Казанский (Приволжский) Федеральный Университет
Институт Физики

Лекция №2
«Вычисления. Библиотека math. Цикл **for**»

И.И. Файрушин

E-mail: *fairushin_ilnaz@mail.ru*

Казань, 2022

Вещественные (действительные) числа

В этом разделе речь пойдет о действительных числах, имеющих тип **float**.

Обратите внимание, что если вы хотите считать с клавиатуры действительное число, то результат, возвращаемый функцией **input()** необходимо преобразовывать к типу **float**:

```
1  x = float(input())
2  print(x)
3
```

Действительные (вещественные) числа представляются в виде чисел с десятичной точкой (а не запятой). Для записи очень больших или очень маленьких по модулю чисел используется так называемая запись «с плавающей точкой» (также называемая «научная» запись). В этом случае число представляется в виде некоторой десятичной дроби, называемой мантиссой, умноженной на целочисленную степень десяти (порядок).

Вещественные (действительные) числа

Числа с плавающей точкой в программах на языке Питон, а также при вводе и выводе записываются так: сначала пишется мантисса, затем пишется буква **e**, затем пишется порядок. Пробелы внутри этой записи не ставятся. Например, 1.496e11 и 2.99e-23. Перед самим числом также может стоять знак минус.

Напомним, что результатом операции деления / всегда является действительное число (**float**), в то время как результатом операции // является целое число (int).

Преобразование действительных чисел к целому производится с округлением в сторону нуля, то есть **int(1.7) == 1**, **int(-1.7) == -1**.

Вещественные (действительные) числа

Библиотека math.

Для проведения вычислений с действительными числами язык Питон содержит много дополнительных функций, собранных в библиотеку (модуль), которая называется **math**.

Для использования этих функций в начале программы необходимо подключить математическую библиотеку, что делается командой

import math

Например, пусть мы хотим округлять вещественные числа до ближайшего целого числа вверх. Соответствующая функция ceil от одного аргумента вызывается, например, так: **math.ceil(x)** (то есть явно указывается, что из модуля **math** используется функция **ceil()**). Вместо числа **x** может быть любое число, переменная или выражение. Функция возвращает значение, которое можно вывести на экран, присвоить другой переменной или использовать в выражении.

Вещественные (действительные) числа

```
1  import math
2
3  x = math.ceil(4.2)
4  y = math.ceil(4.8)
5  print(x)
6  print(y)
7
```

Другой способ использовать функции из библиотеки **math**, при котором не нужно будет при каждом использовании функции из модуля **math** указывать название этого модуля, выглядит так:

```
1  from math import ceil
2
3  x = 7 / 2
4  y = ceil(x)
5  print(y)
6
```

```
1  from math import *
2
3  x = 7 / 2
4  y = ceil(x)
5  print(y)
6
```

Вещественные (действительные) числа

Округление	
`int(x)`	Округляет число в сторону нуля. Это стандартная функция, для ее использования не нужно подключать модуль `math`.
`round(x)`	Округляет число до ближайшего целого. Если дробная часть числа равна 0.5, то число округляется до ближайшего четного числа.
`round(x, n)`	Округляет число `x` до `n` знаков после точки. Это стандартная функция, для ее использования не нужно подключать модуль `math`.
`floor(x)`	Округляет число вниз («пол»), при этом `floor(1.5) == 1`, `floor(-1.5) == -2`
`ceil(x)`	Округляет число вверх («потолок»), при этом `ceil(1.5) == 2`, `ceil(-1.5) == -1`
`abs(x)`	Модуль (абсолютная величина). Это — стандартная функция.
Корни, логарифмы	
`sqrt(x)`	Квадратный корень. Использование: `sqrt(x)`
`log(x)`	Натуральный логарифм. При вызове в виде `log(x, b)` возвращает логарифм по основанию `b`.
`e`	Основание натуральных логарифмов e = 2,71828...

Вещественные (действительные) числа

Тригонометрия	
`sin(x)`	Синус угла, задаваемого в радианах
`cos(x)`	Косинус угла, задаваемого в радианах
`tan(x)`	Тангенс угла, задаваемого в радианах
`asin(x)`	Арксинус, возвращает значение в радианах
`acos(x)`	Арккосинус, возвращает значение в радианах
`atan(x)`	Арктангенс, возвращает значение в радианах
`atan2(y, x)`	Полярный угол (в радианах) точки с координатами (x, y).
`degrees(x)`	Преобразует угол, заданный в радианах, в градусы.
`radians(x)`	Преобразует угол, заданный в градусах, в радианы.
`pi`	Константа π = 3.1415...

Задач

Задача «Улитка»

Условие

Улитка ползет по вертикальному шесту высотой h метров, поднимаясь за день на a метров, а за ночь спускаясь на b метров. На какой день улитка доползет до вершины шеста?

Программа получает на вход натуральные числа h, a, b.

Программа должна вывести одно натуральное число. Гарантируется, что $a > b$.

Задача «Часы - 1»

Условие

С начала суток прошло H часов, M минут, S секунд ($0 \le H < 12$, $0 \le M < 60$, $0 \le S < 60$). По данным числам H, M, S определите угол (в градусах), на который повернулась часовая стрелка с начала суток и выведите его в виде действительного числа.

Задача «Часы - 3»

Условие

С начала суток часовая стрелка повернулась на угол в α градусов. Определите сколько полных часов, минут и секунд прошло с начала суток, то есть решите задачу, обратную задаче «Часы - 1». Запишите ответ в три переменные и выведите их на экран.

Цикл for

Цикл **for**, также называемый циклом с параметром, в языке Питон богат возможностями. В цикле **for** указывается переменная и множество значений, по которому будет пробегать переменная. Множество значений может быть задано *списком, кортежем, строкой или диапазоном*.

Вот простейший пример использования цикла, где в качестве множества значений используется кортеж:

```
1  i = 1
2  for color in 'red', 'orange', 'yellow', 'green', 'cyan', 'blue', 'violet':
3      print('#', i, ' color of rainbow is ', color, sep = '')
4      i += 1
5
```

В этом примере переменная **color** последовательно принимает значения **'red', 'orange'** и т.д. В теле цикла выводится сообщение, которое содержит название цвета, то есть значение переменной **color**, а также номер итерации цикла число, которое сначала равно **1**, а потом увеличивается на один (инструкцией **i += 1** с каждым проходом цикла.

Цикл for

Инструкция **i += 1** эквивалентна конструкции **i = i + 1** (это просто сокращенная запись). Такую сокращенную запись можно использовать для всех арифметических операций: ***=, -=, /=, %=...** В списке значений могут быть выражения различных типов, например:

```
1 for i in 1, 2, 3, 'one', 'two', 'three':
2     print(i)
3
```

При первых трех итерациях цикла переменная **i** будет принимать значение типа **int**, при последующих трех — типа **str**.

Цикл for

Как правило, циклы **for** используются либо для повторения какой-либо последовательности действий заданное число раз, либо для изменения значения переменной в цикле от некоторого начального значения до некоторого конечного.

Для повторения цикла некоторое заданное число раз **n** можно использовать цикл **for** вместе с функцией **range**:

```
1  for i in range(4):  # равносильно инструкции for i in 0, 1, 2, 3:
2      # здесь можно выполнять циклические действия
3      print(i)
4      print(i ** 2)
5  # цикл закончился, поскольку закончился блок с отступом
6  print('Конец цикла')
7
```

В качестве **n** может использоваться числовая константа, переменная или произвольное арифметическое выражение (например, **2**10**). Если значение **n** равно нулю или отрицательное, то тело цикла не выполнится ни разу.

Цикл for

Функция **range** может также принимать не один, а два параметра. Вызов **range(a, b)** означает, что индексная переменная будеть принимать значения от **a** до **b - 1**, то есть первый параметр функции **range**, вызываемой с двумя параметрами, задает начальное значение индексной переменной, а второй параметр — первое значение, которое индексная переменная принимать не будет. Если же **a≥b**, то цикл не будет выполнен ни разу. Например, для того, чтобы просуммировать значения чисел от **1** до **n** можно воспользоваться следующей программой:

```
1  sum = 0
2  n = 5
3  for i in range(1, n + 1):
4      sum += i
5  print(sum)
6
```

Цикл for

В этом примере переменная **i** принимает значения **1, 2, ..., n,** и значение переменной **sum** последовательно увеличивается на указанные значения.

Наконец, чтобы организовать цикл, в котором индексная переменная будет уменьшаться, необходимо использовать функцию **range** с тремя параметрами. Первый параметр задает начальное значение индексной переменной, второй параметр — значение, до которого будет изменяться индексная переменная (не включая его!), а третий параметр — величину изменения индексной переменной. Например, сделать цикл по всем нечетным числам от **1** до **99** можно при помощи функции **range(1, 100, 2)**, а сделать цикл по всем числам от **100** до **1** можно при помощи **range(100, 0, -1)**.

Настройка функции print()

По умолчанию функция **print()** принимает несколько аргументов, выводит их через пробел, после чего ставит перевод строки. Это поведение можно изменить, используя именованные параметры **sep** (разделитель) и **end** (окончание).

```
1  print(1, 2, 3)
2  print(4, 5, 6)
3  print(1, 2, 3, sep=', ', end='. ')
4  print(4, 5, 6, sep=', ', end='. ')
5  print()
6  print(1, 2, 3, sep='', end=' -- ')
7  print(4, 5, 6, sep=' * ', end='.')
8
```

Задачи

Задача «Ряд - 1»

Условие

Даны два целых числа A и B (при этом A ≤ B). Выведите все числа от A до B включительно.

Задача «Ряд - 2»

Условие

Даны два целых числа A и B. Выведите все числа от A до B включительно, в порядке возрастания, если A < B, или в порядке убывания в противном случае.

Задача «Ряд - 3»

Условие

Даны два целых числа A и B, $A > B$. Выведите все нечётные числа от A до B включительно, в порядке убывания. В этой задаче можно обойтись без инструкции if.

Задач

- -

Задача «Сумма N чисел»

Условие

Дано несколько чисел. Вычислите их сумму. Сначала вводите количество чисел N, затем вводится ровно N целых чисел.
Какое наименьшее число переменных нужно для решения этой задачи?

Задача «Лесенка»

Условие

По данному натуральному n ≤ 9 выведите лесенку из n ступенек, i-я ступенька состоит из чисел от 1 до i без пробелов.

Задача «Потерянная карточка»

Условие

Для настольной игры используются карточки с номерами от 1 до N. Одна карточка потерялась. Найдите ее, зная номера оставшихся карточек.
Дано число N, далее N − 1 номер оставшихся карточек (различные числа от 1 до N). Программа должна вывести номер потерянной карточки.

Спасибо за внимание!