

Ссылка на урок:

<https://coreapp.ai/app/player/lesson/673c9b37936b5278df06eb7a> (для учеников)

Занятие 8

Тема. Выражения. Математические функции. Часть 2

1. Теоретическая часть

На этом занятии мы продолжим говорить о функциях. В Python функции можно разделить на четыре типа:

- 1) встроенные;
- 2) именованные;
- 3) анонимные;
- 4) рекурсивные.

Дадим краткое определение каждому типу функции.

Встроенные – это функции, которые доступны по умолчанию, без использования дополнительных модулей или библиотек. Их можно вызывать по имени в любой момент и в любом месте кода. Они предназначены для выполнения базовых математических операций.

Дополнительные математические функции доступны через встроенный модуль **math**. Для их использования необходимо импортировать этот модуль:

```
>>> import math
>>> print(factorial(3)) # 6
```

Именованные (пользовательские) – это функции, которую пользователь создаёт самостоятельно для выполнения определенной задачи, объявление функции начинается со служебного слова **def**.

Анонимные (lambda-функции) – это особый вид функций, которые предназначены для решения конкретной задачи и не используются повторно в других местах кода.

Рекурсивные — это функции, которые вызывают сами себя во время выполнения программы, в них реализован способ вычисления очередного значения функции через вычисление предшествующих значений (например, вычисление факториала).

Независимо от типа функции, при её описании (по-другому, объявлении) указывается имя функции (кроме анонимной) и, в круглых скобках, список параметров. Функция может принимать произвольное количество параметров или не принимать их вовсе.

На этом занятии поговорим о работе со встроенными математическими функциями.

Общий вид встроенной функции: **<имя функции> (список параметров) :**

При вызове функции формальные параметры, прописанные в круглых скобках при её объявлении, заменяются на *конкретные значения или переменные*, называемые *аргументами*.

Аргументы – это значения или переменные, передаваемые в функцию при её вызове (фактические параметры).

В некоторых случаях могут использоваться встроенные функции без аргументов. В Python существует более 60 встроенных функций. Приведём примеры некоторых из них:

- 1) функция **round()** – округляет переданное ей число, в общем виде она выглядит так: **round(number, digits)**.

Через аргумент **number** передаётся дробное число, а в аргументе **digits** указывается количество чисел после запятой. Второй аргумент по умолчанию равен нулю, например, **result = round(10.25) # 10**.

- 2) функция **abs()** – возвращает абсолютную величину (модуль) переданного ей числа, например,

```
number = -10
absolute = abs(number)
print(absolute) # 10
```

- 3) функция **pow()** (произносится как «пау») – возводит указанное число в нужную степень. Она принимает на вход два обязательных аргумента: берет первый аргумент и возводит его в степень, переданную вторым аргументом, например,

```
result = pow(2, 3) # 2 * 2 * 2
print(result) # 8
```

- 4) ***функция **sqrt()** (произносится как «эскьюэрти») – возвращает квадратный корень положительного числа; тип возвращаемого числа – float. Но, перед вызовом этой функции следует импортировать в программу модуль **math**, например,

```
import math
kor = math.sqrt(64)
print(kor) # 8.0
ИЛИ
import math
print(math.sqrt(64)) # 8.0
```

- 5) функция **len()** – считает количество символов в переданной строке, например,

```
# Вызов функции len с аргументом 'Hello!'
result = len('Hello!')
print(result) # 6
```

- 6) функции **min()** и **max()** предназначены для поиска минимального и максимального значения из переданного списка или кортежа чисел, например,

```
numbers = [4, 56, -2, 7, 12]
min_val = min(numbers)
max_val = max(numbers)
print(min_val) # -2
print(max_val) # 56
```

СПРАВКА. Кортеж – это неизменяемый список.

- 7) функция **divmod()** – используется для целочисленного деления двух входных чисел, в общем виде это выглядит так: **divmod(делимое, делитель)**. В

результате функция возвращает кортеж из двух значений, первое из которых – частное от деления аргументов, а второе – остаток от деления, например,

```
print(divmod(20, 5)) # Результат выполнения кода: (4, 0)
print(divmod(21, 5)) # Результат выполнения кода: (4, 1)
print(divmod(20, 7)) # Результат выполнения кода: (2, 6)
print divmod (6, 9) # Результат выполнения кода: (0, 6)
```

Пример расчёта:

$$\begin{array}{r} 6 \overline{) 9} \\ 0 \overline{) 0}, \\ 6 \end{array}$$

```
print divmod (9, 47) # Результат выполнения кода: (0, 9)
```

Если при использовании функции **divmod()** делимое является отрицательным числом, то алгоритм вычисления изменяется, так как математически остаток от деления на положительное число не может быть отрицательным:

- избавляемся от отрицательного остатка, для этого к остатку прибавляем делитель;
- к отрицательному неполному частному прибавляем минус единицу, чтобы привести в соответствие исходное делимое и положительный остаток.

Получившихся два новых значения и будут результатом. Подробное объяснение этой ситуации приведено на сайте YOUR TUTOR:

<https://yourtutor.info/%D0%BE%D1%81%D1%82%D0%B0%D1%82%D0%BE%D0%BA-%D0%BE%D1%82-%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F-%D0%BE%D1%82%D1%80%D0%B8%D1%86%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D1%85>

Пример расчёта:

divmod(-10, 3)

$$\begin{array}{r} -10 \overline{) 3} \\ \underline{9} \overline{) -3}, \\ -1 \end{array}$$

=> (-3, -1), но такой ответ оставить не можем, так как нам нужно получить реальный остаток (положительный) и соответствующее ему исходное делимое. Выполняем действия:

$$-1 + 3 = 2,$$

$$-3 + (-1) = -3 - 1 = -4$$

=> (-4, 2), что и будет являться ответом.

Ещё пример:

divmod(-20, 7)

$$\begin{array}{r} -20 \overline{) 7} \\ \underline{14} \overline{) -2}, \\ -6 \end{array}$$

Выполняем правильные действия:

$$-6 + 7 = 1$$

$$-2 + (-1) = -3$$

Ответ: (-3, 1) .

- 8) ***функция `cos()` – возвращает косинус числа (в радианах), перед вызовом этой функции (а также остальных тригонометрических функций) следует импортировать в программу модуль **math**, например,

```
import math
a = 45
b = math.cos(a)
print(b)    # 0.5253219888177297
```

Примеры программ с использованием встроенных математических функций

1. Написать программу, которая выводит на экран модуль суммы двух переменных целого типа – `num1` и `num2`. Значения переменных ввести с клавиатуры во время выполнения программы. Использовать поясняющий текст.

Решение (с комментариями)

```
num1 = input(' Введите первое число:') # после ввода числа
нажать кл. Enter
num1 = int(num1) # преобразовываем введённую строку символов в
целочисленный тип
num2 = input(' Введите второе число:') # после ввода числа
нажать кл. Enter
num2 = int(num2) # преобразовываем введённую строку символов в
целочисленный тип
abssum = abs(num1 + num2)
print('Ответ: ', abssum)
```

На экране:

main.py		Output
<pre>1 num1 = input(' Введите первое число:') # после ввода числа нажать кл. Enter 2 num1 = int(num1) # преобразовываем введённую строку символов в целочисленный тип 3 num2 = input(' Введите второе число:') # после ввода числа нажать кл. Enter 4 num2 = int(num2) # преобразовываем введённую строку символов в целочисленный тип 5 abssum = abs(num1 + num2) 6 print('Ответ: ', abssum) 7</pre>	Run	<pre>Введите первое число:10 Введите второе число:-13 Ответ: 3 === Code Execution Successful ===</pre>

The screenshot shows a Python IDE with a file named 'abssum.py' located at 'C:/Центринформ/Питон_Мои программы/abssum.py (3.7.4)'. The script contains the following code:

```
num1 = input('Введите первое число: ')
num1 = int(num1)
num2 = input('Введите второе число: ')
num2 = int(num2)
abssum = abs(num1 + num2)
print('Ответ: ', abssum)
```

Below the editor is a 'Python 3.7.4 Shell' window showing the execution output:

```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 20:34:2
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for mor
>>>
===== RESTART: C:/Центринформ/Питон_Мои программы/ak
Введите первое число: 10
Введите второе число: -13
Ответ:  3
>>>
```

2. Написать программу вычисления времени падения камня по формуле:

$t = \sqrt{2\frac{h}{g}}$, где $g=9.81\text{м/с}^2$, h – высота. Значение высоты (в метрах) ввести с клавиатуры во время выполнения программы. Результат вывести с тремя знаками после запятой (точки). При вводе высоты и выводе результата использовать поясняющий текст.

Решение.

```
import math
G = 9.81
h = input('Введите высоту (в метрах): ')
h = float(h)
t = math.sqrt(2*h/G)
print("Время падения камня: ", "{:6.3f}".format(t), "секунд")
```

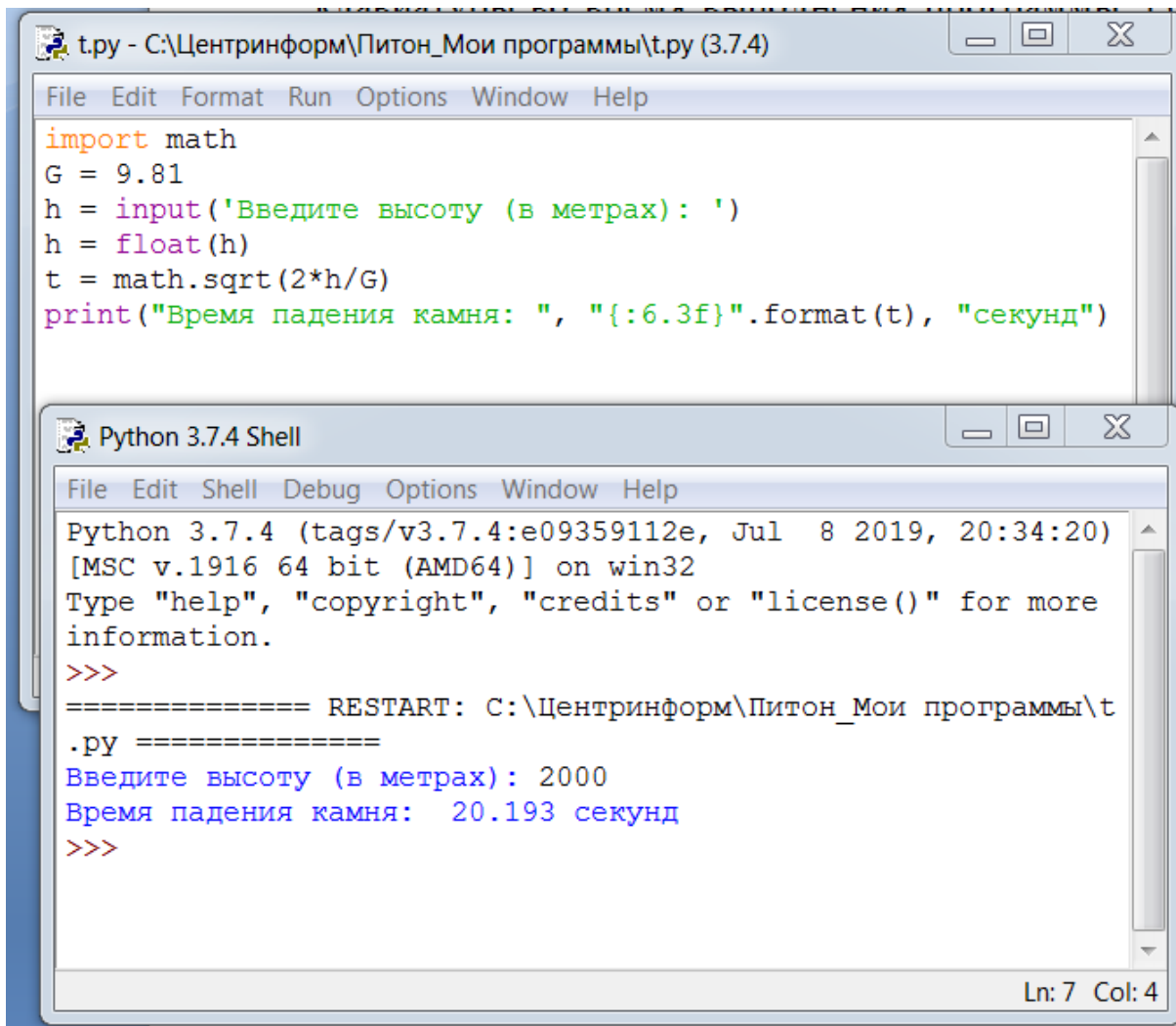
На экране:

The screenshot shows a code editor with a file named 'main.py'. The code is as follows:

```
1 import math
2 G = 9.81
3 h = input('Введите высоту (в метрах): ')
4 h = float(h)
5 t = math.sqrt(2*h/G)
6 print("Время падения камня: ", "{:6.3f}".format(t), "секунд")
```

On the right side, there is an 'Output' window showing the execution results:

```
Введите высоту (в метрах): 2000
Время падения камня: 20.193 секунд
=== Code Execution Successful ===
```



Можно объединить 5 и 6 строки в одну команду:

```
import math
G = 9.81
h = input('Введите высоту (в метрах): ')
h = float(h)
print("Время падения камня: ", "{:6.3f}".format(math.sqrt(2*h/G)),
"секунд")
```

2. Просмотр учебного видеоролика «Встроенные математические функции Python» (с целью расширения знаний, наглядного представления и закрепления материала):

<https://rutube.ru/video/7a1f68211400af9bdbc592ab2dd57679/?r=plemwd>

(длина ролика: 05:41).

3. Закрепление знаний

ЗАДАНИЕ 1.

Что будет выведено на экран в результате выполнения данных команд?

- а) `print(divmod(20, 3))` # Результат выполнения кода: (6, 2)
- б) `print(divmod(20, 20))` # Результат выполнения кода: (1, 0)
- в) `print(divmod(5, 10))` # Результат выполнения кода: (0, 5)
- г) `print(divmod(18, 40))` # Результат выполнения кода: (0, 18)
- д) `print(divmod(-20, 3))` # Результат выполнения кода: (-7, 1)

е) `print(divmod(-15, 2))` # Результат выполнения кода: (-8, 1)
ж) `print(divmod(-45, 3))` # Результат выполнения кода: (-15, 0)
з) `print(divmod(-10, 3))` # Результат выполнения кода: (-4, 2)

ЗАДАНИЕ 2.

Написать программу вычисления значения выражения по формуле:

$\frac{\sin a - b}{|b| + \cos a} + a$, где $a = 90$, $b = -5$ (задаются с помощью оператора присваивания).

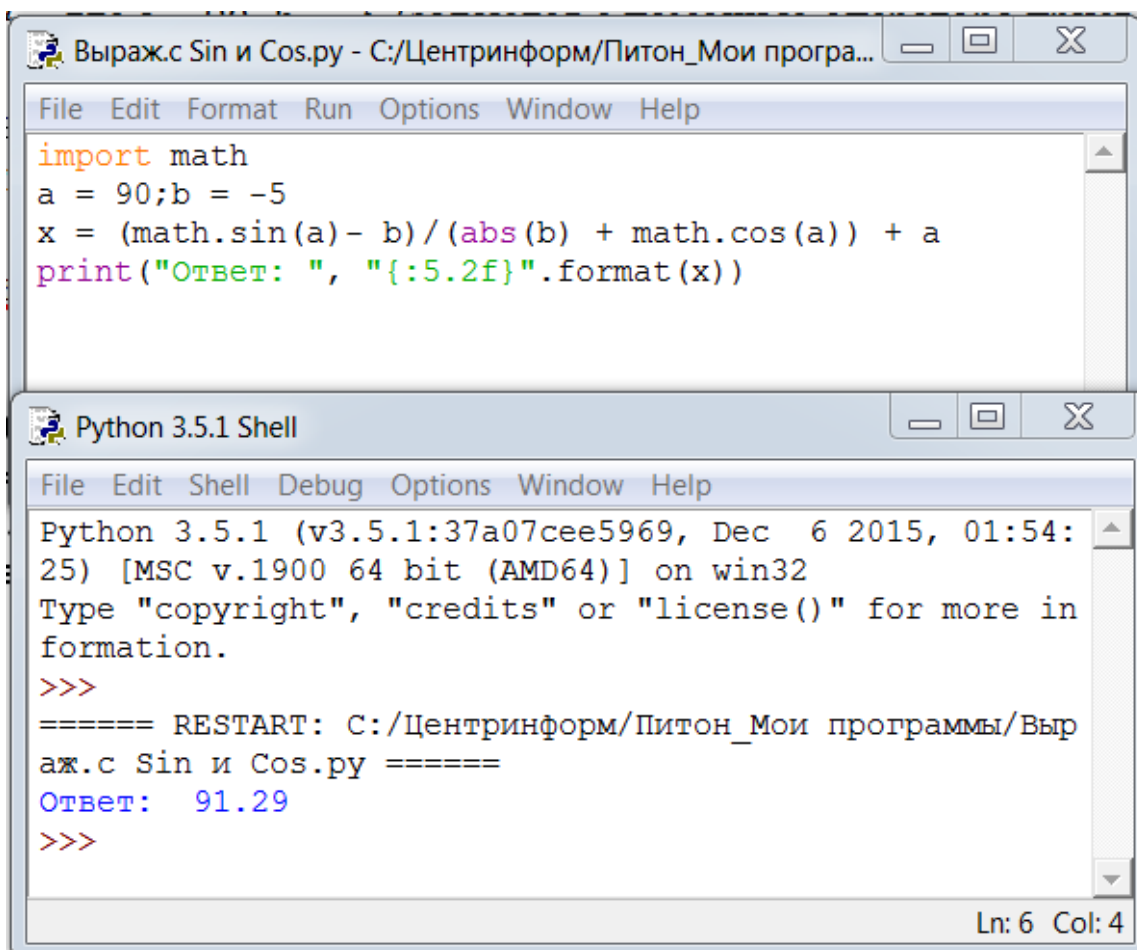
Результат вывести с двумя знаками после запятой (точки). При выводе результата использовать поясняющий текст.

!!! Решение будет на следующей странице

Решение.

```
import math
a = 90;b = -5
x = (math.sin(a) - b) / (abs(b) + math.cos(a)) + a
print("Ответ: ", "{:5.2f}".format(x))
```

На экране:



The screenshot shows two windows from a Python IDE. The top window, titled 'Выраж.с Sin и Cos.py - C:/Центринформ/Питон_Мои програ...', contains the following Python code:

```
import math
a = 90;b = -5
x = (math.sin(a) - b) / (abs(b) + math.cos(a)) + a
print("Ответ: ", "{:5.2f}".format(x))
```

The bottom window, titled 'Python 3.5.1 Shell', shows the execution output:

```
Python 3.5.1 (v3.5.1:37a07cee5969, Dec 6 2015, 01:54:
25) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more in
formation.
>>>
===== RESTART: C:/Центринформ/Питон_Мои программы/Выр
аж.с Sin и Cos.py =====
Ответ:  91.29
>>>
```

The status bar at the bottom right of the shell window indicates 'Ln: 6 Col: 4'.