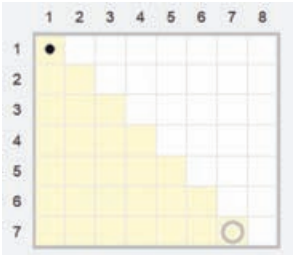
5* Решите задачу upr_16_5, используя вложенные циклы. Размер поля заранее не известен.



6* Придумайте задачу для исполнителя *Робот*, в которой будут использоваться различные алгоритмические конструкции.

§ 17. Язык программирования Python

Компьютер (англ. computer — *вычислитель*) — устройство (или система), способное выполнять заданную чётко определённую изменяемую последовательность операций. Чаще всего это операции численных расчётов.

Гвидо ван Россум приступил к созданию языка Python в декабре 1989 г. в центре математики и информатики в Нидерландах. Ван Россум является основным автором Python. Он продолжал выполнять центральную роль в принятии решений относительно развития языка вплоть до 12 июля 2018 г.

Желание упростить расчёты присуще человеку с древних времён. Создавая различные приспособления для счёта, человек прошёл долгий путь. Сегодняшний компьютер способен выполнять сотни миллионов операций в секунду. Для решения вычислительных задач требуется сначала составить алгоритм, а затем записать его в виде программы, используя какой-либо язык программирования.

Язык программирования устанавливает набор правил, определяющих внешний вид программы и действия, которые выполнит исполнитель под её управлением.

Сегодня язык Python является одним из самых популярных и востребованных языков программирования. Существует большое количество сред программирования с поддержкой языка Python: IDLE, PyCharm, Spyder, Thonny, Eclipse + PyDev, Visual Studio и др. В учебном курсе используется среда IDLE, с которой вы работали, знакомясь с учебными компьютерными исполнителями.

17.1. Команда вывода

Демонстрировать работу любой программы имеет смысл только тогда, когда она выводит какую-либо информацию.

Пример 17.1. Написать программу для вывода слов «Привет, мир!».

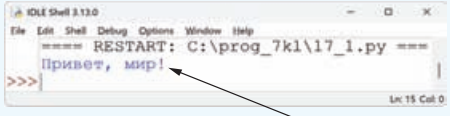
```
print('Привет, мир!')
```

Результат работы программы отражается в главном окне среды IDLE.

Команда `print()` предназначена для **вывода данных**. Она является функцией, аргументы которой — выводимые значения.

Аргументами функции `print` могут быть разделители (пример 17.2).

Пример 17.1. Окно среды IDLE с результатом работы программы.



Результат работы программы

По традиции, начавшейся в 1978 г. с примера из книги Б. Кернигана и Д. Ритчи «Язык программирования Си», первая программа на любом языке программирования должна просто выводить на экран приветствие миру.

Пример 17.2. Аргументы команды `print()` для управления разделителями.

`sep` — задаёт текст, который будет служить разделителем при выводе различных значений в одной команде `print()`. По умолчанию — пробел.

`end` — задаёт текст, который появится на экране после вывода всех значений одной команды `print()`. По умолчанию — перевод строки.

В программе перевод строки задаётся как `'\n'`. Если использовать `sep = '\n'`, то каждое выводимое значение одной команды `print()` будет расположено в новой строке.

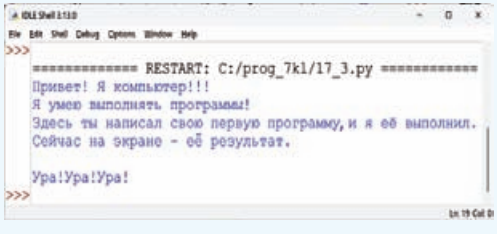
Аргументы, управляющие выводом, могут использоваться в команде `print()` по одному или оба сразу.

У функции print есть ещё два необязательных аргумента: file (определяет, куда выводятся данные, по умолчанию — на экран) и flush (определяет, нужно ли принудительно очищать поток вывода, по умолчанию — false).

Пример 17.3. Текст программы.

```
print('Привет!',
      'Я компьютер!!!')
print('Я умею выполнять',
      end = ' ')
print('программы!', 'Здесь',
      sep = '\n', end = ' ')
print('ты', 'написал свою',
      end = ' ')
print('первую программу,',
      end = ' ')
print('и я её выполнил.')
print('Сейчас на экране',
      end = ' - ')
print('её результат.')
print()
print('Ура!', 'Ура!', 'Ура!',
      sep = '!')
```

Результат работы программы:



Текст, который нужно вывести на экран, заключают в кавычки. Кавычки могут быть одинарными или двойными. Текст выводится на экран в том виде, в котором он записан в программе. Его можно записать как на русском, так и на любом другом языке. Текстом может быть произвольный набор символов.

В программе может быть несколько команд вывода. После выполнения команды вывода курсор переводится на следующую строку. В одной команде можно указать вывод нескольких значений, которые отделяются друг от друга запятыми. Если функция print() используется без аргументов, то будет выведена пустая строка.

Пример 17.3. Вывести на экран компьютера следующий текст, используя различные значения разделителей: «Привет! Я компьютер!!! Я умею выполнять программы! Здесь ты написал свою первую программу, и я её выполнил. Сейчас на экране — её результат. Ура! Ура! Ура!».

Используя сочетания аргументов sep и end, можно управлять размещением текста на экране.

Текст в команде print(), записанный в кавычках, не анализируется. Если кавычки опустить, то производится анализ тех данных, которые записаны в скобках. Например, если в скобках написать арифметическое выражение, то сначала вычисляется его значение, а затем выводится результат.

Пример 17.4. Посчитать значение выражения $2 + 2 \cdot 2$.

Если в команде print() записать выражение в кавычках, то на экран будет выведено само выражение, при отсутствии кавычек — значение данного выражения.

Пример 17.4. Текст программы.

```
print('2 + 2 * 2 =')
print(2 + 2 * 2)
```

Результат работы программы:

```
2 + 2 * 2 =
6
```

Для вывода результата в одной строке программу нужно записать так:

```
print('2 + 2 * 2 =', end = ' ')
print(2 + 2 * 2)
```

Или так:

```
print('2 + 2 * 2 =',2 + 2 * 2)
```

Результат работы программы:

```
2 + 2 * 2 = 6
```

17.2. Понятие типа данных

На практике редко приходится писать программы, которые решают только одну задачу. Обычно программы создаются для решения целого класса задач, которые можно сформулировать в общем виде.

С такими задачами вы уже сталкивались в курсе математики. Например, решение задачи «Найти площадь прямоугольника» можно записать так: $S = a \cdot b$, где переменные a и b обозначают соответственно длину и ширину прямоугольника, а S — площадь.

Программисты ЭВМ до начала 1950-х гг. при создании программ пользовались машинным кодом. Запись программы на машинном коде состояла из единиц и нулей. Типы данных не использовались. Машинный код принято считать языком программирования первого поколения.

Первым языком программирования, в котором появилась возможность создавать переменные, считается Ассемблер. В этом языке вместо машинного кода стали использовать команды, записанные текстом. Ассемблер относится к языкам программирования второго поколения.

Зная эту формулу, можно найти площадь любого прямоугольника.

В программировании для решения задач в общем виде также используют переменные. Для работы компьютера с переменными они должны храниться в его памяти.

Информацию, представленную в формализованном виде и пригодную для обработки на компьютере, называют **данными.**

Переменная в программировании — это именованная ячейка памяти, хранящая значение переменной.

Компьютер может обрабатывать данные разных типов: целые и рациональные числа, строки и др.

Тип данных определяет способ их хранения в памяти компьютера, диапазон возможных значений и операции, которые можно выполнять с этим типом данных.

Каждая переменная задаётся своим именем. Для обозначения имени переменной используют буквы латинского алфавита, арабские цифры и знак «_».

В 1957 г. появился язык Фортан, открывший эру языков программирования третьего поколения. Он позволил использовать разные числовые типы данных, необходимые для сложных расчётов: целые, вещественные (действительные) и комплексные (комплексные числа изучаются на факультативах по математике в старших классах).

Дальнейшее развитие языков программирования позволило добавить возможность работы с другими типами данных. Нынешние языки программирования поддерживают возможность работы с большим количеством разных типов данных.

Пример 17.5. Числовые типы данных.

Язык Python поддерживает работу с целыми и рациональными числами. Значения переменных могут определяться так же, как в математике.

```
a = 2
b_1 = 7.5
radius = 12.0
```

Переменная `a` будет определена как целая, а переменные `b_1` и `radius` — как рациональные. Целая часть дробного числа отделяется от дробной части точкой.

Заглавные и строчные буквы считаются различными.

При использовании переменной выделяется память для хранения значения этой переменной. В процессе выполнения программы значение переменной может изменяться. Тип переменной в языке Python определяется по её значению (пример 17.5 на с. 124). Это происходит в процессе выполнения программы.

17.3. Оператор присваивания

Одной из основных команд для обработки данных в программе является оператор присваивания.

Оператор присваивания предназначен для того, чтобы:

- задавать значения переменных;
- вычислять значение арифметического выражения, результат вычисления которого будет записан как значение переменной.

Формат записи оператора присваивания:

<имя переменной> = <выражение>

В записи арифметического выражения используются знаки математических действий: сложения, вычитания, умножения, деления

Пример 17.5. *Продолжение.*

Задавать значения можно и таким образом:

```
a, b_1, radius = 2, 7.5, 12.0
```

Чтобы убедиться, что переменные получили указанные значения, выведем их с помощью команды print():

```
print(a, b_1, radius)
```

Результат работы программы:

```
2  7.5  12.0
```

Пример 17.6. Запись оператора присваивания.

Запись оператора присваивания в программе и вывод значений переменных:

```
x = 9
x1 = 3.5
a_1 = 20 * (x + x1) - 32
y = 3
  chastnoe = x / y
y = 7 + 2 * y ** 3
print('x =', x)
print('x1 =', x1)
print('a_1 =', a_1)
print('chastnoe =',
      chastnoe)
print('y =', y)
```

Результат работы программы:

```
x = 9
x1 = 3.5
a_1 = 218.0
chastnoe = 3.0
y = 61
```

Если в выражении встречаются переменные разных числовых типов (как целых, так и рациональных), то результат будет рациональным числом (значение переменной a_1).

Пример 17.7. Запись оператора присваивания на Python для математических выражений:

Выражение	Запись на Python
$S = 2(a + b)$	S = 2 * (a + b)
$S = a^2$	S = a * a или S = a ** 2
$a = \frac{x + y}{3}$	a = (x + y) / 3

Пример 17.8*. Изменение значения переменной.

В Python допустимы команды присваивания следующего вида:

```
a = a * 2 или a *= 2
```

Смысл команды следующий: из ячейки памяти извлекается значение переменной a, затем оно умножается на 2, результат записывается в ту же ячейку памяти. Старое значение переменной a будет потеряно.

Запись оператора присваивания для изменения значения переменной b следующая:

```
b = b – 3 или b -= 3
```

и возведение в степень. В языке Python им соответствуют следующие символы:

Математические операции	Запись в Python
+ (сложение)	+
– (вычитание)	–
· (умножение)	*
: (деление)	/
возведение в степень	**

Приоритет выполнения операций соответствует принятому в математике: сначала выполняется возведение в степень, затем умножение и деление, далее — сложение и вычитание. В выражениях для изменения порядка действий можно использовать скобки (примеры 17.6 и 17.7).

Для записи обыкновенной дроби используется знак деления. Знак умножения опускать нельзя. Вычисления, производимые над целыми числами, всегда точные. При вычислениях с рациональными числами результат может быть приблизительным.

Пример 17.8*. Записать оператор присваивания, после выполнения которого значение переменной a увеличится в два раза, а переменной b уменьшится на 3.

17.4. Ввод данных

Начальные значения переменных можно задавать не только с помощью оператора присваивания, но и вводить с клавиатуры. В этом случае текст программы не нужно изменять для нового набора данных.

Команда `input()` предназначена для **ввода данных**. Для того чтобы значение переменной вводилось с клавиатуры, нужно присвоить ей значение функции `input()`.

При использовании в программе команды ввода данных в виде `a = input()` все символы, вводимые с клавиатуры, при выполнении команды воспринимаются как текст (в том числе и числа). Поэтому в команде ввода нужно указать, в какой числовой тип данных мы хотим преобразовать вводимый текст (пример 17.9).

В качестве параметра функции `input()` можно задать текст, который будет служить подсказкой при вводе данных (пример 17.10).

Ввод данных осуществляется в главном окне среды IDLE (пример 17.10). После завершения работы программы в этом же окне будет выведен результат.

Пример 17.9. Ввод чисел.

Для преобразования в целое число используется функция `int()`, а в рациональное число — функция `float()`.

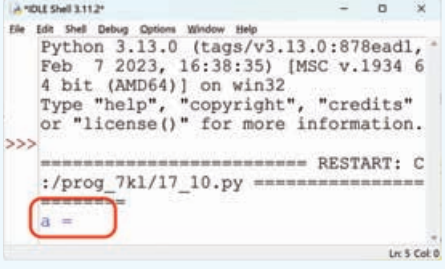
```
x = int(input('целое'))
y = float(input('рациональное'))
```

Пример 17.10. Ввести два рациональных числа, найти их сумму.

Текст программы:

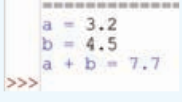
```
a = float(input('a = '))
b = float(input('b = '))
s = a + b
print('a + b =', s)
```

После запуска программы в главном окне среды IDLE появится строка `a =`.



В этой строке нужно набрать числовое значение. Обратите внимание на то, что программа выводит текст синим цветом, а значения, вводимые пользователем, отображаются чёрным.

Результат работы программы:



1. Какая команда языка программирования Python предназначена для вывода данных?
2. Как можно изменить разделитель при выводе данных?
3. Что такое переменная?
4. Что определяет тип данных?
5. Для чего используется команда присваивания?
6. Какая команда языка программирования Python предназначена для ввода данных?

Упражнения

1 Выполните следующие задания для программы из примера 17.3 на с. 122 (файл с программой можно скачать).

1. Замените все аргументы sep на end и выполните программу. Что произошло? Объясните почему.

2. Как изменится результат работы программы, если в исходном тексте заменить все аргументы end на sep?

3. Измените программу так, чтобы текст на экране выглядел следующим образом:

```
Привет! Я компьютер!!! Я умею выполнять программы!
Ты сегодня написал свою первую программу!!!
Я выполнил твою программу. Посмотри на результат!
```

2 Для примера 17.4 на с. 123 внесите изменения в программу так, чтобы действия выполнялись в том порядке, в котором записаны, то есть сначала сложение, а потом умножение.

3 Составьте программу для определения возраста пользователя через 5 лет и выведите результат. Возраст пользователя вводится в годах.

4* Составьте программу, которая позволяет ввести два числа *a* и *b*, затем первое число уменьшает в два раза, а второе увеличивает на 30. Выведите изменённые значения переменных.

5 Напишите программу для вычисления значения числового выражения.

1) $23 + 45 \cdot 11 - 15$. 2) $\frac{37 + 2^5}{41}$. 3) $\frac{5638 - 2347}{49} + \frac{12^3 \cdot (7 + 56)}{455}$.