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

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`	Подключаем функцию `randint()` модуля `random`
`a = randint(1, 10)`	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
round(1.5)	2
round(2.5)	2
round(2.65, 1)	2.6
round(2.75, 1)	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?