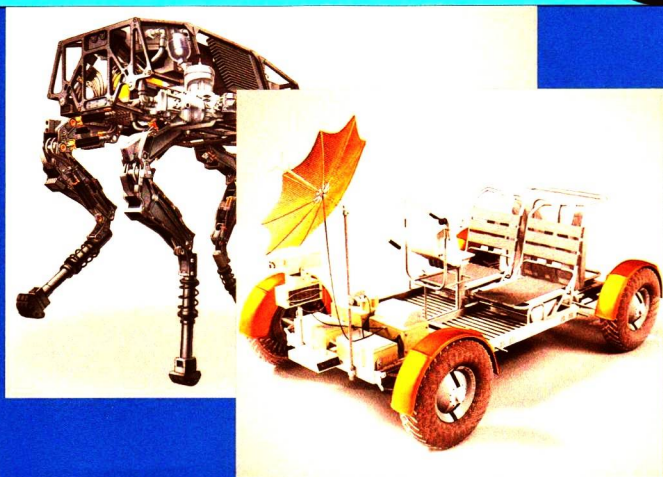




ИНФОРМАТИКА

8-9



Л. Л. Босова
Н. А. Аквилянов
И. О. Кочергин
Ю. Л. Штепа
Т. А. Бурцева

Начала программирования на языке Python

Дополнительные главы
к учебникам



ИЗДАТЕЛЬСТВО

БИНОМ

Л. Л. Босова, Н. А. Аквилянов, И. О. Кочергин,
Ю. Л. Штепа, Т. А. Бурцева

ИНФОРМАТИКА

8–9 классы

Начала программирования на языке Python

Дополнительные главы
к учебникам



Москва
БИНОМ. Лаборатория знаний
2020

УДК 004.9
ББК 32.97
Б85

Б85 Босова, Л. Л. Информатика. 8–9 классы. Начала программирования на языке Python. Дополнительные главы к учебникам / Л. Л. Босова, Н. А. Аквилянов, И. О. Кочергин и др. — М. : БИНОМ. Лаборатория знаний, 2020. — 96 с. : ил. — ISBN 978-5-9963-5091-9

Учебное издание входит в состав УМК по информатике для основной школы авторского коллектива под руководством Л. Л. Босовой, дополняя содержание учебников 8 и 9 классов материалами по программированию на языке Python. Учебное издание создает условия для выбора обучающимися языка программирования, обеспечивает возможность изучения нескольких языков программирования.

Соответствует федеральному государственному образовательному стандарту основного общего образования и примерной основной образовательной программе основного общего образования.

**УДК 004.9
ББК 32.97**

Учебное издание

**Босова Людмила Леонидовна
Аквилянов Никита Александрович
Кочергин Илья Олегович
Штепа Юлия Леонидовна
Бурцева Татьяна Анатольевна**

**ИНФОРМАТИКА
8–9 классы**

**НАЧАЛА ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ PYTHON
Дополнительные главы к учебникам**

Редактор *М. Полежаева*
Оформление: *Н. Новак*
Технический редактор *Е. Денюкова*
Компьютерная верстка: *Е. Голубова*

Подписано в печать 30.08.19. Формат 70х100, 16. Усл. печ. л. 7,8.
Тираж 2000 экз. Заказ 2429

ООО «БИНОМ. Лаборатория знаний»
127473, Москва, ул. Краснопролетарская, д. 16, стр. 3,
тел. (495)181-53-44, e-mail: binom@Lbz.ru, http://Lbz.ru

Приобрести книги издательства «БИНОМ. Лаборатория знаний»
можно в магазине по адресу:

Москва, ул. Краснопролетарская, д. 16, стр. 3,
тел. (495)181-60-77, e-mail: shop@blbz.ru
Время работы: вторник–суббота с 9 до 19 часов

Заявки на оптовые заказы принимаются
Коммерческим департаментом издательства:
тел. (495)181-53-44, доб. 271, 511, e-mail: sales@blbz.ru

Отпечатано в полном соответствии с качеством
предоставленного электронного оригинал-макета в типографии

ОАО «Альянс «Югполиграфиздат», ВПК «Офсет»
400001 г. Волгоград, ул. КИМ, 6. Тел./факс: (8442) 26-60-10, 97-49-40

© ООО «БИНОМ. Лаборатория знаний», 2020
© Художественное оформление
ООО «БИНОМ. Лаборатория знаний», 2020
Все права защищены

ISBN 978-5-9963-5091-9

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 «пайтон», 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 \geq и \leq);

`!=` (знак \neq);

`"""` 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	<code>int (integer)</code>	Сколько угодно большие, размер ограничен оперативной памятью
Вещественный	<code>float</code>	Любые числа с дробной частью
Строковый	<code>str (string)</code>	Любые последовательности символов из таблицы Unicode
Логический	<code>bool (boolean)</code>	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).

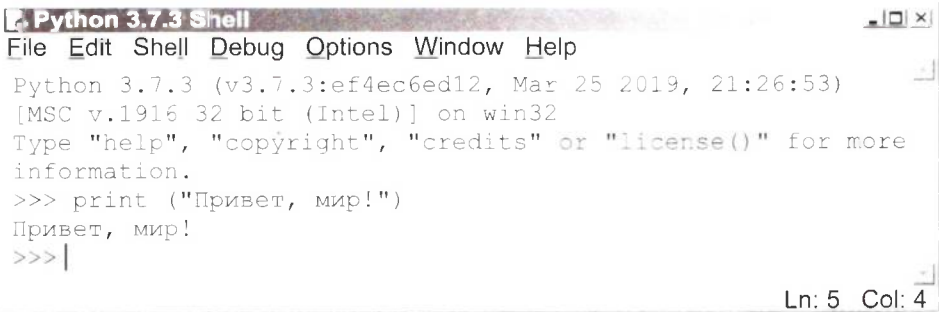


Рис. 1.1. Работа в командном режиме

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*.

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

```
a += b
a -= b
a *= b
a /= b
a **= 2
```

Полная запись

```
a = a + b
a = a - b
a = a * b
a = a / b
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 | 5) break |
| 2) float | 6) while |
| 3) input | 7) else |
| 4) print | 8) string |

4. Каких правил следует придерживаться при выборе имён для различных объектов в языке Python?
5. Отнесите каждую из следующих последовательностей символов в одну 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) lz | е) ELSE | л) n3 |
| б) _lz | ж) sUMMA | м) 3n |
| в) ¹⁾ | з) Summa | н) n 3 |
| г) фy | и) дата | о) n+3 |
| д) z-1 | к) lфy | 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 | б) a = b = 5 | в) a = b = 1 |
| a += 1 | a += b | a *= 10 |
| c = a + b | c = 2 * a - b | c = a / (2 * b) |

- г) $a, b = 3, 5$ д) $a, b = 5, 3$ е) $b, a = 5, 2$
 $b += 2$ $b += a$ $b **= a$
 $c = a + b$ $c = 10 * b / a$ $c = b / a * 4$

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 `s` 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	<code>print(1, 20, 300)</code>	1 20 300
Убрать разделители-пробелы	<code>print(1, 20, 300, sep='')</code>	120300
Добавить разделитель-запятую	<code>print(1, 20, 300, sep=',')</code>	1, 20, 300
Вывод каждого значения с новой строки	<code>print(1, 20, 300, sep='\n')</code>	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
<pre>a = 4 print ("a=", "{:5d} {:5d}".format(a, a * a))</pre>	a= 4 16
<pre>a = 1 / 3; b = 1 / 9 print ("{:7.3f} {:7.4f}".format(a, b))</pre>	0.333 0.1111
<pre>a = 1 / 3 print ("{:10.3e}".format(a))</pre>	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

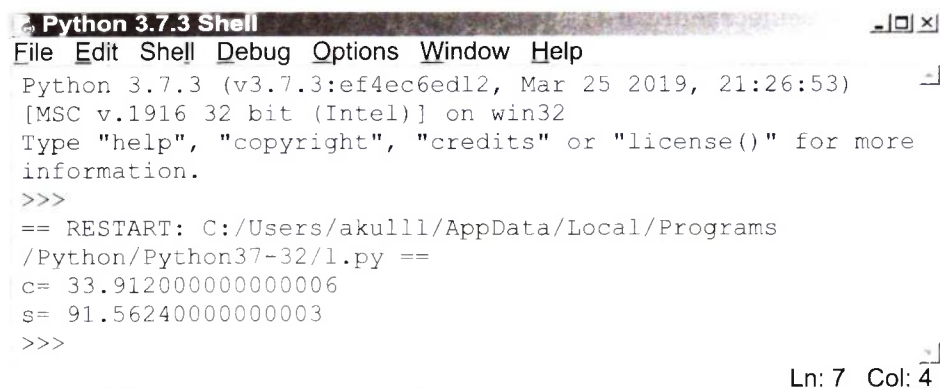
Пользуясь рассмотренными операторами, составим программу, вычисляющую длину окружности и площадь круга с радиусом 5,4 см.

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).



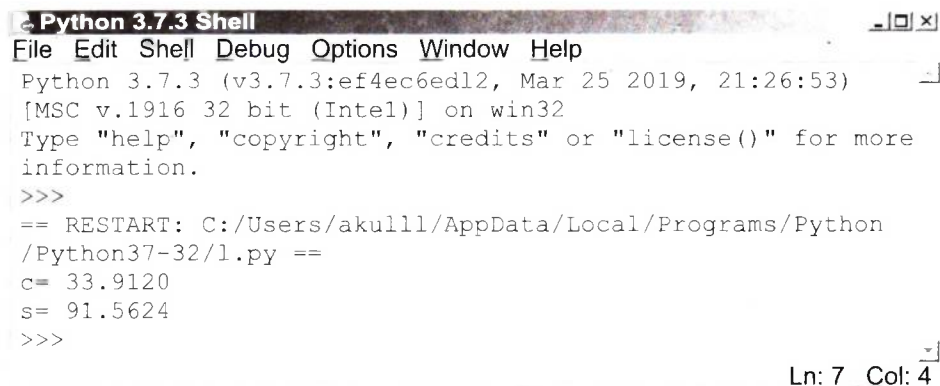
```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53)
[MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more
information.
>>>
== RESTART: C:/Users/akull1/AppData/Local/Programs
/Python/Python37-32/1.py ==
c= 33.912000000000006
s= 91.562400000000003
>>>
```

Ln: 7 Col: 4

Рис. 1.2. Результат вычисления длины окружности и площади круга

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

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



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53)
[MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more
information.
>>>
== RESTART: C:/Users/akull1/AppData/Local/Programs/Python
/Python37-32/1.py ==
c= 33.9120
s= 91.5624
>>>
```

Ln: 7 Col: 4

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:

<code>a, b = input().split()</code>	Ввод двух строковых величин, разделённых пробелом
<code>a, b = int(a), int(b)</code>	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())
```

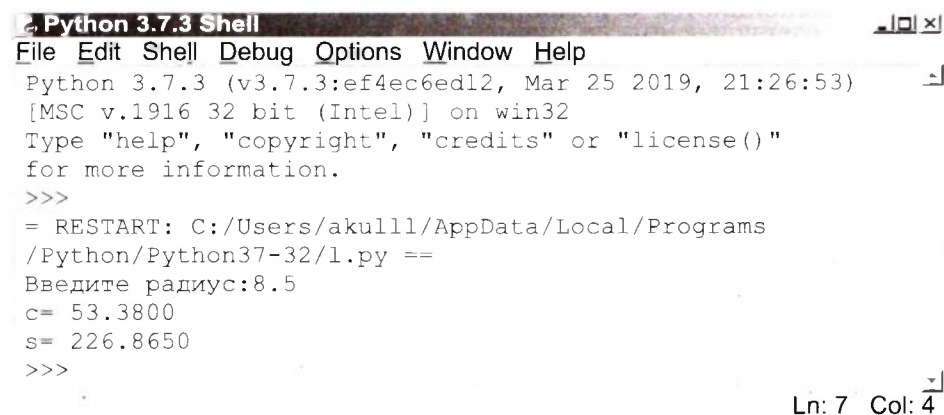
Здесь с помощью функции `map` организовано применение функции `int()` к каждому элементу вводимого списка.

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.



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53)
[MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()"
for more information.
>>>
= RESTART: C:/Users/akull11/AppData/Local/Programs
/Python/Python37-32/1.py ==
Введите радиус:8.5
c= 53.3800
s= 226.8650
>>>
Ln: 7 Col: 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 дружелюбность 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 дружелюбность 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) <code>print(10, 20, 30)</code> | 1) 102030 |
| б) <code>print(10, 20, 30, sep='')</code> | 2) 10, 20, 30 |
| в) <code>print(10, 20, 30, sep=',')</code> | 3) 10:20:30 |
| г) <code>print(10, 20, 30, sep=':')</code> | 4) 10 20 30 |
| д) <code>print(10, 20, 30, sep=',')</code> | 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	б)	1	в)	5
	2 2		212		555
	3 3		31313		55555
	2 2		212		555
	1		1		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. Выберите 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.

Проверочная работа № 2

1. Укажите оператор, используемый для вывода данных в Python.
 - 1) write
 - 2) int
 - 3) float
 - 4) print
 - 5) input
 - 6) read

2. Расположите строки так, чтобы получилась программа, рассчитывающая по двум введенным с клавиатуры вещественным значениям катетов квадрат гипотенузы прямоугольного треугольника. В ответе запишите правильную последовательность номеров.
 - 1) $C = A * A + B * B$
 - 2) `print('Квадрат гипотенузы ', C)`
 - 3) `A = float(input())`
 - 4) `print('Введите длины катетов')`
 - 5) `B = float(input())`

§ 1.3

Программирование линейных алгоритмов

Ключевые слова:

- вещественный тип данных
- целочисленный тип данных
- строковый тип данных
- логический тип данных

Программы, реализующие линейные алгоритмы, считаются наиболее простыми. Все имеющиеся в них операторы выполняются последовательно, один за другим.

Программируя линейные алгоритмы, рассмотрим более подробно целочисленные, логические, символьные и строковые типы данных.

1.3.1. Числовые типы данных

Вы уже знакомы с числовыми типами данных `int` и `float`. К ним применимы многочисленные функции, некоторая их часть приведена в табл. 1.3.

Таблица 1.3

Функция	Назначение	Тип аргумента	Тип результата
<code>abs(x)</code>	Абсолютная величина (модуль) числа <code>x</code>	<code>int, float</code>	Такой же, как у аргумента
<code>round(x)</code>	Округление вещественного <code>x</code> до заданного количества знаков после запятой (по умолчанию — до нуля знаков, т. е. до ближайшего целого)	<code>float</code>	<code>int, float</code>
<code>int(x)</code>	Преобразование вещественного или строкового <code>x</code> к целому	<code>str, float</code>	<code>int</code>
<code>sqrt(x)</code>	Квадратный корень из <code>x</code>	<code>int, float</code>	<code>float</code>
<code>sin(x)</code>	Синус угла <code>x</code> , заданного в радианах	<code>int, float</code>	<code>float</code>
<code>random()</code>	Случайное число от 0 до 1	-	<code>float</code>
<code>randint(a, b)</code>	Случайное целое число <code>n</code> , $a \leq n \leq b$	<code>int</code>	<code>int</code>

Первые три из представленных в таблице 3 функций встроены в язык Python; чтобы их вызвать, не надо выполнять никаких дополнительных действий.

Например, программа ввода вещественного числа и вывода его абсолютной величины может выглядеть так:

```
x = float(input())
print(abs(x))
```

Что касается функций `sqrt(x)` и `sin(x)`, то для их вызова предварительно надо подключить модуль `math`, в котором собраны математические функции; две последние из приведённых в табл. 1.3 функций требуют подключения модуля `random`.

На языке Python написано так много самых разных функций, что встраивать весь этот объем кода в сам язык нецелесообразно. Проблема доступа к дополнительным возможностям языка, обеспечиваемым этими функциями, решается с помощью модулей. Каждый модуль содержит набор функций, предназначенных для решения задач из определенной области. Так модуль `graph` нужен для рисования геометрических фигур, модуль `random` позволяет генерировать случайные числа и т. д. Для доступа к функциям модуля его надо импортировать в программу. После импорта интерпретатор будет «знать» о существовании дополнительных функций и позволит ими пользоваться.



Подключение модуля осуществляется командой `import`. Например, команда `from math import *` подключает к программе все функции (так как стоит знак `*`) модуля `math`. После этого к ним можно будет обращаться так же, как к встроенным функциям:

```
y = sqrt(x)
z = sin(x)
```

Для того, чтобы записать в переменную `a` случайное число в диапазоне от 1 до 10, можно использовать следующие операторы:

```
from random import randint
a = randint(1, 10)
```

Подключаем функцию `randint()` модуля `random`

Обращаемся к функции `randint()` как к встроенной

Исследуем работу функций `round()` и `int()`, применив их к некоторому вещественному `x`. Соответствующая программа будет иметь вид:

```
# Программа 3
print('Исследование функций round, int ')
x = float(input('Введите x>>'))
print('Округление - ', round(x))
print('Целая часть - ', int(x))
```

Запустите программу несколько раз для каждого `x = {10,2; 10,8; -10,2; -10,8}`. Что вы можете сказать о типе результата каждой из этих функций?



Попытайтесь пояснить следующие результаты работы функции `round()`.

Входные данные	Результат
<code>round(1.5)</code>	2
<code>round(2.5)</code>	2
<code>round(2.65, 1)</code>	2.6
<code>round(2.75, 1)</code>	2.8

1.3.2. Целочисленный тип данных

Над целыми числами в языке Python можно выполнять следующие операции:

- сложение (+);
- вычитание (-);
- умножение (*);
- целочисленное деление или получение неполного частного (//);
- взятие остатка или получение остатка от деления (%);
- деление (/);
- возведение в степень (**).

Результаты первых пяти операций — целые числа. Результатом операции деления может быть вещественное число.

Рассмотрим пример использования операций // и %, записав на языке Python программу нахождения суммы цифр вводимого с клавиатуры натурального трёхзначного числа.

Используем тот факт, что положительное трёхзначное число можно представить в виде следующей суммы:

$x = a \cdot 100 + b \cdot 10 + c$, где a , b , c — цифры числа.

```
# Программа 4
print('Нахождение суммы цифр трёхзначного числа')
x = int(input('Введите исходное число>>'))
a = x // 100
b = x % 100 // 10
c = x % 10
s = a + b + c
print('s= ', s)
```



Чему равна сумма цифр числа 123? А числа -123? Совпадают ли ваши результаты с результатами работы программы? Как можно объяснить и исправить ошибку в программе?

1.3.3. Строковый тип данных

Значением строковой величины (тип `str`) является произвольная последовательность символов, заключенная в одинарные или двойные кавычки. Символьная строка рассматривается как единый объект.

Символом в языке Python является любой из символов, который можно получить на экране нажатием на клавиатуре одной из клавиш или комбинации клавиш, а также некоторых других символов, в том числе и невидимых.

В тексте программы переменную строкового типа можно задать, заключив цепочку символов в одинарные или двойные кавычки:

```
d = '5'
c = 'Book'
c1 = "1*"
```

Новое значение может быть записано в строку с помощью оператора ввода с клавиатуры:

```
s = input()
```

Если значение символьной переменной считывается с клавиатуры, то его следует набирать без апострофов.

Встроенная функция `len` определяет длину строки — количество символов в ней:

```
n = len(s)
```

Можно проверить равенство (совпадение) строк (`d == c`) или выяснить, какая из двух строк меньше (при этом используется поочерёдное сравнение кодов символов, образующих слова; меньшим будет то слово, у которого код очередного символа окажется меньше).

Чтобы найти код символа, используют функцию `ord()`, где в качестве параметра указывают символ.

Чтобы по коду узнать символ, используют функцию `chr()`, где в качестве параметра указывают код символа.

В Python (как и в алгоритмическом языке) строки можно сцеплять: `a + b` (к концу строки `a` прикрепляется («приписывается») строка `b`). Пусть

```
d = c1 + d + c1
```

Тогда в переменную `d` будет записана следующая строка: `'1*51*'`.

В результате операции $a * k$, где k — целое число, строка a повторяется k раз. Так, в результате выполнения команды

```
print (d*3)
```

будет выведена строка:

```
1*51*1*51*1*51*
```

Пример. Напишем на языке Python программу, которая запрашивает имя и выводит приветствие.

```
# Программа 5
print('Как тебя зовут?')
name = input()
print('Привет, ', name)
```

Пример. Напишем на языке Python программу, в которой для введённого с клавиатуры символа на экран выводится его код. Затем на экран выводится строка, представляющая собой последовательность из трёх символов используемой кодовой таблицы: символа, предшествующего исходному; исходного символа; символа, следующего за исходным.

```
# Программа 6
a = input()
kod = ord(a)
print(kod)
b = chr(kod - 1) + a + chr(kod + 1)
print(b)
```

1.3.4. Логический тип данных

Как известно, величины логического типа принимают всего два значения; в Python это **False** и **True**. Эти константы определены так, что **False < True**.

Логические значения получаются в результате выполнения операций сравнения числовых, строковых и логических выражений. Поэтому в Python логической переменной можно присваивать результат операции сравнения.

Пример. Напишем программу, определяющую истинность высказывания «Число N является чётным» для произвольного целого числа N .

Пусть `ans` — логическая переменная, а `N` — переменная целого типа. Тогда в результате выполнения оператора присваивания `ans = N % 2 == 0` переменной `ans` будет присвоено значение **True** при любом чётном `N` и **False** в противном случае.

```
# Программа 7
print('Определение истинности высказывания о чётности
числа')
N = int(input('Введите исходное число>>'))
ans = N % 2 == 0
print('Число', N, 'является чётным -', ans)
```

Логическим переменным также можно присваивать значения логических выражений, построенных с помощью известных вам логических операций **И**, **ИЛИ**, **НЕ**, которые в Python обозначаются соответственно **and**, **or**, **not**.

Пример. Напишем программу, определяющую истинность высказывания «Треугольник с длинами сторон *a*, *b*, *c* является равнобедренным» для произвольных целых чисел *a*, *b*, *c*.

```
# Программа 8
print('Определение истинности высказывания
о равнобедренном треугольнике')
a = int(input('Введите значение a>>'))
b = int(input('Введите значение b>>'))
c = int(input('Введите значение c>>'))
ans = (a == b) or (a == c) or (b == c)
print('Треугольник с длинами сторон', a, ', ', b, ', ', c,
', ' является равнобедренным -', ans)
```

САМОЕ ГЛАВНОЕ

В языке Python используются вещественный, целочисленный, строковый, логический и другие типы данных. Для них определены соответствующие операции и функции.

На языке написано большое количество самых разных функций, для использования многих из них необходимо подключать специальные модули.

В языке Python реализованы операции целочисленного деления: для деления нацело используется оператор `//`, для взятия остатка от деления — оператор `%`.

Символьная строка — это последовательность символов, рассматриваемая как единый объект. Длина строки — это количест-

во символов в строке. Знак + при работе со строками означает их сцепление в одну строку; знак * — многократное сложение строк.

Логическим переменным можно присваивать значения логических выражений, построенных с помощью логических операций `and`, `or`, `not`.

Вопросы и задания

1. Определите значения переменных после выполнения фрагмента программы. Составьте таблицу значений переменных.

<p>а) <code>x = 11</code> <code>y = 5</code> <code>z = y</code> <code>y = x % y</code> <code>x = z</code> <code>y = (y + 2) * z</code></p>	<p>б) <code>x = 13</code> <code>y = 3</code> <code>z = x</code> <code>z = z // y</code> <code>y = x</code></p>
---	--

2. Определите значение переменной `c` после выполнения программы:

<p>а) <code>a = 9</code> <code>b = a % 5</code> <code>b = b * 10</code> <code>a = b // 5 - 3</code> <code>b = b // 5 * 2</code> <code>c = a + b</code></p>	<p>б) <code>a = 123</code> <code>b = a // 10</code> <code>b = b / 4 + 2</code> <code>b = b * 25 + 2</code> <code>a = a + b</code> <code>a = a % 5 * 3</code> <code>c = a + b</code></p>
---	---

в) `a = 951`
`b = a // 100 + a % 100`
`a = a // 10`
`a %= 10`
`a += b`
`b = (a + b) % 10 / 2`
`a -= b - 3`
`c = a + b`

3. Дана программа:

```
a, b = map(int, input().split())
c = (a + b + abs(a - b)) // 2
print(c)
```

Определите результат работы программы для следующих входных данных:

- | | |
|--------|-----------|
| а) 4 8 | г) 2 -10 |
| б) 9 3 | д) -3 -9 |
| в) 7 7 | е) -18 -8 |

Подумайте, какую задачу решает эта программа.

4. Установите соответствие между обозначениями функций и их назначением.

- | | |
|-------------------------------|---|
| а) <code>abs(x)</code> | 1) Извлечение квадратного корня из x |
| б) <code>sqrt(x)</code> | 2) Вычисление модуля x |
| в) <code>round(x)</code> | 3) Получение случайного целого числа от 0 до x |
| г) <code>randint(0, x)</code> | 4) Округление x до указанного количества знаков после запятой |

5. Напишите и отладьте программу, которая вычисляет:

- а) дискриминант квадратного уравнения;
- б) площадь кольца, если его толщина t см, а диаметр внутреннего круга — d см.

6. Если сумма налога исчисляется в рублях и копейках, то налоговая служба округляет её до ближайшего рубля (до 50 копеек — с недостатком, свыше 50 копеек (включая 50) — с избытком). Напишите программу для ввода точной суммы налога и вывода суммы, которую следует уплатить.

7. В модуле `random` есть функция `randint(a, b)`, генерирующая случайное целое число из отрезка $[a; b]$. Соответствующая программа имеет вид:

```
print('Функция randint')
from random import randint
a = int(input ('Введите a>>'))
b = int(input ('Введите b>>'))
print('случайное целое число из отрезка [' , a , ' ; ' ,
b , ']=' , randint(a, b))
```

Какие изменения следует внести в программу, чтобы получить случайное целое число из полуинтервала $(a; b]$?

Как можно получить случайное целое число из интервала $(a; b)$?

8. Напишите и отладьте программу для:
- получения случайного целого числа x из полуинтервала $[0; 15)$;
 - получения случайного вещественного числа x из отрезка $[0; 15]$;
 - получения случайного целого числа x из полуинтервала $[-15; 15)$ без использования функции `randint()`;
 - получения случайного вещественного числа x из отрезка $[10; 15]$ с помощью функции `random()`.
9. Компания выпустила лотерейные билеты трёх типов: для молодежи, для взрослых и для пенсионеров. Номера билетов каждого типа лежат в пределах:
- для молодёжи — от 1 до 100;
 - для взрослых — от 101 до 200;
 - для пенсионеров — от 201 до 250.
- Напишите программу для выбора лотерейного билета каждого типа случайным образом.
10. Напишите на языке Python программу, которая для произвольного натурального двузначного числа определяет:
- сумму и произведение его цифр;
 - число, образованное перестановкой цифр исходного числа.
11. Напишите на языке Python программу, реализующую алгоритм работы кассира, выдающего покупателю сдачу (s) наименьшим возможным количеством банкнот по 2000 ($k2000$), 1000 ($k1000$), 500 ($k500$), 100 ($k100$), 50 ($k50$) и 10 ($k10$) рублей. Предусмотрите вывод сообщения о том, что часть сдачи, которую невозможно выдать купюрами, будет выдана монетами.

Пример входных данных	Пример выходных данных
845	Следует сдать: банкнот по 500 руб. — 1 шт. банкнот по 100 руб. — 3 шт. банкнот по 50 руб. — 0 шт. банкнот по 10 руб. — 4 шт. монетами — 5 руб.

12. Идёт k -я секунда суток. Разработайте программу, которая по введённой k -й секунде суток определяет, сколько целых часов h и целых минут m прошло с начала суток. Напри-

мер, если $k = 13\ 257 = 3 \cdot 3600 + 40 \cdot 60 + 57$, то $h = 3$ и $m = 40$. Выведите на экран фразу: «It is ... hours ... minutes». Вместо многоточий программа должна выводить значения h и m , отделяя их от слов ровно одним пробелом.

Пример входных данных	Пример выходных данных
13257	It is 3 hours 40 minutes.

13. Определите результат работы программы. Запишите математическую формулу для вычисления значения s .

```
from math import *
a = 12
b = 5
c = 13
p = (a + b + c) / 2
s = p
s *= p - a
s *= p - b
s *= p - c
s = sqrt(s)
print(s)
```

14. Для заданного x вычислите y по формуле

$$y = x^3 + 2,5x^2 - x + 1.$$

При этом:

- а) операцию возведения в степень использовать запрещено;
- б) в одном операторе присваивания можно использовать не более одной арифметической операции (сложение, умножение, вычитание);
- в) в программе может быть использовано не более пяти операторов присваивания.

Подсказка: преобразуйте выражение к следующему виду:

$$y = ((x + 2,5)x - 1)x + 1.$$

15. По заданным координатам точек A и B вычислите длину отрезка AB .

Подсказка: Расстояние d между точками $A(x_a, y_a)$ и $B(x_b, y_b)$

выражается формулой: $d = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2}$.

Пример входных данных	Пример выходных данных
$x_a = 2$ $y_a = 1$ $x_b = 10$ $y_b = 7$	$ AB = 10.0$

16. Известны длины сторон треугольника a , b , c . Напишите программу, вычисляющую площадь этого треугольника.

Пример входных данных	Пример выходных данных
$a = 3$ $b = 4$ $c = 5$	$S = 6.0$

17. Известны координаты вершин A , B , C треугольника. Напишите программу, вычисляющую площадь этого треугольника.

Пример входных данных	Пример выходных данных
$x_a = 2$ $y_a = 1$ $x_b = 6$ $y_b = 5$ $x_c = 10$ $y_c = 1$	$S = 16.0$

18. Напишите и отладьте программу, которая вводит строку с клавиатуры и выводит на экран её длину.
19. Напишите и отладьте программу, которая запрашивает три строковые величины — взаимосвязанные прилагательное, существительное и глагол, а затем выводит все варианты фраз с использованием введённых слов.

Пример входных данных	Пример выходных данных
ЗЕЛЁНЫЕ ЛИСТЬЯ РАСПУСКАЮТСЯ	ЗЕЛЁНЫЕ ЛИСТЬЯ РАСПУСКАЮТСЯ ЗЕЛЁНЫЕ РАСПУСКАЮТСЯ ЛИСТЬЯ ЛИСТЬЯ ЗЕЛЁНЫЕ РАСПУСКАЮТСЯ ЛИСТЬЯ РАСПУСКАЮТСЯ ЗЕЛЁНЫЕ РАСПУСКАЮТСЯ ЗЕЛЁНЫЕ ЛИСТЬЯ РАСПУСКАЮТСЯ ЛИСТЬЯ ЗЕЛЁНЫЕ

20. Даны значения целочисленных переменных: $a = 10$, $b = 20$. Чему будет равно значение логической переменной `rez` после выполнения операции присваивания?
- а) `rez = (a == 10) or (b > 10)`
 - б) `rez = (a > 5) and (b > 5) and (a < 20) and (b < 30)`
 - в) `rez = (not (a < 15)) or (b > 20)`
21. Составьте программу, вводящую **True**, если высказывание является истинным, и **False** в противном случае:
- а) сумма цифр трёхзначного числа x является чётным числом;
 - б) треугольник с длинами сторон a , b , c является разносторонним

Проверочная работа № 3

1. Укажите истинные высказывания.

- а) `3 > 2 and 5 > 6 == True`
- б) `'a' < 'b' or 1 > 0 == True`
- в) `60 > 20 == True`
- г) `61 // 5 * 3 == 410 // 3 == 3`
- д) `2 // 3 == 1`

2. Напишите программу вычисления площади прямоугольного треугольника, значения катетов которого A и B вводятся с клавиатуры.

Пример входных данных	Пример выходных данных
4 6	Для значений катетов 4 и 6 площадь прямоугольного треугольника равна 12.

3. Дана программа:

```
print('Введите две стороны четырёхугольника:')
a = int(input())
b = int(input())
print('Введите диагонали четырёхугольника:')
d1 = int(input())
d2 = int(input())
ans = a != b and d1 == d2
print('Этот четырёхугольник является
прямоугольником - ', ans)
```

При каких исходных данных высказывание «Этот четырёхугольник является прямоугольником» будет:

- а) истинным;
- б) ложным?

§ 1.4

Программирование

разветвляющихся алгоритмов

Ключевые слова:

- условный оператор
- неполный условный оператор
- составной оператор
- каскадное ветвление

1.4.1. Условный оператор

При записи на языке Python разветвляющихся алгоритмов используют **условный оператор**. Его общий вид:

```
if <условие>:  
    <блок_операторов_1>  
else:  
    <блок_операторов_2>
```

Слова **if** и **else** начинаются на одном уровне, а все команды внутренних блоков сдвинуты относительно этого уровня вправо на одно и то же расстояние. Начало и конец блока, который выполняется при истинности (ложности) условия, определяется именно этими сдвигами.

Обратите внимание! В Python сдвиги (отступы) операторов относительно левой границы влияют на работу программы. Для отступа можно использовать пробелы (обычно не меньше двух) или символы табуляции (вставляются при нажатии на клавишу *Tab*).

Если при истинности (ложности) какого-то условия предполагается выполнить по одному действию, то условный оператор может быть записан в одну строку:

```
if <условие>: <оператор_1> else:<оператор_2>
```

Для записи неполных ветвлений используется неполная форма условного оператора:

```
if <условие>:  
    <оператор>
```

В качестве условий используются логические выражения:

- простые — записанные с помощью операций отношения (<, >, >=, <=, != (не равно), == (равно));
- составные — записанные с помощью логических операций (and, or, not).

В языке Python разрешены двойные неравенства, например $A < B < C$.

Пример. Напишем на языке Python рассмотренный в п. 2.4.2 (пример 8) учебника для 8 класса алгоритм определения принадлежности точки x отрезку $[a, b]$.

```
# Программа 9  
print('Определение принадлежности точки отрезку')  
a = int(input('Введите a: '))  
b = int(input('Введите b: '))  
x = int(input('Введите x: '))  
if x >= a and x <= b:  
    print('Точка принадлежит отрезку')  
else:  
    print('Точка не принадлежит отрезку')
```

Пример. Воспользуемся неполным условным оператором для записи на языке Python рассмотренного в п. 2.4.2 (пример 9) учебника для 8 класса алгоритма присваивания переменной y значения наибольшей из трёх величин a , b и c .

```
# Программа 10  
print('Нахождение наибольшей из трёх величин')  
a = int(input('Введите a: '))  
b = int(input('Введите b: '))  
c = int(input('Введите c: '))  
y = a  
if b > y:  
    y = b  
if c > y:  
    y = c  
print('y=', y)
```

Дополните эту программу так, чтобы её выполнение приводило к нахождению минимального значения из четырёх величин a , b , c и d .



Пример. Уравнение вида $ax^2 + bx + c = 0$, где x — переменная, a , b и c — некоторые числа, причём $a \neq 0$, называется квадратным уравнением. Алгоритм решения квадратного уравнения вам хорошо известен. Запишем соответствующую программу на языке Python:

```
# Программа 11
from math import *
print('Решение квадратного уравнения')
print('Введите коэффициенты a, b, c>>')
a = float(input('a='))
b = float(input('b='))
c = float(input('c='))
d = b * b - 4 * a * c
if d < 0:
    print ('Корней нет')
if d == 0:
    x = - b / (2 * a)
    print('Корень уравнения x=', "{:6.4f}".format(x))
if d > 0:
    x1 = (-b + sqrt(d)) / (2 * a)
    x2 = (-b - sqrt(d)) / (2 * a)
    print('Корни уравнения:')
    print('x1=', "{:6.4f}".format(x1))
    print('x2=', "{:6.4f}".format(x2))
```

1.4.2. Многообразие способов записи ветвлений

Внутри условного оператора могут находиться любые операторы, в том числе и другие условные операторы. Условные операторы, находящиеся внутри блоков, следующих после **if** («если») или **else** («иначе»), называются **вложенными условными операторами**:

```
if <условие_1>:
    <блок_операторов_1>
else:
    if <условие_2>:
        <блок_операторов_2>
    else:
        <блок_операторов_3>
```

Если после **else** сразу следует ещё один оператор **if**, можно использовать так называемое «каскадное» ветвление с ключевыми словами **elif** (сокращение от **else-if**). Если очередное условие ложно, то выполняется проверка следующего условия и т. д.

Пример. Воспользуемся каскадным ветвлением для записи на языке Python рассмотренного в п. 2.4.2 (пример 10) учебника для 8 класса алгоритма решения линейного уравнения.

```
# Программа 12
print('Решение линейного уравнения')
a = float(input('Введите коэффициент a>>'))
b = float(input('Введите коэффициент b>>'))
if a != 0:
    x = -b / a
    print('Корень уравнения x=', x)
elif b != 0:
    print('Корней нет')
else:
    print('x - любое число')
```

Как правило, для решения одной и той же задачи можно предложить несколько алгоритмов. Убедимся в этом, написав программу для решения линейного уравнения, используя неполное ветвление:

```
# Программа 13
print('Решение линейного уравнения')
a = float(input('Введите коэффициент a>>'))
b = float(input('Введите коэффициент b>>'))
if a != 0:
    x = - b / a
    print ('Корень уравнения x =', x)
if a == 0 and b != 0:
    print ('Корней нет')
if a == 0 and b == 0:
    print ('x - любое число')
```

Возможно, второй вариант программы покажется вам более наглядным. Но и у первого варианта есть свои преимущества: в нём делается меньше проверок.

Воспользуйтесь каскадным ветвлением для записи алгоритма решения квадратного уравнения на языке Python.



САМОЕ ГЛАВНОЕ

При записи разветвляющихся алгоритмов на языке Python используют условный оператор, позволяющий выбрать один из двух вариантов действий в зависимости от выполнения некоторого условия:

```
if <условие>:  
    <блок_операторов_1>  
else:  
    <блок_операторов_2>
```

Условие, которое нужно проверить, записывается после слова **if**. Если условие верно, выполняются все команды, записанные после строки с оператором **if** (со сдвигом вправо). Если условие неверно, выполняются все команды, записанные после строки с оператором **else** (со сдвигом вправо).

Для записи неполных ветвлений используется неполный условный оператор:

```
if <условие>:  
    <операторы>
```

В обеих частях условного оператора можно использовать любые операторы, в том числе и другие (вложенные) условные операторы.

Для выбора из нескольких вариантов используют следующую конструкцию условного оператора:

```
if <условие>:  
    <операторы>  
elif <условие>:  
    <операторы>  
else:  
    <операторы>
```

Вопросы и задания

1. Отметьте условия, записанные на языке Python правильно.

- 1) $a \geq 0$
- 2) $x \leq 3$
- 3) $x > 0$ **or** $y < 0$
- 4) $c \neq 0$
- 5) $a \neq b$

- 6) $-5 < a < 10$
- 7) $x > 0, y < 0$
- 8) $a > 10$ и $b < 5$
- 9) $x == 6$
- 10) $x = y = z$
- 11) $x <> 0$

2. Запишите на языке Python следующие условия:

- 1) $y \neq 0$;
- 2) x не кратно 7;
- 3) $-5 < x < 10$;
- 4) $x \in [-1; 1]$.

3. Как на языке Python записывается полное и неполное ветвление?

4. Является ли условным оператором следующая последовательность символов?

- а) `if x < y: x = 0 else input(y)`
- б) `if x >= y: x = 0; y := 0`
`else: print(z)`
- в) `if x < y < z: a = a + 1`

5. Дана программа на языке Python:

```
print('Введите три числа: ')
a, b, c = map(float, input().split())
x = a
if b < x:
    x = b
if c < x:
    x = c
print('x=', x)
```

Что будет выведено в результате работы программы при следующих входных данных?

- а) 10 5 1
- б) 10 5 7
- в) 2 10 5

Постройте блок-схему, соответствующую программе.

6. Дана программа на языке Python:

```
print('Введите три числа: ')
x, y, z = map(int, input().split())
if x <= y <= z:
    x *= 2
    y *= 2
    z *= 2
else:
    x -= 2
    y -= 2
    z -= 2
print(x, y, z)
```

Приведите пример входных данных, при котором исходные значения:

- увеличиваются в 2 раза;
 - уменьшаются на 2.
7. Запишите следующий фрагмент программы, используя два условных оператора:

```
if a > b: c = 1
if a > b: d = 2
if a <= b: c = 3
if a <= b: d = 4
```

8. Найдите ошибки в операторах на языке Python.

<p>а) <code>if 1 < x, x < 2:</code> <code> x = x + 1;</code> <code> y := 0</code></p>	<p>б) <code>if 1 < x and x < 2</code> <code> x += 1</code> <code> y = 0</code> <code> else: x = 0, y = y + 1</code></p>
---	---

Предложите правильный вариант записи ветвлений и составьте соответствующие им блок-схемы.

9. Дано трёхзначное число. Напишите программу, которая определяет:

- есть ли среди цифр заданного целого трёхзначного числа одинаковые;

Пример входных данных	Пример выходных данных
123	Нет
121	Да
222	Да

б) является ли число «перевёртышем» (палиндромом), т. е. числом, десятичная запись которого читается одинаково слева направо и справа налево.

Пример входных данных	Пример выходных данных
122	Нет
121	Перевёртыш
222	Перевёртыш

10. Даны две точки в плоской прямоугольной системе координат. Напишите программу, определяющую, которая из точек находится ближе к началу координат.

Пример входных данных	Пример выходных данных
Координаты 1-й точки >> 1 2 Координаты 2-й точки >> 3 4	1-я точка ближе

11. Даны три натуральных числа. Напишите программу, определяющую, существует ли треугольник с такими длинами сторон. Если такой треугольник существует, то определите его тип (равносторонний, равнобедренный, разносторонний).

Пример входных данных	Пример выходных данных
a b c >> 1 2 1	Не существует
a b c >> 2 2 2	Равносторонний
a b c >> 20 20 30	Равнобедренный
a b c >> 3 4 5	Разносторонний

12. Имеются данные о количестве полных лет четырёх призёров спартакиады. Напишите программу, выбирающую и выводящую возраст самого младшего призёра.

13. Напишите программу, определяющую, лежит ли точка $A(x_a, y_a)$ на прямой $y = kx + m$, над ней или под ней.

Пример входных данных	Пример выходных данных
к, m>>-1 5 ха, уа >>1 2	Точка лежит под прямой.
к, m>>-1 5 ха, уа >>1 10	Точка лежит над прямой.
к, m>>-1 5 ха, уа >>1 4	Точка лежит на прямой.

14. Напишите программу, которая производит обмен значений переменных x и y , если x больше y .

Пример входных данных	Пример выходных данных
x>>5 y>>6	x= 5 y= 6
x>>6 y>>5	x= 5 y= 6

15. Дан фрагмент программы:

```
x, y = map(int, input().split())
z = 0
if x > 0:
    if y > 0:
        z = 1
    else:
        z = 2
```

Составьте блок-схему, соответствующую этому фрагменту программы, и определите значение переменной z при следующих значениях x и y :

- а) 1, 1;
- б) 1, -1;
- в) -1, 1;
- г) -1, -1.

16. Дан условный оператор:

```
if a < 5:
    c = 1
elif a > 5:
    c = 2
else:
    c = 3
```

Какое значение имеет переменная a , если в результате выполнения условного оператора переменной c присваивается значение 3?

17. Напишите программу, вычисляющую значение функции:

$$y = \begin{cases} 1 & \text{при } < 0, \\ 0 & \text{при } = 0, \\ 1 & \text{при } > 1. \end{cases}$$

Пример входных данных	Пример выходных данных
-5	y= -1
0	y= 0
5	y= 1

18. Составьте программу для решения задачи № 21 к § 2.4 учебника для 8 класса (определение дня недели).

19. Поле шахматной доски определяется парой натуральных чисел, каждое из которых не превосходит 8. Напишите программу, которая по введённым координатам двух полей (k, l) и (m, n) определяет, имеют ли эти поля один цвет.

Пример входных данных	Пример выходных данных
Координаты 1-го поля>>2 2 Координаты 2-го поля>>3 3	Поля одного цвета
Координаты 1-го поля>>2 3 Координаты 2-го поля>>3 3	Поля разного цвета
Координаты 1-го поля>>2 7 Координаты 2-го поля>>5 4	Поля одного цвета

20. Напишите программу, в которой пользователю предлагается дополнить до 100 некоторое целое число a (a — случайное целое число из отрезка $[0; 100]$). Ответ пользователя проверяется и комментируется.

21. С помощью программы сравните тройки слов и сделайте выводы о том, как происходит сравнение: KAWAI — Kawai — kawai; инФорматика — информатика — информатикА, 50_ кг — 50_kg — 200_кг; яблоко — яблоки — яблоня.

22. Ниже приведена программа, записанная на двух языках программирования.

Алгоритмический язык	Python
<pre> алг нач цел i, x, y ввод x ввод y если (x>5) или (mod(y, 2) = 0) то вывод "ДА" иначе вывод "НЕТ" все кон </pre>	<pre> x = int(input()) y = int(input()) if x > 5 or y % 2 == 0: print("ДА") else: print("НЕТ") </pre>

Было проведено 10 запусков программы, при которых в качестве значений переменных x и y вводились следующие пары чисел: (1, 7); (13, 6); (1, 1); (4, 12); (-16, -2); (-10, 13); (-17, 17); (10, 9); (1, 5); (-8; -6). Сколько было запусков, при которых программа вывела «ДА»?

Проверочная работа № 4

1. Для каждой записи в левом столбце подберите соответствующее ей составное условие, записанное на языке Python, из правого столбца.

- | | |
|--------------------|-------------------------------|
| а) x не кратно 4 | 1) $x \geq -1$ and $x \leq 5$ |
| б) $x \in [-1; 5]$ | 2) $x \% 4 \neq 0$ |
| в) $5 < x < -2$ | 3) $x \% 4 \neq 0$ |
| | 4) $x > -5$ and $x < -2$ |
| | 5) $x > -5$ or $x < -2$ |
| | 6) $x \geq -1$ or $x \leq 5$ |

2. Какие ошибки допущены в программе?

```

print(Введите число a)
a = int(input())
if a >= 0:
    if a = 0:
        a = 18
    else a += 1
    else: a -= 6
print(a)
input
                    
```

Найдите все ошибки и исправьте их. Опишите на естественном языке, с помощью формулы или блок-схемы алгоритм преобразования исходных данных, используемый в данной программе.

Определите выходные данные при следующих исходных данных:

- а) -10
- б) 0
- в) 10

3. Квадраты при игре в крестики-нолики пронумерованы, как показано на рисунке:

1	2	3
4	5	6
7	8	9

Заданы номера трёх квадратов, N_1, N_2, N_3 , причём $N_1 < N_2 < N_3$. Напишите программу, проверяющую, лежат ли квадраты на одной вертикали.

§ 1.5

Программирование циклических алгоритмов

Ключевые слова:

- цикл **while**
- цикл **for**

1.5.1. Программирование циклов с известным условием продолжения работы

В языке Python цикл с заданным условием продолжения работы записывается с помощью оператора **while** (в переводе с английского языка — «до тех пор, пока»). Общий вид оператора:

```
while <условие>:
    <операторы>
```


Здесь:

<условие> — логическое выражение, условие продолжения работы цикла; пока оно истинно, выполняется тело цикла; если условие ложно с самого начала, тело цикла не выполнится ни разу;

<операторы> — один или несколько операторов, с помощью которых записано тело цикла.

Запишем на языке Python рассмотренный в п. 2.4.3 (пример 14) учебника для 8 класса алгоритм получения частного q и остатка r от деления натурального числа x на натуральное число y без использования операции деления.

```
# Программа 14
print('Частное и остаток')
x = int(input('Введите делимое x>>'))
y = int(input('Введите делитель y>>'))
r = x
q = 0
while r >= y:
    r = r - y
    q += 1
print('Частное q =', q)
print('Остаток r =', r)
```



Выполните программу при $x = 25$ и $y = 4$. Каким будет результат выполнения программы при $x = -10$ и $y = 3$? Как вы можете объяснить этот результат?

1.5.2. Программирование циклов с известным условием окончания работы

В языке Python нет специального оператора для записи циклов с заданным условием окончания работы, но его можно организовать с помощью цикла **while**:

```
while True:
    <операторы>
    if <условие>: break
```

Цикл

```
while True:
    <операторы>
```

будет выполняться бесконечно, потому что условие **True** всегда истинно. Выйти из такого цикла можно только с помощью специального оператора **break** (в переводе с англ. — «прервать»).

Запишем на языке Python рассмотренный в п. 2.4.3 (пример 17) учебника для 8 класса алгоритм решения задачи о графике тренировок спортсмена.

```
# Программа 15
print('График тренировок')
i = 1; x = 10
while True:
    i += 1
    x = x + 0.1 * x
    if x >= 25: break
print('Начиная с ', i, '-го дня спортсмен будет пробегать 25 км' sep''')
```

1.5.3. Программирование циклов с известным числом повторений

Цикл с известным числом повторений в языке Python записывается с помощью оператора **for** (в переводе с английского — «для»). Его общий вид:

```
for <параметр> in range(k, n, m):
    <операторы>
```

Здесь:

<параметр> — переменная целого типа;

range() — функция, описывающая необходимое количество повторов тела цикла; в скобках может быть указано от одного до трёх чисел:

- одно число (n) указывает на то, что нужно последовательно перебрать все целые числа от 0 до n - 1;
- два числа (k, n) говорят о том, что нужно последовательно перебрать все целые числа, находящиеся в диапазоне от k (начальное значение) до n - 1 (конечное значение);
- три числа (k, n, m) указывают на то, что параметр должен изменяться от k до n - 1 с шагом, равным m;

<операторы> — один или несколько операторов, составляющих тело цикла.

При выполнении цикла **for** с тремя параметрами после каждого выполнения тела цикла происходит увеличение параметра

цикла на шаг m . Если $k = n$ или $k > n$, цикл не выполнится ни разу.

Запишем на языке Python рассмотренный в п. 2.4.3 (пример 19) учебника для 8 класса алгоритм вычисления степени с натуральным показателем n для любого вещественного числа a .

```
# Программа 16
print('Возведение в степень')
a = float(input('Введите основание a>>'))
n = int(input('Введите показатель n>>'))
y = 1
for i in range(n):
    y = y * a
print('y=', y)
```

Здесь параметр цикла i будет изменяться от 0 до $n - 1$ включительно, следовательно, тело цикла выполнится ровно n раз.

1.5.4. Различные варианты программирования циклического алгоритма

Для решения одной и той же задачи могут быть предложены разные алгоритмы, на основе которых могут быть разработаны разные программы. Вы могли убедиться в этом, программируя ветвления. Рассмотрим пример, показывающий, что реализация цикла может быть запрограммирована разными способами.

Пример. Напишем программу, в которой осуществляется ввод целых чисел до тех пор, пока не будет введён ноль, и подсчёт количества введённых положительных и отрицательных чисел.

Так как здесь в явном виде задано условие окончания работы, то воспользуемся конструкцией **while True**:

```
# Программа 17
k1 = k2 = 0
while True:
    n = int(input('Введите целое число>>'))
    if n > 0:
        k1 += 1
    if n < 0:
        k2 += 1
    if n == 0: break
```

```
print ('Введено:')
print ('положительных чисел -', k1)
print ('отрицательных чисел -', k2)
```

Имеющееся условие окончания работы можно достаточно просто преобразовать в условие продолжения работы — работа продолжается, пока n не равно 0, значит, мы можем воспользоваться оператором **while** с таким условием:

```
# Программа 18
k1 = k2 = 0
n = int(input('Введите целое число>>'))
while n != 0:
    if n > 0:
        k1 += 1
    if n < 0:
        k2 += 1
    n = int(input('Введите целое число>>'))
print('Введено:')
print('положительных - ', k1);
print('отрицательных - ', k2)
```

В рассмотренном примере число повторений тела цикла заранее неизвестно, и поэтому оператор **for** здесь применить нельзя. Если число повторений тела цикла известно, то лучше воспользоваться оператором **for**. Вместе с тем любая задача, в которой число повторений тела цикла определено заранее, может быть запрограммирована с помощью любого из двух рассмотренных выше циклов.

САМОЕ ГЛАВНОЕ

В языке Python имеются два вида операторов цикла: **while** (цикл с условием) и **for** (цикл с параметром).

Цикл с условием выполняется до тех пор, пока некоторое условие (условие продолжения работы цикла) не станет ложным. Если условие в заголовке цикла ложно с самого начала, тело цикла не выполнится ни разу.

Если условие в заголовке цикла всегда остаётся истинным, цикл работает бесконечно. Для досрочного выхода из цикла используют оператор **break**.

Цикл с параметром применяют тогда, когда количество повторений цикла известно заранее или может быть вычислено до

начала цикла. Переменная, изменение которой определяет работу цикла, называется параметром (переменной) цикла. При вызове функции `range()` указывают от одного до трёх параметров: начальное значение, значение-ограничитель (не входящее в диапазон) и шаг изменения переменной цикла; обязательный параметр — значение-ограничитель.

Если число повторений тела цикла известно, то лучше воспользоваться оператором `for`; в остальных случаях используется оператор `while`.

Вопросы и задания

1. Проанализируйте работу программы:

```
x = 1
y = 1
while x < 5:
    y *= 2
    x += 1
```

Ответьте на вопросы.

- Сколько раз выполнится тело цикла?
 - Какое значение примет `x` после завершения программы?
 - Какое значение примет `y` после завершения программы?
 - Сколько раз выполнится тело цикла, если заменить условие на `x <= 5`?
 - Сколько раз выполнится тело цикла, если заменить условие на `x >= 5`?
 - Сколько раз выполнится тело цикла, если заменить условие на `x > 0`?
 - Что произойдёт, если из тела цикла убрать команду `x += 1`?
 - Сколько раз выполнится тело цикла, если заменить команду `x += 1` на `x += 2`?
 - Сколько раз выполнится тело цикла, если заменить команду `x += 1` на `x -= 1`?
2. Дана последовательность операторов:

```
a = 1; b = 2
while a + b < 8:
    a += 1
    b += 2
    s = a + b
```

Сколько раз будет выполнено тело цикла и какими будут значения переменных a , b , s после выполнения этой последовательности операторов?

3. Определите значения переменных s и i после выполнения фрагмента программы.

а) $s = 0$ $i = 0$ while $i < 5$: $i += 1$ $s += i$	б) $s = 0$ $i = 0$ while $i < 5$: $i += 1$ $s += i$	в) $s = 0$ $i = 2$ while $i > 1$: $s = s + 1 / i$ $i = i - 1$
---	---	---

4. Определите значение переменной s после выполнения фрагмента программы при указанных значениях a . Составьте таблицы значений переменных.

```

p = a
s = 0
while p > 0:
    s = s + p % 10
    p = p // 10
    
```

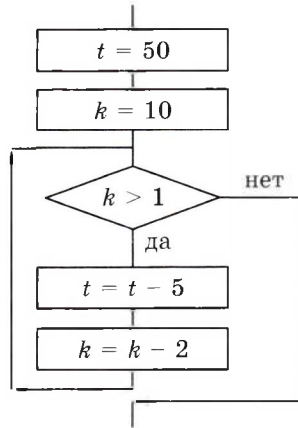
- а) $a = 23$;
- б) $a = 32$;
- в) $a = 109$;

5. От программы, записанной на алгоритмическом языке, перейдите к записи программы на языке Python. Определите, что будет выведено в результате работы программы. Составьте таблицу значений переменных.

```

алг
    цел s, k
нач
    s := 0
    k := 0
    нц пока k < 6
        s := s + 2
        k := k + 1
    кц
    вывод s
кон
    
```

6. Запишите на языке Python фрагмент программы, соответствующий блок-схеме. Определите значения переменных k и t после её выполнения. Составьте таблицу значений переменных.



7. Требовалось написать программу вычисления факториала числа n (факториал числа n — это произведение всех целых чисел от 1 до n). Программист торопился и написал программу неправильно. Ниже приведён фрагмент его программы, в котором содержатся три ошибки:

```

k = 1
f = 0
while k < n:
    f = f * k
k += 1
  
```

Найдите ошибки. Допишите необходимые операторы и выполните программу на компьютере.

Пример входных данных	Пример выходных данных
Введите $n \gg 5$	$5! = 120$
Введите $n \gg 6$	$6! = 720$

8. Проанализируйте следующий цикл:

```

while a < b:
    c = a == b
  
```

В чём его особенность?

9. Запишите на языке Python программы решения № 26–30 из § 2.4 из учебника для 8 класса. Используйте оператор **while**.

10. Дана последовательность операторов:

```
a = 1
b = 1
while True:
    a += 1
    b *= 2
    if b > 8: break
s = a + b
```

Сколько раз будет выполнено тело цикла и какими будут значения переменных *a*, *b*, *s* после выполнения этой последовательности операторов?

11. Определите значение переменных *s* и *i* после выполнения следующих операторов:

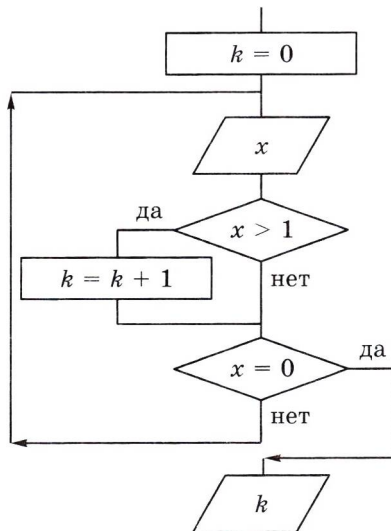
а) `s = 0`
`i = 3`
while True:
`s = s + 5 // i`
`i -= 1`
if i < 1:
`break`

б) `s = 0`
`i = 1`
while True:
`s = s + 1 // i`
`i -= 1`
if i <= 1:
`break`

12. От программы, записанной на алгоритмическом языке, перейдите к записи программы на языке Python. Определите, что будет выведено в результате работы программы, если были введены следующие числа: 1, 5, -10, 3, -8, 6, 4, 0. Составьте таблицу значений переменных.

```
алг
  цел s, x
нач
  s := 0
нц
  ввод x
  s := s + x
кц при x = 0
вывод s
кон
```


13. Запишите на языке Python фрагмент программы, соответствующий блок-схеме. Определите значение переменной k после его выполнения при следующих значениях переменной x : 1, 5, -10, 3, -8, 6, 1, 2, -7, 4, 0. Составьте таблицу значений переменных.



14. Дана программа на языке Python:

```

k1 = 0
k2 = 0
while True:
    print('Введите целое число')
    x = int(input())
    if x < 0: k1 += 1
    if x > 0: k2 += 1
    if x == 0: break
print('k1=', k1, 'k2=', k2)
    
```

Составьте блок-схему, соответствующую программе. Какая задача решается с помощью этой программы?

15. Напишите программу, в которой осуществляется ввод целых чисел до тех пор, пока не будет введён ноль, и подсчёт суммы и среднего арифметического введённых положительных чисел.

16. Напишите программу, в которой осуществляется ввод целых чисел до тех пор, пока не будет введён ноль, и определение максимального (наибольшего) из введённых чисел.

17. Напишите программу вычисления наибольшего общего делителя двух целых чисел.

18. Сколько раз будет выполнено тело цикла?

- а) `for i in range (15): s += 1`
- б) `for i in range (10, 15): s += 1`
- в) `for i in range (-1, 1): s += 1`
- г) `for i in range (1, 1): s += 11`
- д) `k = 5`
`for i in range (k - 1, k + 1): s += 1`

19. Определите значения переменных `s` и `i` после выполнения следующих операторов. Составьте таблицы значений переменных.

- | | |
|--|--|
| а) <code>s = 0</code>
<code>for i in range(6):</code>
<code> s += i</code> | б) <code>s = 1</code>
<code>n = 1</code>
<code>for i in range(2, n + 1):</code>
<code> s += 1 / i</code> |
| в) <code>s = 1</code>
<code>n = 1</code>
<code>for i in range(1, 4):</code>
<code> s += 1 / n</code>
<code>n = n + 2</code> | г) <code>s = 1</code>
<code>n = 1</code>
<code>for i in range(1, 4):</code>
<code> s += 1 / n</code>
<code>n = n + 2</code> |

20. Что будет выведено в результате выполнения цикла?

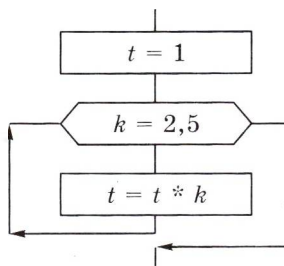
- а) `for x in range(1, 6): print('#', end= '')`
- б) `for x in range(6): print('#', end= '')`
- в) `for x in range(2, 8): print('#', end= '')`
- г) `for x in range(5, 6): print('#', end= '')`
- д) `for x in range(6, 6): print('#', end= '')`
- е) `for x in range(5, 0, 1): print('#', end= '')`
- ж) `for x in range(5, 4, -1): print('#', end= '')`
- з) `for x in range(5, 1, -1): print('#', end= '')`
- и) `for x in range(1, 4, -1): print('#', end= '')`

21. От программы, записанной на алгоритмическом языке, перейдите к записи программы на языке Python. Определите, что будет выведено в результате работы программы. Составьте таблицу значений переменных.

```

алг
  цел s, k
нач
  s := 0
  нц для k от 1 до 5
    s := s + 2 * k
  кц
  вывод s
кон
    
```

22. Запишите на языке Python фрагмент программы, соответствующий блок-схеме. Определите значение переменных k и t после его выполнения. Составьте таблицу значений переменных.



23. Определите результат работы программы, записанной на языке Python.

а)

```
m = 0
for i in range(1, 7):
    m -= 6
print(m)
```

б)

```
m = 0
for i in range(3, 8):
    m -= 6 - i
print(m)
```

24. Проанализируйте фрагменты программ. Запишите результат их работы. Для каждого случая запишите фрагмент программы, обеспечивающий такой же результат, но с использованием другого оператора цикла.

- а) `x = 1`
`while x <= 5:`
`print(x)`
`x += 1`
- б) `for x in range(-2, 3):`
`y = abs(x)`
`print(y)`
- в) `x = 10`
`while x >= 5:`
`print(x)`
`x -= 1`
- г) `for x in range(5, -1, -1):`
`y = x * x`
`print(y)`

25. Определите, что будет выведено в результате работы следующей программы. Текст программы приведен на двух языках программирования.

Алгоритмический язык	Python
<pre> алг нач цел i, a, b a := 2 b := 3 нц для i от 5 до 9 a := a + 3 + div(a, 3) b := b + a - div(b, 4) кц вывод a + b кон </pre>	<pre> a = 2 b = 3 for i in range(5, 10): a = a + 3 + a // 3 b = b + a - b // 4 print(a + b) </pre>

26. Напишите программу, которая 10 раз выводит на экран ваши имя и фамилию.
27. Напишите программу, выводящую на экран изображение шахматной доски, где чёрные клетки изображаются звёздочками, а белые — пробелами. Рекомендуемый вид экрана после выполнения программы:

```

*      *      *      *      *
      *      *      *      *
*      *      *      *      *
      *      *      *      *
*      *      *      *      *
      *      *      *      *
*      *      *      *      *
      *      *      *      *
                    
```

28. Напишите программу, которая вычисляет сумму:
- первых n натуральных чисел;
 - квадратов первых n натуральных чисел;
 - всех чётных чисел в диапазоне от 1 до n ;
 - всех двузначных чисел.
29. Напишите программу, которая генерирует 10 случайных чисел в диапазоне от 1 до 20, выводит эти числа на экран и вычисляет их среднее арифметическое.
30. Напишите на языке Python программы решения задач № 32, 33 из § 2.4 учебника для 8 класса. Используйте оператор `for`.
31. Напишите программу, которая выводит на экран таблицу степеней двойки (от нулевой до десятой). Рекомендуемый вид экрана после выполнения программы:

Таблица степеней двойки:

0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024

32. Напишите программу, которая выводит на экран таблицу умножения на n (n — целое число в диапазоне от 2 до 10, вводимое с клавиатуры).

Пример входных данных	Пример выходных данных
Введите $n >> 5$	$5 * 2 = 10$ $5 * 3 = 15$ $5 * 4 = 20$ $5 * 5 = 25$ $5 * 6 = 30$ $5 * 7 = 35$ $5 * 8 = 40$ $5 * 9 = 45$ $5 * 10 = 50$

33. Какой из двух рассмотренных операторов цикла является, по вашему мнению, основным, т. е. таким, что им можно заменить другой оператор? Обоснуйте свою точку зрения.

Проверочная работа № 5

1. Определите значение переменной *s* после выполнения операторов:

```
i = 0
s = 0
while i < 3 :
    i += 1
    s += 2 * i
```

2. Определите, что будет выведено на экран в результате выполнения алгоритма. Запишите условие задачи, для решения которой составлен данный алгоритм.

```
n = 200
while n % 18 != 0:
    n += 1
print('Ответ: ', n)
```

3. Используя цикл **while**, напишите программу определения суммы всех нечётных чисел от 1 до 99 включительно.

4. Сколько раз выполнится тело следующего цикла?

```
i = 21
while True:
    i -= 5
    if i > 21:
        break
```

5. Какое число будет выведено на экран в результате работы следующей программы?

```
k = 1
s = 0
while True:
    s += k
    k += 2
    if k < 8:
        break
print(s)
```

6. Запишите значение переменной `sum` после выполнения фрагмента программы.

```
sum = 0
for i in range(5, 10):
    sum += i
```

7. Запишите результат выполнения программы.

```
p = 1
for i in range(6, 2, -1):
    p *= i
print(p)
```

8. *Дополнительное задание.* Напишите программу, которая по двум натуральным числам a и b , не превосходящим 30 000, подсчитывает количество натуральных чисел, кратных 10, на отрезке $[a, b]$.

Программа получает на вход два натуральных числа a и b , при этом гарантируется, что $1 < a < b < 30\,000$. Проверять входные данные на корректность не нужно.

Программа должна вывести одно число: количество натуральных чисел, кратных 10, на отрезке $[a, b]$.

Входные данные	Выходные данные
7 37	3

Тестовые задания для самоконтроля за курс 8 класса

1. Разработчиком языка Python является:
- а) Блез Паскаль
 - б) Никлаус Вирт
 - в) Норберт Винер
 - г) Гвидо ван Россум

2. Что из нижеперечисленного входит в алфавит языка Python?
- а) латинские строчные и прописные буквы
 - б) служебные слова
 - в) русские строчные и прописные буквы
 - г) знак подчёркивания
3. Какая последовательность символов не может служить именем в языке Python?
- а) `_mas`
 - б) `maSl`
 - в) `d2`
 - г) `2d`
4. Вещественные числа имеют тип данных:
- а) `float`
 - б) `int`
 - в) `bool`
 - г) `str`
5. Языковые конструкции, с помощью которых в программах записываются действия, выполняемые в процессе решения задачи, называются:
- а) операндами
 - б) операторами
 - в) выражениями
 - г) данными
6. Разделителями между операторами в одной строке служит:
- а) точка
 - б) точка с запятой
 - в) пробел
 - г) запятая
7. При присваивании изменяется:
- а) имя переменной
 - б) тип переменной
 - в) значение переменной
 - г) значение константы

8. Для вывода результатов в Python используется оператор
- а) while
 - б) input()
 - в) print()
 - г) and
9. Для вычисления квадратного корня из x используется функция:
- а) abs(x)
 - б) sqr(x)
 - в) sqrt(x)
 - г) int(x)
10. Для генерации случайного целого числа из отрезка [10, 20] необходимо использовать выражение:
- а) randint(2*10)
 - б) randint(1020)
 - в) randint(10, 20)
 - г) randint(10) * 2
11. В каких условных операторах допущены ошибки?
- а) `if b = 0: print('Деление невозможно.')`
 - б) `if a < b: min = a; else min = b`
 - в) `if a > b : max = a
 else max = b`
 - г) `if a > b and b > 0: c = a + b`
12. Определите значение переменной c после выполнения следующего фрагмента программы.
- ```
a = 100
b = 30
a = a - b * 3
if a > b:
 c = a - b
else:
 c = b - a
```
- а) 20
  - б) 70
  - в) -20
  - г) 180

## 13. Условный оператор

```
if a % 2 == 0:
 print('Да')
else:
 print('Нет')
```

позволяет определить, является ли число  $a$ :

- а) целым
- б) двузначным
- в) чётным
- г) простым

## 14. Какого оператора цикла не существует в языке Python?

- а) for
- б) while
- в) repeat...until

## 15. Тело цикла в фрагменте программы

```
a = 1
b = 1
while a + b < 8:
 a += 1
 b += 2
```

выполнится:

- а) 0 раз
- б) 2 раза
- в) 3 раза
- г) бесконечное число раз

16. Определите значения переменных  $s$  и  $i$  после выполнения фрагмента программы.

```
s = 0
i = 5
while i > 0:
 s += i
 i -= 1
```

- а)  $s = 0$ ;  $i = -1$
- б)  $s = 5$ ;  $i = 0$
- в)  $s = 15$ ;  $i = 5$
- г)  $s = 15$ ;  $i = 0$

17. В данном фрагменте программы

```
s = 0
for i in range(1, 11):
 s = s + 2 * i
```

вычисляется:

- а) сумма целых чисел от 1 до 10
- б) сумма чётных чисел от 1 до 10
- в) удвоенная сумма целых чисел от 1 до 10
- г) сумма первых десяти натуральных чётных чисел

## Глава 2

# АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ

### § 2.1

## Одномерные массивы целых чисел

#### **Ключевые слова:**

- массив
- элемент массива
- индекс элемента
- значение элемента
- заполнение массива
- вывод массива
- обработка массива
- последовательный поиск
- сортировка

До сих пор мы работали с простыми типами данных. Для решения многих практических задач из простых типов образуют составные типы данных, так называемые структуры данных. Примером структуры является массив.

**Массив** — это совокупность фиксированного количества однотипных элементов, которым присвоено общее имя. Доступ к отдельному элементу массива осуществляется по его номеру (индексу).

С подобными объектами — числовыми последовательностями — вы уже встречались на уроках математики. Например, члены арифметической прогрессии обозначались так:  $a_1, a_2, a_3, \dots, a_n$ .

Размерность массива — это количество индексов, необходимое для однозначного доступа к элементу массива. Массивы с одним индексом называют одномерными, с двумя — двумерными и т. д. Мы будем рассматривать одномерные массивы.



В языке Python нет такой структуры данных, как «массив»; для хранения группы однотипных объектов используют списки — объекты типа `list`. Далее, говоря о списках, мы будем использовать слово «массив».



В отличие от массивов список — это динамическая структура: количество элементов списка можно изменять во время выполнения программы, удаляя и добавляя элементы. Кроме того, в отличие от массива в списке можно хранить данные разных типов.

### 2.1.1. Обращение к элементу массива

Нумерация элементов массивов в Python всегда начинается с нуля.

Для того чтобы обратиться к элементу массива, записывают имя массива, а после него в квадратных скобках указывают индекс нужного элемента.

Рассмотрим пример массива (рис. 2.1):

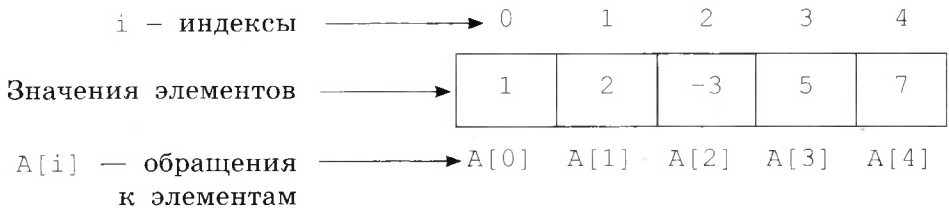


Рис. 2.1. Одномерный массив

Здесь:

$A$  — имя массива;

$A[0] = 1$  (значение элемента массива  $A$  с индексом 0 равно 1),  $A[1] = 2$ ,  $A[2] = -3$ ,  $A[3] = 5$ ,  $A[4] = 7$ .

Индексом может быть не только целое число, но и целое значение переменной или арифметического выражения. Так, в нашем примере  $A[4 * i - 2] = -3$  при  $i = 1$ .

Индексом может быть даже значение элемента массива. В нашем примере  $A[A[1]] = -3$ .

При обращении к элементу массива с несуществующим индексом происходит серьёзная ошибка — выход за границы массива — и программа завершается аварийно.

Длина массива (количество элементов массива) определяется с помощью функции `len()`:  $N = \text{len}(A)$ .

5 — длина массива, представленного на рис. 2.1.

Иногда размер массива хранят в отдельной переменной:  $N = 5$ .

Перед использованием в программе массив необходимо создать, в противном случае обращение к несуществующему элементу вызовет ошибку и аварийное завершение программы.

## 2.1.2. Заполнение массива

Заполнять массив в Python можно несколькими способами. Рассмотрим некоторые из них.

1. Перечисление значений элементов. Массив можно создать перечислением элементов через запятую в квадратных скобках, например, так:

```
A = [1, 2, -3, 5, 7].
```

С помощью записи

```
D = [1] * 5
```

будет создан массив из пяти элементов, каждый из которых равен 1.

2. Ввод значений элементов с клавиатуры. Небольшие массивы можно вводить с клавиатуры. Для этого можно использовать цикл с параметром, выполняющий оператор ввода отдельно для каждого элемента массива:

```
for i in range(N):
 A[i] = int(input())
```

При каждом повторении цикла строка, введенная пользователем, преобразуется в целое число с помощью функции `int()`, и это число добавляется в массив. Пользователь сам должен следить за тем, значение какого именно элемента он вводит в тот или иной момент.

Чтобы перед вводом значения очередного элемента на экране появлялась подсказка с индексом этого элемента, можно использовать следующий цикл:

```
for i in range(N):
 print("A[{}]=".format(i), end="")
 A[i] = int(input())
```

В этом случае, например, при вводе значения элемента с индексом 2 на экран будет выведено `A[2]=` и справа от знака «=» будет мигать курсор — приглашение к вводу.

3. Заполнение массива случайными числами. Для работы со случайными числами сначала нужно подключить функцию `randint()` модуля `random()`, генерирующую случайное целое число в заданном диапазоне:

```
from random import randint
```

Далее можно использовать цикл с параметром:

```
for i in range(N):
 A[i] = randint(10, 110)
```

Массив из  $N$  элементов будет заполнен случайными числами, принадлежащими отрезку  $[10, 110]$ .

4. Задание значений элементов массива по формуле.

Программа

```
for i in range(N):
 A[i] = i
```

заполняет массив целыми числами от 0 до  $N-1$ .

Программа

```
for i in range(N):
 A[i] = i ** 2
```

заполняет элементы массива числами, равными квадратам их индексов.

### 2.1.3. Вывод массива

Во многих случаях бывает полезно вывести значения элементов массива на экран. Так, если значения массива генерировались случайным образом, то необходимо знать, какими именно значениями заполнен исходный массив. Также нужно видеть, как изменились значения элементов массива после обработки.

Самый простой способ — вывести список как один объект:

```
print(A)
```

В этом случае весь массив выводится в квадратных скобках, а его элементы разделяются запятыми.

Можно вывести элементы массива на экран по одному, используя цикл:

```
for i in range(len(A)):
 print(A[i], end=" ")
```

Параметр `end` определяет, что после вывода каждого элемента добавляется пробел, а не символ перехода на новую строку.

Значения элементов массива можно вывести в столбик:

```
for i in range(len(A)):
 print(A[i])
```

Более наглядным является следующий вариант вывода с комментариями:

```
for i in range(N):
 print ('A[' + i + ']= ' + A[i])
```

На основании рассмотренных примеров напомним программу, в которой осуществляется: заполнение целочисленного массива `A`, состоящего из 10 элементов, случайными числами, значения которых изменяются в диапазоне от 0 до 99; вывод массива `A` на экран.

```
Программа 19
```

```
N = 10
```

```
A = [0] * N
```

```
from random import randint
```

```
for i in range(N):
 A[i] = randint(0,99)
```

```
for i in range(N):
 print ('A[' + i + ']= ' + A[i])
```

Задание массива

Подключение генератора случайных чисел

Заполнение массива

Вывод массива

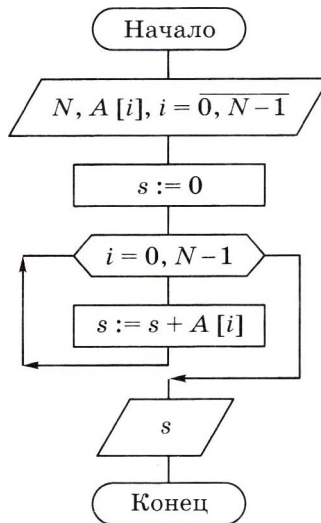
### 2.1.4. Вычисление суммы элементов

**Пример.** В некотором населённом пункте `N` домов. Известно, сколько людей проживает в каждом из домов. Составим алгоритм подсчёта количества жителей населённого пункта.



Исходные данные (количество жильцов) представим в виде одномерного массива  $A$ , содержащего  $N$  элементов:  $A[0]$  — количество жильцов дома 1,  $A[1]$  — количество жильцов дома 2,  $A[N-1]$  — количество жильцов дома  $N$ .

В общем случае  $A[i]$  — количество жильцов дома  $i+1$ , где  $i$  принимает целочисленные значения от 0 до  $N-1$  (кратко обозначим это в блок-схеме как  $i = 0, N-1$ ). Результат работы алгоритма обозначен через  $s$  (рис. 2.2).



**Рис. 2.2.** Блок-схема задачи вычисления суммы элементов массива

Суммирование элементов массива осуществляется по тому же принципу, что и суммирование значений простых переменных — за счёт поочерёдного добавления слагаемых:

- 1) определяется ячейка памяти (переменная  $s$ ), в которой будет последовательно накапливаться результат суммирования;
- 2) переменной  $s$  присваивается начальное значение 0 — число, не влияющее на результат сложения;
- 3)  $N$  раз текущее значение переменной  $s$  складывается со значением текущего элемента массива  $A[i]$ ; полученный результат присваивается переменной  $s$ .

Описанный процесс наглядно можно изобразить так:

|       |            |                                 |
|-------|------------|---------------------------------|
|       | s=0        | s=0                             |
| i=0   | s=s+A[0]   | s=0+A[0]                        |
| i=1   | s=s+A[1]   | s=0+A[0]+A[1]                   |
| i=2   | s=s+A[2]   | s=0+A[0]+A[1]+ [2]              |
|       | ...        | ...                             |
| i=N-1 | s=s+A[N-1] | s=0+A[0]+A[1]+A[2]+ ...+ A[N-1] |

Запишем соответствующую программу на языке Python.

|                                                                     |                                        |
|---------------------------------------------------------------------|----------------------------------------|
| # Программа 20<br>N = 10<br>A = [0] * N                             | Задание массива                        |
| <b>from</b> random <b>import</b> randint                            | Подключение генератора случайных чисел |
| <b>for</b> i <b>in</b> range(N):<br>A[i] = randint(50, 200)         | Заполнение массива                     |
| <b>for</b> i <b>in</b> range(N):<br>print ('A[' , i, ' ]= ' , A[i]) | Вывод массива                          |
| s = 0<br><b>for</b> i <b>in</b> range(N):<br>s += A[i]              | Вычисление суммы элементов массива     |
| print('s=', s)                                                      | Вывод результата                       |

Рассмотрим работу этого алгоритма для массива

A = [100, 120, 130, 80, 70].

|     |         |       |
|-----|---------|-------|
|     | s=0     | s=0   |
| i=0 | s+=A[0] | s=100 |
| i=1 | s+=A[1] | s=220 |
| i=2 | s+=A[2] | s=350 |
| i=3 | s+=A[3] | s=430 |
| i=4 | s+=A[4] | s=500 |



Сравните программы 2 и 3. Выделите в них общие блоки. Обратите внимание на различия.

Каким образом в программе 3 уточнена информация, представленная в примере о домах населённого пункта?

### 2.1.5. Последовательный поиск в массиве

В программировании поиск — одна из наиболее часто встречающихся задач невычислительного характера.

Можно выделить следующие типовые задачи поиска:

- 1) найти наибольший (наименьший) элемент массива;
- 2) найти элемент массива, значение которого равно заданному значению.

Для решения таких задач в программе необходимо организовать последовательный просмотр элементов массива и сравнение значения очередного просматриваемого элемента с неким образцом.

Рассмотрим подробно решение задач первого типа: нахождение наибольшего (наименьшего) элемента.

Представим себе одномерный массив в виде стопки карточек, на каждой из которых написано число. Тогда идея поиска наибольшего элемента массива может быть представлена следующим образом:



- 1) возьмём верхнюю карточку (первый элемент массива), запомним имеющееся на карточке число (запишем его мелом на доске) как наибольшее из просмотренных; уберём карточку в сторону;
- 2) возьмём следующую карточку; сравним числа, записанные на карточке и на доске; если число на карточке больше, то сотрём число, записанное на доске, и запишем там то же число, что и на карточке; если же новое число не больше, то на доске оставим имеющуюся запись; уберём карточку в сторону;
- 3) повторим действия, описанные в п. 2, для всех оставшихся карточек в стопке.

В итоге на доске будет записано наибольшее значение элемента просмотренного массива.



В программировании при обосновании корректности циклических алгоритмов используется понятие инварианта цикла.

*Инвариант цикла* — логическое выражение (условие), зависящее от переменных, изменяющихся в теле цикла; оно истинно непосредственно перед началом выполнения цикла и после каждого прохода тела цикла.

Условие «записанное на доске число — самое большое из всех просмотренных до сих пор» является инвариантом цикла для рассмотренного алгоритма.

Так как доступ к значению элемента массива осуществляется по его индексу, то при организации поиска наибольшего элемента в одномерном массиве можно запоминать (хранить) его индекс. Обозначим искомый индекс *imax*. Тогда описанный выше алгоритм в сформированном нами массиве *A* на языке Python можно записать так:

|                                                                                            |                                        |
|--------------------------------------------------------------------------------------------|----------------------------------------|
| <pre># Программа 21 N = 10 A = [0] * N</pre>                                               | Задание массива                        |
| <pre>from random import randint</pre>                                                      | Подключение генератора случайных чисел |
| <pre>for i in range(N):     A[i] = randint(0, 99)     print ('A[' + i + ']= ', A[i])</pre> | Заполнение и вывод массива             |
| <pre>imax = 0 for i in range(1,N):     if A[i] &gt; A[imax]: imax = i</pre>                | Поиск индекса наибольшего элемента     |
| <pre>print('Наибольший элемент: ', A[imax])</pre>                                          | Вывод результата                       |

Рассмотрим работу этого алгоритма для массива

*A* = [100, 120, 130, 80, 70].

| <i>imax</i>             | <i>i</i> | <i>A[i] &gt; A[imax]</i> |
|-------------------------|----------|--------------------------|
| 0                       | 1        | 120 > 100 (Да)           |
| 1                       | 2        | 130 > 120 (Да)           |
| 2                       | 3        | 80 > 130 (Нет)           |
| 2                       | 4        | 70 > 13 (Нет)            |
| Наибольший элемент: 130 |          |                          |



Если в массиве есть несколько элементов, значения которых равны максимальному значению, то данная программа найдёт первый из них (первое вхождение). Подумайте, что следует изменить в программе, чтобы в ней находился последний из максимальных элементов. Как следует преобразовать программу, чтобы с её помощью можно было найти минимальный элемент массива?

Результатом решения задачи второго типа (нахождение элемента массива, значение которого равно заданному значению) может быть:

- $k$  — индекс элемента массива такой, что  $A[k] = x$ , где  $x$  — заданное число;
- сообщение о том, что искомого элемента в массиве не обнаружено.

Программа поиска в сформированном нами массиве  $A$  значения, равного  $x$ , может выглядеть так:

|                                                                                                                                                                          |                                        |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------|
| <pre># Программа 22 N = 10 A = [0] * N</pre>                                                                                                                             | Задание массива                        |
| <pre>from random import randint</pre>                                                                                                                                    | Подключение генератора случайных чисел |
| <pre>for i in range(N):     A[i] = randint(0, 99)     print ('A[' + i + ']= ' + A[i])</pre>                                                                              | Заполнение и вывод массива             |
| <pre>x = int(input('x='))</pre>                                                                                                                                          | Ввод значения $x$                      |
| <pre>nx = -1 for i in range(0, N):     if A[i] == x: nx = i</pre>                                                                                                        | Поиск индекса элемента, равного $x$    |
| <pre>if nx == -1:     print('Элемента со           значением, равным ',           x, ' нет') else:     print('Индекс элемента,           равного заданному, ', nx)</pre> | Вывод результата                       |

В этой программе последовательно просматриваются все элементы массива. Номер найденного элемента сохраняется в переменной `nx`. Если в массиве есть несколько элементов, значения которых равны заданному числу, то программа найдёт последний из них. Если значение переменной `nx` не изменилось в ходе выполнения цикла и осталось равным `-1`, то это означает, что в массиве нет элемента, равного `x`.

Во многих случаях требуется найти первый из элементов, имеющих соответствующее значение, и дальнейший просмотр массива прекратить. Для этой цели можно использовать следующий фрагмент программы:

```
nx = -1
for i in range(N):
 if A[i] == x:
 nx = i
 break
if nx >= 0:
 print("A[{}]={}".format(nx, x))
else:
 print("Элемент не найден")
```

Используя цикл с параметром, мы начали последовательный перебор элементов массива и завершим его досрочно, как только будет найдено требуемое значение. В таком случае для выхода из цикла используется оператор **break**.

Здесь выполнение алгоритма будет завершено (прервано) в одном из двух случаев:

- 1) в массиве найден первый из элементов, равный заданному значению;
- 2) все элементы массива просмотрены.

Запишите полный текст программы и выполните её на компьютере.



Зачастую требуется определить количество элементов, удовлетворяющих некоторому условию. В этом случае вводится переменная, значение которой увеличивается на единицу каждый раз, когда найден нужный элемент.

Определите, количество каких элементов подсчитывается с помощью следующего фрагмента программы.



```
k = 0
for i in range(10):
 if A[i] > 50: k += 1
print("k=", k)
```

Если требуется определить сумму значений элементов, удовлетворяющих некоторому условию, то вводят переменную, к значению которой прибавляют значение найденного элемента массива.



Определите, какому условию удовлетворяют элементы массива, значения которых суммируются с помощью следующего фрагмента программы.

```
s = 0
for i in range(10):
 if A[i] > 50 and A[i] < 60: s += A[i]
print("s=", s)
```



Запишите полные тексты двух программ и выполните их на компьютере.

### 2.1.6. Сортировка массива



Под **сортировкой (упорядочением)** массива понимают перераспределение значений его элементов в некотором определённом порядке.

Порядок, при котором в массиве первый элемент имеет самое маленькое значение, а значение каждого следующего элемента не меньше значения предыдущего элемента, называют **неубывающим**.

Порядок, при котором в массиве первый элемент имеет самое большое значение, а значение каждого следующего элемента не больше значения предыдущего элемента, называют **невозрастающим**.

Цель сортировки — облегчить последующий поиск элементов: искать нужный элемент в упорядоченном массиве легче.

Рассмотрим один из возможных алгоритмов сортировки массивов — сортировку выбором.

Сортировка выбором (например, по невозрастанию) осуществляется следующим образом:

- 1) в массиве выбирается максимальный элемент;
- 2) максимальный и первый элементы меняются местами; первый элемент считается отсортированным;

3) в неотсортированной части массива снова выбирается максимальный элемент; он меняется местами с первым неотсортированным элементом массива;

4) действия, описанные в п. 3, повторяются с неотсортированными элементами массива до тех пор, пока не останется один неотсортированный элемент (его значение будет минимальным).

Рассмотрим процесс сортировки выбором на примере массива

$$A = [0, 1, 9, 2, 4, 3, 6, 5].$$

| Индекс   | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |   |
|----------|---|---|---|---|---|---|---|---|---|
| Значение | 0 | 1 | 9 | 2 | 4 | 3 | 6 | 5 |   |
| Шаги     | 1 | 0 | 1 | 9 | 2 | 4 | 3 | 6 | 5 |
|          | 2 | 9 | 1 | 0 | 2 | 4 | 3 | 6 | 5 |
|          | 3 | 9 | 6 | 0 | 2 | 4 | 3 | 1 | 5 |
|          | 4 | 9 | 6 | 5 | 2 | 4 | 3 | 1 | 0 |
|          | 5 | 9 | 6 | 5 | 4 | 2 | 3 | 1 | 0 |
|          | 6 | 9 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|          | 7 | 9 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Итог:    | 9 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |   |

В этом массиве из 8 элементов операцию выбора максимального элемента мы проводили 7 раз. В массиве из  $n$  элементов такая операция будет проводиться  $n - 1$  раз. Объясните почему.



Приведём фрагмент программы, реализующий описанный алгоритм:

```
for i in range(n - 1):
 imax = i
 for j in range(i + 1, n):
 if A[j] > A[imax]: imax = j
 A[i], A[imax] = A[imax], A[i]
```



Здесь мы использовали один цикл внутри другого. Такая конструкция называется **вложенным циклом**.



Запишите полный текст программы и выполните её на компьютере для рассмотренного в примере массива *A*.

### САМОЕ ГЛАВНОЕ

Для работы с большим количеством однотипных данных используются массивы.

**Массив** — это совокупность фиксированного количества однотипных элементов, которым присвоено общее имя. Доступ к отдельному элементу массива осуществляется по его номеру (индексу).

Нумерация элементов массива в Python начинается с нуля.

При обращении к элементу массива индекс записывают в квадратных скобках. Это может быть число, имя переменной целого типа или арифметическое выражение, результат которого — целое число.

Перед использованием в программе массив необходимо создать, в противном случае обращение к несуществующему элементу вызовет ошибку и аварийное завершение программы.

Заполнять массив можно либо вводя значение каждого элемента с клавиатуры, либо присваивая элементам некоторые значения в программе. При заполнении массива и выводе его элементов на экран используется цикл с параметром, который изменяется от минимального до максимального значения индекса.

При решении разнообразных задач, связанных с обработкой массивов, используются такие типовые алгоритмы, как: суммирование элементов массива; поиск элемента с заданными свойствами; сортировка массива.

Для вычисления суммы элементов массива используется переменная, в которой накапливается сумма. Начальное значение этой переменной равно нулю.

При подсчёте элементов, удовлетворяющих условию, используется переменная, которая увеличивается на 1 каждый раз, когда найден новый подходящий элемент. Начальное значение этой переменной равно нулю.

## Вопросы и задания

1. Объясните разницу между понятиями «индекс элемента массива» и «значение элемента массива».
2. Может ли массив одновременно содержать целые и вещественные значения?
3. Для чего предназначены массивы?
4. Запишите в следующем формате:

`A[<индекс>] = <значение элемента массива>`

значения элементов массивов, сформированных следующим образом:

а) `for i in range(7): A[i] = 1`

б) `for i in range(7): A[i] = i`

в) `for i in range(7): A[i] = i * i - 4`

г) `A=[3, 4, -1, 5, 0, 10, -12]`

д) `for i in range(8):`

`if i % 2 == 0: A[i] = i / 2`

`else: A[i] = 0`

5. Что вы можете сказать о массиве, сформированном следующим образом?

а) `for i in range(10): A[i] = random.randint(-50, 50)`

б) `for i in range(20): A[i] = i`

в) `for i in range(0, 5): A[i] = 2 * i - 1`

6. Чему равна сумма элементов массива `A[0]` и `A[5]`, сформированного следующим образом?

| Алгоритмический язык                                 | Python                                            |
|------------------------------------------------------|---------------------------------------------------|
| <pre>нц для i от 0 до 9   A[i] := i * i - 5 кц</pre> | <pre>for i in range(10):   A[i] = i * i - 5</pre> |

7. Известны значения элементов одномерного целочисленного массива А, состоящего из 5 элементов:

|             |   |   |    |   |   |
|-------------|---|---|----|---|---|
| <b>i</b>    | 0 | 1 | 2  | 3 | 4 |
| <b>A[i]</b> | 4 | 1 | -5 | 7 | 2 |

Чему равно значение A[A[4]]?

8. Чему равно среднее арифметическое значение элементов массива A[3] и A[4], сформированного следующим образом?

| Алгоритмический язык                                                                                                           | Python                                                                                                                    |
|--------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------|
| <pre> нц для i от 0 до 9   если mod(i, 2)=0     то A[i] := i / 2   иначе A[i] := (i + 1) / 2 все кц                     </pre> | <pre> for i in range(10):     if i % 2 == 0:         A[i] = i / 2     else: A[i] = (i + 1) / 2                     </pre> |

9. Найдите сумму значений элементов A[1] и A[4] массива, сформированного следующим образом:

```
for i in range(6): A[i] = i * (i + 1)
```

10. Массив задан следующим образом:

```
B = [2, 1, 2, 3, 5, 11].
```

Найдите значение выражения

```
B[5] * B[4] - B[2] - B[3] * B[1].
```

11. Массив А из десяти элементов сформирован следующим образом:

| Алгоритмический язык                                                   | Python                                                                |
|------------------------------------------------------------------------|-----------------------------------------------------------------------|
| <pre> нц для i от 0 до 9   A[i] := i * i кц                     </pre> | <pre> for i in range(10):     A[i] = i * i                     </pre> |

К данному массиву был применён следующий алгоритм:

| Алгоритмический язык                                                                              | Python                                                                                      |
|---------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------|
| <pre> b := A[9] нц для i от 0 до 8   A[9 - i] := A[8 - i] кц A[0] := b                     </pre> | <pre> b = a[9] for i in range(9):   A[9 - i] = A[8 - i] A[0] = b                     </pre> |

Чему равно значение седьмого элемента обработанного массива?

12. Массив A из десяти элементов сформирован следующим образом:

| Алгоритмический язык                                                    | Python                                                               |
|-------------------------------------------------------------------------|----------------------------------------------------------------------|
| <pre> нц для i от 0 до 9   A[i] := 11 - i кц                     </pre> | <pre> for i in range(10):   A[i] = 11 - i                     </pre> |

К данному массиву был применён следующий алгоритм:

| Алгоритмический язык                                                      | Python                                                                |
|---------------------------------------------------------------------------|-----------------------------------------------------------------------|
| <pre> нц для i от 0 до 8   A[i + 1] := A[i] кц                     </pre> | <pre> for i in range(9):   A[i + 1] = A[i]                     </pre> |

Чему равно среднее арифметическое значений элементов обработанного массива?

13. Определите, что будет выведено в результате выполнения следующей программы:

а) 

```

A = [1, 7, 3, 6, 0, 10]
s = 0
for i in range(6):
 s = s + a[i]
sr = s / 6
print('sr= ', sr)

```

б) 

```

A = [0]*7
for i in range(7):
 A[i] = i * 3
k = 0
for i in range(7):
 if A[i] > 10: k += 1
print('k= ', k)

```



Запишите соответствующую программу на языке программирования Python.

15. Установите соответствие между записанными на языке Python фрагментами программ обработки одномерного массива и результатами их работы.

- |                                                                                    |                                                                 |
|------------------------------------------------------------------------------------|-----------------------------------------------------------------|
| а) <pre>y = 0 for i in range(10):     if A[i] == 0:         y += 1</pre>           | 1. Произведение ненулевых элементов массива                     |
| б) <pre>s = 0 for i in range(15):     if A[i] % 2 == 0:         s += a[i]</pre>    | 2. Значение наибольшего элемента массива                        |
| в) <pre>n = 0 for i in range(1, 10):     if A[i] &lt; a[n]:         n = i</pre>    | 3. Сумма всех элементов массива с чётными номерами              |
| г) <pre>k = 0 for i in range(100):     if A[i] % 5 == 0:         k += 1</pre>      | 4. Номер (индекс) минимального элемента массива                 |
| д) <pre>m = a[0] for i in range(1, 10):     if A[i] &gt; m:         m = A[i]</pre> | 5. Количество всех элементов массива, значение которых кратно 5 |
| е) <pre>z = 1 for i in range(10):     if A[i] &lt;&gt; 0:         z *= A[i]</pre>  | 6. Количество ненулевых элементов массива                       |

16. В массиве `Dat` хранятся данные измерений среднесуточной температуры за неделю в градусах (`Dat[1]` — данные за понедельник, `Dat[2]` — данные за вторник и т. д.). Определите, что будет выведено в результате работы программы, записанной на алгоритмическом языке. Запишите эту программу на языке Python.

Алгоритмический язык:

**алг**

**нач**

```

целтаб Dat[1..7]
цел m, k
Dat[1] = 12; Dat[2] = 14;
Dat[3] = 13; Dat[4] = 15;
Dat[5] = 15; Dat[6] = 12;
Dat[7] = 16
m := 0
нц для k от 1 до 7
 если Dat[k] > 14 то
 m := m + 1
 все
кц
вывод m

```

**кон**

17. Напишите программу, которая вычисляет среднюю температуру воздуха за неделю с точностью до двух знаков после запятой. Исходные данные вводятся с клавиатуры.

| Пример входных данных                                                                                                        | Пример выходных данных                  |
|------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------|
| Введите температуру:<br>Понедельник 12<br>Вторник 10<br>Среда 16<br>Четверг 18<br>Пятница 17<br>Суббота 16<br>Воскресенье 14 | Средняя температура за<br>неделю: 14.71 |

18. Дан массив из десяти целых чисел. Напишите программу подсчёта количества элементов этого массива, имеющих максимальное значение.
19. В классе 20 учеников писали диктант по русскому языку. Напишите программу, подсчитывающую количество двоек, троек, четвёрок и пятёрок, полученных за диктант.

20. Объявлен набор в школьную баскетбольную команду. Известен рост каждого из  $N$  учеников, желающих попасть в эту команду. Составьте алгоритм подсчёта количества претендентов, имеющих шанс попасть в команду, если рост игрока команды должен быть не менее 170 см. Запишите программу на языке Python. Считайте рост претендента в команду случайным числом из диапазона от 150 до 200 см, а число претендентов  $N = 50$ .
21. В целочисленных массивах  $A$  и  $B$  содержатся длины катетов десяти прямоугольных треугольников ( $A[i]$  — длина первого катета,  $B[i]$  — длина второго катета  $i$ -го треугольника). Найдите треугольник с наибольшей площадью. Выведите его номер, длины катетов и площадь. Предусмотрите случай, когда таких треугольников несколько.
22. Занесите информацию о десяти европейских странах в три массива:  $Name$  (название стран),  $K$  (численность населения),  $S$  (площадь страны). Выведите названия стран в порядке возрастания плотности их населения.

### Проверочная работа № 6

1. Дан одномерный массив  $A$  из шести элементов:

|      |     |    |   |    |    |
|------|-----|----|---|----|----|
| -125 | 200 | 10 | 5 | 43 | 11 |
|------|-----|----|---|----|----|

- 1) Как можно создать этот массив в программе, записанной на языке Python?
  - 2) Чему равно значение элемента массива с индексом 4?
  - 3) Чему равно значение элемента массива  $A[A[3]]$ ?
2. Программа обрабатывает одномерный целочисленный массив  $Dat$ :

|          |    |    |    |     |    |    |    |    |    |    |
|----------|----|----|----|-----|----|----|----|----|----|----|
| $i$      | 0  | 1  | 2  | 3   | 4  | 5  | 6  | 7  | 8  | 9  |
| $Dat[i]$ | 70 | 80 | 90 | 100 | 80 | 40 | 40 | 70 | 80 | 90 |

Заполните в тетради трассировочную таблицу и определите, какие числа будут выведены в результате выполнения следующего фрагмента программы.



```

k = 1
m = Dat[k]
for i in range(10):
 if Dat[i] < m:
 m = Dat[i]
 k = i
print('m=', m)
print('k=', k)

```

| k   | m   | i   | dat[i] < m |
|-----|-----|-----|------------|
| ... | ... | ... | ...        |

3. Программисту было поручено написать программу нахождения суммы отрицательных элементов одномерного целочисленного массива. Программист разработал программу, но допустил в ней одну ошибку. Текст программы с ошибкой представлен ниже:

```

for i in range(8):
 A[i] = int(input())
s = 0;
for i in range(8):
 if A[i] < 0:
 s = s + A[1];
print(s)

```

Каким окажется ответ после выполнения этой программы, если в качестве элементов массива будут введены числа

1, 2, 3, 4, 0, -1, -2, -3, -4?

Как исправить программу, чтобы она решала поставленную перед программистом задачу?

## § 2.2

### Запись вспомогательных алгоритмов на языке Python

**Ключевые слова:**

- подпрограмма
- процедура
- функция
- рекурсивная функция

Запись вспомогательных алгоритмов в языках программирования осуществляется с помощью подпрограмм.

В Python различают два вида подпрограмм: **процедуры** и **функции**.

### 2.2.1. Процедуры

**Процедура** — это подпрограмма, выполняющая некоторые действия; она может иметь произвольное количество входных параметров.

Описание процедуры имеет вид:

```
def <имя процедуры> () :
 <операторы>
```

Процедура начинается со служебного слова **def** (от англ. *define* — определить). После этого записывается имя процедуры, скобки и двоеточие.

Операторы, которые входят в тело процедуры, записываются с отступом. Так мы показываем, какие команды входят в процедуру.

Для того чтобы процедура заработала, её необходимо вызвать по имени; причём таких вызовов может быть сколько угодно.

Процедура должна быть определена к моменту её вызова, т. е. должна быть выполнена команда **def**, которая создаёт объект-процедуру в памяти. Если процедура вызывается из основной программы, то нужно поместить определение процедуры раньше точки её вызова.

**Пример.** Предположим, что требуется вывести четыре строки, каждая из которых состоит из семи единиц. Для этого создадим процедуру, которая выполняет вывод одной строки из семи единиц, и вызовем её четыре раза.

```
def digit() :
 print(1111111)
print('Четыре строки из семи единиц')
digit()
digit()
digit()
digit()
```

Предположим, что требуется вывести четыре строки, состоящие из семи, восьми, девяти и десяти единиц соответственно.



Для этого создадим процедуру с параметром, определяющим длину строки.

Чтобы вывести единицу  $n$  раз воспользуемся командой:

```
print('1' * n)
```

Чтобы использовать эту команду в процедуре, укажем в скобках переменную (параметр), значение которой и будет определять длину строки.

```
def digit(n):
 print('1' * n)
```

Чтобы вывести строку требуемой длины, нужно вызвать процедуру, указав в скобках значение параметра  $n$ , т. е. количество символов '1':

```
def digit (n):
 print('1' * n)
digit(7) #выводит 7 единиц: 1111111
digit(8) #выводит 8 единиц: 11111111
digit(9) #выводит 9 единиц: 111111111
digit(10) #выводит 10 единиц: 1111111111
```

Процедура может зависеть от нескольких параметров. Давайте немного улучшим процедуру `digit()`: сделаем так, чтобы можно было изменять не только длину строки, но и цифры, из которых она строится. Для этого добавим в процедуру ещё один параметр, который назовём  $d$ :

```
def digit(d, n):
 print(d * n)
```

Обозначение переменных, значения которых мы вводим с клавиатуры, не обязательно должно совпадать с обозначением параметров процедуры. Мы могли бы назвать их любыми другими именами, например:

```
def digit(d, n):
 print(d * n)
x=(input('Введите цифру:'))
y=int(input('Введите длину строки:'))
digit(x, y)
```

Переменные  $d$  и  $n$  — это локальные переменные; они определены и используются только внутри процедуры `digit()`. Обращаться к ним вне этой процедуры нельзя. Как только работа

процедуры будет закончена, все её локальные переменные удалятся из памяти.

В тех случаях, когда значение переменной, полученное в подпрограмме, должно быть использовано в основной программе, эту переменную следует объявить как глобальную.

**Пример.** Напишем процедуру для нахождения наибольшего общего делителя двух чисел с помощью алгоритма Евклида. Используем её для нахождения наибольшего общего делителя следующих шести чисел: 16, 32, 40, 64, 80 и 128.

```
def nod (a, b):
 global x
 while a != b:
 if a > b:
 a = a - b
 else:
 b = b - a
 x = a
```

```
m = [16, 32, 40, 64, 80, 128]
x = m[0]
for i in range(1, 6):
 y = m[i]
 nod(x, y)
print('НОД=', x)
```

Процедура

Основная программа

Вызов процедуры

- Измените программу так, чтобы с её помощью можно было найти:
- а) наибольший общий делитель следующих пяти чисел: 12, 24, 30, 48 и 51;
  - б) наибольший общий делитель произвольных десяти целых двузначных чисел.



### 2.2.2. Функции

**Функция** — это подпрограмма, имеющая единственный результат, записываемый в ячейку памяти. В отличие от процедуры, функция не только выполняет какие-то команды, но и возвращает результат в виде числа, символьной строки или др.



Описание функции имеет вид:

```
def <имя функции>():
 <операторы>
 return <результат>
```

Функция начинается со служебного слова **def**. После этого записывается имя функции, скобки и двоеточие. Операторы, которые входят в тело функции, записываются с отступом. После оператора **return** записывается результат, который возвращает функция.

В языке Python есть встроенная функция `max()`, вычисляющая максимальное значение из нескольких аргументов.

**Пример.** Запишем функцию, которая возвращает значение наибольшего из двух целых чисел.

```
def max(a, b):
 if a > b:
 m = a
 else:
 m = b
 return m
```

Результат функции можно сразу вывести на экран:  
`print(max(6, 8))`

Также мы можем присвоить результат работы функции любой глобальной переменной: `x = max(6, 8)`. Одна функция может вызывать другую.

**Пример.** Напишем программу нахождения максимального из четырёх целых чисел, использующую функцию поиска максимального из двух чисел:

```
def max(a, b):
 if a > b:
 m = a
 else:
 m = b
 return m
a, b, c, d = map(int, input().split())
f = max(max(a, b), max(c, d))
print('f=', f)
```

Измените программу так, чтобы с её помощью можно было найти:

- а) максимальное из чисел  $a, b, c$ ;
- б) максимальное из чисел  $b, c, d$ ;
- в) минимальное из четырёх чисел;
- г) разность максимального и минимального из четырёх чисел.

**Пример.** В январе Саше подарили пару новорождённых кроликов. Через два месяца они дали первый приплод — новую пару кроликов, а затем давали приплод по паре кроликов каждый месяц. Каждая новая пара также даёт первый приплод (пару кроликов) через два месяца, а затем — по паре кроликов каждый месяц. Сколько пар кроликов будет у Саши в декабре?

Составим математическую модель этой задачи. Обозначим через  $f(n)$  количество пар кроликов в месяце с номером  $n$ . По условию задачи,  $f(1) = 1$ ,  $f(2) = 1$ ,  $f(3) = 2$ . Из двух пар, имеющих в марте, дать приплод в апреле сможет только одна:  $f(4) = 3$ . Из пар, имеющих в апреле, дать приплод в мае смогут только пары, родившиеся в марте и ранее:  $f(5) = f(4) + f(3) = 3 + 2 = 5$ . В общем случае:  $f(n) = f(n - 1) + f(n - 2)$ ,  $n \geq 3$ .

Числа 1, 1, 2, 3, 5, 8, ... образуют так называемую последовательность Фибоначчи, названную в честь итальянского математика, впервые решившего соответствующую задачу ещё в начале XIII века.

Оформим в виде функции вычисление члена последовательности Фибоначчи.

```
def f(n):
 if n == 1 or n == 2:
 rez = 1
 else: rez = f(n - 1) + f(n - 2)
 return rez
```

Полученная функция — **рекурсивная**; в ней реализован способ вычисления очередного значения функции через вычисление её предшествующих значений.

Напишите программу, вычисляющую и выводящую 10 первых членов последовательности Фибоначчи.



## САМОЕ ГЛАВНОЕ

Запись вспомогательных алгоритмов в языках программирования осуществляется с помощью подпрограмм. В Python различают два вида подпрограмм: процедуры и функции.

Процедура — это подпрограмма, выполняющая некоторые действия, она может иметь произвольное количество входных параметров. Процедура не возвращает результат, который можно присвоить переменной, а только обрабатывает входные данные.

В процедуру могут быть переданы глобальные переменные, которые она обрабатывает, изменяет.

В отличие от процедуры, функция не только выполняет какие-то команды, но и возвращает результат в виде числа, символической строки или др.

Вызов функции можно использовать в арифметических выражениях и условиях так же, как и переменную того типа, который возвращает функция. В теле любой подпрограммы можно вызывать другие функции и процедуры. Рекурсивная функция — это функция, которая вызывает сама себя, напрямую или через другие процедуры и функции.

## Вопросы и задания

1. Для чего используются подпрограммы?
2. Дан текст процедуры на языке Python:

```
def f(n):
 if n > 1: f(n // 2)
 print('**')
```

Определите, сколько звёздочек будет выведено в результате вызова `f(7)`. Вычисления фиксируйте в таблице:

|      |   |   |   |   |   |   |   |
|------|---|---|---|---|---|---|---|
| n    | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| f(n) |   |   |   |   |   |   |   |

3. Напишите процедуру с параметрами `n` (целое число) и `a` (символ), которая выводит на экран `n` строк, каждая из которых содержит `n` символов `a`.
4. Напишите процедуру с параметрами `w` (ширина), `h` (высота), `a` (символ), которая выводит на экран «прямоугольник» из символов `a`, ширина которого равна `w`, а высота — `h`.
5. В чём основное различие процедур и функций?
6. Напишите функцию `KDN()`, которая вычисляет количество цифр вводимого целого числа.
7. Напишите функцию `KBDN()`, которая вычисляет количество цифр в двоичной записи вводимого десятичного числа.

8. Напишите программу вычисления наименьшего общего кратного следующих четырёх чисел: 36, 54, 18 и 15. Используйте процедуру вычисления наибольшего общего делителя двух чисел.
9. Напишите программу перестановки значений переменных  $a$ ,  $b$ ,  $c$  в порядке возрастания, т. е. так, чтобы  $a < b < c$ . Используйте функцию `swap()`. Исходные данные вводятся с клавиатуры.

| Пример входных данных | Пример выходных данных |
|-----------------------|------------------------|
| 1 2 3                 | 1 2 3                  |
| 2 1 3                 | 1 2 3                  |
| 3 1 2                 | 1 2 3                  |
| 2 3 1                 | 1 2 3                  |

10. Видоизмените программу сортировки массива выбором так, чтобы в ней использовалась процедура выбора наибольшего элемента массива.
11. Напишите программу вычисления выражения
 
$$s = 1! + 2! + 3! + \dots + n!$$
 Здесь  $n!$  — факториал числа  $n$ .  $n! = 1 \cdot 2 \cdot \dots \cdot (n - 1) \cdot n$ . Используйте функцию вычисления факториала.
12. Напишите программу вычисления выражения
 
$$s = x^3 + x^5 + x^n,$$
 где  $x$  и  $n$  вводятся с клавиатуры. Используйте подпрограмму вычисления степени.
13. Напишите функцию, вычисляющую длину отрезка по координатам его концов. Напишите программу, вычисляющую периметр треугольника по координатам его вершин с помощью этой функции.
14. Напишите функцию, вычисляющую площадь треугольника по целочисленным координатам его вершин. Напишите программу вычисления площади четырёхугольника по координатам его вершин с помощью этой функции.



# Оглавление

|                                                                              |    |
|------------------------------------------------------------------------------|----|
| <b>Глава 1. НАЧАЛА ПРОГРАММИРОВАНИЯ</b> .....                                | 3  |
| § 1.1. Общие сведения о языке программирования Python .....                  | 3  |
| 1.1.1. Алфавит и словарь языка .....                                         | 4  |
| 1.1.2. Типы данных, используемые в языке Python .....                        | 6  |
| 1.1.3. Режимы работы интерпретатора Python .....                             | 6  |
| 1.1.4. Оператор присваивания .....                                           | 8  |
| § 1.2. Организация ввода и вывода данных .....                               | 13 |
| 1.2.1. Вывод данных .....                                                    | 13 |
| 1.2.2. Первая программа на языке Python .....                                | 15 |
| 1.2.3. Ввод данных с клавиатуры .....                                        | 17 |
| § 1.3. Программирование линейных алгоритмов .....                            | 23 |
| 1.3.1. Числовые типы данных .....                                            | 24 |
| 1.3.2. Целочисленный тип данных .....                                        | 26 |
| 1.3.3. Строковый тип данных .....                                            | 27 |
| 1.3.4. Логический тип данных .....                                           | 28 |
| § 1.4. Программирование разветвляющихся алгоритмов .....                     | 36 |
| 1.4.1. Условный оператор .....                                               | 36 |
| 1.4.2. Многообразие способов записи ветвлений .....                          | 38 |
| § 1.5. Программирование циклических алгоритмов .....                         | 47 |
| 1.5.1. Программирование циклов с известным условием продолжения работы ..... | 47 |
| 1.5.2. Программирование циклов с известным условием окончания работы .....   | 48 |
| 1.5.3. Программирование циклов с известным числом повторений .....           | 49 |
| 1.5.4. Различные варианты программирования циклического алгоритма .....      | 50 |
| Тестовые задания для самоконтроля за курс 8 класса .....                     | 62 |
| <b>Глава 2. АЛГОРИТМИЗАЦИЯ И ПРОГРАММИРОВАНИЕ</b> ...                        | 67 |
| § 2.1. Одномерные массивы целых чисел .....                                  | 67 |
| 2.1.1. Обращение к элементу массива .....                                    | 68 |
| 2.1.2. Заполнение массива .....                                              | 69 |
| 2.1.3. Вывод массива .....                                                   | 70 |
| 2.1.4. Вычисление суммы элементов .....                                      | 71 |
| 2.1.5. Последовательный поиск в массиве .....                                | 74 |
| 2.1.6. Сортировка массива .....                                              | 78 |
| § 2.2. Запись вспомогательных алгоритмов на языке Python .....               | 88 |
| 2.2.1. Процедуры .....                                                       | 89 |
| 2.2.2. Функции .....                                                         | 91 |