Глава 1
НАЧАЛА ПРОГРАММИРОВАНИЯ

§ 1.1
Общие сведения о языке программирования Python

Ключевые слова:
- язык программирования
- программа
- алфавит
- служебные слова
- типы данных
- структура программы
- оператор присваивания

Языки программирования — это формальные языки, предназначенные для записи алгоритмов, исполнителем которых является компьютер. Алгоритмы, записанные на языках программирования, называют **программами**.

Существует несколько тысяч языков программирования. Один из самых популярных современных языков программирования называется Python (произносится «па́йтон», хотя в России многие называют язык просто «питон»). Его разработал в 1991 году нидерландский программист Гвидо ван Россум. Язык Python непрерывно совершенствуется, и сейчас большинство программистов используют его третью версию — Python 3. Именно с этой версией будем работать и мы.

Python — язык программирования высокого уровня, предназначенный для решения самого широкого круга задач. С его помощью можно обрабатывать различные данные, проводить математические вычисления, создавать изображения, работать с базами данных, разрабатывать веб-сайты.

Для того, чтобы разрабатывать программы на языке Python, нужно установить на компьютер интерпретатор Python. Во многих операционных системах, например в macOS и Linux, этот интерпретатор входит в стандартную поставку и устанавливается вместе с операционной системой. Чтобы установить Python в операционной системе Microsoft Windows, скачайте последнюю версию программы-установщика Python 3 для Windows с официального сайта **http://www.python.org/** (для этого зайдите в меню **Downloads** и выберите **Windows**), запустите загрузившийся файл и следуйте указаниям установщика. Обратите внимание: в установщик Python для Windows встроена интегрированная среда разработки IDLE (произносится «*айдл*»), предназначенная для ввода, просмотра, редактирования, запуска или отладки программы на языке Python.

1.1.1. Алфавит и словарь языка

Основой языка программирования Python, как и любого другого языка, является **алфавит** — набор допустимых символов, которые можно использовать для записи программы. Это:

- латинские прописные и строчные буквы (A, B, C, …, X, Y, Z, a, b, c, …, x, y, z);
- арабские цифры (0, 1, 2, …, 7, 8, 9);
- специальные символы (знак подчёркивания; круглые, квадратные скобки; знаки арифметических операций; # — знак начала однострочного комментария и др.).

В качестве неделимых элементов (составных символов) рассматриваются следующие последовательности символов:

>= и <= (знаки ≥ и ≤);
!= (знак ≠);
""" или ''' (утроенные двойные или одинарные кавычки, ставящиеся в начале и в конце многострочного комментария).

В языке также существует некоторое количество различных цепочек символов, рассматриваемых как единые смысловые элементы с фиксированным значением. Такие цепочки символов называются **служебными словами**. В таблице 1.1 приведены основные служебные слова, которые мы будем использовать при записи программ на языке Python.

Таблица 1.1

Служебное слово языка Python	Значение служебного слова
and	и
break	прервать
elif	иначе если
else	иначе
False	ложь
float	вещественный тип данных (с плавающей точкой)
for	для
if	если
input	ввод
integer	целый
list	список
not	не
or	или
print	печать
string	строковый тип данных (цепочка символов)
True	истина
while	пока

Для обозначения переменных, программ и других объектов используются **имена (идентификаторы)** — любые отличные от служебных слов последовательности букв, цифр и символа подчёркивания, начинающиеся с буквы или символа подчёркивания.

Прописные и строчные буквы в именах различаются, например, `f` и `F` — две разные переменные.

Длина имени может быть любой. Для удобства рекомендуется использовать имена, передающие смысл объектов, с длиной не более 15 символов.

В программах на языке Python (начиная с версии 3) есть возможность использовать в именах буквы национальных алфавитов (от русских до китайских иероглифов). Но это считается очень плохим стилем, так делать не рекомендуется. Подумайте почему.

1.1.2. Типы данных, используемые в языке Python

В языке Python используются различные типы данных (табл. 1.2).

Таблица 1.2

Название	Обозначение	Допустимые значения
Целочисленный	`int` (`integer`)	Сколь угодно большие, размер ограничен оперативной памятью
Вещественный	`float`	Любые числа с дробной частью
Строковый	`str` (`string`)	Любые последовательности символов из таблицы Unicode
Логический	`bool` (`boolean`)	False и True

В вещественном числе целая часть от дробной отделяется точкой, при этом перед точкой и после неё должно быть, по крайней мере, по одной цифре. Пробелы внутри числа недопустимы.

Произвольный набор символов, заключённый в одинарные или двойные кавычки, считается строковой величиной (строкой). Строка может содержать любые символы, набираемые на клавиатуре, в том числе буквы национальных алфавитов.

В отличие от многих других языков программирования переменные в языке Python не нужно объявлять. *Тип переменной определяется автоматически в тот момент, когда ей присваивается новое значение.*

Тип каждой переменной может динамически изменяться по ходу выполнения программы. Определить, какой тип имеет переменная в текущий момент, можно с помощью функции (команды) `type()`.

1.1.3. Режимы работы интерпретатора Python

Интерпретатор Python может работать в двух режимах:

- через командную строку (в командном, или интерактивном режиме), когда каждая введённая команда сразу выполняется;

- в программном режиме, когда программа сначала записывается в файл (обычно имеющий расширение *.py*) и при запуске выполняется целиком.

Изучение языков программирования принято начинать с программы, выводящей на экран надпись: «Привет, мир!». На Python соответствующая программа будет иметь вид:

```
print("Привет, мир!")
```

Для вывода на экран последовательности символов (текста, надписи) используется встроенная команда `print`. Последовательность символов, которые должны быть выведены на экран, заключается в двойные кавычки и записывается в круглых скобках. Вместо двойных кавычек можно использовать одинарные кавычки.

В начале строки (левее команды `print()`) не должно быть пробелов — таково требование языка Python.

Для запуска программы выбираем в меню **Пуск** → **Программы** → **Python 3.7** → **IDLE**. В результате откроется окно *Python Shell*, в котором символы >>> означают приглашение ввести команду. После ввода строки нажимаем клавишу *Enter*. На следующей строке сразу отобразится результат, а далее — приглашение для ввода новой команды (рис. 1.1).

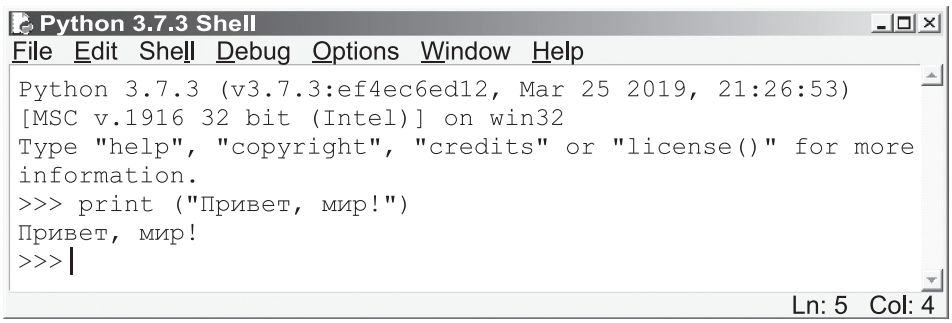

Рис. 1.1. Работа в командном режиме

Для создания файла с программой в меню **File** выбираем пункт **New File**. В открывшемся окне набираем текст программы, а затем сохраняем его под каким-нибудь именем (например, *test.py*), выбрав пункт меню **File** → **Save As**. Запустить программу на выполнение можно, выбрав пункт меню **Run** → **Run Module** или нажав клавишу *F5*.

www

В сети Интернет существуют ресурсы для запуска и отладки программ на Python в режиме online. Вот некоторые из них:

http://pythontutor.com/visualize.html#mode=edit

http://rextester.com/l/python3_online_compiler

https://www.jdoodle.com/python3-programming-online

https://ideone.com/

1.1.4. Оператор присваивания

Программа на языке программирования представляет собой последовательность операторов (инструкций, команд).

Оператор — языковая конструкция, представляющая один шаг из последовательности действий или набор описаний.

Запись значения в переменную выполняет **оператор присваивания**. Общий вид оператора:

```
<имя переменной> = <значение или вычисляемое выражение>
```

Операция присваивания допустима для всех приведённых в табл. 1.2 типов данных. Выражения в языке Python конструируются по рассмотренным в учебнике для 8 класса правилам для алгоритмического языка.

Примеры:

```
a = 25
b = "Привет!"
c = 1.4 + 5.7 * a
d = a < c
e = "Мир! " + b
f = x * (a + c) / 3
```

В правой части оператора присваивания нельзя указывать переменные, которые не были заранее созданы (определены). Так, для переменных `c` и `d` все входящие в соответствующие выражения переменные были заданы выше. Последняя строка содержит ошибку, так как переменная `x` из правой части ранее не была создана (определена).

В языке Python разрешено множественное присваивание. Запись

```
a, b = 19, 25
```

равносильна паре операторов присваивания:

```
a = 19
b = 25
```

При этом считается, что эти два действия происходят параллельно, т. е. одновременно. Если двум переменным присваивается одно и то же значение, можно применить множественное присваивание «по цепочке»:

```
a = b = 5
```

Эта запись равносильна паре операторов `b = 5` и `a = 5`.

Для основных арифметических операций в языке Python используются те же обозначения, что и в алгоритмическом языке:

`+` — сложение;
`-` — вычитание;
`*` — умножение;
`/` — деление;
`**` — возведение в степень.

В языке Python можно использовать сокращённую запись арифметических операций.

Сокращённая запись	**Полная запись**
`a += b`	`a = a + b`
`a -= b`	`a = a - b`
`a *= b`	`a = a * b`
`a /= b`	`a = a / b`
`a **= 2`	`a = a ** 2`

САМОЕ ГЛАВНОЕ

Python — один из самых популярных современных языков программирования. Это язык программирования высокого уровня, предназначенный для самого широкого круга задач.

В Python можно работать в двух режимах:

- через командную строку (в интерактивном режиме), когда каждая введённая команда сразу выполняется;
- в программном режиме, когда программа сначала записывается в файл (обычно имеющий расширение *.py*).

В языке Python используются различные типы данных: целочисленный (`int`), вещественный (`float`), строковый (`str`), логический (`bool`) и другие.

Переменные в языке Python объявлять не нужно; тип переменной автоматически определяется в тот момент, когда ей присваивается новое значение.

Для обозначения переменных, программ и других объектов используются имена (идентификаторы) — любые отличные от служебных слов последовательности букв, цифр и символа подчёркивания, начинающиеся с буквы или символа подчёркивания.

В программах на языке Python есть возможность использовать в именах буквы национальных алфавитов, но это считается очень плохим стилем, и делать так не рекомендуется.

Вопросы и задания

1. Почему язык программирования Python считается универсальным?
2. Что входит в состав алфавита языка Python?
3. Перед вами слова, которые встречаются во многих программах на языке Python. Как эти слова можно перевести на русский язык?

   1) `integer`
   2) `float`
   3) `input`
   4) `print`
   5) `break`
   6) `while`
   7) `else`
   8) `string`

4. Каких правил следует придерживаться при выборе имён для различных объектов в языке Python?
5. Отнесите каждую из следующих последовательностей символов в к одной из трёх групп: 1 — рекомендуемые имена переменных в языке Python; 2 — допустимые имена переменных в языке Python; 3 — недопустимые имена переменных в языке Python.

   а) `1z`
   б) `_1z`
   в) `□`[1)]
   г) `фу`
   д) `z-1`
   е) `ELSE`
   ж) `sUMMA`
   з) `Summa`
   и) `дата`
   к) `1фу`
   л) `n3`
   м) `3n`
   н) `n 3`
   о) `n+3`
   п) `_1_4_5_aAb12_as555`

1) Здесь и далее □ обозначает пробел.

6. Установите соответствие между названиями типов данных и их обозначениями.

а) Целочисленный	1) `str`
б) Вещественный	2) `bool`
в) Строковый	3) `int`
г) Логический	4) `float`

7. В чём разница между числами 100 и 100.0 в языке Python?

8. Охарактеризуйте режимы работы интерпретатора Python:

   1) командный;
   2) программный.

9. В командном режиме введите последовательно следующие строки:

```
a = 10
type(a)
a = '10 10'
type(a)
a = False
type(a)
a = 12.0
type(a)
```

   Сделайте вывод о том, как изменялся тип переменной `a`.

10. Какая ошибка допущена в следующей программе?

```
a = 3
b = 4
s = a * b * d
print(s)
```

11. Какое значение будет присвоено переменной `c` в результате выполнения программы?

```
a, b = 11, 63
c = b = 55
d = b + c - a
```

12. Чему будет равно значение переменной `c` после выполнения программы?

а)
```
a = b = 3
a += 1
c = a + b
```

б)
```
a = b = 5
a += b
c = 2 * a - b
```

в)
```
a = b = 1
a *= 10
c = a / (2 * b)
```

г)
```
a, b = 3, 5
b += 2
c = a + b
```

д)
```
a, b = 5, 3
b += a
c = 10 * b / a
```

е)
```
b, a = 5, 2
b **= a
c = b / a * 4
```

13. Чему будут равны значения переменных `a` и `b` после выполнения программы при указанных начальных значениях? Какими будут типы переменных `a` и `b`?

а) `a = 4` и `b = 0`; б) `a = 0` и `b = 0`.

```
a += 1
b += a
a *= b
b /= 5
a -= a
```

14. Запишите оператор для:
   а) вычисления среднего арифметического `sred` переменных `x1` и `x2`;
   б) уменьшения на единицу значения переменной `k`;
   в) увеличения на единицу значения переменной `i`;
   г) вычисления стоимости покупки `sum`, состоящей из нескольких тетрадей, нескольких ручек и нескольких карандашей.

Проверочная работа № 1

1. Какие утверждения ложны?
   а) `125` — целое число;
   б) `-12.0` — отрицательное целое число;
   в) `'Число Пи'` — вещественное число;
   г) `7 < 6` — логическое значение;
   д) `123.124` — вещественное число;
   е) **`True`** — строковое значение.

2. По сокращённой записи восстановите полную запись оператора присваивания.
   1) `B += 7`
   2) `A -= c`
   3) `C -= a - 5`
   4) `A *= b`
   5) `B /= a + 3`
   6) `C /= a + b * 2`
   7) `A **= (3 - c) * 2`

§ 1.2 Организация ввода и вывода данных

Ключевые слова:
- оператор вывода `print()`
- формат вывода
- оператор ввода `input()`

1.2.1. Вывод данных

В предыдущем параграфе мы познакомились с типами данных и рассмотрели оператор присваивания. Этого достаточно для того, чтобы записать программу преобразования данных. Но результат этих преобразований нам виден не будет.

Для вывода данных из оперативной памяти на экран компьютера используется **оператор (функция) вывода** `print()`:

```
print(<выражение 1>, <выражение 2>,…, <выражение N>)
```

Здесь в круглых скобках помещается список вывода — список выражений, значения которых выводятся на экран. Это могут быть числовые, символьные и логические выражения, в том числе константы и переменные.

Пример. Оператор `print('s=', s)` выполняется так:

1) на экран выводятся символы, заключённые в одинарные кавычки: `s=`
2) на экран выводится значение переменной с именем `s`.

Если значение переменной `s` равно 15, и она имеет целочисленный тип, то на экране появится: `s=␣15`

Обратите внимание: по умолчанию выводимые выражения разделяются одним пробелом, иначе говоря, разделителем между ними является пробел. Оператор `print()` вставляет между выводимыми значениями так называемый разделитель (сепаратор, от англ. *separator*). Мы можем его изменять, указывая новый разделитель после слова `sep`.

Вариант организации вывода	Оператор (функция) вывода	Результат
По умолчанию	`print(1, 20, 300)`	1□20□300
Убрать разделители-пробелы	`print(1, 20, 300, sep='')`	120300
Добавить разделитель-запятую	`print(1, 20, 300, sep=', ')`	1,□20,□300
Вывод каждого значения с новой строки	`print(1, 20, 300, sep='\n')`	1 20 30

Предположим, что мы работаем с натуральными числами, каждое из которых меньше 100. Тогда на одно число на экране достаточно выделить 3 позиции: две позиции на запись самого числа и ещё одну позицию на пробел слева, разделяющий числа. Записывается это так:

```
print("{:3}{:3}{:3}".format(a, b, c))
```

Это **форматный вывод**: строка для вывода строится с помощью функции `format`. Аргументы этой функции (`a`, `b` и `c`) — это те величины, которые выводятся; они указываются в круглых скобках.

Символьная строка слева от точки — это форматная строка, которая определяет **формат вывода**, т. е. как именно величины будут представлены на экране. Фигурные скобки обозначают место для вывода очередного элемента; число после двоеточия — количество позиций, которые отводятся на число; на первом месте выводится значение `a`, на втором — значение `b`, на третьем — `c`. Если цифр в числе меньше, чем зарезервированных под него позиций на экране, то свободные позиции дополняются пробелами слева от числа. Если указанное в формате вывода число меньше, чем необходимо, то оно автоматически будет увеличено до минимально необходимого.

Например, числа 12, 5 и 15 будут выведены так:

□12□□5□15

Числа 12, 5 и 1500 будут выведены следующим образом:

□12□51500

Для вывода вещественного числа в списке вывода для каждого выражения указываются два параметра:

1) общее количество позиций, отводимых на число;
2) количество позиций в дробной части числа:
d — целые числа (int);
f — вещественные (float);
e — экспоненциальный формат.

Фрагмент программы	Результат выполнения оператора вывода
`a = 4` `print ("a=", "{:5d}{:5d}".format(a, a * a))`	a= □□□□4□□□16
`a = 1 / 3; b = 1 / 9` `print ("{:7.3f}{:7.4f}".format(a, b))`	□□0.333□0.1111
`a = 1 / 3` `print ("{:10.3e}".format(a))`	□3.333e-01

При выполнении очередного оператора print() по умолчанию вывод продолжается в новой строке. Чтобы убрать переход к новой строке, используется параметр end:

```
print(a, end="")    # убран переход на новую строку
print(b)
```

1.2.2. Первая программа на языке Python

Пользуясь рассмотренными операторами, составим программу, вычисляющую длину окружности и площадь круга с радиусом 5,4 см.

Исходным данным в этой задаче является радиус: r = 5,4 см. Результатом работы программы должны быть величины c и s: c — длина окружности и s — площадь круга. c, s и r — величины вещественного типа.

Исходное данное и результаты связаны соотношениями, известными из курса математики: $c = 2\pi r$, $s = \pi r^2$. Программа, реализующая вычисления по этим формулам, будет иметь вид:

```
# Программа 1
r = 5.4
c = 2 * 3.14 * r
s = 3.14 * r ** 2
print ('c=', c)
print ('s=', s)
```

Эта программа верна и решает поставленную задачу. Запустив её на выполнение, мы получим следующий результат (рис. 1.2).

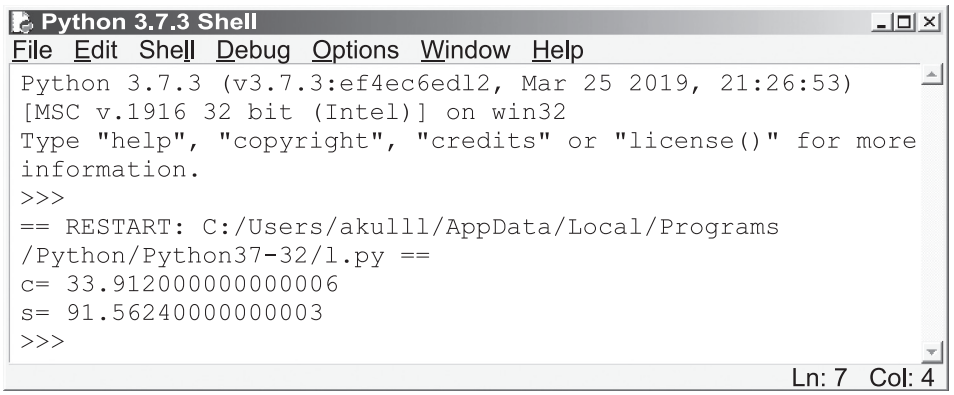

Рис. 1.2. Результат вычисления длины окружности и площади круга

Улучшим внешний вид результата, использовав вывод по формату (рис. 1.3).

```
print("c=", "{:7.4f}".format(c))
print("s=", "{:7.4f}".format(s))
```

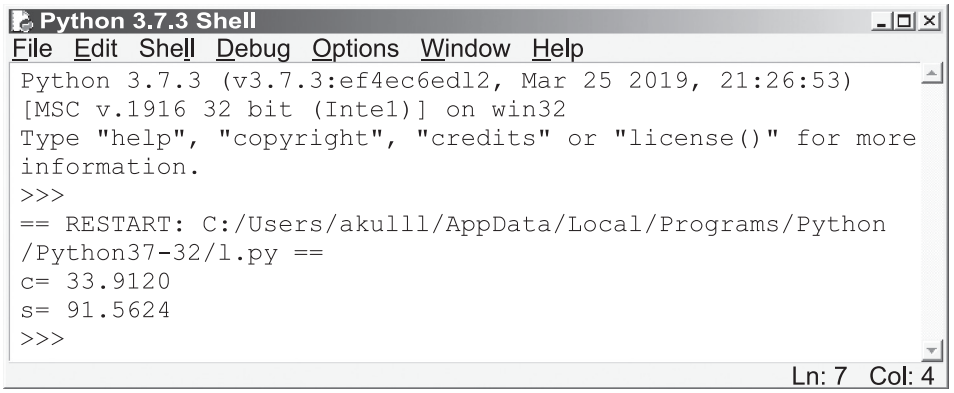

Рис. 1.3. Форматный вывод

И всё-таки составленная нами программа имеет существенный недостаток: она находит длину окружности и площадь круга для единственного значения радиуса (5,4 см).

Для того чтобы вычислить длину окружности и площадь круга для другого значения радиуса, потребуется вносить изменения

непосредственно в текст программы, а именно изменять оператор присваивания. Внесение изменений в существующую программу, по меньшей мере, не всегда удобно (например, когда программа большая и операторов присваивания много). Ниже вы познакомитесь с оператором, позволяющим вводить исходные данные в процессе работы программы, не прибегая к изменению текста программы.

1.2.3. Ввод данных с клавиатуры

Для ввода в оперативную память значений переменных используется **оператор (функция) ввода** `input()` (от англ. *input* — ввод):

```
a = input()
```

Пара скобок говорит о том, что мы вызываем функцию. Их надо писать обязательно, даже если в скобках ничего нет.

При выполнении этой команды программа ожидает от пользователя ввода последовательности символов с клавиатуры; после того, как пользователь нажимает клавишу *Enter*, набранная им символьная строка записывается в переменную с именем `a`. Это значит, что в памяти выделяется область необходимого размера, с ней связывается имя `a`, и в этой области сохраняются все полученные символы.

Если мы планируем работать не со строками, а с числами, то сразу же после считывания необходимо выполнить преобразование типов при помощи соответствующей функции:

- `a = int(a)` — для целых чисел;
- `a = float(a)` — для вещественных чисел.

Считывание строк и преобразование типов рекомендуется объединять:

- `a = int(input())` — для целых чисел;
- `a = float(input())` — для вещественных чисел.

Экспериментально убедитесь в истинности утверждения: «Функции `int()` и `float()` работают без ошибок, если введённая строка состоит только из цифр».

Можно совмещать вывод подсказки и ввод данных, указывая текст подсказки в скобках как аргумент функции `input()`:

```
r = float(input('Введите радиус '))
```

Каждый оператор ввода `input()` захватывает только одну строку данных, причем захватывает её целиком. Для того, чтобы ввести в одной строке два целых числа, разделённых пробелом (например, `10 20`), используют функцию `split()` (от англ. *split* — расщепить). Можно воспользоваться следующей последовательностью команд:

`a, b = input().split()`	Ввод двух строковых величин, разделённых пробелом
`a, b = int(a), int(b)`	Преобразование к целому типу

Теперь рассмотрим ситуацию, когда входные данные заданы в одной строке, но разделены особыми разделителями, отличными от пробела. Типичным примером таких входных данных являются показания времени (10:33).

В таких случаях надо для `split()` указывать конкретный символ разделителя, взятый в двойные или одинарные кавычки. В нашем примере:

```
hours, minutes = input().split(':')
```

Аналогично организуется считывание трёх и более переменных:

```
a, b, c = input().split()
```

Для преобразования к целому типу переменных `a`, `b`, `c` можно использовать конструкцию:

```
a, b, c = int(a), int(b), int(c)
```

Сократить запись считывания нескольких значений и их преобразования в числовой тип можно с помощью функции `map`, которая применяет к каждому элементу списка заданное правило.

```
a, b, c = map(int, input().split())
```

Здесь с помощью функции `map` организовано применение функции `int()` к каждому элементу вводимого списка.

Усовершенствуем программу 1, организовав в ней ввод данных с помощью оператора `input()`, включив строку с приглашением для ввода:

```
# Программа 2
r = float(input('Введите радиус:'))
c = 2 * 3.14 * r
```

```
s = 3.14 * r ** 2
print("c=", "{:7.4f}".format(c))
print("s=", "{:7.4f}".format(s))
```

Результат работы усовершенствованной программы представлен на рис. 1.4.

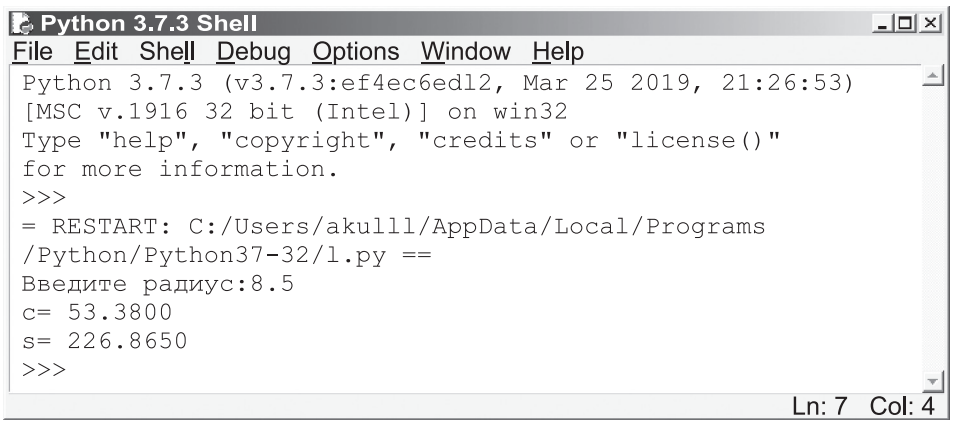

Рис. 1.4. Программа с реализованным вводом данных с клавиатуры

Теперь наша программа может вычислять длину окружности и площадь круга для любого целого значения `r`. Иначе говоря, она решает не единичную задачу, а целый класс задач. Кроме того, в программе понятно и удобно организован ввод исходных данных и вывод получаемых результатов. Это обеспечивает дружественность пользовательского интерфейса.

САМОЕ ГЛАВНОЕ

Оператор ввода (функция) `input()` вводит с клавиатуры символьную строку. Для преобразования строки в целое число её обрабатывают функцией `int()`, в вещественное число — функцией `float()`.

Сократить запись считывания нескольких значений и их преобразования в числовой тип можно с помощью функции `map()`, которая применяет к каждому вводимому элементу заданное правило.

Для вывода данных из оперативной памяти на экран монитора используется оператор вывода (функция) `print()`. Элементы

списка вывода разделяются запятыми. По умолчанию при выводе данные разделяются пробелами; после вывода всех данных функция `print()` переводит курсор в начало следующей строки.

Формат вывода — это указываемое общее количество знакомест, отводимое на число, определяющее, сколько позиций на экране должна занимать выводимая величина. Форматный вывод данных выполняется с помощью функции `format()`.

Ввод исходных данных и вывод результатов должны быть организованы понятно и удобно; это обеспечивает дружественность пользовательского интерфейса.

Вопросы и задания

1. Что является результатом выполнения оператора?

   а) `print(a)`
   б) `print(' a')`
   в) `print('a=', a)`

2. Напишите программу, выводящую на экран следующее забавное изображение:

```
 (\_/)
(='.'=)
(")_(")
```

3. Какой тип имеет переменная `f`, если после выполнения оператора `print(f)` на экран было выведено следующее число?

   а) `125`
   б) `125.0`

4. Дан фрагмент программы:

```
a = 10; b = a + 1; a = b - a; print(a, b)
```

   Какие числа будут выведены на экран компьютера?

5. Для каждого оператора `print()` укажите соответствующий ему результат работы:

а) `print(10, 20, 30)`	**1)** `102030`
б) `print(10, 20, 30, sep='')`	**2)** `10, 20, 30`
в) `print(10, 20, 30, sep=', ')`	**3)** `10:20:30`
г) `print(10, 20, 30, sep=':')`	**4)** `10 20 30`
д) `print(10, 20, 30, sep=',')`	**5)** `10,20,30`

6. Что будет выведено в результате работы следующей программы?

```
a = 1; b = 2; c = 3
print("{:3}".format(a))
print("{:2}{:1}{}".format(b, b, b))
print("{}{}{}{}{}".format(c, c, c, c, c))
print("{:2}{:1}{}".format(b, b, b))
print("{:3}".format(a))
```

7. Внесите изменения в программу из предыдущего задания так, чтобы в результате её выполнения выводились следующие изображения:

а)
```
  1
 2 2
3   3
 2 2
  1
```

б)
```
  1
 212
31313
 212
  1
```

в)
```
  5
 555
55555
 555
  5
```

8. Что будет выведено в результате работы следующей программы?

```
x = 143.511
print(x)
print("{:8.2f}".format(x))
print("{:.6f}".format(x))
print("{:10.3e}".format(x))
print("{:12.3e}".format(x))
```

9. Определите результат работы программы, если переменным a и b были присвоены значения 2 и 4 соответственно.

```
a = int(input())
b = int(input())
a = a * a
b **= 2
k = a * b
k *= 2
k += a + b
print(k)
```

10. Целочисленным переменным i, j, k нужно присвоить соответственно значения 10, 20 и 30. Напишите оператор ввода, соответствующий входной строке:

а) 20 10 30
б) 30 20 10
в) 10 30 20

11. Найдите ошибку в программе, которая должна вывести сумму двух введенных чисел.

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

Проверьте правильность своего решения, выполнив программу на компьютере.

12. С клавиатуры вводятся два целых числа в строку через пробел. Выберете фрагмент программы, в котором переменным a и b будут присвоены соответствующие целочисленные значения:

1)
```
a, b = map(int(input()).split())
```
2)
```
a, b = int(input()).map(split())
```
3)
```
a = int(input())
b = int(input())
```
4)
```
a, b = map(split().int(input()))
```
5)
```
a, b = map(int(input()).int(input()))
```
6)
```
a, b = map(int, input().split())
```
7)
```
a, b = int(map( input().split()))
```
8)
```
a, b = map(int, input(),split())
```
9)
```
a, b = map(int. input().split())
```
10)
```
a, b = map(int.input(),split())
```

13. Напишите оператор, обеспечивающий ввод с клавиатуры необходимых исходных данных для вычисления дискриминанта квадратного уравнения по трём целочисленным значениям его коэффициентов.

14. Дан фрагмент программы:

```
a = input(); b = input(); d = input()
a = float(a)
b = float(b)
d = float(d)
c = a + b; print (a, b, c, end=""); print(d)
```

Упростите его, сократив число операторов.

15. Напишите программу, которая вычисляет площадь и периметр прямоугольника по длинам двух его сторон.