell).

This screenshot shows the "Precompiled Binaries" section of the SQLite download page. It is divided into two main sections: "Precompiled Binaries for Mac OS X (x86)" and "Precompiled Binaries for Windows".

- Precompiled Binaries for Mac OS X (x86):**
  - sqlite-tools-osx-x86-** A bundle of command-line tools for managing SQLite database files, including the [command-line shell](#) program, the [sqldiff](#) program, [3190300.zip](#) (1.14 MB) (sha1: 1012be9d387f2d0adb7b27e59676046566c798c)
- Precompiled Binaries for Windows:**
  - sqlite-dll-win32-x86-** 32-bit DLL (x86) for SQLite version 3.19.3. [3190300.zip](#) (434.80 KB) (sha1: 92df84c8f528cf92993f346a7b3375f92419b7c)
  - sqlite-dll-win64-x64-** 64-bit DLL (x64) for SQLite version 3.19.3. [3190300.zip](#) (722.63 KB) (sha1: 80024d5736996e07bac72fbfd6d0b0cd59d2782a)
  - sqlite-tools-win32-x86-** A bundle of command-line tools for managing SQLite database files, including the [command-line shell](#) program, the [sqldiff.exe](#) program, and the [sqlite3\\_analyzer.exe](#) program. [3190300.zip](#) (1.56 MB) (sha1: 21a42e8103a5a49a7305af2f58174cebeb34d1c6)

Чтобы использовать SQL, необходимо загрузить программу DB Browser for SQLite с сайта <https://sqlitebrowser.org>.



The screenshot shows the DB Browser for SQLite application window. The interface includes a menu bar with File, Edit, View, Help. Below the menu is a toolbar with New Database, Open Database, Write Changes, and Revert Changes. The main area has tabs for Database Structure, Browse Data, Edit Prefs, and Execute SQL. On the right side, there are panes for DB Schema (listing tables), SQL Log, and an Edit Database Cell dialog. The bottom status bar shows "Type of data currently in cell: NULL" and "Apply".

# Понятие о реляционных базах данных

Для примера мы воспользуемся данными небольшой производственной компании, которая хранит информацию о своих работниках в базе данных **SQL**.

Ниже приведена таблица Employees для хранения информации обо всех работниках. Содержимое таблицы можно просмотреть на вкладке Browse Data.

Table: Employees				
	ID	Name	Dept	Salary
	Filter	Filter	Filter	Filter
1	1	Bob	Sales	25000
2	2	Sue	IT	28500
3	3	Tim	Sales	25000
4	4	Anne	Admin	18500
5	5	Paul	IT	28500
6	6	Simon	Sales	22000
7	7	Karen	Manufacturing	18500
8	8	Mark	Manufacturing	19000
9	9	George	Manufacturing	18500
10	10	Keith	Manufacturing	15000



Таблица состоит из четырех полей (ID, Name, Dept и Salary) и десяти записей (по одной для каждого работника.) Присмотревшись к списку работников, вы заметите, что таблица включает несколько человек с одинаковыми значениями отдела (Dept). Подобные повторения встречаются в большинстве баз данных. Чтобы база данных работала более эффективно, повторяющиеся записи часто хранятся в отдельной таблице. В данном примере можно создать таблицу всех отделов, в которой хранится вся информация о каждом из них, чтобы ее не приходилось полностью повторять для каждого работника.

Table: Departments

Dept	Manager
Filter	Filter
1 Manufacturing	Kenith
2 Sales	James
3 IT	Connor
4 Admin	Sally

На схеме изображена таблица Departments с информацией об отделах. Мы упростили таблицу, чтобы в ней хранился только один фрагмент данных для каждого отдела (в данном случае имя начальника). Тем не менее она сохраняет необходимость вводить имя начальника при каждой записи, что мы бы и делали, будь все данные в одной большой таблице.

Благодаря разделению данных на две таблицы мы сможем в случае необходимости обновить имя начальника в одном месте вместо того, чтобы проделывать это несколько раз при хранении данных в единой таблице.

Эта особенность называется **«один ко многим»**, так как несколько работников могут относиться к одному отделу.



**Первичный ключ** – это поле в таблице (обычно первое), в котором хранится уникальный идентификатор определенной записи. Таким образом, в таблице Employees первичным ключом является столбец ID, а в таблице Departments – столбец Dept.

При создании таблицы для каждого поля необходимо указать следующую информацию:

- имя поля (оно не может включать пробелы и должно подчиняться тем же правилам, что и имена переменных);
- является ли оно первичным ключом;
- тип данных поля.

Поддерживаются следующие типы данных:

- **integer**: целочисленные значения;
- **real**: значения с плавающей точкой;
- **text**: значения, представляющие собой текстовые строки;
- **blob**: значения, которые хранятся точно в таком виде, в котором они были введены.



Также можно указать, что поле обязательно должно содержать значение, то есть не может быть пустым; для этого в конец поля при создании следует добавить выражение **NOT NULL**.

## Примеры кода

```
import sqlite3
```

Эта строка должна находиться в самом начале программы, чтобы Python мог использовать библиотеку SQLite3.

```
with sqlite3.connect("company.db") as db:  
    cursor=db.cursor()
```

Подключается к базе данных **company**. Если база данных не существует, то она будет создана. Файл будет создан в одной папке с программой.

```
cursor.execute("""CREATE TABLE IF NOT EXISTS employees(  
    id integer PRIMARY KEY,  
    name text NOT NULL,  
    dept text NOT NULL,  
    salary integer);""")
```

Создает таблицу **employees** с четырьмя полями (**id**, **name**, **dept** и **salary**). Задает тип данных каждого поля, определяет, какое поле является первичным ключом и какие поля нельзя оставлять пустыми. Тройные кавычки обеспечивают перенос кода по строкам, чтобы он легче воспринимался (вместо вывода в одну строку).

```
cursor.execute("""INSERT INTO employees(id, name, dept, salary)  
    VALUES("1", "Bob", "Sales", "25000")""")  
db.commit()
```

Вставляет данные в таблицу **employees**. Команда **db.commit()** сохраняет изменения.

```
newID = input("Enter ID number: ")  
newName = input("Enter name: ")  
newDept = input("Enter department: ")  
newSalary = input("Enter salary: ")  
cursor.execute("""INSERT INTO employees(id, name, dept, salary)  
    VALUES(?, ?, ?, ?)""", (newID, newName, newDept, newSalary))  
db.commit()
```

Запрашивает у пользователя новые данные, которые позже вставляются в таблицу.

```
cursor.execute("SELECT * FROM employees")  
print(cursor.fetchall())
```

Выводит все данные из таблицы **employees**.

```
db.close()
```

Это должна быть последняя строка в программе для закрытия базы данных.



```
cursor.execute("SELECT * FROM employees")
for x in cursor.fetchall():
    print(x)
```

Выводит все данные из таблицы **employees** и выводит каждую запись в отдельной строке.



```
cursor.execute("SELECT * FROM employees ORDER BY name")
for x in cursor.fetchall():
    print(x)
```

Выбирает все записи из таблицы **employees**, сортирует их по имени и выводит каждую запись в отдельной строке.

```
cursor.execute("SELECT * FROM employees WHERE salary > 20000")
```

Выбирает из таблицы **employees** все записи, у которых значение поля **salary** превышает 20 000.

```
cursor.execute("SELECT * FROM employees WHERE dept = 'Sales'")
```

Выбирает из таблицы **employees** все записи, у которых поле **dept** содержит значение «Sales».

```
cursor.execute("""SELECT employees.id, employees.name, dept.manager
    FROM employees.dept WHERE employees.dept = dept.dept
    AND employees.salary > 20000""")
```

Выбирает поля **ID** и **name** из таблицы **employees**, и поле **manager** из таблицы **dept**, если значение поля **salary** превышает 20 000.

```
cursor.execute("SELECT id, name, salary FROM employees")
```

Выбирает поля **ID**, **name** и **salary** из таблицы **employees**.



```
whichDept = input("Enter a department: ")
cursor.execute("SELECT * FROM employees WHERE dept=?",[whichDept])
for x in cursor.fetchall():
    print(x)
```

Разрешает пользователю ввести название отдела и выводит записи всех работников заданного отдела.

```
cursor.execute("""SELECT employees.id, employees.name, dept.manager
    FROM employees, dept WHERE employees.dept=dept.dept""")
```

Выбирает поля **ID** и **name** из таблицы **employees** и поле **manager** из таблицы **departments**, используя поле **dept** для связывания данных. Если способ связывания таблиц не указан, Python предполагает, что все работники относятся к одному отделу, и вы не получите ожидаемых результатов.

```
cursor.execute("UPDATE employees SET name = 'Tony' WHERE id=1")
db.commit()
```

Обновляет данные в таблице (с заменой исходных данных), в результате чего работнику с **ID 1** назначается имя "**Tony**".

```
cursor.execute("DELETE employees WHERE id=1")
```

Удаляет из таблицы **employees** записи, у которых поле **ID** содержит 1.

A handwritten signature or a stylized piece of text, possibly "RFF", written in black ink.

# Задачи

**139**

Создайте базу данных SQL с именем **PhoneBook**. База данных должна содержать таблицу **Names** со следующими данными:

ID	First Name	Surname	Phone Number
1	Simon	Howeis	01223 349752
2	Karen	Phillips	01954 295773
3	Darren	Smith	01583 749012
4	Anne	Jones	01323 567322
5	Mark	Smith	01223 855534

**141**

Создайте базу данных SQL с именем **BookInfo**, предназначеннную для хранения списка авторов и написанных ими книг. Она состоит из двух таблиц. Первая таблица с именем **Authors** содержит следующие данные:

Name	Place of Birth
Agatha Christie	Torquay
Cecelia Ahern	Dublin
J. K. Rowling	Bristol
Oscar Wilde	Dublin

Вторая таблица **Books** должна содержать следующие данные:

ID	Title	Author	Date Published
1	De Profundis	Oscar Wilde	1905
2	Harry Potter and the chamber of secrets	J. K. Rowling	1998
3	Harry Potter and the prisoner of Azkaban	J. K. Rowling	1999
4	Lyrebird	Cecelia Ahern	2017
5	Murder on the Orient Express	Agatha Christie	1934
6	Perfect	Cecelia Ahern	2017
7	The marble collector	Cecelia Ahern	2016
8	The murder on the links	Agatha Christie	1923
9	The picture of Dorian Gray	Oscar Wilde	1890
10	The secret adversary	Agatha Christie	1921
11	The seven dials mystery	Agatha Christie	1929
12	The year I met you	Cecelia Ahern	2014

**140**

Используя базу данных **PhoneBook** из программы 139, напишите программу, которая выводит следующее меню:

Main Menu

- 1) View phone book
- 2) Add to phone book
- 3) Search for surname
- 4) Delete person from phone book
- 5) Quit

Enter your selection:

Если пользователь выбирает пункт 1, он сможет просмотреть всю телефонную книгу. Если он выбирает пункт 2, он сможет добавить новую запись в телефонную книгу. Если выбран пункт 3, программа предлагает ввести фамилию, а затем выводит записи всех людей с заданной фамилией. При выборе пункта 4 программа предлагает ввести идентификатор и удаляет соответствующую запись из таблицы. При выборе пункта 5 программа завершается. Наконец, при вводе недопустимого числа должно выводиться соответствующее сообщение об ошибке. После каждого действия пользователь должен возвращаться к меню, пока не будет выбран пункт 5.

**142**

Используя базу данных **BookInfo** из программы 141, выведите список авторов с датами рождения. Предложите пользователю ввести место рождения, а затем выведите название, дату издания и имя автора для всех книг авторов, родившихся в указанном месте.

**143**

Используя базу данных **BookInfo** из программы 141, предложите пользователю ввести год. Выведите все книги, изданные после этого года; список должен быть упорядочен по году издания.

**144**

Используя базу данных **BookInfo** из программы 141, предложите пользователю ввести имя автора. Сохраните все книги этого автора в текстовом файле; поля должны разделяться дефисами, так что выводимая информация должна выглядеть примерно так:

- 5 - Murder on the Orient Express - Agatha Christie - 1934
- 8 - The murder on the links - Agatha Christie - 1923
- 10 - The secret adversary - Agatha Christie - 1921
- 11 - The seven dials mystery - Agatha Christie - 1929

Откройте текстовый файл и убедитесь в том, что программа работает правильно.



Вы уже так много узнали. Вспомните все задачи и приемы программирования, которые вы выучили.  
Просто потрясающе!

**145**

Напишите программу, которая отображает следующий экран:

The screenshot shows a window titled "TestScores". It contains two text input fields: one labeled "Enter student's name:" and another labeled "Enter student's grade:". Below these fields are two buttons: "Add" and "Clear". The window has standard OS X-style window controls (minimize, maximize, close) at the top right.

При нажатии кнопки **Add** данные должны сохраняться в базе данных SQL с именем **TestScores**. Кнопка **Clear** стирает текущее содержимое окна.

# Ответы

139

```
import sqlite3

with sqlite3.connect("PhoneBook.db") as db:
    cursor = db.cursor()

    cursor.execute(""" CREATE TABLE IF NOT EXISTS Names(
        id integer PRIMARY KEY,
        firstname text,
        surname text,
        phonenumber text); """)

    cursor.execute(""" INSERT INTO Names(id, firstname, surname, phonenumber)
        VALUES("1", "Simon", "Howels", "01223 349752")""")
    db.commit()

    cursor.execute(""" INSERT INTO Names(id, firstname, surname, phonenumber)
        VALUES("2", "Karen", "Phillips", "01954 295773")""")
    db.commit()

    cursor.execute(""" INSERT INTO Names(id, firstname, surname, phonenumber)
        VALUES("3", "Darren", "Smith", "01583 749012")""")
    db.commit()

    cursor.execute(""" INSERT INTO Names(id, firstname, surname, phonenumber)
        VALUES("4", "Anne", "Jones", "01323 567322")""")
    db.commit()

    cursor.execute(""" INSERT INTO Names(id, firstname, surname, phonenumber)
        VALUES("5", "Mark", "Smith", "01223 855534")""")
    db.commit()

db.close()
```

**140**

```
import sqlite3

def viewphonebook():
    cursor.execute("SELECT * FROM Names")
    for x in cursor.fetchall():
        print(x)

def addtophonebook():
    newid = int(input("Enter ID: "))
    newfname = int(input("Enter first name: "))
    newsname = int(input("Enter surname: "))
    newpnum = int(input("Enter phone number: "))
    cursor.execute(""" INSERT INTO Names(id, firstname, surname, phonenumber)
VALUES (?, ?, ?, ?)""", (newid, newfname, newsname, newpnum))
    db.commit()

def selectname():
    selectsurname = input("Enter a surname: ")
    cursor.execute(""" SELECT * FROM Names WHERE surname = ? ", [selectsurname])
    for x in cursor.fetchall():
        print(x)

def deletedata():
    selectid = int(input("Enter ID: "))
    cursor.execute("DELETE FROM Names WHERE id = ? ", [selectid])
    cursor.execute("SELECT * FROM Names")
    for x in cursor.fetchall():
        print(x)
    db.commit()

with sqlite3.connect("PhoneBook.db") as db:
    cursor = db.cursor()

def main():
    again = "y"
    while again == "y":
        print()
        print("Main Menu")
        print()
        print("1) View phone book")
        print("2) Add to phone book")
        print("3) Search for surname")
        print("4) Delete person from phone book")
        print("5) Quit")
        print()
        selection = int(input("Enter your selection: "))
        print()

        if selection == 1:
            viewphonebook()
        elif selection == 2:
```

Продолжение на следующей странице

```
    addtophonebook()
elif selection == 3:
    selectname()
elif selection == 4:
    deletedata()
elif selection == 5:
    again = "n"
else:
    print("Incorrect selection entered")

main()
db.close()
```

**141**

```
import sqlite3

with sqlite3.connect("BookInfo.db") as db:
    cursor = db.cursor()

cursor.execute(""" CREATE TABLE IF NOT EXISTS Authors(
Name text PRIMARY KEY,
PlaceofBirth text); """)

cursor.execute(""" INSERT INTO Authors(Name, PlaceofBirth)
VALUES("Agatha Christie", "Torquay")""")
db.commit()

cursor.execute(""" INSERT INTO Authors(Name, PlaceofBirth)
VALUES("Cecilia Ahern", "Dublin")""")
db.commit()
cursor.execute(""" INSERT INTO Authors(Name, PlaceofBirth)
VALUES("J.K.Rowling", "Bristol")""")
db.commit()
cursor.execute(""" INSERT INTO Authors(Name, PlaceofBirth)
VALUES("Oscar Wilde", "Dublin")""")
db.commit()

cursor.execute(""" CREATE TABLE IF NOT EXISTS Books(
ID integer PRIMARY KEY,
Title text,
Author text,
DatePublished integer); """)

cursor.execute(""" INSERT INTO Books(ID, Title, Author, DatePublished)
VALUES("1", "De Profundis", "Oscar Wilde", "1905")""")
db.commit()
cursor.execute(""" INSERT INTO Books(ID, Title, Author, DatePublished)
VALUES("2", "Harry Potter and the chamber of secrets", "J.K.Rowling", "1998")""")
db.commit()
cursor.execute(""" INSERT INTO Books(ID, Title, Author, DatePublished)
VALUES("3", "Harry Potter and the prisoner of Azkaban", "J.K.Rowling", "1999")""")
db.commit()
```

Продолжение на следующей странице

```

cursor.execute(""" INSERT INTO Books(ID, Title, Author, DatePublished)
VALUES("4", "Lyrebird", "Cecilia Ahern", "2017")""")
db.commit()
cursor.execute(""" INSERT INTO Books(ID, Title, Author, DatePublished)
VALUES("5", "Murder on the Orient Express", "Agatha Christie", "1934")""")
db.commit()
cursor.execute(""" INSERT INTO Books(ID, Title, Author, DatePublished)
VALUES("6", "Perfect", "Cecilia Ahern", "2017")""")
db.commit()
cursor.execute(""" INSERT INTO Books(ID, Title, Author, DatePublished)
VALUES("7", "The marble collector", "Cecilia Ahern", "2016")""")
db.commit()
cursor.execute(""" INSERT INTO Books(ID, Title, Author, DatePublished)
VALUES("8", "The murder on the links", "Agatha Christie", "1923")""")
db.commit()
cursor.execute(""" INSERT INTO Books(ID, Title, Author, DatePublished)
VALUES("9", "The picture of Dorian Gray", "Oscar Wilde", "1890")""")
db.commit()
cursor.execute(""" INSERT INTO Books(ID, Title, Author, DatePublished)
VALUES("10", "The secret adversary", "Agatha Christie", "1921")""")
db.commit()
cursor.execute(""" INSERT INTO Books(ID, Title, Author, DatePublished)
VALUES("11", "The seven dials mystery", "Agatha Christie", "1929")""")
db.commit()
cursor.execute(""" INSERT INTO Books(ID, Title, Author, DatePublished)
VALUES("12", "The year I met you", "Cecilia Ahern", "2014")""")
db.commit()

db.close()

```

**142**

```

import sqlite3

with sqlite3.connect("BookInfo.db") as db:
    cursor = db.cursor()

    cursor.execute("SELECT * FROM Authors")
    for x in cursor.fetchall():
        print(x)

    print()
    location = input("Enter a place of birth: ")
    print()

    cursor.execute("""SELECT Books.Title, Books.DatePublished, Books.Author
    FROM Books,Authors WHERE Authors.Name = Books.Author AND Authors.PlaceofBirth = ?
    """,[location])
    for x in cursor.fetchall():
        print(x)

db.close()

```

**143**

```
import sqlite3

with sqlite3.connect("BookInfo.db") as db:
    cursor = db.cursor()

selectionyear = int(input("Enter a year: "))
print()

cursor.execute("""SELECT Books.Title, Books.DatePublished, Books.Author
FROM Books WHERE DatePublished > ? ORDER BY DatePublished """, [selectionyear])
for x in cursor.fetchall():
    print(x)

db.close()
```

**144**

```
import sqlite3

file = open("BooksList.txt", "w")

with sqlite3.connect("BookInfo.db") as db:
    cursor = db.cursor()

cursor.execute("SELECT Name from Authors")
for x in cursor.fetchall():
    print()

print()
selectauthor = input("Enter an author's name: ")
print()

cursor.execute("""SELECT * FROM Books WHERE Author = ?""" , [selectauthor])
for x in cursor.fetchall():
    newrecord = str(x[0]) + " - " + x[1] + " - " + x[2] + " - " + str(x[3]) + "\n"
    file.write(newrecord)

file.close()

db.close()
```

**145**

```
import sqlite3
from tkinter import *

def addtolist():
    newname = sname.get()
    newgrade = sgrade.get()
    cursor.execute(""" INSERT INTO Scores (name, score)
VALUES(?, ?)""", (newname, newgrade))
    db.commit()
    sname.delete(0, END)
    sgrade.delete(0, END)
    sname.focus()

def clearlist():
    sname.delete(0, END)
    sgrade.delete(0, END)
    sname.focus()

with sqlite3.connect("TestScore.db") as db:
    cursor = db.cursor()

cursor.execute(""" CREATE TABLE IF NOT EXISTS Scores(
id integer PRIMARY KEY, name text, score integer); """)

window = Tk()
window.title("TestScores")
window.geometry("450x200")

label1 = Label(text = "Enter student's name: ")
label1.place(x = 30, y = 35)
sname = Entry(text = "")
sname.place(x = 150, y = 35, width = 200, height = 25)
sname.focus()
label2 = Label(text = "Enter student's grade: ")
label2.place(x = 30, y = 80)
sgrade = Entry(text = "")
sgrade.place(x = 150, y = 80, width = 200, height = 25)
sgrade.focus()
addbtn = Button(text = "Add", command = addtolist )
addbtn.place(x = 150, y = 120, width = 75, height = 25)
clearbtn = Button(text = "Clear", command = clearlist)
clearbtn.place(x = 250, y = 120, width = 75, height = 25)

window.mainloop()
db.close()
```

# Часть II

Учебные проекты



# Введение в часть II



В этом разделе содержатся крупные программные проекты. На их выполнение у вас уйдет больше времени, чем на предыдущие задачи; скорее всего, вам придется обращаться к предыдущим разделам книги, чтобы вспомнить некоторые ключевые темы. Не огорчайтесь, если вам приходится искать ключевые строки кода в предыдущих разделах – даже опытный программист не сразу разберется в хитроумном коде, который ему не знаком. Все это является частью процесса обучения, и предполагалось, что читатель будет работать с книгой именно так.

К каждой задаче прилагается список навыков, которые понадобятся для ее решения. Вы можете сами определить готовность ее решить. Также к каждой задаче приводится краткое описание и раздел, перечисляющий те проблемы, с которыми вам предстоит справиться. Решения в этой части занимают намного больше места, а некоторые даже растянуты на несколько страниц, но все они образуют одну непрерывную программу. Если программу нужно разбивать на несколько страниц, мы постараемся делать это по границам подпрограмм или по естественным точкам разрыва, если это возможно.

Прочтите каждую задачу от начала и до конца, прежде чем браться за них, – так вы узнаете обо всех подводных камнях. После прочтения отложите книгу и подумайте над тем, как вы собираетесь подходить к решению. Возможно, стоит сделать заметки, а если вы чувствуете себя особо компетентным, то можете даже нарисовать блок-схему. Не стоит переходить к написанию кода, не имея ни малейшего понятия, куда вы двигаетесь. Так вы только зайдете в тупик и начнете сомневаться в своих способностях. Составьте план, разбейте одну большую задачу на несколько мелких и более доступных, а затем последовательно отработайте каждую из них. А сейчас налейте себе чего-нибудь, возьмите блокнот и карандаш, вдохните поглубже, переверните страницу и беритесь за первую задачу.



# Задача 146. Код сдвига

Для выполнения этой задачи необходимы следующие навыки:

- ввод и вывод данных;
- списки;
- разбиение и соединение строк;
- инструкции **if**;
- циклы (**while** и **for**);
- подпрограммы.



## Задача

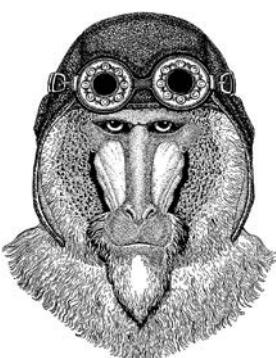
Код сдвига – один из самых простых кодов, который может использоваться для шифрования сообщений. Каждая буква заменяется другой буквой, полученной сдвигом вперед по алфавиту на заданную величину. Например, при сдвиге на один символ строка «abc» преобразуется в «bcd» (то есть каждая буква в алфавите сдвигается вперед на одну позицию).

Напишите программу, которая выводит следующее меню:

- 1) Make a code
- 2) Decode a message
- 3) Quit

Enter your selection:

Если пользователь выбирает вариант 1, он получает возможность ввести сообщение (с пробелами), а затем число. Python выводит закодированное сообщение, полученное с применением заданного сдвига.



Если пользователь выбирает вариант 2, он вводит закодированное сообщение и правильное число, а программа выводит декодированное сообщение (то есть смешается на нужное количество букв в обратном направлении по алфавиту).

При выборе варианта 3 программа завершает работу.

После того как пользователь закодирует или декодирует сообщение, меню выводится снова, пока пользователь не завершит работу с программой.

## Проблемы, которые придется преодолеть

Решите, хотите ли вы поддерживать буквы как верхнего, так и нижнего регистра, или же программа будет преобразовывать данные к одному регистру.

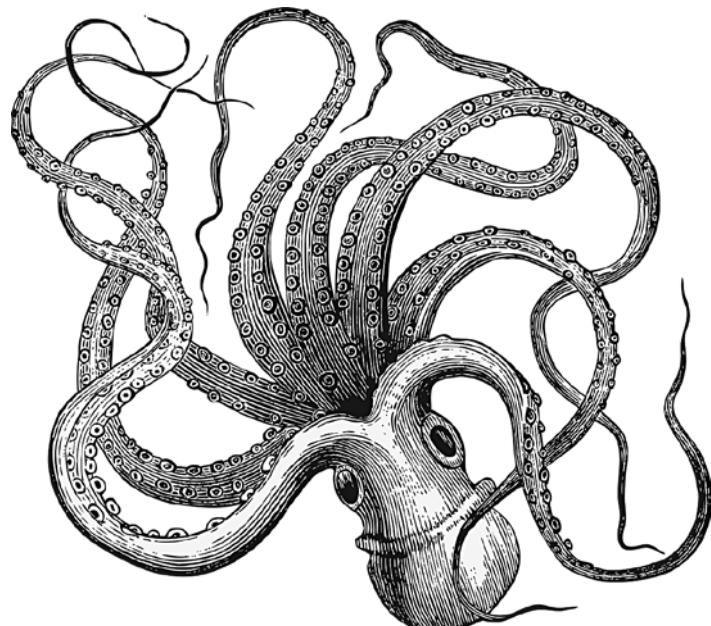
Решите, будут ли поддерживаться знаки препинания в строках.

Если в результате сдвига происходит выход за конец алфавита, позиция должна возвращаться к началу. Например, если пользователь вводит «*xxyz*» со сдвигом 5, должна выводиться закодированная строка «*bcd*». При декодировании сообщения сдвиг должен работать в обратном направлении; таким образом, значение «*a*» должно вернуться к «*w*».

Проследите за тем, чтобы при выборе пользователем недопустимого пункта меню или числа для сдвига выводилось соответствующее сообщение.



Протестируйте функциональность декодирования на сообщении «*weovugjohsslunl*», которое было создано со сдвигом 7 для алфавита «*abcdefghijklmnopqrstuvwxyz*» (обратите внимание, что в конце алфавита используется пробел).



## Ответ

```
alphabet = ["a", "b", "c", "d", "e", "f", "g", "h", "i", "j",
            "k", "l", "m", "n", "o", "p", "q", "r", "s", "t",
            "u", "v", "w", "x", "y", "z", " "]

def get_data():
    word = input("Enter your message: ")
    word = word.lower()
    num = int(input("Enter your number (1-26): "))
    if num > 26 or num == 0:
        while num > 26 or num == 0:
            num = int(input("Out of range, please enter a number (1-26): "))
    data = (word, num)
    return(data)

def make_code(word, num):
    new_word = ""
    for x in word:
        y = alphabet.index(x)
        y = y + num
        if y > 26:
            y = y - 27
        char = alphabet[y]
        new_word = new_word + char
    print(new_word)
    print()

def decode(word, num):
    new_word = ""
    for x in word:
        y = alphabet.index(x)
        y = y - num
        if y < 0:
            y = y + 27
        char = alphabet[y]
        new_word = new_word + char
    print(new_word)
    print()

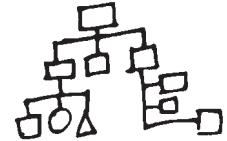
def main():
    again = True
    while again == True:
        print("1) Make a code")
        print("2) Decode a message")
        print("3) Quit")
        print()
        selection = int(input("Enter your selection: "))
        if selection == 1:
            (word, num) = get_data()
            make_code(word, num)
        elif selection == 2:
            (word, num) = get_data()
            decode(word, num)
        elif selection == 3:
            again = False
        else:
            print("Incorrect selection")

main()
```

# Задача 147. Mastermind

Для выполнения этой задачи необходимы следующие навыки:

- ввод и вывод данных;
- списки;
- случайный выбор из списка;
- инструкции **if**;
- циклы (**while** и **for**);
- подпрограммы.



## Задача

В этом разделе мы займемся созданием электронной версии настольной игры «Mastermind» (на отечественном рынке известной как «Властелин разума». – *Примеч. ред.*). Компьютер автоматически генерирует комбинацию из четырех цветов из списка возможных (предусмотрите возможность компьютера многократно выбирать любой цвет). Например, компьютер может выбрать комбинацию «красный», «синий», «красный», «зеленый». Эта последовательность остается **скрытой** от пользователя.

Пользователь вводит комбинацию из четырех цветов из списка, использованного компьютером. Например, он может выбрать цвета «розовый», «синий», «желтый» и «красный».

Программа сообщает, сколько цветов находятся в правильной позиции и сколько правильно угаданных цветов находятся в неправильных позициях. Так, в приведенном примере должны выводиться сообщения «Правильный цвет в правильной позиции: 1» и «Правильный цвет в неправильной позиции: 1».

Пользователь продолжает гадать, пока не введет все четыре цвета в правильном порядке. В конце игры программа выводит соответствующий текст и сообщает, сколько попыток ему понадобилось.

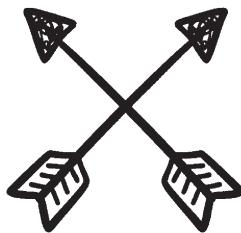
## Проблемы, которые придется преодолеть

Самая сложная часть этой игры – логика проверки того, сколько цветов пользователь угадал верно и сколько из них находится в ошибочных местах. В предыдущем примере, если пользователь введет комбинацию «синий», «синий», «синий», «синий», программа должна сообщить, что 1 цвет находится в правильной позиции и 0 цветов угадано правильно, но находятся в неправильной позиции.

Решите, существует ли более удобный способ ввода пользователем его решений (например, выражение цвета с помощью одной буквы). Если вы используете первую букву, проследите за тем, чтобы все цвета начинались с разных букв. Убедитесь в том, что ваши инструкции понятны пользователю.

Решите, хотите ли вы поддерживать буквы как верхнего, так и нижнего регистра, или же проще преобразовать все к одному регистру.

Не забудьте встроить код, проверяющий, что пользователь вводит только подходящие данные, и выведите соответствующее сообщение, если это не так. Если пользователь не угадал цвет, предоставьте возможность повторить ввод вместо того, чтобы прямо указывать на неверный выбор.



## Ответ

```
import random

def select_col():
    colours = ["r", "b", "o", "y", "p", "g", "w"]
    c1 = random.choice(colours)
    c2 = random.choice(colours)
    c3 = random.choice(colours)
    c4 = random.choice(colours)
    data = (c1, c2, c3, c4)
    return data

def tryit(c1, c2, c3, c4):
    print("The colours are: (r)ed, (b)lue, (o)range, (y)ellow, (p)ink, (g)reen and (w)hite.")
    try_again = True
    while try_again == True:
        u1 = input("Enter your choice for place 1: ")
        u1 = u1.lower()
        if u1 != "r" and u1 != "b" and u1 != "o" and u1 != "y" and u1 != "p" and u1 != "g" and u1 != "w":
            print("Incorrect selection")
        else:
            try_again = False
    try_again = True
    while try_again == True:
        u2 = input("Enter your choice for place 2: ")
        u2 = u2.lower()
        if u2 != "r" and u2 != "b" and u2 != "o" and u2 != "y" and u2 != "p" and u2 != "g" and u2 != "w":
            print("Incorrect selection")
        else:
            try_again = False
    try_again = True
    while try_again == True:
        u3 = input("Enter your choice for place 3: ")
        u3 = u3.lower()
        if u3 != "r" and u3 != "b" and u3 != "o" and u3 != "y" and u3 != "p" and u3 != "g" and u3 != "w":
            print("Incorrect selection")
        else:
            try_again = False
    try_again = True
    while try_again == True:
        u4 = input("Enter your choice for place 4: ")
        u4 = u4.lower()
```

```
if u4 != "r" and u4 != "b" and u4 != "o" and u4 != "y" and u4 != "p" and u4 != "g" and u4 != "w":
    print("Incorrect selection")
else:
    try_again = False
correct = 0
wrong_place = 0
if c1 == u1:
    correct = correct + 1
elif c1 == u2 or c1 == u3 or c1 == u4:
    wrong_place = wrong_place + 1
if c2 == u2:
    correct = correct + 1
elif c2 == u1 or c2 == u3 or c2 == u4:
    wrong_place = wrong_place + 1
if c3 == u3:
    correct = correct + 1
elif c3 == u1 or c3 == u2 or c3 == u4:
    wrong_place = wrong_place + 1
if c4 == u4:
    correct = correct + 1
elif c4 == u1 or c4 == u2 or c4 == u3:
    wrong_place = wrong_place + 1
print("Correct colour in the correct place: ", correct)
print("Correct colour but in the wrong place: ", wrong_place)
print()
data2 = [correct, wrong_place]
return data2

def main():
    (c1, c2, c3, c4) = select_col1()
    score = 0
    play = True
    while play == True:
        (correct, wrong_place) = tryit(c1, c2, c3, c4)
        score = score + 1
        if correct == 4:
            play = False
    print("You win!")
    print("You took ", score, " guesses")

main()
```

# Задача 148. Пароли

Для выполнения этой задачи необходимы следующие навыки:

- ввод и вывод данных;
- списки;
- инструкции **if**;
- циклы (**while** и **for**);
- запись и чтение файлов .csv.



## Задача

Напишите программу, которая сохраняет идентификаторы пользователей и их пароли. Программа должна выводить следующее меню:

- 1) Create a new User ID
- 2) Change a password
- 3) Display all User IDs
- 4) Quit

Enter Selection:

Если пользователь выбирает пункт 1, программа предлагает ввести идентификатор пользователя. Она должна проверить, существует ли введенный идентификатор в списке. Если это так, программа выводит соответствующее сообщение и предлагает выбрать другой идентификатор. После того как пользователь введет допустимый идентификатор, программа запрашивает пароль. Паролю начисляется по одному баллу за соответствие перечисленным ниже условиям:

- пароль должен содержать не менее 8 символов;
- пароль должен включать буквы верхнего регистра;
- пароль должен включать буквы нижнего регистра;
- пароль должен включать цифры;
- пароль должен включать хотя бы один специальный символ: !, £, \$, %, &, <, \* или @.

Если пароль получает всего 1 или 2 балла, он должен быть отклонен с формулировкой, что он является слабым. Если у пароля 3 или 4 балла, выводится сообщение о том, что его можно улучшить. Спросите пользователя, хочет ли он повторить попытку. Если пароль набрал 5 баллов, сообщите, что он является сильным. В конец файла .csv должны добавляться только допустимые идентификаторы и пароли.

Если пользователь выберет пункт 2, он должен будет ввести идентификатор пользователя. Проверьте, существует ли идентификатор в списке. Если это так, предложите пользователю изменить пароль и сохраните изменения в файле .csv. Убедитесь в том, что программа только изменяет существующий пароль, а не создает новую запись.

Если пользователь выберет пункт 3, выведите только идентификаторы пользователей без паролей.

Наконец, при выборе пункта 4 программа должна завершиться.



## Проблемы, которые придется преодолеть

Файлы .csv с существующими данными не могут редактироваться. Данные можно только просматривать и добавлять. Поэтому вам необходимо будет импортировать данные во временный список Python и внести изменения до того, как данные будут записаны в файл .csv.

Убедитесь в том, что изменяться могут только пароли, связанные с существующим идентификатором пользователя.

Выводите сообщения, упрощающие работу пользователя с системой.

Повторно выводите меню, пока пользователь не выйдет из программы.



## Ответ

Для этой задачи необходимо сначала создать файл .csv с именем passwords.csv. Либо сделайте это с помощью программного кода, либо просто создайте файл Excel и сохраните его в формате .csv. Файл должен находиться в одной папке с программой.

```
import csv

def get_data():
    file = list(csv.reader(open("password.csv")))
    tmp = []
    for x in file:
        tmp.append(x)
    return tmp

def create(userID):
    name_again = True
    while name_again == True:
        userID = input("Enter a new user ID: ")
        userID.lower()
        inlist = False
        row = 0
        for y in tmp:
            if userID in tmp[row][0]:
                print(userID, " has already been allocated")
                inlist = True
            row = row + 1
        if inlist == False:
            name_again = False
    return userID

def create_password():
    sclist = ["!", "£", "$", "%", "^", "&", "*", "(", ")", "?", "@", "#"]
    nclist = ["1", "2", "3", "4", "5", "6", "7", "8", "9", "0"]
    tryagain = True
    while tryagain == True:
        score = 0
        uc = False
        lc = False
        sc = False
        nc = False
        password = input("Enter Password: ")
        length = len(password)
        if length >= 8:
            score = score + 1
        for x in password:
```

Продолжение на следующей странице

```
if x.islower():
    lc = True
if x.isupper():
    uc = True
if x in sclist:
    sc = True
if x in nclist:
    nc = True
if sc == True:
    score = score + 1
if lc == True:
    score = score + 1
if uc == True:
    score = score + 1
if nc == True:
    score = score + 1
if score == 1 or score == 2:
    print("This is a weak password, try again")
if score == 3 or score == 4:
    print("This password could be improved")
    again = input("Do you want to try for a stronger password? (y/n)")
    again.lower()
    if again == "n":
        tryagain = False
if password != password2:
    print("Passwords do not match. File not saved")
    main()
else:
    return password

def find_user(tmp)
    ask_name_again = True
    userID = ""
    while ask_name_again == True:
        searchID = input("Enter the user ID you are looking for ")
        searchID.lower()
        inlist = False
        row = 0
        for y in tmp:
            if searchID in tmp[row][0]:
                inlist = True
            row = row + 1
        if inlist == True:
            userID = searchID
            ask_name_again = False
        else:
            print(searchID, " is NOT in the list")
    return userID
```

```
def change_password(userID, tmp):
    if userID != "":
        password = create_password()
        ID = userID.index(userID)
        tmp[ID][1] = password
        file = open("passwords.csv", "w")
        x = 0
        for row in tmp:
            newrecord = tmp[x][0] + ", " + tmp[x][1] + "\n"
            file.write(newrecord)
            x = x + 1
        file.close()

def display_all(userID):
    tmp = get_data()
    x = 0
    for row in tmp:
        print(tmp[x][0])
        x = x + 1

def main():
    tmp = get_data()
    go_again = True
    while go_again == True:
        print()
        print("1) Create a new User ID")
        print("2) Change a password")
        print("3) Display all User IDs")
        print("4) Quit")
        print()
        selection = int(input("Enter Selection: "))
        if selection == 1:
            userID = create_userID(tmp)
            password = create_password()
            file = open("passwords.csv", "a")
            newrecord = userID + ", " + password + "\n"
            file.write(str(newrecord))
            file.close()
        elif selection == 2:
            userID = find_userID(tmp)
            change_password(userID, tmp)
        elif selection == 3:
            display_all(userID())
        elif selection == 4:
            go_again = False
        else:
            print("Incorrect selection")

main()
```

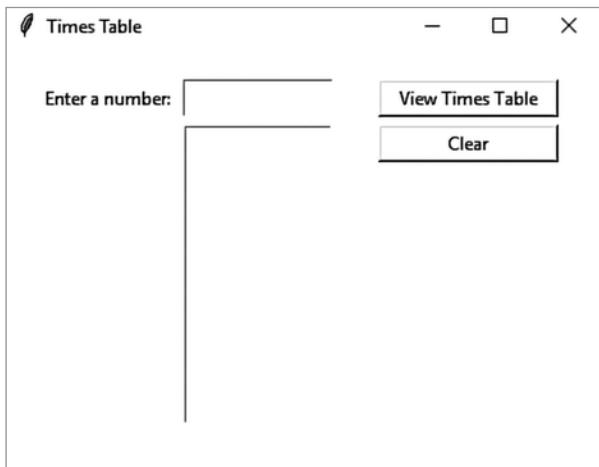
# Задача 149. Таблица умножения

Для выполнения этой задачи необходимы следующие навыки:

- циклы (**while** и **for**);
- подпрограммы;
- библиотека Tkinter.

## Задача

Напишите программу, которая выводит следующее окно:



 Times Table

Enter a number:

- □ ×

1 x 99 = 99
2 x 99 = 198
3 x 99 = 297
4 x 99 = 396
5 x 99 = 495
6 x 99 = 594
7 x 99 = 693
8 x 99 = 792
9 x 99 = 891
10 x 99 = 990
11 x 99 = 1089
12 x 99 = 1188

Если ввести число в первом поле и нажать кнопку View Times Table, в области списка выводится таблица умножения. Например, если ввести значение 99, выводится список, изображенный справа.

Кнопка Clear стирает содержимое обоих полей.



## Проблемы, которые придется преодолеть

В списке должны выводиться не только ответы, но и умножаемые числа. Следующая строка кода может помочь вам:

```
num_list.insert(END,(i, "x", num, "=", answer))
```

Убедитесь в том, что основные элементы расположены правильно, чтобы таблица была как можно более удобной для пользователя.

## Ответ

```
from tkinter import *

def show_table():
    num = num_box.get()
    num = int(num)
    value = 1
    for i in range(1, 13):
        answer = i * num
        num_list.insert(END,(i, "x", num, "=", answer))
        value = value + 1
    num_box.delete(0, END)
    num_box.focus()

def clear_list():
    num_box.delete(0, END)
    num_list.delete(0, END)
    num_box.focus()

window = Tk()
window.title("Times Table")
window.geometry("400x280")

label1 = Label(text = "Enter a number: ")
label1.place(x = 20, y = 20, width = 100, height = 25)

num_box = Entry(text = 0)
num_box.place(x = 120, y = 20, width = 100, height = 25)
num_box.focus()

button1 = Button(text = "View Times Table", command = show_table)
button1.place(x = 250, y = 20, width = 120, height = 25)

num_list = Listbox()
num_list.place(x = 120, y = 50, width = 100, height = 200)

button2 = Button(text = "Clear", command = clear_list)
button2.place(x = 250, y = 50, width = 120, height = 25)

window.mainloop()
```

# Задача 150. Картичная галерея

Для выполнения этой задачи необходимы следующие навыки:

- библиотека Tkinter;
- SQLite 3.



## Задача

Небольшая картинная галерея, продающая работы разных художников, хочет хранить информацию о картинах в базе данных SQL. Создайте для хранения данных удобную систему с графическим интерфейсом. Ниже приведены данные, которые должны храниться в базе.

### Информация о художниках

ArtistID	Name	Address	Town	County	Postcode
1	Martin Leighton	5 Park Place	Peterborough	Cambridgeshire	PE32 5LP
2	Eva Czarniecka	77 Warner Close	Chelmsford	Essex	CM22 5FT
3	Roxy Parkin	90 Hindhead Road		London	SE12 6WM
4	Nigel Farnworth	41 Whitby Road	Hunlty	Aberdeenshire	AB54 5PN
5	Teresa Tanner	70 Guild Street		London	NW7 1SP

### Картины

PieceID	ArtistID	Title	Medium	Price
1	5	Woman with black Labrador	Oil	220
2	5	Bees & thistles	Watercolour	85
3	2	A stroll to Westminster	Ink	190
4	1	African giant	Oil	800
5	3	Water daemon	Acrylic	1700
6	4	A seagull	Watercolour	35
7	1	Three friends	Oil	1800
8	2	Summer breeze 1	Acrylic	1350
9	4	Mr Hamster	Watercolour	35
10	1	Pulpit Rock, Dorset	Oil	600
11	5	Trawler Dungeness beach	Oil	195

PieceID	ArtistID	Title	Medium	Price
12	2	Dance in the snow	Oil	250
13	4	St Tropez port	Ink	45
14	3	Pirate assassin	Acrylic	420
15	1	Morning walk	Oil	800
16	4	A baby barn swallow	Watercolour	35
17	4	The old working mills	Ink	395

## Проблемы, которые придется преодолеть

Программа должна предоставлять возможность добавления новых записей о художниках и картинах.

После того как картина будет продана, данные о ней должны быть удалены из главной базы данных SQL и сохранены в отдельном текстовом файле.

Пользователь должен иметь возможность искать картины по имени художника, технике исполнения или цене.



## Ответ

```
import sqlite3
from tkinter import *

def addartist():
    newname = artistname.get()
    newaddress = artistadd.get()
    newtown = artisttown.get()
    newcounty = artistcounty.get()
    newpostcode = artistpostcode.get()
    cursor.execute("""INSERT INTO Artists (name, address, town, county, postcode)
VALUES (?, ?, ?, ?, ?)""", (newname, newaddress, newtown, newcounty, newpostcode))
    db.commit()
    artistname.delete(0, END)
    artistadd.delete(0, END)
    artisttown.delete(0, END)
    artistcounty.delete(0, END)
    artistpostcode.delete(0, END)
    artistname.focus()

def clearartist():
    artistname.delete(0, END)
    artistadd.delete(0, END)
    artisttown.delete(0, END)
    artistcounty.delete(0, END)
    artistpostcode.delete(0, END)
    artistname.focus()

def addart():
    newartname = artname.get()
    newtitle = arttile.get()
    newmedium = medium.get()
    newprice = artprice.get()
    cursor.execute("""INSERT INTO Art (artistID, title, medium, price)
VALUES (?, ?, ?, ?)""", (newartname, newtitle, newmedium, newprice))
    db.commit()
    artname.delete(0, END)
    arttile.delete(0, END)
    medium.set("")
    artprice.delete(0, END)
    artistname.focus()

def clearwindow():
    outputwindow.delete(0, END)

def viewartlist():
    cursor.execute("SELECT * FROM Artists")
```

Продолжение на следующей странице

```
for x in cursor.fetchall():
    newrecord = str(x[0]) + ", " + str(x[1]) + ", " + str(x[2]) + ", " + str(x[3]) +
    ", " + str(x[4]) + ", " + x[5] + "\n"
    outputwindow.insert(END, newrecord)

def viewart():
    cursor.execute("SELECT * FROM Art")
    for x in cursor.fetchall():
        newrecord = str(x[0]) + ", " + str(x[1]) + ", " + str(x[2]) + ", " + str(x[3]) +
        ", €" + str(x[4]) + "\n"
        outputwindow.insert(END, newrecord)

def searchartistoutput():
    selectedartist = searchartist.get()
    cursor.execute("SELECT name FROM Artists WHERE artistid = ?", [selectedartist])
    for x in cursor.fetchall():
        outputwindow.insert(END, x)
    cursor.execute("SELECT name FROM Art WHERE artistid = ?", [selectedartist])
    for x in cursor.fetchall():
        newrecord = str(x[0]) + ", " + str(x[1]) + ", " + str(x[2]) + ", " +
        str(x[3]) + ", €" + str(x[4]) + "\n"
        outputwindow.insert(END, newrecord)
    searchartist.delete(0, END)
    searchartist.focus()

def searchmediumoutput():
    selectedmedium = medium2.get()
    cursor.execute("""SELECT Art.pieceid, Artists.name, Art.title, Art.medium, Art.
    price FROM Artists, Art WHERE Artists.artistid = Art.artistid AND Art.medium = ?""",
    [selectedmedium])
    for x in cursor.fetchall():
        newrecord = str(x[0]) + ", " + str(x[1]) + ", " + str(x[2]) + ", " + str(x[3]) +
        ", €" + str(x[4]) + "\n"
        outputwindow.insert(END, newrecord)
    medium2.set("")

def searchbyprice():
    minprice = selectmin.get()
    maxprice = selectmax.get()
    cursor.execute("""SELECT Art.pieceid, Artists.name, Art.title, Art.medium, Art.price
    FROM Artists, Art WHERE Artists.artistid = Art.artistid AND Art.price >= ? AND Art.price
    <=?""", [minprice, maxprice])
    for x in cursor.fetchall():
        newrecord = str(x[0]) + ", " + str(x[1]) + ", " + str(x[2]) + ", " + str(x[3]) +
        ", €" + str(x[4]) + "\n"
        outputwindow.insert(END, newrecord)
    selectmin.delete(0, END)
    selectmax.delete(0, END)
    selectmin.focus()
```

```
def sold():
    file = open("SoldArt.txt", "a")
    selectedpiece = soldpiece.get()
    cursor.execute("SELECT * FROM Art WHERE pieceid = ?", [selectedpiece])
    for x in cursor.fetchall():
        newrecord = str(x[0]) + ", " + str(x[1]) + ", " + str(x[2]) + ", " + str(x[3]) +
", " + str(x[4]) + "\n"
        file.write(newrecord)
    file.close()
    cursor.execute("DELETE FROM Art WHERE pieceid = ?", [selectedpiece])
    db.commit()

with sqlite3.connect("Art.db") as db:
    cursor = db.cursor()

cursor.execute("""CREATE TABLE IF NOT EXISTS Artists(
artistid integer PRIMARY KEY, name text, address text, town text, county text, postcode
text); """)

cursor.execute("""CREATE TABLE IF NOT EXISTS Art(
pieceid integer PRIMARY KEY, artistid integer, title text, medium text, price integer);
""")

window Tk()
window.title("Art")
window.geometry("1220x600")

title1 = Label(text = "Enter new details: ")
title1.place(x = 10, y = 10, width = 100, height = 25)
artistnamelbl = Label(text = "Name: ")
title1.place(x = 30, y = 40, width = 80, height = 25)
artistname = Entry(text = "")
artistname.place(x = 110, y = 40, width = 200, height = 25)
artistname.focus()
artistaddlbl = Label(text = "Address: ")
artistaddlbl.place(x = 310, y = 40, width = 80, height = 25)
artistadd = Entry(text = "")
artistadd.place(x = 390, y = 40, width = 200, height = 25)
artisttownlbl = Label(text = "Town: ")
artisttownlbl.place(x = 590, y = 40, width = 80, height = 25)
artisttown = Entry(text = "")
artisttown.place(x = 670, y = 40, width = 100, height = 25)
artistcountylbl = Label(text = "County: ")
artistcountylbl.place(x = 770, y = 40, width = 80, height = 25)
artistcounty = Entry(text = "")
artistcounty.place(x = 850, y = 40, width = 100, height = 25)
artistpostcodelbl = Label(text = "Postcode: ")
artistpostcodelbl.place(x = 950, y = 40, width = 80, height = 25)
```

```
artistpostcode = Entry(text = "")  
artistpostcode.place(x = 1030, y = 40, width = 100, height = 25)  
addbtn = Button(text = "Add Artist", command = addartist)  
addbtn.place(x = 110, y = 80, width = 130, height = 25)  
clearbtn = Button(text = "Clear Artist", command = clearartist)  
clearbtn.place(x = 250, y = 80, width = 130, height = 25)  
artnamelbl = Label(text = "Artist ID: ")  
artnamelbl.place(x = 30, y = 120, width = 80, height = 25)  
artname = Entry(text = "")  
artname.place(x = 110, y = 120, width = 50, height = 25)  
arttitlelbl = Label(text = "Title: ")  
arttitlelbl.place(x = 200, y = 120, width = 80, height = 25)  
arttitle = Entry(text = "")  
arttitle.place(x = 280, y = 120, width = 280, height = 25)  
artmediumlbl = Label(text = "Medium: ")  
artmediumlbl.place(x = 590, y = 120, width = 80, height = 25)  
medium = StringVar(window)  
artmedium = OptionMenu(window, medium, "Oil", "Watercolour", "Ink", "Acrylic")  
artmedium.place(x = 670, y = 120, width = 100, height = 25)  
artpricelbl = Label(text = "Price:")  
artpricelbl.place(x = 770, y = 120, width = 80, height = 25)  
artprice = Entry(text = "")  
artprice.place(x = 850, y = 120, width = 100, height = 25)  
addartbtn = Button(text = "Add Piece", command = addart)  
arrartbtn.place(x = 110, y = 150, width = 130, height = 25)  
clearartbtn = Button(text = "Clear Piece", command = clearart)  
clearartbtn.place(x = 250, y = 150, width = 130, height = 25)  
  
outputwindow = Listbox()  
outputwindow.place(x = 10, y = 200, width = 1000, height = 350)  
  
clearoutputwindow = Button(text = "Clear Output", command = clearwindow)  
clearoutputwindow.place(x = 1020, y = 200, width = 155, height = 25)  
viewallartists = Button(text = "View All Artists", command = viewartists)  
viewallartists.place(x = 1020, y = 230, width = 155, height = 25)  
viewallart = Button(text = "View All Art", command = viewart)  
viewallart.place(x = 1020, y = 260, width = 155, height = 25)  
searchartist = Entry(text = "")  
searchartist.place(x = 1020, y = 300, width = 50, height = 25)  
searchartistbtn = Button(text = "Search by Artist", command = searchartistoutput)  
searchartistbtn.place(x = 1075, y = 300, width = 100, height = 25)  
medium2 = StringVar(window)  
searchmedium = OptionMenu(window, medium2, "Oil", "Watercolour", "Ink", "Acrylic")  
searchmedium.place(x = 1020, y = 330, width = 100, height = 25)  
searchmediumbtn = Button(text = "Search", command = searchmediumoutput)  
searchmediumbtn.place(x = 1125, y = 330, width = 50, height = 25)  
minlbl = Label(text = "Min:")  
minlbl.place(x = 1020, y = 360, width = 75, height = 25)
```

```
maxlbl = Label(text = "Max:")
maxlbl.place(x = 1100, y = 360, width = 75, height = 25)
selectmin = Entry(text = "")
selectmin.place(x = 1020, y = 380, width = 75, height = 25)
selectmax = Entry(text = "")
selectmax.place(x = 1100, y = 380, width=75, height=25)
searchpricebtn = Button(text = "Search by Price", command = searchbyprice)
searchpricebtn.place(x = 1020, y = 410, width=155, height=25)
soldpiece = Entry(text = "")
soldpiece.place(x = 1020, y = 450, width=50, height=25)
soldbtn = Button(text = "Sold", command = sold)
soldbtn.place(x = 1075, y = 450, width=100, height=25)

window.mainloop()
db.close()
```

# Что дальше?



Если вы как следует проработали все примеры в книге, то сейчас уже хорошо понимаете основы программирования на Python. Вероятно, вы хорошо освоили синтаксис языка и начали мыслить как программист, разбивая большую задачу на мелкие, более удобные части, которые вы уже умеете решать. Оглянитесь и припомните все, что вы узнали, — ваша довольная улыбка будет совершенно оправданной. Чтобы научиться программировать, необходимо упорство. Свою решимость вы уже доказали, и теперь вы владеете всеми базовыми навыками для того, чтобы продолжить свой путь к вершинам Python.

Со знаниями, полученными вами в этой книге, вы уже сможете создавать полезные программы, но сейчас не время расслабляться. Вам предстоит путешествие в большой мир программирования, и вы должны узнать, как работают другие программисты. Поищите информацию в интернете, найдите новые задачи. В ходе исследований вам попадется незнакомый код, так как любую задачу можно решить несколькими способами. Например, в Tkinter существует метод `pack`, и многие программисты предпочитают использовать именно его. Этот метод позволяет строить экраны по принципу сетки, но в отличие от метода `place`, он не позволяет точно настраивать позицию объекта. Попробуйте — возможно, этот способ покажется вам более удобным. Тем не менее будьте внимательны — некоторые приемы плохо сочетаются с другими. Если вы хотите использовать метод `pack`, не пытайтесь смешивать его с `place` в одной программе. Python не любит работать с двумя разными системами позиционирования одновременно, и программа аварийно завершится.



Лучший способ освоить более сложные приемы программирования — опробовать их на практике. Читайте чужой код, посещайте чаты. Программисты — народ отзывчивый и, скорее всего, охотно придут вам на помощь (если только вы не задаете вопрос, на который уже отвечали на этом форуме.) Если вы оказались в тупике, работая над каким-либо кодом, попросите помочь на форуме: решение задач — любимое занятие всех программистов. Возможно, вы не согласитесь с предложенным решением или другой подход покажется вам более подходящим, но по крайней мере вы сможете взглянуть на задачу с другой стороны и увидеть путь к решению, который ранее вам не приходил в голову.

Независимо от того, удовлетворены ли вы полученными знаниями или же собираетесь заняться дальнейшими исследованиями, хочется надеяться, что ваше путешествие в мир программирования было приятным, а книга оказалась полезной.



# Глоссарий

Понятие	Определение
and	Операция, результат которой равен <b>True</b> только в том случае, если истинны оба условия: <pre>if num &gt; 10 and num &lt; 20:     print("In range") else:     print("Out of range")</pre>
blob	Тип данных, который хранится точно в том виде, в котором эти данные были введены. См. SQL и базы данных.
capitalize	Метод для изменения регистра символов, при котором все слова начинаются с символа верхнего регистра, а все остальные символы относятся к нижнему регистру. <pre>print(name.capitalize())</pre>
choice	Метод для выбора случайного варианта из списка. <pre>selection = random.choice(['a', 'b', 'c'])</pre>
count	Метод для подсчета вхождений значения в списке, кортеже, словаре или массиве. <pre>print(names_list.count('Sue'))</pre>
CSV	Тип файлов для хранения данных, упорядоченных по строкам и столбцам.
def	Определение подпрограммы. <pre>def menu():     print("1) Open")     print("2) Close")     selection = int(input("Selection: "))</pre>
del	Удаление элементов из списка. Пример: <pre>del names_list[2]</pre> Удаляет элемент 2 из списка <b>names_list</b> .
elif	Используется в командах <b>if</b> для проверки нового условия, если предыдущие условия не выполняются. <pre>if num &lt; 10:     print("Too low") elif num &gt; 20:     print("Too high") else:     print("In range")</pre>

Понятие	Определение
else	Используется в инструкциях <b>if</b> для определения того, что должно происходить, если все предыдущие условия не выполняются.  <pre>if num &lt; 10:     print("Too low") elif num &gt; 20:     print("Too high") else:     print("In range")</pre>
else...if	См. <b>elif</b> .
extend	Метод для добавления нескольких элементов в конец списка, кортежа, словаря, строки или массива.  <code>names_list.extend(more_names)</code>
for, цикл	Разновидность цикла, в котором блок кода повторяется заданное количество раз.  <code>for i in range(1, 5)     print(i)</code>
forward	Метод для перемещения черепахи вперед; если перо включено, то за черепахой остается след — на экране рисуется прямая линия.  <code>turtle.forward(50)</code> В этом примере черепаха перемещается на 50 единиц.
IDLE	Интегрированная среда разработки; несложный редактор и интерпретатор для языка Python.
if, инструкция	Проверяет, выполняется ли некоторое условие; если оно выполняется, то выполняются следующие строки кода.  <code>if num &lt; 100:     print("too low")</code>
in	Может использоваться для проверки вхождения символа в строке. Ключевое слово <b>in</b> может использоваться в командах <b>for</b> и <b>if</b> . В следующем примере цикла <b>for</b> выводит каждый символ в отдельной строке:  <code>for i in msg:     print(i)</code> А этот пример проверяет, присутствует ли буква в строке:  <code>msg = input("Enter text: ") letter = input("Enter letter: ") if letter in msg:     print("Thank you") else:     print("Not in string")</code>
input	Функция для ввода значения. Обычно присваивается имени переменной.  <code>name = input("Enter name: ")</code>

Понятие	Определение
insert	Вставляет элемент в заданную позицию списка и смещает все последующие элементы, чтобы освободить место. Индексы изменяются в соответствии с их новой позицией в списке.  <code>names_list.insert(1, "Gary")</code>
int	Используется для определения целого числа.  <code>num = int(input("Enter number: "))</code>
islower	Проверяет, что строка содержит только буквы нижнего регистра.  <code>if msg.islower():     print("This message is in lowercase")</code>
isupper	Проверяет, что строка содержит только буквы верхнего регистра.  <code>if msg.isupper():     print("This message is in uppercase")</code>
left	Поворачивает черепаху против часовой стрелки.  <code>turtle.left(120)</code> В данном примере выполняется поворот на 120°.
len	Функция для определения длины значения.  <code>print(len(name))</code>
lower	Метод для преобразования строки к нижнему регистру.  <code>name = name.lower()</code>
Not null	Во время создания таблицы SQL можно указать, что поле не может оставаться пустым при добавлении новой записи:  <code>cursor.execute("""CREATE TABLE IF NOT EXISTS employees(     id integer PRIMARY KEY,     name text NOT NULL,     dept text NOT NULL,     salary integer); """)</code> См. SQL, база данных, таблица и поле.
or	Операция, результат которой равен <b>True</b> в том случае, если истинно хотя бы одно из условий:  <code>if choice == "a" or choice == "b":     print("Thank you") else:     print("Incorrect selection")</code>
pendown	Опускает перо на страницу, чтобы при перемещении черепахи за ней оставался след. По умолчанию перо опущено.  <code>turtle.pendown()</code>

Понятие	Определение
penup	Поднимает перо, чтобы при перемещении черепахи за ней не оставалось следа. <code>turtle.penup()</code>
pi	Возвращает значение числа пи ( $\pi$ ) с точностью до 15 знаков. <pre>import math radius = int(input("Enter the radius: ")) r2 = radius ** 2 area = math.pi * r2 print(area)</pre>
pop	Удаляет последний элемент из списка, кортежа, словаря, строки или массива. <code>names_list.pop()</code>
print	Вывод значений в скобках на экран. <code>print("Hello", name)</code>
randint	Генерирует случайное число. <code>num = random.randint(1, 10)</code>
random	Генерирует случайное вещественное число в диапазоне от 0 до 1. <code>num = random.random()</code>
random, библиотека	Чтобы использовать библиотеку <b>random</b> в Python, необходимо разместить команду <b>import random</b> в начале программы. Также см. randint, choice, random и randrange. <pre>import random num = random.randint(1, 10) correct = False while correct == False:     guess = int(input("Enter a number: "))     if guess == num:         correct = True     elif guess &gt; num:         print("Too high")     else:         print("Too low")</pre>
randrange	Метод для выбора числа из заданного диапазона. Предусмотрена даже возможность выбора шага, разделяющего допустимые значения из диапазона, например: <code>num = random.randrange(0, 100, 5)</code> Команда выбирает случайное число от 0 до 100 с шагом 5, то есть выбираться будут только значения 0, 5, 10, 15 и т. д.

Понятие	Определение
range	<p>Определяет начальную и конечную границу диапазона с возможностью дополнительного определения шага (разности между числами в последовательности). Обычно используется в циклах <b>for</b>:</p> <pre>for i in range(1, 10, 2):     print(i)</pre> <p>Результат:</p> <pre>&gt;&gt;&gt; 1 3 5 7 9</pre>
real	Тип данных, используемый в базах данных SQL для хранения дробных чисел. См. число с плавающей точкой, SQL и база данных.
remove	Удаление элемента из списка. Метод удобен в ситуациях, в которых неизвестен индекс элемента. Если данные присутствуют в нескольких экземплярах, то удаляется только первый экземпляр.
	<code>names_list.remove("Tom")</code>
reverse	Переставляет в обратном порядке элементы списка, кортежа, словаря, строки или массива.
	<code>names_list.reverse()</code>
right	Поворачивает черепаху по часовой стрелке.
	<code>turtle.right(90)</code>
	В данном примере выполняется поворот на 90°.
round	Метод округляет значение до заданного числа знаков.
	<code>newnum = round(num, 2)</code>
sort	Метод упорядочивает список по алфавиту и сохраняет его в новом порядке. Не может использоваться для сортировки списков, содержащих данные разных типов — например, содержащих как строки, так и числовые данные.
	<code>names_list.sort()</code>
sorted	Метод возвращает список, отсортированный в алфавитном порядке. Порядок элементов исходного списка при этом не изменяется — они по-прежнему хранятся в исходном виде. Не может использоваться для списков, содержащих данные разных типов — например, содержащих как строки, так и числовые данные.
	<code>print(sorted(names_list))</code>

Понятие	Определение
SQL	Сокращение от «Structured Query Language», то есть «язык структурированных запросов». SQL используется для работы с базами данных. База данных может содержать несколько таблиц, связанных друг с другом; такие базы данных называются реляционными. Каждая таблица состоит из полей, содержащих определенные данные: идентификатор, имя, адрес и т. д. Каждая строка таблицы называется записью. См. база данных, поле, запись, таблица и запрос.
SQLite	Простая база данных; распространяется бесплатно и хорошо работает в сочетании с Python.
sqrt	Метод для вычисления квадратного корня из числа. Чтобы использовать метод, импортируйте библиотеку <b>math</b> в начале программы. <pre>import math num = math.sqrt(100) print(num)</pre>
str	Преобразование значения в строку. См. строка. <pre>year = str(year)</pre>
strip	Удаляет лишние символы в начале и в конце строки. <pre>text = "    This is some text.    " print(text.strip(" "))</pre>
title	Метод для изменения регистра символов, при котором все слова начинаются с символа верхнего регистра, а все остальные символы относятся к нижнему регистру. <pre>name = name.title()</pre>
Tkinter	Самая популярная библиотека для создания графических интерфейсов в Python.
upper	Метод для преобразования строки к верхнему регистру. <pre>name = name.upper()</pre>
while, цикл	Разновидность цикла, в которой блок кода (обозначенный строками с отступом) повторяется, пока некоторое условие остается истинным. <pre>total = 0 while total &lt;= 50:     num = int(input("Enter a number: "))     total = total + num     print("The total is...",total)</pre>
аргумент	Значение, передаваемое подпрограмме. В следующем примере <b>UserAns</b> – аргумент, который должен определяться за пределами подпрограммы. <pre>def CheckAnswer(UserAns):     if UserAns == 20:         print("Correct")     else:         print("Wrong")</pre>

Понятие	Определение
база данных	Структурированный набор данных. Данные хранятся в таблицах и состоят из полей и записей. См. SQL, таблицы, поля и записи.
библиотека	Подборка кода, связанного с выполнением конкретной функции. Этот код не входит в стандартные блоки кода Python, но может импортироваться по мере необходимости. Для этого импортируйте библиотеку в начале программы. <pre>import math radius = int(input("Enter the radius: ")) r2 = radius ** 2 area = math.pi * r2 print(area)</pre>
больше	Для проверки того, что одно значение больше другого, используется оператор <b>&gt;</b> num1 > num2
больше или равно	Для проверки того, что одно значение больше другого или равно ему, используется оператор <b>&gt;=</b> : num1 >= num2
вложенная команда	Размещение одной команды внутри другой; например, цикл <b>for</b> может быть вложен в инструкцию <b>if</b> . <pre>if num &lt; 20:     for i in range(1, num):         print(i)     else:         print('Too high')</pre>
возведение в степень	Операция возведения в степень выполняется оператором <b>**</b> .
вычитание	Одно значение вычитается из другого. <pre>&gt;&gt;&gt; 5 - 2 3</pre>
графический интерфейс	Графический интерфейс использует окна, текстовые поля и списки, с которыми можно взаимодействовать с помощью мышки. См. Tkinter.
двойная точность	Формат чисел с плавающей точкой, обеспечивающий представление чисел от -10 308 до 10 308.

Понятие	Определение
двумерный список	<p>Список с несколькими измерениями. Например, для создания следующей таблицы данных:</p> <pre data-bbox="384 358 410 440">0 1 2</pre> <pre data-bbox="384 472 410 592">0 23 16 34</pre> <pre data-bbox="384 624 410 725">1 45 29 48</pre> <p>используется следующий код:</p> <pre data-bbox="384 815 941 840">number_list = [[23, 16, 34], [45, 29, 48]]</pre>
деление	<p>Одно значение делится на другое, а результат представляется в виде значения с плавающей точкой.</p> <pre data-bbox="384 929 483 981">&gt;&gt;&gt; 5/2 2.5</pre>
длинное целое	<p>Целое число в диапазоне от -2 147 483 648 до 2 147 483 647.</p>
запись	<p>В терминологии баз данных записью называется полный набор полей — например, данные одного работника хранятся в одной строке таблицы. См. SQL, база данных, таблица и поле.</p>
запись в несуществующий файл	<p>Режим, при котором создается новый файл для записи информации. Если файл с заданным именем уже существует, выдается ошибка:</p> <pre data-bbox="384 1234 797 1343">file = open("newlist.csv", "x") newrecord = "Tim, 43 \n" file.write(str(newrecord)) file.close()</pre> <p>См. запись в файл, присоединение к файлу, чтение файла.</p>
запись в файл	<p>Режим, при котором создается новый текстовый файл или файл .CSV; если файл с заданным именем уже существует, он заменяется новым:</p> <pre data-bbox="384 1482 827 1621">file = open("Countries.txt", "w") file.write("Italy \n") file.write("Germany \n") file.write("Spain \n") file.close()</pre> <p>См. запись в несуществующий файл, присоединение к файлу, чтение файла.</p>

Понятие	Определение
запрос	<p>Используется для получения информации из базы данных.</p> <pre>cursor.execute("""SELECT employees.id, employees.name, dept.manager     FROM employees.dept WHERE employees.dept = dept.dept     AND employees.dept='Sales' """) for x in cursor.fetchall():     print(x)</pre> <p>См. SQL, база данных, таблица и поле.</p>
запуск программы	Откройте меню Run и выберите команду Run Module или нажмите клавишу F5. Перед запуском программа должна быть сохранена.
изображения	<p>В графический интерфейс могут включаться изображения. Существуют два основных режима вывода графики. В первом блоке кода выводится логотип, который не изменяется за время выполнения программы:</p> <pre>logo = PhotoImage(file = "logo.gif") logoimage = Label(image = logo) logoimage.place(x = 30, y = 20, width = 200, height = 120)</pre> <p>Во втором блоке кода изображение изменяется в зависимости от значения, выбранного в списке:</p> <pre>photo = PhotoImage(file = "logo.gif") photobox = Label(window, image = photo) photobox.image = photo photobox.place(x = 200, y = 200, width = 200, height = 120)</pre> <p>См. Tkinter и список.</p>
индекс	<p>Число, задающее позицию отдельного значения в списке, кортеже, словаре или строке. В Python нумерация позиций начинается с 0, а не с 1, так что при автоматическом генерировании индексов первому элементу будет присвоен индекс 0.</p> <pre>colours = ["red", "blue", "green"] print(colours.index("blue"))</pre>
интерпретация	Выполнение программы с последовательной обработкой ее строк.
итерация	Повторение выполнения кода (например, в цикле <b>while</b> или <b>for</b> ).
кавычки	<p>Используются для определения набора символов как строки. Использовать можно как двойные кавычки ("), так и одинарные ('), но тот символ, который был выбран для пометки начала строки, должен использоваться и для ее завершения.</p> <pre>print("This is a string")</pre> <p>Тройные кавычки используются для сохранения форматирования (например, разрывов строк).</p> <pre>address = """ 123 Long Lane Oldtown AB1 23CD print(address)</pre>

Понятие	Определение
кнопка	Используется в графических интерфейсах на базе Tkinter. Следующая команда создает кнопку для выполнения подпрограммы <b>click</b> :  <pre>button1 = Button(text = "Click here", command = click)</pre> <p>См. Tkinter.</p>
командная оболочка	Первое окно, которое вы видите при запуске Python.
комментарии	Используются для пояснения логики работы программы или временного блокирования кода для тестирования. Комментарии начинаются с символа #.  <pre>if salary &gt; 50000: # Комментарий     print("Too high")</pre> <pre># Еще один комментарий</pre>
компилятор	Преобразует программу, написанную на языке высокого уровня (таком как Python), на язык низкого уровня (например, машинный код).
конкатенация	Соединение двух строк с созданием новой строки (см. сложение):  <pre>name = firstname + surname</pre>
кортеж	Список значений, который не может изменяться во время выполнения программы. Обычно используется для определения команд меню, изменение которых в процессе работы маловероятно.  <pre>menu = ('Open', 'Print', 'Close')</pre>
круглые скобки	Значения в круглых скобках определяются как кортеж. См. кортеж.  <pre>tuple = ('a', 'b', 'c') for i in tuple:     print(i)</pre>
логические ошибки	Ошибки, вызванные некорректной логикой программы (а не нарушением синтаксиса), — например, вместо знака > в программе используется <.
массив	В Python массивы являются аналогами списков, но они могут использоваться только для хранения чисел. Пользователь определяет конкретный числовой тип (целое, длинное целое, с плавающей точкой или число двойной точности).  <pre>nums = array('i', [45, 324, 654, 45, 264]) print(nums)</pre> <p>Если в массиве должны храниться строки, необходим список.</p>
меньше	Для проверки того, что одно значение меньше другого, используется оператор <  <pre>num1 &lt; num2</pre>
меньше или равно	Для проверки того, что одно значение меньше другого или равно ему, используется оператор <=:  <pre>num1 &lt;= num2</pre>

Понятие	Определение
надпись	Используется в графических интерфейсах для вывода текста. Следующая команда создает надпись с указанным сообщением:  <pre>label1 = Label(text = "Enter a number: ")</pre> <p>См. Tkinter.</p>
не равно	Для проверки значений на неравенство используется оператор !=  <pre>num1 != num2</pre>
неизменяемый	Значение неизменяемых данных не может модифицироваться после их создания. Например, данные в кортеже неизменяемы, поэтому после запуска программы кортеж изменить не удастся.
окно	Экран, используемый в графических интерфейсах для вывода. Следующий фрагмент кода создает окно, добавляет заголовок и определяет размер окна.  <pre>window = Tk() window.title = ("Add title here") window.geometry("450x100")</pre> <p>См. Tkinter.</p>
определение подпрограммы	Создание подпрограммы, которая может использоваться в других частях программы. См. def.
остаток	Оператор % вычисляет остаток после целочисленного деления:  <pre>&gt;&gt;&gt; 5 % 2 1</pre>
отладка	Процесс поиска и устранения ошибок программирования.
отступы	Используются в Python для обозначения строк кода, принадлежащих другой команде. Например, строки под заголовком цикла <b>for</b> снабжаются отступом, потому что они составляют тело цикла. Строки без отступа не входят в цикл.  <pre>for n in range(0,10):     count = n + 1     print(count) print("The end")</pre> <p>Для ввода отступов в строках можно использовать клавишу табуляции или пробел.</p>
ошибка времени выполнения	Ошибка, возникающая при попытке запуска программы. Например, Python не сможет работать с переменной, хранящейся в виде строки, там, где ожидает получить число. Ошибки времени выполнения приведут к аварийному завершению программы и выводу сообщений об ошибках следующего вида:  <pre>Traceback (most recent call last):   File "C:/Python34/CHALLENGES/testingagain.py", line 2, in &lt;module&gt;     total = num + 100 TypeError: Can't convert 'int' object to str implicitly</pre>

Понятие	Определение
первичный ключ	Поле базы данных, содержащее уникальный идентификатор каждой записи. <pre>cursor.execute("""CREATE TABLE IF NOT EXISTS employees(     id integer PRIMARY KEY,     name text NOT NULL,     dept text NOT NULL,     salary integer); """)</pre> См. SQL, база данных, таблица, запись и поле.
передача переменных	Создание или изменение переменной в одной подпрограмме, при котором эта переменная может использоваться в другой части программы. См. подпрограмма.
переменные	Области памяти для хранения текста и чисел. Для присваивания значений используется знак =. <pre>num = 54</pre>
подпрограмма	Блок кода, который может вызываться для выполнения из другой части программы и возвращать значение. <pre>def get_data():     user_name = input("Enter your name: ")     user_age = int(input("Enter your age: "))     data_tuple = (user_name, user_age)     return data_tuple  def message(user_name,user_age):     if user_age &lt;= 10:         print("Hi", user_name)     else:         print("Hello", user_name)  def main():     user_name,user_age = get_data()     message(user_name,user_age)  main()</pre>
поле	В базе данных полем называется отдельный фрагмент данных (имя, дата рождения, телефон и т. д.), хранящийся в таблице. См. SQL, база данных, таблица и запись.
приглашение	Символы >>> в окне командной оболочки Python. Используется для ввода данных.
присоединение	Добавление одного элемента в конец списка, кортежа, словаря, строки или массива. <pre>names_list.append("Timothy")</pre>

Понятие	Определение
присоединение к файлу	<p>Режим, при котором существующий текстовый файл или файл .CSV открывается для добавления информации в конец существующего содержимого:</p> <pre>file = open("Countries.txt", "a") file.write("France \n") file.close</pre> <p>Также см. запись в файл, запись в несуществующий файл, чтение файла.</p>
пробелы, удаление	См. strip.
равно	<p>Для проверки значений на равенство используется оператор ==</p> <pre>if guess == num:</pre>
разрыв строки	<p>Обеспечивает перенос текста на новую строку.</p> <pre>print("Hello \n How are you?")</pre> <p>Запуск данного кода выдает следующий результат:</p> <pre>Hello How are you?</pre>
раскрывающийся список	<p>Элемент графического интерфейса.</p> <pre>selectname = StringVar(window) selectname.set("Select Name")  nameslist = OptionMenu(window, selectname, "Bob", "Sue", "Tim") nameslist.place(x = 30, y = 250)</pre> <p>См. Tkinter.</p>
решетка	См. комментарии.
синтаксическая ошибка	Ошибка программирования, которая возникает в том случае, если команды располагаются в неправильном порядке или содержат опечатки.
словарь	<p>Тип списка, в котором значения связываются с индексами, определяемыми пользователем.</p> <pre>scores = {"Tim":20, "Sue":35, "Bob":29}</pre>
сложение	<p>Два значения суммируются, если они являются числами:</p> <pre>total = num1 + num2</pre> <p>или соединяются, если значения содержат текст:</p> <pre>name = firstname + surname</pre>
список	<p>Аналог массивов из других языков программирования. Списки позволяют хранить наборы данных под одним именем переменной и могут изменяться во время выполнения программы.</p> <pre>list=['a','b','c'] for i in list:     print(i)</pre>

Понятие	Определение
список (элемент)	Используется в графических интерфейсах на базе Tkinter. Следующая команда создает список, который может использоваться только для вывода.  <code>list_box = Listbox()</code> См. Tkinter.
степень	См. возведение в степень.
строка	Последовательность букв, цифр и других символов, заключенная в одинарные или двойные кавычки. Строки не могут использоваться в математических вычислениях, даже если они содержат только цифры. Тем не менее они могут участвовать в конкатенации и объединяться с другими строками для создания больших строк. См. конкатенация.
таблица	Контейнер для данных. База данных может содержать несколько таблиц, которые могут связываться друг с другом. Ниже приведен пример таблицы для хранения данных работников.  ID Name Dept Salary  1 1 Bob Sales 25000  2 2 Sue IT 28500  3 3 Tim Sales 25000  4 4 Anne Admin 18500

Понятие	Определение
	<p>5 5 Paul IT 28500</p> <p>6 6 Simon Sales 22000</p> <p>7 7 Karen Manufacturing 18500</p> <p>8 8 Mark Manufacturing 19000</p> <p>9 9 George Manufacturing 18500</p> <p>10 10 Keith Manufacturing 15000</p> <p>См. SQL, база данных, поле и запись.</p>
текстовая область	<p>Используется в графических интерфейсах на базе Tkinter; создает поле для вывода текстового сообщения.</p> <pre>output_box = Message(text = 0)</pre> <p>См. Tkinter.</p>
текстовое поле	<p>Используется в графических интерфейсах с Tkinter для ввода или вывода данных. Следующая команда создает пустое текстовое поле:</p> <pre>entry_box = Entry(text = 0)</pre> <p>См. Tkinter.</p>

Понятие	Определение
текстовый файл	<p>Объект, который импортируется в Python и используется программой для чтения и записи данных в этот файл.</p> <pre>file = open("Names.txt", "a") newname = input("Enter a new name: ") file.write(newname + "\n") file.close</pre> <pre>file = open("Names.txt", "r") print(file.read())</pre> <p>См. запись в файл, чтение файла и присоединение к файлу.</p>
умножение	<p>Одно значение умножается на другое.</p> <pre>&gt;&gt;&gt; 3 * 4 12</pre>
условие	<p>Конструкция, используемая для проверки условия. Обычно используется в инструкциях <b>if</b>, циклах <b>while</b> и <b>for</b>.</p> <pre>if guess == num:</pre>
фигурные скобки	<p>Используются для определения значений в словаре:</p> <pre>scores = {"Tim":20, "Sue":35, "Bob":29}</pre>
целое число	<p>Тип данных для представления чисел от -32 768 до 32 767.</p>
целочисленное деление	<p>Вычисление результата от деления нацело одного числа (делимое) на другое (делитель).</p> <pre>&gt;&gt;&gt; 15 // 7 2</pre>
цикл	<p>См. <b>for</b>, цикл и <b>while</b>, цикл.</p>
черепаха	<p>Инструмент для рисований линий на экране.</p> <pre>import turtle for i in range(0, 4):     turtle.forward(100)     turtle.right(90) turtle.exitonclick()</pre> <p>См. <b>forward</b>, <b>left</b>, <b>right</b>, <b>penup</b>, <b>pendown</b>.</p>
число с плавающей точкой	<p>Формат дробных чисел, обеспечивающий представление от -10<sup>38</sup> до 10<sup>38</sup> (то есть до 38 цифровых символов, включая разделитель дробной части в произвольной позиции числа; число может быть как положительным, так и отрицательным).</p> <pre>num = float(input("Enter number: "))</pre>

Понятие	Определение
чтение файла	Открывает существующий текстовый файл или файл .CSV для чтения данных: <pre>file = open("Countries.txt", "r") print (file.read())</pre> Также см. запись в файл, запись в несуществующий файл, присоединение к файлу.
язык структурированных запросов	См. SQL.