

Python

Лекция 3

Функция eval()

- Язык Python содержит много хороших и полезных функций. Но есть одна, с которой имеет смысл познакомиться сразу. Речь о функции eval(). Функция уникальна тем, что возвращает в качестве результата значение выражения, переданного ей в виде текстовой строки.
- Другими словами, если имеется некоторая текстовая строка, в которую «спрятана» команда, имеющая смысл в языке Python, то, передав эту текстовую строку функции eval(), мы получим в результате значение выражения, «спрятанного» в строке.

Функция eval()

- Пример использования функции:

```
main.py ×  
1 text = "(2+3)/0.25-4*2.1"  
2 print(eval(text))  
3 text2 = input("Введите выражение:")  
4 print("Значение введённого выражения", "=", eval(text2))  
5
```

Функция eval()

- В программе командой переменной `text` значением присваивается текст, содержащий арифметическое выражение.
- Тем не менее это всё же текст, и именно так обрабатывается значение переменной `text`. Она используется в команде `print(text, "=", eval(text))`, причём в двух аргументах функции `print()`.
- Первым аргументом передана собственно переменная `text`. Поскольку переменная текстовая, то отображается её значение — как оно есть. Затем отображается знак равенства (вторым аргументом функции `print()` является текст `"="`).

Функция eval()

- Третий аргумент – это выражение eval(text), в переменная text передаётся аргументом функции eval(). Что происходит в таком случае?
- Вычисляется выражение, которое содержится в строке, являющейся значением переменной text. Как следствие, инструкция eval(text) возвращает результат выражения.

Функция `eval()`

- Соответственно второй пример программы представляет собой сначала использование команды `text2 = input()`.
- Выводится запрос и считывается значение, введённое пользователем. Введённое значение будет присваиваться переменной. Что бы пользователь ни ввел – это будет текст.
- Но неявно предполагается, что введённый пользователем текст содержит некоторую команду, корректную с точки зрения синтаксиса языка Python.

Функция eval()

- Например, пользователь на запрос программы вводит значение $2^{**}3+4*5$. Это корректная с точки зрения синтаксиса языка Python команда (оператор $**$ означает возведение в степень), соответствующее выражению 2^3+4*5 (значение выражения равно 28). Но данное значение будет считано как текст – то есть переменная `text` получит к качестве значения ссылку на текст “ $2^{**}3+4*5$ ”. Зато когда выполняется команда `print(“Значение выражение:”, eval(text))`, то благодаря инструкции `eval(text)` значение выражения, «спрятанного» в текст из переменной `text`, вычисляется, и вычисленное значение отображается в окне вывода.

Нормальное знакомство со списками

- Далее мы познакомимся со списками. Это один из множественных типов данных в Python. Помимо списков есть еще множества, кортежи и словари. Их мы тоже обсудим, но немного позже.
- Списки заслуживают особого внимания, поскольку в Python они обычно играют ту роль, которую в других языках программирования играют массивы.

Нормальное знакомство со списками

- Памятка.
- Список представляет собой упорядоченный набор значений. Причём значения, формирующие список, могут быть разного типа. Существуют различные способы создания списка. Самый простой, пожалуй, состоит в том, чтобы перечислить в квадратных скобках через запятую значения, входящие в список.
- Например, командой `nums = [1, 5, 10]`, а ссылка на него записывается в переменную `nums`. Нередко для создания списков используют функцию `list()`. Если в качестве аргумента функции передать текст, то результатом возвращается список из букв (символов) данного текста.

Нормальное знакомство со списками

- Возможны и другие варианты использования этой функции для создания списков – некоторые мы рассмотрим далее.
- Для определения длины списка используют функцию `len()`. Аргументом функции передаётся ссылка на список, а результатом функция возвращает количество элементов в списке. Доступ к элементу списка можно получить по индексу.
- Индекс указывается в квадратных скобках после имени списка. Индексация элементов начинается с нуля – то есть первый элемент в списке имеет нулевой индекс.

Нормальное знакомство со списками

- Индекс может быть отрицательным. В таком случае он определяет положение элемента в списке, начиная с конца списка. Индекс -1 соответствует последнему элементу в списке. У предпоследнего элемента индекс -2, и так далее.
- Также есть очень полезная операция, часто на практике выполняемая со списками, - речь идёт о получении среза. Например, имеется некоторый список, и нам нужно получить последовательность его элементов, начиная с элемента с индексом i и заканчивая элементом с индексом j .

Нормальное знакомство со списками

- В таком случае мы можем выполнить срез командой вида список $[i: j+1]$. Здесь мы указываем индекс элемента, начиная с которого выполняется срез, а затем через двоеточие – индекс элемента, который следует за последним из элементов, входящих в срез.
- Результатом выражения вида список $[i: j+1]$ является список из элементов, которые в исходном списке имеют индексы от i до j включительно. При этом сам исходный список не изменяется.

Нормальное знакомство со списками

- Инструкция вида список[i :] означает создание среза из элементов списка, начиная с элемента с индексом i и до конца списка.
- Инструкция в формате список [: $j+1$] возвращает результатом срез из элементов, начиная с первого элемента в списке и до элемента с индексом j включительно.
- Наконец, значением выражения список [:] является копия списка (срез состоит из всех элементов списка).

Нормальное знакомство со списками

- Памятка.
- Мы рассмотрели наиболее простые способы индексации элементов списка и получения среза. Вообще же соответствующие инструкции могут быть и более замысловатыми. Их мы рассмотрим немного позже.
- Также следует отметить, что существует ряд встроенных функций, предназначенных для работы со списками. Например, при работе с числовыми списками полезными могут оказаться функции `max()`, `min()` и `sum()`, предназначенные, соответственно, для вычисления наибольшего и наименьшего числа в списке, а также суммы элементов списка.

Нормальное знакомство со списками

- Функция `sorted()` позволяет отсортировать список (точнее, создаёт его отсортированную копию), а функция `reversed()` совместно с функцией `list()` даёт возможность создать список, в котором элементы следуют в обратном порядке (по сравнению с исходным списком).

Нормальное знакомство со списками

- Небольшие примеры с листингом кода:

Нормальное знакомство со списками

```
main.py x
1  nums = [5, 10, 1, 69, 42, 30]
2  print("Список чисел: ", nums)
3  print("Длина списка: ", len(nums))
4  print("Первый элемент: ", nums[0])
5  print("Последний элемент: ", nums[-1])
6  print("Наибольшее значение: ", max(nums))
7  print("Наименьшее значение: ", min(nums))
8  print("Сумма всех чисел списка: ", sum(nums))
9  print("Числа в обратном порядке: ", list(reversed(nums)))
10 print("Числа по убыванию: ", sorted(nums, reverse=True))
11 print("Числа по возрастанию: ", sorted(nums))
12 nums[0] = "Text"
13 print("Изменение значения первого элемента списка: ", nums[0])
14 print("Получение среза: ", nums[1: 4])
15 nums[1:-1] = ["A", "B"]
16 print("Замена части элементов списка: ", nums)
17 name = list(range(1, 5))
18 print(name)
19 nums[1:2] = []
20 print(nums)
21 del nums[len(nums)-1]
22 print(nums)
```

Нормальное знакомство со списками

- Проанализируем код этой программы и результат её выполнения. В программе командой `nums [5, 10, 1, 60, 25, 3]` создаётся список из чисел. Чтобы проверить содержимое списка, используем инструкцию `print("Список из чисел", nums)`. По умолчанию при отображении списка его элементы выводятся через запятую, а вся конструкция заключена в квадратные скобки. Длина списка (количество элементов в списке) вычисляется выражением `len(nums)`.
- Считывание значения первого (начального) элемента в списке выполняется с помощью инструкции `nums[0]`

Нормальное знакомство со списками

- Командой `list(reversed(nums))` вычисляется список получающийся из списка `nums` обращением порядка следования элементов. При этом создаётся новый список, а исходный список не меняется.

Нормальное знакомство со списками

- Подробности.
- Выражением `reversed(nums)` является объект итерационного (или итерируемого) типа (его элементы можно перебирать), который соответствует инвертированному списку `nums`. Этот итерируемый объект передается в качестве аргумента функции `list()`, в результате чего мы получаем новый список.

Нормальное знакомство со списками

- Для создания списка, отсортированного в порядке возрастания значений элементов, используем инструкцию `sorted(nums)`, а для сортировки в порядке убывания значений элементов задействована инструкция `sorted(nums, reverse = True)`.
- Здесь вторым аргументом функции `sorted()` передается инструкция `reverse = True`, являющаяся индикатором того, что сортировка должна выполняться в обратном порядке (в порядке убывания значений).

Нормальное знакомство со списками

- И в том и в другом случае список `nums` не меняется. Убеждаемся в этом с помощью команды `print(“Исходный список:”, nums)`.

Нормальное знакомство со списками

- Команда `nums[1]`="текст" показывает, как можно изменить значение элемента списка: второй по порядку (с индексом 1) элемент получает значение «текст» Здесь мы сталкиваемся с ситуацией, когда список содержит значения разных типов. При получении среза с помощью команды `nums[1: len(nums)-1]` получаем список из всех элементов списка `nums`, за исключением первого и последнего элементов.
- Мы учитываем, что количество элементов в списке `nums` может быть вычислено инструкцией `len(nums)`, а поскольку индексация элементов начинается с нуля, то индекс последнего элемента равен `len(nums)-1` (это тот элемент, который уже не попадает в срез).

Нормальное знакомство со списками

- На заметку
- Для последнего элемента вместо индекса $\text{len}(\text{nums})-1$ можно было использовать индекс -1 .

Нормальное знакомство со списками

- Заметка
- Когда переменной в качестве значения присваивается список, то на самом деле переменная получает в качестве значения ссылку на этот список. При присваивании переменной другого значения (например, другого списка) переменная получает значением новую ссылку.
- На исходных список она больше не ссылается. Если на этот исходный список в программе других ссылок нет, то такой список автоматически удаляется из памяти.

Знакомство с условным оператором

- Условный оператор позволяет выполнять разные блоки команд в зависимости от истинности или ложности некоторого условия. Синтаксис условного оператора вы все знаете.
- После ключевого слова `if` указывается выражение (условие), истинность которого проверяется. Далее следует двоеточие и блок команд.
- Каждая команда в блоке выделяется отступом (обычно, это четыре пробела). Команды данного блока выполняются в том случае, если условие истинно.

Знакомство с условным оператором

- После первого блока команд следует ключевое слово `else`, которое завершается двоеточием. Ключевое слово `else` по горизонтали размещается на том же расстоянии, что и ключевое слово `if`. Под ключевым словом `else` размещается блок команд, выполняемых при ложном условии.
- Команды выделяются отступом (обычно в четыре пробела).

Знакомство с условным оператором

- Подробности
- Здесь мы впервые сталкиваемся с механизмом, используемым в Python для структурирования программного кода. В таких языках программирования, как java, c++, c# и некоторых других, блоки команд выделяются с помощью фигурных скобок. В языке Python код структурируется с помощью отступов. В принципе, количество пробелом при выделении блока команд может быть любым, но оно должно быть одним и тем же для всей программы. Однако рекомендуется выполнять отступ в четыре пробела. Именно такие отступы мы и будем использовать в программных кодах.

Знакомство с условным оператором

- Пример кода с чётностью/нечётностью чисел.
- При запуске программы на выполнение появляется сообщение с предложением ввести целое число. Далее все зависит от того, чётное или нечётное число вводит пользователь. Ниже показано, каким будет результат выполнения программы, если пользователь вводит чётное число.

Знакомство с условным оператором

- Памятка
- Мы рассмотрели достаточно простую форму условного оператора. Но вообще ситуация не такая тривиальная. Например условные операторы могут быть вложенными, причём для таких случаев есть специальная форма условного оператора.
- Существуют некоторые особенности, связанные с описанием условия в условном операторе. Всё это мы рассмотрим типа позже, но вы и так опять таки всё знаете.

Знакомство с оператором цикла

- Нередко случается так, что необходимо определённое количество раз выполнить некоторый блок команд. В таких случаях полезно использовать операторы цикла. Далее мы кратко рассмотрим оператор цикла `while`. Синтаксис следующий: `while условие: команды`
- После ключевого слова `while` указывается условие, двоеточие и затем блок команд, которые выполняются, пока условие истинно. Команды, относящиеся к оператору цикла, выделяются отступом в четыре пробела.

Знакомство с оператором цикла

- Выполняется оператор цикла `while` следующим образом. Сначала проверяется условие, указанное после ключевого слова `while`. Если условие истинно, то выполняются команды в теле оператора цикла (т.е команды из блока, относящегося к оператору цикла). Затем снова проверяется условие. Если оно истинно, снова выполняются команды в теле оператора цикла, опять проверяется условие и так далее. Процесс продолжается до тех пор, пока при проверке условия не окажется, что оно ложно.
- В таком случае выполнение оператора цикла завершается, и выполняется команда, следующая после оператора цикла.

Знакомство с оператором цикла

- Оператор цикла для вычисления суммы:

```
24
25     text = int(input("Укажите верхнюю границу суммы: "))
26     s = 0
27     k = 0
28     while k < text:
29         k = k + 1
30          s = k + s
31     print("Сумма числе от 1 до", text, "равна", s)
```

Знакомство с оператором цикла

- Командами $s = 0$ и $k = 0$ в программе создаются две переменные с начальными нулевыми значениями. В первую предполагается записывать значение суммы натуральных чисел, а с помощью второй переменной (которую мы будем называть индексной) выполняется подсчёт количества слагаемых, добавленных в сумму.
- В операторе цикла после ключевого слова `while` в качестве условия указано выражение $k < n$. Значение выражения равно `True`, если значение переменной k меньше значения переменной n . Если условие истинно, то команда $k = k + 1$ в теле оператора цикла увеличивает на единицу текущее значение переменной k , после чего команда $s = s + k$ прибавляет новое значение k к сумме.

Знакомство с оператором цикла

- Второй вариант исполнения:

```
text = input("Укажите количество слагаемых: ")
txt = "1"
k = 1
while str(k) != text:
    k = k + 1
    txt = txt + "+" + str(k)
print(txt, "=", eval(txt))
```

Знакомство с оператором цикла

- Как и в предыдущем случае, в этой программе мы сначала учитываем значение для количества слагаемых в сумме (верхняя граница суммы). Специфика ситуации в том, что считанное значение к целочисленному формату не преобразуется, а сохраняется в текстовом виде.
- Таким образом, переменная `n`, значение которой определяется командой `n = input("Укажите количество слагаемых:")`, ссылается на текстовое значение.

Знакомство с функциями

- Функция представляет собой именованный блок программного кода, который можно выполнить, вызвав функцию. При вызове функции могут передаваться параметры, которые называются аргументами функции. В Python функции описываются исключительно просто. Общий шаблон описания функции представлен ниже.
- Def имя (аргументы):
 - Команды

Знакомство с функциями

- Начинается описание с ключевого слова `def`, после которого указывают название функции, её аргументы (в круглых скобках), двоеточие и, наконец, блок команд, которые выполняются при вызове функции. Как и в случае с условным оператором и оператором цикла, блок команд, относящихся к телу функции, выделяется отступом (в четыре пробела). Для вызова функции необходимо в соответствующей команде указать имя функции и, при необходимости, аргументы, которые передаются функции (в круглых скобках после имени функции). Вызов функции означает, что выполняются команды из тела функции, причём в качестве аргументов используются те значения, которые переданы функции при вызове

Знакомство с функциями

- Функция может возвращать значение, а может не возвращать. Если функция не возвращает значение, то речь идет о выполнении определенного набора команд при вызове функции. Если функция возвращает результат, то инструкцию вызова функции можно использовать как операнд в более сложном выражении. При этом инструкция вызова функции отождествляется со значением, возвращаемым функцией. Чтобы в теле функции определить значение, которое функция возвращает, используют инструкцию `return`, после которой указывается возвращаемое функцией значение.

Знакомство с функциями

- Выполнение инструкции `return` в теле функции приводит к завершению выполнения кода функции. Поэтому очень часто (но не всегда) команда с `return`-инструкцией в теле функции является последней. Инструкция `return` может использоваться и в функции, которая не возвращает значение. В таком случае после инструкции `return` никакое значение не указывается, а ее выполнение просто приводит к завершению выполнения кода функции.

Знакомство с функциями

- Описание функции может размещаться в любом месте программы, но только до того места, где функция впервые вызывается. В листинге ниже представлен программный код, в котором объявляются (а затем вызываются) две функции.

Знакомство с функциями

Знакомство с функциями

- Программный код начинается с описания функции, которая называется `show()` (здесь и далее названия функций будут приводиться с пустыми круглыми скобками). В описании функции у нее объявлен один аргумент, который называется `txt`. Неявно предполагается, что это текстовое значение. В теле функции командой `syms=sorted(list(txt))` на основе текстового (так мы думаем) аргумента создается список, состоящий из символов текстовой строки, причем список сортируется в порядке возрастания значений (для символов сравниваются их коды в кодировочной таблице). После создания списка он отображается командой `print(syms)`.

На заметку

- Переменные, которые «появляются» в теле функции, включая аргумент (или аргументы) функции, доступны только в теле функции. Такие переменные называются локальными.

Знакомство с Python

- Примером вызова функции `show()` служит команда `show("Python")`, в которой функции передается текстовая строка "Python". В результате на основе текста формируется упорядоченный список и отображается в области вывода. Функция для вычисления суммы квадратов натуральных чисел называется `sqsum()`, и ее аргумент, обозначающий верхнюю границу суммы, обозначен как `n`.

Подробности

- Речь идет о функции для вычисления суммы квадратов натуральных чисел $1^2 + 2^2 + 3^2 + \dots + n^2 = n(n + 1)(2n + 1) / 6$. Аргументом функции передается значение n . Результатом функция возвращает значение для суммы $1^2 + 2^2 + 3^2 + \dots + n^2$, причем формула $n(n + 1)(2n + 1) / 6$ при вычислении результата функции не используется.

Знакомство с Python

- После того как список из квадратов натуральных чисел создан и ссылка на него записана в переменную `nums`, командой `return sum(nums)` вычисляется сумма квадратов натуральных чисел, и полученное значение возвращается в качестве результата функции.

Еще на заметку

- Здесь мы использовали встроенную функцию `sum()`, которая позволяет вычислить сумму значений элементов списка, переданного аргументом функции.

Знакомство с Python

- Функция для вычисления суммы квадратов чисел используется в команде `print(«Сумма квадратов числе от 1 до», str(m) + “:”,sqsum(m))`. Функции `sqsum()` в качестве аргумента передается переменная `m` со значением 10. Легко проверить, что значение для суммы квадратов вычислено правильно.

Резюме

- Программа – это последовательность команд. В процессе выполнения программы команды выполняются одна за другой.
- В языке Python есть несколько полезных встроенных функций, предназначенных для ввода и вывода данных: вывод данных в окно интерпретатора осуществляется с помощью функции `print()`, для ввода данных используют функцию `input()`. Функция `eval()` позволяет вычислить выражение в текстовой строке, которая передается аргументом функции.

Резюме

- Переменные в Python не имеют типа. Для использования переменной в программе ей просто присваивается значение. На разных этапах выполнения программы переменная может ссылаться на данные разных типов. Для удаления переменной используют инструкцию `del`, после которой указывается имя удаляемой переменной.
- В программе могут использоваться комментарии. Комментарий начинается с символа `#` и игнорируется при выполнении программы.

Резюме

- Целочисленные значения относятся к типу `int`, текстовые значения относятся к типу `str`. Для преобразования к целочисленному типу используют функцию `int()`, для преобразования к текстовому типу используют функцию `str()`. Текстовые литералы заключаются в двойные или одинарные кавычки.

Резюме

- Список представляет собой упорядоченный набор элементов, которые, в принципе, могут относиться к разным типам. Создается список перечислением значений элементов в квадратных скобках или, например, с использованием функции `list()`. Могут использоваться и иные подходы (например, с помощью генератора списка). Длина списка определяется с помощью функции `len()`. Для получения доступа к элементу массива после имени массива в квадратных скобках указывается индекс элемента.

Резюме

- Индексация элементов начинается с нуля. Отрицательный индекс используется для определения позиции элемента, начиная с конца списка. При работе со списками можно получать срезы, представляющие собой список из элементов исходного массива. Индекс первого входящего в срез элемента и первого не входящего в срез элемента указываются в квадратных скобках после имени массива и разделяются двоеточием.

Резюме

- Условный оператор `if` позволяет выполнять разные блоки команд в зависимости от истинности или ложности определенного условия. Оператор цикла `while` используется для многократного выполнения определенного блока команд. Команды выполняются, пока истинно условие, указанное в операторе цикла.

Резюме

- Функция представляет собой именованный блок программного кода, который можно вызвать по имени. У функции могут быть аргументы. При описании функции после ключевого слова `def` указывается имя функции, ее аргументы (в круглых скобках) и через двоеточие — блок команд, которые выполняются при вызове функции. Инструкция `return` в теле функции завершает выполнение функции. Если после этой инструкции указано значение, оно возвращается результатом функции. Для вызова функции указывается ее значение и, если необходимо, в круглых скобках аргументы, которые передаются функции.