Глава 3 НАЧАЛА ПРОГРАММИРОВАНИЯ

§3.1 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 будет компьютер. Записи алгоритмов на языках программирования называются программами.

Как и люди, компьютеры «говорят» на разных языках, только языки эти — компьютерные. Компьютерный язык служит для того, чтобы переговариваться с компьютером, используя команды, понятные и компьютеру, и человеку. Некоторые языки программирования названы в честь людей (например, Ада и Паскаль), другие названия являются простыми акронимами, то есть аббревиатурой (к примеру, BASIC — от англ. Beginner's All-purpose Symbolic Instruction Code, универсальный код символических инструкций для начинающих), и уж совсем немногие языки названы в честь телевизионных шоу — как Python. О да, язык программирования Python (произносится «Пайтон», с ударением на первый слог, хотя имейте в виду, что в России многие называют язык просто «питон») получил свое имя благодаря телешоу «Летающий цирк Монти Пайтона», так что змея питон здесь вовсе ни при чем.

Python – язык программирования высокого уровня, предназначенный для самого широкого круга задач. С его помощью можно обрабатывать различные данные, создавать изображения, работать с базами данных, разрабатывать Web-сайты.

Разработка языка Python была начата в конце 1980-х годов сотрудником голландского института CWI Гвидо ван Россумом. Опубликован исходный текст в феврале 1991 года.

Чтобы установить Python в системе Microsoft, откройте веб-браузер, введите адрес http://www.python.org/ и скачайте последнюю версию программы-установщика Python 3 для Windows (для этого зайдите в меню Downloads и выберите Windows).

3.1.1. 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);

русские прописные и строчные буквы (можно использовать, но это является очень плохим стилем);

арабские цифры (0, 1, 2, ..., 7, 8, 9);

специальные символы (знак подчёркивания; знаки препинания; круглые, квадратные скобки; знаки арифметических операций, # - знак комментарий и др.).

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

>= и <= (знаки ≥ и ≤);

""" и """ или ''' и ''' – утроенные кавычки или апострофы (начало и конец комментария).

В языке существует также некоторое количество различных цепочек символов, рассматриваемых как единые смысловые элементы с фиксированным значением. Такие цепочки символов называются **служебными словами**. В таблице 3.1 приведены основные служебные слова, которые мы будем использовать при записи программ на языке Python.

Для обозначения переменных, программ и других объектов используются **имена** — 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 мы будем использовать имена, передающие смысл объекта.

Таблица 3.1

Служебное слово языка Python	Значение служебного слова
and	и
break	прервать
else	иначе
False	ложь
float	вещественный (с плавающей точкой)
for	для
if	если
input	ввод
integer	целый
list	список
or	или
print	печать
string	строковый (цепочка символов)
True	истина
while	пока

3.1.2. Типы данных, используемые в языке Python

В языке Python используются различные типы данных. Мы будем пользоваться некоторыми из так называемых простых типов данных (табл. 3.2).

Таблица 3.2

Название	Обозначение	Допустимые значения	Пример
Целочисленный	int (integer)	сколь угодно большие, размер ограничен оперативной памятью	325
Вещественный	float	Любые числа с дробной частью	9.23 0.0 -1.7e-6
Строковый	str (string)	Любые символы из таблицы Unicode	"hello! " "x="
Логический	bool (boolean)	False и True	

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

Тип переменной определяется в тот момент, когда ей присваивается новое значение.

3.1.3. Первая программа на языке Python

Изучение языков программирования принято начинать с программы, выдающей надпись: «Привет, мир!». Посмотрим, как это будет выглядеть на Python:

Программа 0

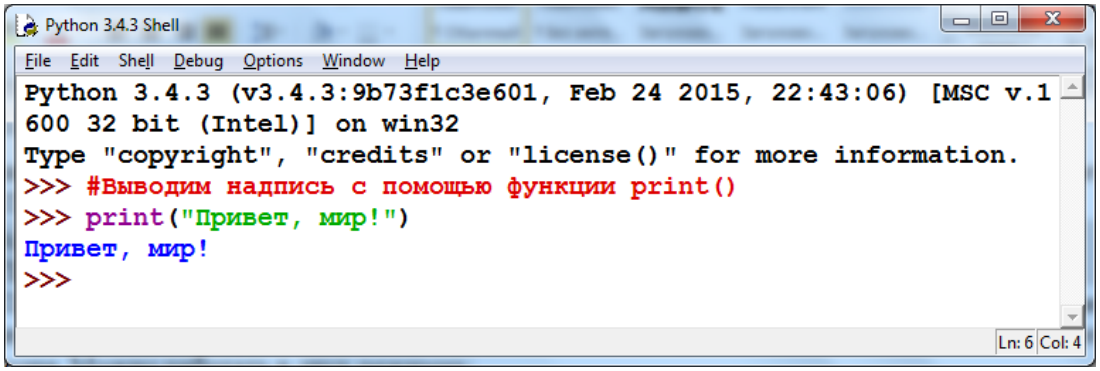
```
# Выводим надпись с помощью функции print()
print ("Привет, мир! ")
```

Программы на языке Python чаще всего выполняются *интерпретатором,* который читает очередную команду и сразу её выполняет, не переводя всю программу в машинный код конкретного процессора. 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**)

Для запуска программы выбираем в меню Пуск **Программы-Python 3.? -IDLE**. В результате откроется окно **PythonShell**, в котором символы >>> означают приглашение ввести команду. После ввода строки нажимаем клавишу Enter. На следующей строке сразу отобразиться результат, а далее приглашение для ввода новой команды. Последовательность выполнения показана ниже:



Примечание: символы >>> вводить не нужно, они вставляются автоматически

Для создания файла с программой в меню **File** выбираем пункт **NewFile.** В открывшемся окне набираем код Программы 0, а затем сохраняем его под именем test.py, выбрав пункт меню **File-SaveAs.** Запустить программу на выполнение можно, выбрав пункт меню **Run-RunModule** или нажав клавишу F5.

Существуют ресурсы для запуска и отладки программ на Pythononline. Например:

- ✓ http://pythontutor.com/visualize.html#mode=edit
- ✓ http://rextester.com/l/python3_online_compiler
- ✓ https://www.jdoodle.com/python3-programming-online
- ✓ https://ideone.com/

3.1.4. Оператор присваивания. Вычисления

Общий вид оператора:

<имя переменной> = <значение или вычисляемое выражение>

Операция присваивания допустима для всех приведённых в табл. 3.2 типов данных. Примеры:

```
a = 25
b = "Привет"
c = 1.4 + 5.7 * a
d = a < c
```

Выражения в языке Python конструируются по рассмотренным ранее правилам для алгоритмического языка.

Нельзя указывать в правой части выражения переменные, которые не были заранее созданы (определены). Так для переменных *c* и *d* все входящие переменные были заданы выше. Следующая строка ошибочна

```
f = x * (a + c) / 3
```

так как переменная *x* из правой части ранее не была создана.

В Python разрешено множественное присваивание.

Запись **a = b = 0** равносильна паре операторов **b = 0; a = 0**

Так же часто используют сокращенную запись арифметических операций:

сокращенная запись	*полная запись*
a += b	*a = a + b*
a -= b	*a = a - b*
*a *= b*	*a = a * b*
a /= b	*a = a / b*

САМОЕ ГЛАВНОЕ

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

Тип переменной определяется в тот момент, когда ей присваивается новое значение.

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**)

Вопросы и задания

1. Почему язык программирования Python считается универсальным?
2. Что входит в состав алфавита языка Python?
3. Каких требований следует придерживаться при выборе имён для различных объектов в языке Python?
4. В чём разница между числами 100 и 100.0 в языке Python?
5. Какой тип данных возможен для вычисления:
   а) значения функции $y = x^2$;
   б) площади прямоугольника;
   в) стоимости покупки, состоящей из нескольких тетрадей и такого же количества обложек;
   г) стоимости покупки, состоящей из нескольких тетрадей, нескольких ручек и нескольких карандашей.
6. Запишите оператор для:
   а) вычисления среднего арифметического переменных *x1* и *x2*;
   б) уменьшения на единицу значения переменной *k*;
   в) увеличения на единицу значения переменной *i*;
   г) вычисления стоимости покупки, состоящей из нескольких тетрадей, нескольких ручек и нескольких карандашей.

§3.2 Организация ввода и вывода данных

Ключевые слова:

- оператор вывода print
- формат вывода
- оператор ввода input

3.2.1. Вывод данных

В предыдущем параграфе мы познакомились с простой программой на языке Python, научились определять тип данных, 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 монитора используется оператор 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 переменные.

Произвольный набор символов, заключённый в апострофы или кавычки, считается строковой переменной. Строковая переменная может содержать любые символы, набираемые на клавиатуре.

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

на экран выводятся символы, заключённые в апострофы: *s=*

на экран выводится значение переменной с именем *s*.

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

Оператор print вставляет между выводимыми значениями так называемый *разделитель* (или сепаратор, англ. *separator).* По умолчанию разделитель – это пробел, но мы можем его изменить, указав новый разделитель после слова **`sep`**

Вариант организации вывода	Оператор вывода	Результат
По умолчанию	print (1, 20, 300)	1 20 300
Убрать разделители — пробелы	print (1, 20, 300, sep="")	120300
Добавить другой разделитель	print (1, 20, 300, sep=",")	1,20,300

Формат вывода — это указываемое общее количество знакомест, отводимое на число, определяющее, сколько позиций на экране должна занимать выводимая величина. 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.

Для вывода вещественного числа в списке вывода для каждого выражения указываются два параметра: 1) общее количество позиций, отводимых под число; 2) количество позиций в дробной части числа: (◦ - обозначение пробела)

d – целые числа (int)

f – вещественные (float)

e – экспоненциальный формат (см. стр 19 учебника)

Оператор вывода	Результат выполнения оператора
a = 4 print ("a=", "{:5d}{:5d}".format(a, a * a))	a= ◦◦◦◦4◦◦◦16

a = 1 / 3 b = 1 / 9 print ("{:7.3f}{:7.3f}".format(a, b))	◦◦0.333◦◦0.111
a=1/3 b=1/9 print ("{:10.3e}{:10.3e}".format(a, b))	◦3.333e-0.1◦◦1.111e-01

При выполнении нового оператора **print** вывод продолжается в новой строке. Чтобы убрать переход к новой строке, используется параметр **end**

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

3.2.2. Первая программа на языке Python

Пользуясь рассмотренными операторами, составим программу, вычисляющую длину окружности и площадь круга радиуса 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.

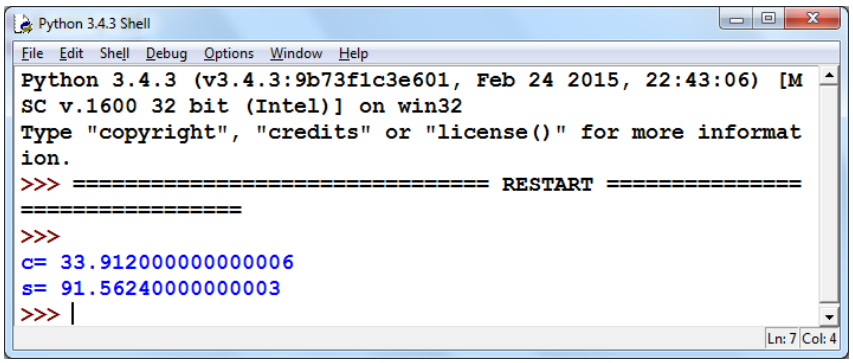
Исходные данные и результаты связаны соотношениями, известными из курса математики:

$c = 2\pi^2$, $s = \pi r^2$. Программа, реализующая вычисления по этим формулам, будет иметь вид:

Программа 1

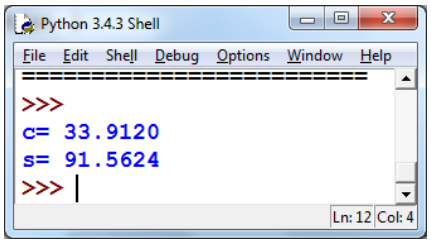
```
r = 5.4
c = 2 * 3.14 * r
s = 3.14 * r * r
print ('c=', c)
print ('s=', s)
```

Эта программа верна и решает поставленную задачу. Запустив её на выполнение, вы получите следующий результат:



Вывод по формату:

```
print ( "c=", "{:6.4f}".format (c))
print ( "s=", "{:6.4f}".format (s))
```



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.

3.2.3. Ввод данных с клавиатуры

Для ввода в оперативную память значений переменных используется оператор ввода **input**. Каждая функция **input()** захватывает только ОДНУ строку данных и воспринимает ее как ТЕКСТ (строковый тип):

a=input ()

Тип переменной можно преобразовать налету:

- для целых: a = int (input())
- для вещественных: a = float (input())

При выполнении функции **input** компьютер переходит в режим ожидания данных: пользователь должен ввести данные с клавиатуры и нажать клавишу Enter, система запишет это значение в переменную.

Есть задачи, в которых вводимые данные заданы в одной строке, например, 10 20. Чтобы была возможность работать с такими данными, их надо разделить.

Для этого нужно применить следующий синтаксис:

a, b = input().split()

Здесь есть правило: количество данных должно соответствовать количеству переменных. В примере выше в одной строке передаются два целых числа, разделенных пробелом. Соответственно, нужно эти данные передать также двум переменным. Осталось не забыть, что определенные таким образом переменные имеют строковый тип. Для арифметического сложения их нужно привести к типу целого числа.

a = int(a)

b = int(b)

или одной строкой

a, b = int(a), int(b)

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

В таких случаях надо для split() указывать конкретный символ разделителя. Разделителем будет символ ':', взятый в кавычки или апострофы.

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

Также добавим, что если в одной строке введены данные одного единственного типа (только целые числа, вещественные числа или строки), то определение переменных и задание им одного и того же типа можно выполнить одной строкой кода, используя следующий синтаксис:

```
a, b = map(int, input().split())   # назначает a и b тип целого числа, данные разделены пробелом
c, d = map(float, input().split(';')) # назначает c и d тип вещественного числа, данные разделены ';'
e, f, h = map(str, input().split('_'))   # назначает e, f, h строковый тип, данные разделены знаком '_'
```

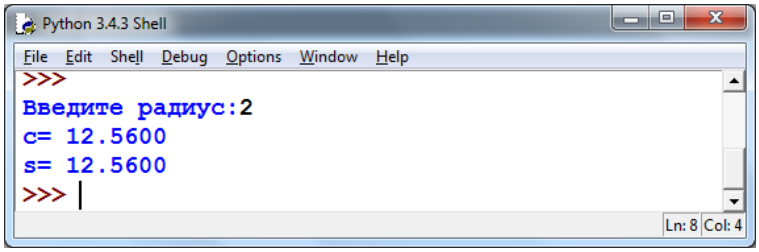
При вызове функции input в скобках можно записать сообщение-подсказку:

a=input("Введите число:")

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 * r
print ("c=", "{:6.4f}".format(c))
print ("s=", "{:6.4f}".format(s))
```



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

САМОЕ ГЛАВНОЕ

Для ввода в оперативную память значений переменных используется оператор ввода **`input`**.

Для вывода данных из оперативной памяти на экран монитора используется оператор вывода **`print`**.

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

Вопросы и задания

1. Целочисленным переменным *i*, *j*, *k* нужно присвоить соответственно значения 10, 20 и 30. Запишите оператор ввода, соответствующий входному потоку:
   а) 20 10 30
   б) 30 20 10
   в) 10 30 20

2. Опишите оператор, обеспечивающий ввод необходимых исходных данных, для вычисления площади треугольника по его трём сторонам.

3. Что является результатом выполнения оператора?
   а) print (a)
   б) print (' a')
   в) print ('a=', a)

4. Какой тип имеет переменная *f*, если после выполнения оператора *print (f)* на экран было выведено следующее число?
   а) 125
   б) 1.25E+2

5. Дан фрагмент программы:

```
input (a)
input (b)
c =a + b
print (a, b)
print (c)
```

   Упростите его, сократив число операторов.

6. Дан фрагмент программы:

```
a = 10
b = a + l
a = b – a
print (a, b)
```

   Какие числа будут выведены на экран компьютера?

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

§3.3 Программирование линейных алгоритмов

Ключевые слова:

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

Программы, реализующие линейные алгоритмы, являются простейшими. Все имеющиеся в них операторы выполняются последовательно, один за другим.

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

3.3.1. Числовые типы данных

Вы уже знакомы с основными числовыми типами данных int и float. К ним применимы стандартные функции, часть из которых приведена в табл. 3.3.

Таблица 3.3

Функция	Назначение	Тип аргумента	Тип результата
abs (x)	Модуль *x*	int, float	Такой же, как у аргумента
round (x)	Округление вещественного *x* до ближайшего целого (можно задать количество знаков после запятой	float	int, float
int (x)	Преобразование вещественного или строкового *x* к целому	str, float	int

Большинство стандартных функций языка Python разбиты на группы по назначению, и каждая группа записана в отдельный файл, который называется *модулем.* Математические функции собраны в модуле **math:**

```
import math
```

После этого обращение должно выглядеть как:

```
y = math.sqrt (x)
```

Можно загрузить в рабочее пространство все функции модуля:

```
from math import *
```

Теперь к функциям модуля **math** можно обращаться так же, как к встроенным функциям:

```
y=sqrt (x)
```

Функция	Назначение	Тип аргумента	Тип результата
sqrt (x)	Квадратный корень из *x*	int, float	float
sin (x)	Синус угла *x*, заданного в радианах	int, float	float

Для работы со случайными числами нужно использовать модуль **random**

Функция	Назначение	Тип аргумента	Тип результата
random()	Случайное число от 0 до 1	-	float
randint(a,b)	Случайное число n, $a \le n \le b$	int	int

Пример: 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)
```

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?

3.3.2. Целочисленный тип данных

Над целыми числами в языке Python выполняются следующие операции:

- ✓ сложение (+)
- ✓ вычитание (-)
- ✓ умножение (*)
- ✓ получение целого частного (//)
- ✓ получение целого остатка 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: $x = a * 100 + b * 10 + c$, 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?

3.3.3. Строковый тип данных

Значением строковой величины (тип str) является произвольная последовательность символов, заключенная в апострофы. Символами 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. Множество таких символов состоит из 256 элементов, каждому из которых согласно используемой кодовой таблице поставлен в соответствие код — число 0 до 255.

Символы, соответствующие первым 32 кодам, являются управляющими, а остальные — изображаемыми. К изображаемым символам относится и пробел, имеющий код 32.

Знакам препинания, знакам арифметических операций, цифрам, прописным и строчным латинским буквам соответствуют коды от 33 до 127. Буквам национального алфавита соответствуют коды с номерами 128 и далее.

В тексте программы переменную строкового типа можно задать, заключив цепочку символов в апострофы или кавычки: *s* = '5'; *c* = 'Book'; *c1* = "1 * ".

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

В Python (как и в алгоритмическом языке) строки можно сцеплять.

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

Программа 5

```
print ('Как тебя зовут?')
n = input()
print ('Привет,', n)
```

3.3.4. 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, символьных, 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* — целая переменная. 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 6

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

Логическим переменным можно присваивать значения логических выражений, построенных с помощью известных вам логических функций **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 7

```
print ('Определение истинности высказывания о равнобедренном треугольнике')
a = int (input('Введите значение a>>'))
b = int (input('Введите значениеb>>'))
c = int (input('Введите значениес>>'))
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.

Вопросы и задания

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

2. 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
xa = 2 ya = 1 xb = 10 yb = 7	|AB|= 10.0

3. 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

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

Пример входных данных	Пример выходных данных
xa = 2 ya = 1 xb = 6 yb = 5 xc = 10 yc = 1	S = 16.0

5. 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). Используйте компьютер, чтобы ввести точную сумму налога и вывести, сколько следует уплатить.

6. Исследуйте работу функции **`random`**, запустив многократно на выполнение программу:

Программа 8

```
print ('Исследование функции random')
from random import randint
a = int (input ('Введите a >>'))
b = int (input ('Введитеb>>'))
print ('randint в интервале (', a, ', ', b, ')=', randint (a, b))
print (randint (a, b))
```

   Как можно получить случайное число из интервала (a, b)?

   Как можно получить случайное число из интервала (a, b]?

7. Одна компания выпустила лотерейные билеты трёх разрядов: для молодежи, для взрослых и для пенсионеров. Номера билетов каждого разряда лежат в пределах:

   для молодёжи — от 1 до 100;

   для взрослых — от 101 до 200;

   для пенсионеров — от 201 до 250.

   С помощью компьютера выберите случайным образом лотерейный билет в каждом разряде.

8. Запишите на языке Python программу, которая для произвольного натурального двузначного числа определяет:

а) сумму и произведение его цифр;
б) число, образованное перестановкой цифр исходного числа.

9. Запишите на языке Python программу, реализующую алгоритм работы кассира, выдающего покупателю сдачу (*s*) наименьшим возможным количеством банкнот по 500 (*k*500), 100 (*k*100), 50 (*k*50) и 10 (*k*10) рублей.

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.

10. 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. Например, 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.

11. Разработайте 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 РАСПУСКАЮТСЯЗЕЛЁНЫЕ РАСПУСКАЮТСЯ ЗЕЛЁНЫЕ ЛИСТЬЯ РАСПУСКАЮТСЯ ЛИСТЬЯ ЗЕЛЁНЫЕ

12. 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 <2 0) and (b < 30)
в) rez = (not (a < 15)) or (b > 20)

13. Составьте программу, вводящую True, если высказывание является истинным, и False в противном случае:
а) сумма цифр трёхзначного числа *x* является чётным числом;
б) треугольник со сторонами *a, b, c* является разносторонним

§3.4 Программирование разветвляющихся алгоритмов

Ключевые слова:

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

3.4.1. Условный оператор

При записи на языке Python разветвляющихся алгоритмов используют условный оператор. В условном операторе после условия **if,** и после **else** можно использовать много операторов: для этого отступ у всех операторов должен быть одинаковый, таким образом, они объединяются в один составной оператор. Его общий вид:

```
if <условие>:
    <оператор_1>
    <оператор_2>
      ……..
    <оператор_n>
else:
    <оператор_1>
    <оператор_2>
      ……..
    <оператор_k>
```

Если оператор только один, то иногда бывает удобно записать его в той же строке:

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

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);
сложные — записанные с помощью логических операций: **and, or, not**.

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

Пример 1. Запишем на языке Python рассмотренный в п. 2.4.2 (пример 8) 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: '))
if x >= a and x <= b:
    print ('Точка принадлежит отрезку')
else:
    print ('Точка не принадлежит отрезку')
```

Пример 2. Воспользуемся неполным условным оператором для записи на языке Python рассмотренного в п. 2.4.2 (пример 9) 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)
```

Более короткая запись:

```
a, b, c = map (int, input ('Введите a, b, c>>').split())
y = a
if b > y: y = b
if c > y: y = c
print ('y=',  y)
```

В языке Python имеются встроенные функции **`max`** и **`min`** для нахождения максимального и минимального из двух или нескольких значений. И тогда программа будет иметь вид:

```
a, b, c = map (int, input ('Введите a, b, c>>').split())
y = max (a, b, c)
print ('y=', y)
```

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

3.4.2. Каскадное ветвление

В Python существует разновидность условного оператора, которая позволяет выбрать один из нескольких (а не только из двух) вариантов. Если после **`else`** сразу следует еще один оператор **`if`**, можно использовать каскадное ветвление со служебным словом **`elif`** (сокращение от **`else-if`**): если очередное условие ложно, выполняется проверка следующего условия и т. д.

Пример. 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')
elif d == 0:
    x = – b/2/ a
    print ('Корень уравнения x=',"{:6.4f}". format(x))
else:
    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))
```

Пример. Запишем на языке Python рассмотренного в п. 2.4.2 (пример 10) 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, записав программу 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.

САМОЕ ГЛАВНОЕ

При записи на языке Python разветвляющихся алгоритмов используют условный оператор:

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

Если при некотором условии требуется выполнить определённую последовательных операторов, то их объединяют в один составной оператор, имеющий одинаковый отступ. Оптимальным отступом считается 4 пробела.

Для записи неполных ветвлений используется неполный условный оператор:

```
if <условие>:
    <операторы>
```

Для выбора из нескольких вариантов используют следующую конструкцию условного оператора:

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

Вопросы и задания

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

2. 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 + l
3. Что такое составной оператор? Для чего он используется в условном операторе?
4. Используя составной оператор, упростите следующий фрагмент программы:
   if a > b: c = 1
   if a > b: d = 2
   if a <= b: c = 3
   if a <= b: d = 4
5. Дано трёхзначное число. Напишите программу, которая определяет:
   а) 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», 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

6. 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

7. 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

8. Имеются данные о количестве полных лет четырёх призёров спартакиады. Напишите программу, выбирающую и выводящую возраст самого младшего призёра.
9. Напишите программу, определяющую, лежит ли точка *A(xa, ya)* 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.

10. 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

11. 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?

12. Напишите программу, вычисляющую значение функции:

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

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

13. Составьте программу для решения задачи № 21 к § 2.4 (определение дня недели).

14. 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

15. Напишите программу, в которой пользователю предлагается дополнить до 100 некоторое целое число *a* (*a* — случайное число, меньшее 100). Ответ пользователя проверяется и комментируется.

§3.5 Программирование циклических алгоритмов

Ключевые слова:

- while (цикл ПОКА)
- for (цикл с параметром)

3.5.1. Программирование циклов с заданным условием продолжения работы

Цикл с заданным условием продолжения работы (цикл-ПОКА) программируется в языке Python с помощью оператора **while**. Общий вид оператора:

```
While <условие>:
    <оператор>
```

Здесь:

<условие> — логическое выражение; пока оно истинно, выполняется тело цикла;

<оператор> — простой или составной оператор, с помощью которого записано тело цикла.

Запишем на языке Python рассмотренный в п. 2.4.3 (пример 14) 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)
```

Каким будет результат выполнения программы при x = -10 и y = 3? Как вы можете объяснить этот результат?

3.5.2. Программирование циклов с заданным условием окончания работы

В языке Python нет цикла с заданным условием окончания работы, но его можно организовать с помощью цикла **while**:

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

Такой цикл 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», досрочный выход из цикла).

Запишем на языке Python рассмотренный в п. 2.4.3 (пример 17) 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')
```

3.5.2 Программирование циклов с заданным числом повторений

Цикл с заданным числом повторений (цикл ДЛЯ) программируется в языке Python с помощью оператора **for**. Его общий вид:

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

<параметр> — переменная целого типа;
<оператор> — простой или составной оператор — тело цикла.

Здесь слово **for** означает «для», переменная (её называют переменной цикла) изменяется в диапазоне (**in range**) от **k** (начальное_значение) до **n-1** (конечное_значение - **n** не включается в диапазон) с шагом **m**. Начальное значение можно не указывать – тогда оно равно 0, шаг тоже можно не указывать, по умолчанию равен 1.

При выполнении этого оператора после каждого выполнения тела цикла происходит увеличение на шаг параметра цикла. Если **k = n** или **k > n** – цикл не выполнится ни разу.

Запишем на языке Python рассмотренный в п. 2.4.3 (пример 19) 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):  # цикл будет работать от 0 до n-1 ровно n раз
    y = y * a
print('y=', y)
```

3.5.3 Различные варианты программирования циклического алгоритма

Особенностью программирования является то, что для решения одной и той же задачи могут быть созданы разные программы. Вы могли убедиться в этом, программируя ветвления. Рассмотрим пример, показывающий, что и циклический алгоритм может быть запрограммирован разными способами.

Пример: Напишем программу, в которой осуществляется ввод целых чисел (ввод осуществляется 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 оператором 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, мы можем воспользоваться оператором **while**:

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 не известно, 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-ПОКА), **for** (цикл с параметром). Если число повторений тела цикла известно, то лучше воспользоваться оператором **for**; в остальных случаях используется оператор **while**.

Вопросы и задания

1. Дана последовательность операторов:

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

Сколько раз будет повторен цикл и какими будут значения переменных *a*, *b*, *s* после исполнения этой последовательности операторов?

2. Требовалось написать программу вычисления факториала числа *n* (факториал числа *n* есть произведение всех целых чисел от 1 до *n*). Программист торопился и написал программу неправильно. Ниже приведён фрагмент его программы, в котором содержатся четыре 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>>5	5! = 120
Введите n>>6	6! = 720

3. Проанализируйте следующий цикл:

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

В чём его особенность?

4. Запишите на языке Python программы решения № 26-30 из § 2.4. Используйте оператор **`while`**.

5. Дана последовательность операторов:
```
a = 1
b = 1
while True:
    a += 1
    b *= 2
    if b > 8: break
s = a + b
```
Сколько раз будет повторён цикл и какими будут значения переменных *a*, *b*, *s* после исполнения этой последовательности операторов?

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

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

8. Напишите программу вычисления наибольшего общего делителя двух целых чисел

9. Сколько раз будет выполнен цикл?
   а) `for i in range (15):  s += l`
   б) `for i in range (10, 15):  s += l`
   в) `for i in range (-1, 1):  s += l`
   г) `for i in range (1, 1):  s += l`
   д)
```
k = 5
for i in range (k - l, k + 1): s += l
```

10. Напишите программу, которая 10 раз выводит на экран ваши имя и фамилию.

11. Напишите программу, выводящую на экран изображение шахматной доски, где чёрные клетки изображаются звёздочками, а белые — пробелами. Рекомендуемый вид экрана после выполнения программы:
```
 * * * *
* * * *
 * * * *
* * * *
```

12. Напишите программу, которая вычисляет сумму:
   а) первых *n* натуральных чисел;
   б) квадратов первых *n* натуральных чисел;
   в) всех чётных чисел в диапазоне от 1 до *n*;
   г) всех двузначных чисел.

13. Напишите программу, которая генерирует 10 случайных чисел в диапазоне от 1 до 20, выводит эти числа на экран и вычисляет их среднее арифметическое.

14. Запишите на языке Python программы решения задач № 32, 33 из § 2.4. Используйте оператор **`for`**.

15. 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

16. 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

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

Тестовые задания для самоконтроля

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 интервала [10, 20] необходимо использовать выражение:
   а) randint (2 *10)
   б) randint (1020)
   в) randint (10, 20)
   г) randint (10)*2
11. В каких из условных операторов 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. Цикл в фрагменте программы

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