◀ ШКОЛА ЮНОГО ПРОГРАММИСТА ▶

А. В. Щерба

Программирование на Python®

Первые шаги

Москва
Лаборатория знаний

Оглавление

Введение

Программист — профессия, о которой многие наверняка что-то слышали. На ум приходит образ человека, постоянно сидящего за компьютером и «клацающего» по клавиатуре. И если не каждый учащийся после школы собирается стать программистом, то зачем учиться этому в школе? Попробуем ответить на этот вопрос.

Программа и программирование

Если говорить простыми словами, то ***программа*** — это инструкции для машины, ***язык программирования*** — способ их передачи, а ***программирование*** — сам процесс написания подобных инструкций на выбранном языке.

Как и для людей, для машины одна и та же инструкция может быть записана на различных языках (C++, Python, JavaScript, Ruby и др.).

В данной книге мы рассмотрим язык программирования Python. Этот язык характеризуется огромным количеством решаемых задач, простотой изучения и удобством работы с кодом, а также пользуется большим спросом среди работодателей и в IT-сообществе. Например, язык Python применяют для создания таких web-приложений, как Gmail, Google Maps и YouTube.

В чем ценность умения программировать?

Стоит отметить, что любой навык не обязательно используется именно в профессиональной сфере. Так, умение быстро считать, приобретаемое на уроках математики, способствует развитию быстрой реакции на поставленную задачу, составление таблиц по биологии или истории учит навыку выбирать главное и прослеживать взаимосвязи, изучение языков показывает наличие совершенно другого типа мышления.

Тогда вопрос о важности приобретения этих навыков в школе становится практически риторическим и сводится к вопросу, важно ли научиться быстро реагировать, находить главное из

большого потока информации и смотреть на вещи под разным углом. Можете ответить на этот вопрос самостоятельно и поразмышлять о том, какие еще навыки развиваются в школе.

Умение структурировать и оптимизировать информацию/процессы, создавать понятный интерфейс приложения, доступный и удобный онлайн-сервис, организовывать диалог между пользователем и системой — все это находится на расстоянии вытянутой руки в самой распространенной профессиональной области XXI в.

Подобным навыкам начинают обучаться уже в школьные годы. Программирование как процесс помогает научиться выделять главное, раскладывать сложное на простое, развивает дальновидность и креативность. Что вовсе может не касаться компьютерной науки.

Как оптимизировать свои действия? С чего начать написание проекта или доклада?

Программирование — зерно, которое в скором времени обязательно даст свои плоды в виде уменьшения времени, которое затрачивается, например, на выполнение домашнего задания или создание проекта. Глобально же навык программирования можно рассматривать как инструмент развития личности, который пригодится во всех сферах жизни.

Предлагаем смело начать изучение основ программирования на языке Python, а в процессе самостоятельно решить, хотите вы начать заниматься этим на более глубоком и профессиональном уровне или применять освоенные навыки в повседневной жизни.

Глава 1. Знакомство со средой программирования IDLE и первая программа

IDLE — это среда, которая позволяет просматривать, редактировать, запускать и производить отладку программ на языке Python.

Данная среда программирования является свободно распространяемым программным обеспечением, доступным для скачивания с сайта www.python.org, поэтому ею может воспользоваться любой пользователь сети Интернет.

Установка среды IDLE

Пользователям ***Windows*** необходимо скачать установочный файл на официальном сайте https://www.python.org/downloads/windows/, кликнуть дважды по загруженному файлу и следовать инструкциям установщика (рис. 1).

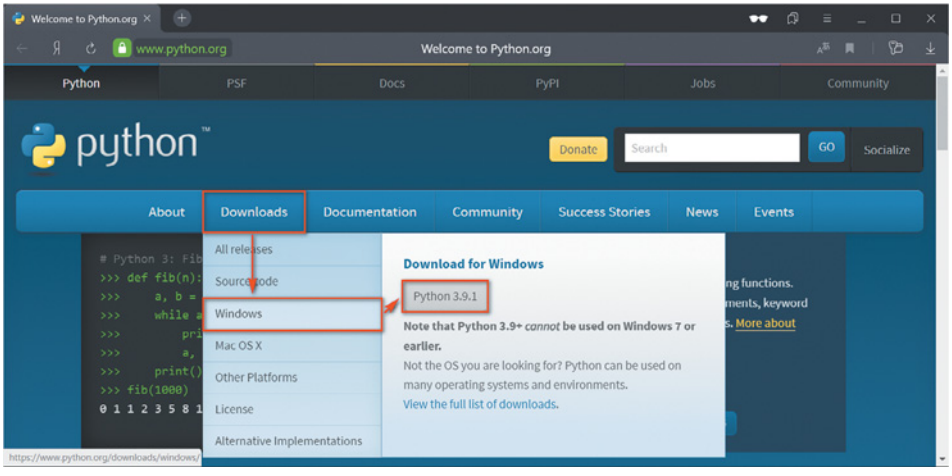


Рис. 1. Загрузка установочного файла

На ***Linux*** или ***Mac OS*** язык Python может быть уже установлен и готов к использованию, поскольку является стандартным компонентом этих операционных систем. Если его все же нет,

его можно скачать на официальном сайте, в том же разделе, что и для Windows.

В ***Linux*** для установки также достаточно двух команд в терминале:

```
$ sudo apt-get update
$ sudo apt-get install idle3
```

Интерфейс среды IDLE

После загрузки и установки Python откройте IDLE. На экране появится следующее окно (рис. 2):

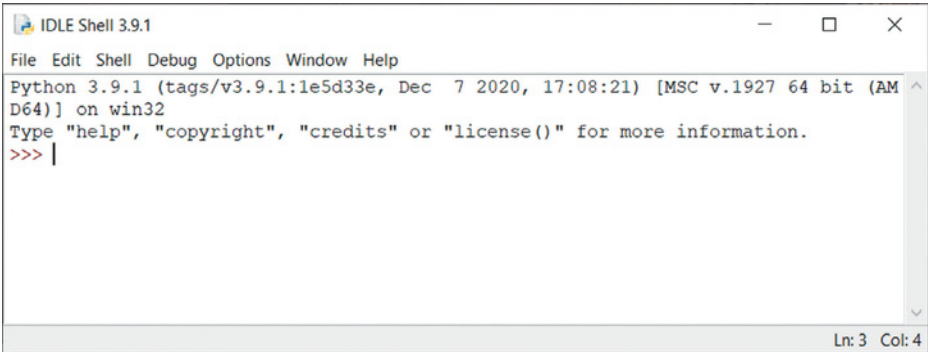


Рис. 2. Интерактивный режим IDLE

Перед нами ***интерпретатор Python*** — специальный модуль, который позволяет процессору считывать команды, записанные на языке программирования, и исполнять их. Другими словами, когда мы пишем код на языке Python, интерпретатор «читает» нашу программу и покомандно выполняет написанные в ней инструкции, опуская промежуточный этап сборки (компиляцию единого файла), в отличие от многих других языков программирования.

Существует два вида работы в IDLE: *интерактивный* и *с помощью создания отдельного файла*.

Интерактивный сеанс в IDLE начинается с вывода двух строк информационного текста о дате, времени и разрядности операционной системы, которые можно видеть на рис. 2, затем выводится приглашение к вводу команды >>>.

Ввод каждой инструкции завершается нажатием клавиши **Enter**, после чего интерпретатор Python выполняет эту операцию и выдает результат или сообщение об ошибке.

В интерактивном режиме можно вводить любое число команд, и каждая из них будет выполняться сразу же после ввода. Такой тип работы также называют работой в ***интерактивной оболочке***.

Второй тип работы в IDLE мы рассмотрим позже в данной главе.

Первая программа на языке Python

Традиционно первой программой при изучении нового языка программирования является вывод строки «Hello, world!», символизирующей начало диалога между машиной и пользователем.

В строке приглашения к вводу введем первую инструкцию:

```
>>>  print ('Hello, world!')
```

Теперь нажмем **Enter** и увидим, как интерпретатор Python моментально выполнит указание напечатать строку «Hello, world!» (рис. 3). Наша первая программа (инструкция для Python) готова!

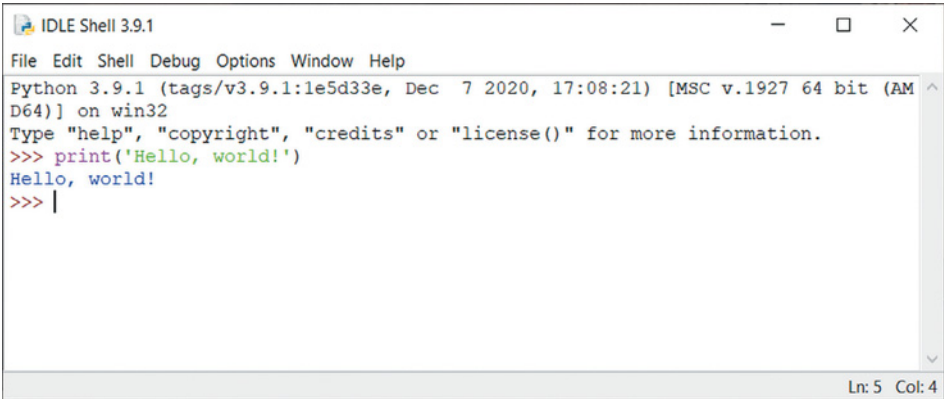


Рис. 3. Запуск первой программы на Python

Оператор print() — вывод данных на экран

Вывод на экран, пожалуй, одна из самых важных функций в программировании. С ее помощью осуществляется передача информации пользователю или программисту.

Обратите внимание, что `'Hello, world!'` в нашей первой программе записано в кавычках. Так мы показываем Python, что необходимо вывести данную последовательность символов (включая пробелы) без изменений. Это также означает, что последовательность символов может быть любой и даже не иметь смысловой нагрузки. Например, `'!'`, `'Error'`, `'  '` или `'ytrewq'`.

Последовательность символов, заключенная в кавычки, называется ***строкой*** (не путать со строчкой в тексте) и является *неизменяемым типом данных*. Поэтому команда `print('1')` выведет на экран не число 1, а символ «1», с которым, например, нельзя будет производить арифметические операции.

Важно!
Интерактивная оболочка позволяет не использовать `print()`, поскольку в данном режиме нажатие клавиши **Enter** подразумевает автоматический вывод результата команды (рис. 4).

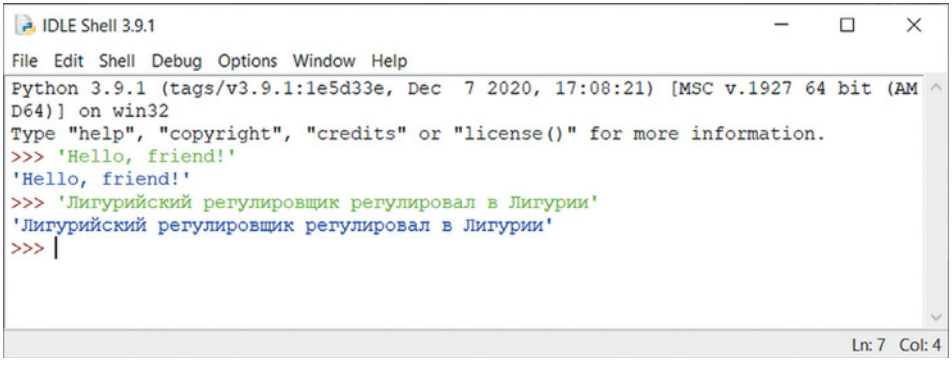


Рис. 4. Возможность печатать строки без команды `print()`

Эксперименты в интерактивной оболочке

Благодаря тому что программный код выполняется немедленно, интерактивный режим превращается в замечательный инструмент для экспериментов.

Предположим, что мы изучаем некоторый фрагмент программы на языке Python и наталкиваемся на выражения:

```
'Spam!'*8
'1'*8
1*8
```

Можно потратить с десяток минут в попытках выяснить, что же делают эти инструкции, а можно выполнить их в интерактивной оболочке — так будет намного быстрее и проще (рис. 5).

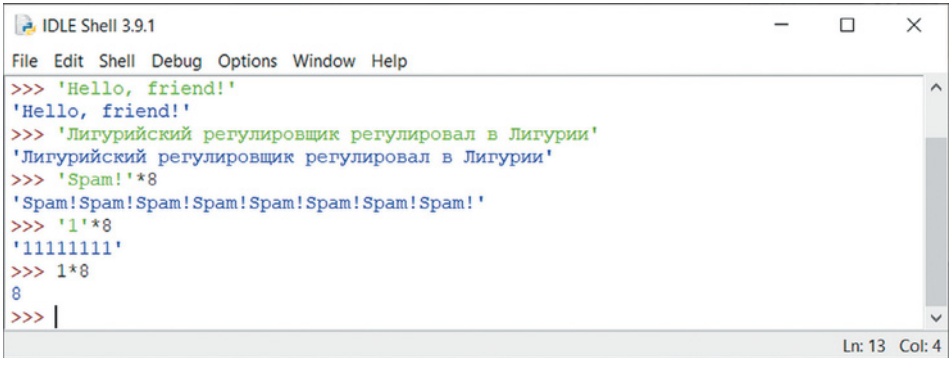


Рис. 5. Инструкция для Python вывести на экран `'Spam!'*8, '1'*8, 1*8`

Первый эксперимент наглядно показывает, что произошло умножение строки `'Spam!'` на число `8`: в языке Python оператор `*` выполняет операцию умножения над числами, но, если левый операнд является строкой, он действует как оператор многократной *конкатенации* строки с самой собой.

Не совсем понятно, правда? Разберем несколько новых понятий: «переменная», «операция», «оператор», «операнд», «выражение», «конкатенация».

Переменная — это *объект* (реализуемый как именованая область памяти), который может принимать различные значения. Название переменной начинается с одной или нескольких латинских букв (например, *b*, *sum*), может содержать цифры и знаки подчеркивания (например, *num*1, *num_2*, *num_3_1* и т. д.). При этом имена переменных в Python чувствительны к регистру (например, *Number*, *NUMBER*, *number* — это три различные переменные).

Операция — это некоторое *действие*, которое необходимо совершить над числами и/или переменными (например, *сложение*, *вычитание*, *умножение*, *деление* и т. д.).

Оператор — это объект (символ), который *выполняет операцию* и имеет привычную символьную запись (например, +, –, *, /).

Операнд — это объект (число, символ, строка или переменная), *над которым* оператор выполняет операцию.

Таким образом, ***выражение*** — совокупность *операций*, которые выполняются *операторами* над *операндами*.

Например:

b + 5 — выражение;
сложение — операция;
\+ — оператор;
b, 5 — операнды.

Конкатенация строк — операция *присоединения*, «склеивания» символов или их наборов.

Операция конкатенации строк возможна не только с помощью оператора *, но и с помощью оператора +. Также их можно использовать вместе (рис. 6).

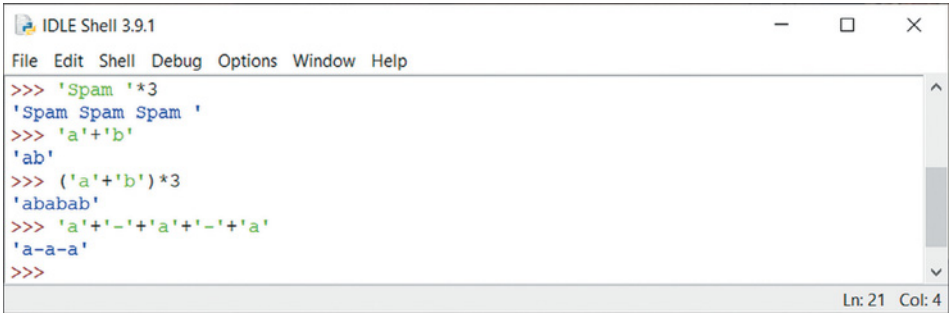


Рис. 6. Пример конкатенации строк с помощью операторов * и +

Эксперименты с кавычками

Уделим особое внимание одинарным и двойным кавычкам, которые используются при выводе строк, через эксперимент в интерактивном режиме:

```
>>> 'Одинарные кавычки уже были рассмотрены нами'
'Одинарные кавычки уже были рассмотрены нами'

>>> "Как насчет двойных? - Ого! Так тоже работает!"
'Как насчет двойных? - Ого! Так тоже работает!'

>>> "Python считал наши двойные кавычки, но вывел одинарные...хм"
'Python считал наши двойные кавычки, но вывел одинарные...хм'
```

```
>>> "Хотим двойные кавычки! Python, выведи "Москва". Пожалуйста!"
SyntaxError: invalid syntax

>>> 'Москва не подсвечивается зеленым цветом. А это значит, что Python видит только строки между одинарными кавычками или между двойными. "Попробуем совместить"'
'Москва не подсвечивается зеленым цветом. А это значит, что Python видит только строки между одинарными кавычками или между двойными. "Попробуем совместить"'

>>> "Получилось! А если 'внутри' предложения взять одинарные кавычки, а снаружи - двойные?"
"Получилось! А если 'внутри' предложения взять одинарные кавычки, а снаружи - двойные?"

>>> 'А если все одинарные'?':'
SyntaxError: invalid syntax

>>> "No :("
'No :('

>>> 'ВЫВОД'
'ВЫВОД'

>>> "ВЫВОД"
'ВЫВОД'

>>> "ВЫВОД'вывод'ВЫВОД"
"ВЫВОД'вывод'ВЫВОД"

>>> 'вывод"ВЫВОД"вывод'
'вывод"ВЫВОД"вывод'
```

Вывод:

- Чтобы вывести строку, не содержащую текста с кавычками, можно использовать как одинарные, так и двойные кавычки.
- Если необходимо вывести строку, содержащую текст с кавычками, внешние и внутренние кавычки *не* должны совпадать.

В некоторых языках программирования одинарные и двойные кавычки предназначены для разных целей. Python же позволяет использовать оба варианта, но строка обязательно должна начинаться и заканчиваться одним и тем же типом кавычек.

Говорят, что бывают тройные одинарные и тройные двойные кавычки. Пробуем! Эксперимент продолжается:

```
>>> """Попробуем просто написать что-нибудь в тройных
двойных кавычках"""
'Попробуем просто написать что-нибудь в тройных двойных
кавычках'

>>> '''А теперь в тройных одинарных'''
'А теперь в тройных одинарных'

>>> """Говорят, что у тройных кавычек есть своя особен-
ность…
Кстати, почему мысли растекаются на такие большие пред-
ложения, что даже в одну строку не вмещаются?"""
'Говорят, что у тройных кавычек есть своя особенность…
\nКстати, почему мысли растекаются на такие большие
предложения, что даже в одну строку не вмещаются?'

>>> "Ребята, вы видели?! Перед словом 'Кстати' была на-
жата кнопка Enter, а Python записал все в одну строку"
"Ребята, вы видели?! Перед словом 'Кстати' была нажата
кнопка Enter, а Python записал все в одну строку"

>>> "Может, каждый перенос строки обозначается \n? Про-
верим:
SyntaxError: EOL while scanning string literal

>>> 'Ах да! Нужно не забывать закрывать кавычки. Итак,
проверяем:'
'Ах да! Нужно не забывать закрывать кавычки. Итак, про-
веряем:'

>>> """abra
kadabra
dabra
badra
bodra
bobra
dobra!"""
'abra\nkadabra\ndabra\nbadra\nbodra\nbobra\ndobra!'

>>> 'Значит, так и есть. С тройными одинарными так же?'
'Значит, так и есть. С тройными одинарными так же?'
```

```
>>> '''
poly
moly
usy
pusy'''
'poly\nmoly\nusy\npusy'

>>> 'Хорошо, тут то же самое. Так чем же отличаются одинарные и двойные от тройных одинарных и тройных двойных кавычек?'
'Хорошо, тут то же самое. Так чем же отличаются одинарные и двойные от тройных одинарных и тройных двойных кавычек?'

>>> 'Между прочим, хотелось сейчас продолжить писать на другой строке, но после нажатия Enter строка с одинарными кавычками была моментально выполнена Python. Это значит, что только с тройными кавычками можно переходить на другую строку при нажатии клавиши Enter. Вот и разобрались.'
'Между прочим, хотелось сейчас продолжить писать на другой строке, но после нажатия Enter строка с одинарными кавычками была моментально выполнена Python. Это значит, что только с тройными кавычками можно переходить на другую строку при нажатии клавиши Enter. Вот и разобрались.'
```

Вывод:

Тройные одинарные и тройные двойные кавычки позволяют делать перенос строки с помощью клавиши **Enter**, который после выполнения инструкции обозначается *специальным символом* `\n`.

Арифметические операторы

История первого компьютера рассказывает о том, что изначально компьютер был создан как большая вычислительная машина и только спустя некоторое время появились дополнительные устройства ввода и вывода информации (монитор, клавиатура, мышь и т. д.).

Поэтому использование Python в качестве калькулятора — базовая возможность, и реализуется она с помощью арифметических операторов:

+	-	*	/	**	//	%

Ниже представлены некоторые варианты вычислений с различными видами чисел (натуральные, целые, дробные) в интерактивном режиме:

+	-	*	/
>>>2+5 7 >>>-2+5 3 >>>2+(-5.0) -3.0 >>>2+-5 -3 >>>2+2.3 4.3 >>>5.2+3.03 8.23	>>>2-5 -3 >>>-2-5 -7 >>>2-(-5.0) 7.0 >>>2--5 7 >>>2.3-2 0.3 >>>-5.2-3.03 -8.23	>>>2*5 10 >>>-2*5 -10 >>>2*(-7.2) -14.4 >>>2*-5 -10 >>>-2*2.3 -4.6 >>>5.2*3.03 15.756	>>>2/5 0.4 >>>-2/5 -0.4 >>>20/-4 -5.0 >>>4.6/2.3 2.0 >>>-5/25 -0.2 >>>-24/-25 0.96

Важно!

- Все дробные числа записаны с точкой, а не с запятой.
- В отличие от привычной математики, в Python могут ставиться два арифметических знака подряд. Например, 2+-5. Первый знак будет считываться как операция, второй — как знак числа.

В общем случае Python различает приоритет выполнения операций:

1) унарные операции (унарный минус);
2) операции умножения и деления;
3) операции сложения и вычитания.

Важно!
Для изменения порядка выполнения используются круглые скобки.

Рассмотрим следующую таблицу с менее знакомыми операторами:

**	//	%
>>>2**5 32 >>>-3**2 9 >>>-1**9 -1	>>>2//5 0 >>>16//13 1 >>> -5//2 -3	>>>2%5 2 >>>16%13 3 >>>-20%7 1

Оператор ** отвечает за операцию возведения в степень. Таким образом, число слева от оператора ** — это *основание степени*, а число справа — *показатель степени*:

```
 2 ** 5 = 32,
-3 ** 2 =  9,
-1 ** 9 = -1,
```

так как $2^5 = 32$;
так как $(-3)^2 = 9$;
так как $(-1)^9 = -1$.

Оператор // отвечает за целую часть при делении первого числа на второе и округляет результат до ближайшего наименьшего целого числа[1]:

```
 2 //  5 = 0,
```

так как $\frac{2}{5} = 0\frac{2}{5}$ (ближайшие целые числа 0 и 1, 0 — наименьшее);

```
16 // 13 = 1,
```

так как $\frac{16}{13} = 1\frac{3}{13}$ (ближайшие целые числа 1 и 2, 1 — наименьшее);

```
-5 //  2 = -3,
```

так как $-\frac{5}{2} = -2\frac{1}{2}$ (ближайшие целые числа −2 и −3, −3 — наименьшее).

[1] До 3-й версии языка при делении целых чисел с помощью оператора / тоже отбрасывалась дробная часть (2/4 = 0, 2//4 = 0). Разницу можно было увидеть только при делении целых и дробных чисел (2.0/4 = 0.5, 2.0//4 = 0.0). Затем разработчики перераспределили роли данных операторов, и за отбрасывание дробной части стал отвечать только оператор //.

Оператор `%` отвечает за остаток при делении первого числа на второе и выводит количество единиц (остаток) до ближайшего наименьшего целого числа:

`2 % 5 = 2,` — так как 2 : 5 = 0 (ост. 2) (2 единицы до наименьшего целого числа 0);

`16 % 13 = 3,` — так как 16 : 13 = 1 (ост. 3) (3 единицы до наименьшего целого числа 1);

`-20 % 3 = 1,` — так как –20 : 3 = (–21+1) : 3 = –7 (ост. 1) (1 единица до наименьшего целого числа –7).

Экспериментируйте с числами, чтобы проверить, правильно ли вы поняли ту или иную новую операцию.

Разработка программ в IDLE

Обратной стороной интерактивного режима является неудобство в создании многострочных программ. Поэтому пришло время поговорить о втором, самом используемом способе работы с IDLE — создании отдельных файлов, что позволяет многократно обращаться, исправлять, дополнять и снова запускать код для получения и проверки результата.

Итак, первой программой в интерактивном режиме был вывод строки «`Hello, world!`» с помощью оператора `print()`. Создадим файл для нашей первой программы.

1. В верхнем поле окна IDLE выберем **File** → **New File** (рис. 7).

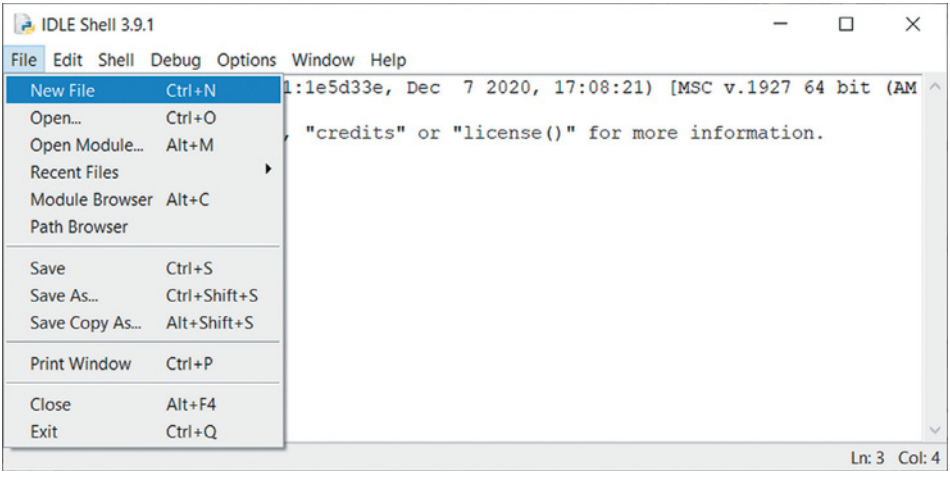
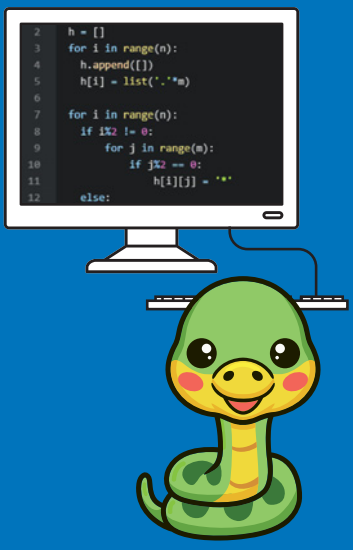


Рис. 7. Создание нового файла в IDLE



Программирование – это грамотность XXI века!

Книги новой серии «Школа юного программиста» издательства «Лаборатория знаний» построены на методике пошагового обучения программированию. Следуя этой методике, любой желающий, от школьника до студента вуза, сможет научиться писать программы, разрабатывать мобильные приложения и компьютерные игры и даже освоить технологии машинного обучения и нейросетей.

Издательство предлагает следующие учебные пособия:

- «Учимся вместе со Scratch: программирование, игры, робототехника» (5–6 классы)
- «Scratch 2.0: от новичка к продвинутому пользователю. Пособие для подготовки к Scratch-Олимпиаде» (1–11 классы)
- «Творческие задания в среде Scratch. Рабочая тетрадь для 5–6 классов»
- «Scratch 3.0: творческие проекты на вырост. Рабочая тетрадь для 7–8 классов»
- «Создаем игры с Kodu Game Lab» (4–5 классы)
- «Разработка мобильных приложений. Первые шаги» (8–11 классы)
- «Компьютерное зрение на PYTHON. Первые шаги» (4–9 классы)