## 12. Арифметические выражения и функции. Целочисленное деление. Строковые выражения.

Вы уже знакомы с числовыми типами данных int и float и строковым типом str. К ним применимы многочисленные функции, некоторые из них приведены в таблице.

| **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 |
| pow(x,y) | Вычисляет x в степени y | int, float | int, float |
| sin(x) | Синус угла X, заданного в радианах | int, float | float |
| cos(x) | Косинус угла X, заданного в радианах | int, float | float |
| tan(x) | Тангенс угла X, заданного в радианах | int, float | float |
| random() | Случайное число от 0 до 1 | - | float |
| randint(a, b) | Случайное целое число n из промежутка от a до b включительно | int | int |

***Надо запомнить! Аргумент функции всегда записывается в скобках.***

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 функциями, решается с помощью модулей. Каждый модуль содержит набор функций, предназначенных для решения задач из определенной области.

Так, первые три из представленных в таблице 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 *pow(x,y)*, *sqrt(x)*, *sin(x)*, *cos(x)*, *tan(x)*, то для их вызова предварительно надо подключить модуль *math*, в котором собраны математические функции. Две последние из приведённых в таблице функций требуют подключения модуля *random*, позволяющего генерировать случайные числа. Модуль *graph* нужен для рисования геометрических фигур, 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
```

### Запись арифметических выражений в языке Python

Допустим нам необходимо вычислить выражение, записанное в математическом виде таким образом:

$$\frac{2 * 17{,}56^2}{7 * 2{,}47 * 0{,}43}$$

Прежде чем написать программу, которая подсчитает нам результат, сформулируем правила записи алгебраических выражений на языке программирования:

1. Выражения содержат числа, имена других переменных, знаки операций, круглые скобки, имена функций.
2. Разделителем целой и дробной части является точка.

3. Выражение записывается в одну строку (линейная запись выражений), символы последовательно выстраиваются друг за другом, проставляются ВСЕ знаки операций, используются круглые скобки.
4. Учитывается приоритет операций (порядок действий), для изменения которого используются круглые скобки. Действия выполняются в следующем порядке:
    1) действия в скобках;
    2) возведение в степень;
    3) умножение и деление, слева направо;
    4) сложение и вычитание, слева направо.

Таким образом, умножение и деление имеют одинаковый приоритет, более высокий, чем сложение и вычитание.

Следуя правилам записи арифметических выражений, мы должны перевести данную (математическую запись) дроби в линейную запись, то есть записать дробь в одну строчку. Так как и числитель, и знаменатель у нас сложные (то есть содержат два и более множителя), то при записи в линейную форму необходимо выражения в числителе и знаменателе взять в скобки. Таким образом, линейная запись такого выражения будет выглядеть следующим образом:

```
(2*17.56*17.56)/(7*2.47*0.43)
```

Напишем программу для вычисления данного выражения.

Так как у нас имеется дробь, то результатом будет являться вещественное число

```
print((2 * 17.56 * 17.56) / (7 * 2.47 * 0.43))
```

При запуске программы на экране будет выведено число 82.94984330235246

Если в качестве операндов некоторого арифметического выражения используются только целые числа, то результат тоже будет целое число. Исключением является операция деления, результатом которой является вещественное число. При совместном использовании целочисленных и вещественных переменных, результат будет вещественным.

Если в выражении операторы имеют одинаковый приоритет, то они выполняются в направлении слева направо за исключением оператора возведения в степень (**), который выполняется справа налево.

Определим порядок действий компьютера при вычислении выражения:

a = c + b – 1 / 2 * 5

Первым будет выполняться деление, ½, затем умножение полученного числа на 5. После этого сложение c + b и, наконец, из полученной суммы вычитается результат ранее выполненного умножения.

Запишем ещё одно арифметическое выражение на языке программирования.

$$\frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

Это выражение содержит и степень, и функцию вычисления квадратного корня.

*d = (-b + sqrt(b**2 – 4 * a * c))/ (2 * a)*

Составим программу для его вычисления:

```
from math import *
a,b,c = map(float, input().split() )
d = (-b + sqrt(b**2 – 4 * a * c)/ (2 * a)
print('d=', d)
```

**Целочисленная арифметика.**

Вы уже знаете, что над целыми числами в языке Python можно выполнять следующие операции:
сложение ( + ), вычитание (-), умножение (*), возведение в степень (**),
деление (/), целочисленное деление или получение неполного частного (//),
взятие остатка или получение остатка от деления (%).
Остановимся более подробно на трёх последних операциях.

Операция / - обычное деление. Ответ может быть дробным (тип float). Например, 9 / 4 = 2,25. Такой ответ можно получить при привычном нам делении на калькуляторе.

Операция // (во многих языках программирования обозначается словом **div)** - целочисленное деление. При таком делении дробная часть отбрасывается (ответ получается типа integer).
Например, 9 // 4 = 2

Операция % (во многих языках программирования обозначается словом **mod)** вычисляет остаток от деления (ответ получается типа integer). Например, 9 % 4 = 1

Чтобы лучше понять, вспомним как делили начальной школе, когда не знали о дробях и сравним с вычислениями выше. 9 : 4 = 2 (ост. 1)

Например, 435 сантиметров = 4 метра 35 сантиметров.
Этого можно добиться, если
1) число 435 поделим нацело на 100, 435 // 100 = 4 метра.
2) возьмем остаток от деления 435 на 100, 435 % 100 = 35 сантиметров.
3) Выведем оба ответа.

```
A = int(input('введите число сантиметров'))
print(A, ' сантиметров = ', A // 100, ' метров ', A % 100, ' сантиметров ')
```

Запишем на языке Python программу нахождения суммы цифр вводимого с клавиатуры натурального двухзначного числа.

Используем тот факт, что положительное двухзначное число можно представить в виде следующей суммы: x = a · 10 + b , где a – количество десятков в числе и его первая цифра, b – количество единиц в числе и его вторая цифра. Если положительное двухзначное число x разделить нацело на 10, то получится первая цифра числа (число десятков). А остаток от такого деления есть последняя цифра числа (число единиц). Например, 25 : 10 = 2 (ост 5) значит, 25 // 10 = 2 ; 25 % 10 = 5

```
print (' Нахождение суммы цифр двухзначного числа ')
x = int(input (' Введите двузначное число:  ' ))
a = x // 10
b = x % 10
s = a + b
print ( ' s=  ' , s)
```

Эту программу можно записать и короче, сократив количество используемых переменных:

```
print (' Нахождение суммы цифр двухзначного числа ')
x = int(input (' Введите двузначное число:  ' ))
print ( ' s=  ' , (x // 10) + (x % 10))
```

А если число X трехзначное, то как отделить его цифры?
Команда *a = X // 100* отделит число сотен.
Команда *b = X % 10* отделит число единиц.
Команда *c = (X % 100) // 10* или *(X // 10) % 10* отделит число десятков.

А как перевернуть трехзначное число (например, было 123, а стало321)?
После отделения цифр, собрать его «наоборот», сделав единицы сотнями, а сотни единицами.
*X1 = b * 100 + c * 10 + a*

В Python операция вычисления остатка выполняется по математическим правилам, то есть, как принято считать в Теории Чисел, остаток - это неотрицательное число. **Знак остатка совпадает со знаком делителя.**

Пример

```
c = 10 // 3       # Ответ: c = 3
d = 10 % 3        # Ответ: d = 1
e = -7 // 4       # Ответ: e = -2
f = -7 % 4        # Ответ: f = 1
```

Значения переменных e и f получились такими, потому что -7 = (-2*4)+1

## 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. Например:

```
b = '1+'
c = 'Book'
d = '5'
```

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)*

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.

| **Функция** | **Назначение** | **Тип аргумента** | **Тип результата** |
|---|---|---|---|
| len(x) | Длина строки в переменной X | str | int |
| ord(x) | Код символа в переменной X | str | int |
| chr(y) | Символ по коду в переменной y | int | str |

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).

В Python строки можно сцеплять: *a + b* (к концу строки *a* прикрепляется («приписывается») строка *b*). Эта операция называется **конкатенация**. Пусть *e = b + d + b*

Тогда в нашем примере в переменную *e* будет записана следующая строка: *' 1+51+ '*.

В результате операции *a * k*, где *k* - целое число, строка *a* повторяется *k* раз. Так, при *b = ' 1+ '*, в результате выполнения команды *print (b * 3)* будет выведена строка: *1+1+1+*

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

```
# Программа Приветствие
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 три символа
a = input(' Введите символ ') # вводится символ в переменную a
k = ord(a)                    # определяется код символа, лежащего в a и присваивается переменной k
print ('код символа ', k)     # выводится код символа a, лежащий в переменной k
b = chr(k - 1) + a + chr(k + 1) # b= символ по коду (k-1)+ символ из перем-й a+символ по коду (k + 1)
print(b)                      # выводится строка из трех символов, лежащая в переменной b
```

**Контрольные вопросы и задания**

1. Из таблицы в параграфе перепишите и выучите функции и их назначение.
2. Как подключить модули *math, random, graph* со всеми их функциями? Запишите соответствующие команды с пояснениями, какая из них что делает.
3. Как сгенерировать случайное целое число из промежутка от x до y? от 1 до 5? Запишите соответствующие команды с комментарием, какая из них что делает.
4. Каков приоритет операций при вычислении значений выражений?
5. Запишите программу для вычисления значения выражения $x = \frac{-b-\sqrt{b^2-4ac}}{2a}$
6. Какие операции обозначаются символами / // % ?
7. Используя команды и примеры, написанные в параграфе, запишите программу, вычисляющую произведение цифр трехзначного числа. Прокомментируйте каждую строку программы.
8. Как определить длину строки n? Запишите соответствующую команду с комментарием.
9. Как по коду Z узнать соответствующий символ? А как предыдущий? Как последующий? Запишите соответствующие команды с комментарием.
10. Как по символу узнать его код? Запишите соответствующую команду с комментарием.
11. Как вывести значение переменной s строкового типа k раз подряд? Запишите соответствующую команду с комментарием.