Moscow Coding School

Python как первый язык

Преподаватель: Захарчук Сергей Сергеевич

Сегодня

- То_что_хотели_знать=[’Операторы сравнения’, ‘Операторы присваивания’, ‘Логические операторы’, ‘Операторы принадлежности’, ‘Операторы идентичности’, ‘Приоритеты операторов’, ‘Бинарные операторы’]
- Функции_Python=[‘Именные функции’, ‘инструкция def’, ‘Анонимные функции’, ‘инструкция lambda’]
- Понятие модулей (повторение) + встроенные функции

Операторы сравнения

Предположим, переменная a = 10, а переменная b = 20.

ОПЕРАТОР	ОПИСАНИЕ	ПРИМЕР
==	проверяет одинаково ли значение операндов, если одинаковы – то условие является истинной (true);	(a == b) false
!=	проверяет одинаково ли значение операндов, если НЕ одинаковы – то условие является истинной (true);	(a != b) true
<>	проверяет одинаково ли значение операндов, если НЕ одинаковы – то условие является истинной (true);	(a <> b) true (этот оператор аналогичен !=)
>	проверяет значение левого оператора, если он больше, чем правый – то условие является истинной;	(a > b) false
<	проверяет значение левого оператора, если он меньше, чем правый – то условие является истинной;	(a < b) true.
>=	проверяет значение левого оператора, если он больше или равен правому – то условие является истинной;	(a >= b) false
<=	проверяет значение левого оператора, если он меньше или равен правому – то условие является истинной;	(a <= b) true.

Операторы присваивания

Предположим, переменная a = 10, а переменная b = 20.

ОПЕРАТОР	ОПИСАНИЕ	ПРИМЕР
=	простой оператор присваивания, присваивает значение с правой стороны операнду с левой стороны;	c = a + b присвоит результат a + b переменной c
+=	оператор присваивания “сложение И”, добавляет значение правого оператора левому, и присваивает результат левому операнду;	c += a аналогичен c = c + a
-=	оператор присваивания “вычитание И”, вычитает правый оператор из левого и результат присваивает левому операнду;	c -= a аналогичен c = c – a
*=	оператор присваивания “умножение И”, умножает правый оператор на левый и присваивает результат левому операнду;	c *= a аналогичен c = c * a
/=	оператор присваивания “деление И”, делит левый оператор на правый и присваивает результат левому операнду;	c /= a аналогичен c = c / a
%=	оператор присваивания “модуль И”, получает модуль из обоих операторов и присваивает значение левому операнду;	c %= a аналогичен c = c % a
**=	оператор присваивания “возведение в степень И”, выполняет возведение в степень операндов и присваивает рузультат левому операнду;	c **= a аналогичен = c ** a
//=	оператор присваивания “деление с остатком И”, выполняет деление с остатком и присваивает результат левому операнду;	c //= a аналогичен c = c // a

Логические операторы

Предположим, переменная a = 10, а переменная b = 20.

В языке **Python** поддерживаются следующие логические операторы *and (и), or (или), not (не)*.

OPERATOR	DESCRIPTION	EXAMPLE
and	называется логическим оператором AND (И). Если оба операнда = true выражение будет так же true;	(a and b) is true.
or	называется логическим оператором OR (ИЛИ). Если хотя бы один из двух операторов не пустой (не равен 0) – выражение истинно;	(a or b) is true.
not	называется логическим оператором NOT (НЕ). Используется для обратного изменения логического результата выражения. Если выражение истинно – с помощью этого оператора оно станет ложным (false).	not(a and b) is false.

Операторы принадлежности

OPERATOR	DESCRIPTION	EXAMPLE
in	Считается истиной (true), если находит переменную в заданной последовательности, и ложью (false) в противном случае;	for x in y: Если x принадлежит последовательности
not in	Считается истиной (true), если не находит переменную в заданной последовательности, и ложью (false) в противном случае.	for x not in y: Если x НЕ принадлежит последовательности

Операторы идентичности

Операторы идентичности сравнивают расположение двух объектов в памяти. Таких операторов два:

OPERATOR	DESCRIPTION	EXAMPLE
is	Считается истиной (true), если переменные по обе стороны от оператора указывают на один объект, и ложью (false) в противном случае;	If x is y: … if id(x) equals id(y).
is not	Считается ложью (false), если переменные по обе стороны от оператора указывают на один объект, и истиной (true) в противном случае/	x is not y: … if id(x) is not equal to id(y).

Приоритеты операторов

OPERATOR	DESCRIPTION
**	Возведение в степень
~ + -	Комплиментарный оператор
* / % //	Умножение, деление, деление по модулю, целочисленное деление.
+ -	Сложение и вычитание.
>> <<	Побитовый сдвиг вправо и побитовый сдвиг влево.
&	Бинарный "И".
^ |	Бинарный "Исключительное ИЛИ" и бинарный "ИЛИ"
<= < > >=	Операторы сравнения
<> == !=	Операторы равенства
= %= /= //= -= += *= **=	Операторы присваивания
is is not	Тождественные операторы
in not in	Операторы членства
not or and	Логические операторы

Бинарные операторы

Оператор	Описание	Примеры
&	Бинарный "И" оператор, копирует бит в результат только если бит присутствует в обоих операндах.	(a & b) даст нам 12, которое в двоичном формате выглядит так 0000 1100
|	Бинарный "ИЛИ" оператор копирует бит, если тот присутствует в хотя бы в одном операнде.	(a | b) даст нам 61, в двоичном формате 0011 1101
^	Бинарный "Исключительное ИЛИ" оператор копирует бит только если бит присутствует в одном из операндов, но не в обоих сразу.	(a ^ b) даст нам 49, в двоичном формате 0011 0001
~	Бинарный комплиментарный оператор. Является унарным (то есть ему нужен только один операнд) меняет биты на обратные, там где была единица становиться ноль и наоборот.	(~a) даст в результате -61, в двоичном формате выглядит 1100 0011.
<<	Побитовый сдвиг влево. Значение левого операнда "сдвигается" влево на количество бит указанных в правом операнде.	a << 2 в результате даст 240, в двоичном формате 1111 0000
>>	Побитовый сдвиг вправо. Значение левого операнда "сдвигается" вправо на количество бит указанных в правом операнде.	a >> 2 даст 15, в двоичном формате 0000 1111

Именные функции, инструкция def

Функция в python - объект, принимающий аргументы и возвращающий значение. Обычно функция определяется с помощью инструкции **def**.

Определим простейшую функцию:

```
>>> def add(x, y):
...     return x + y
```

Теперь мы ее можем вызвать:

```
>>> add(1, 10)
11
>>> add('abc', 'def')
'abcdef’
```

Именные функции, инструкция def

pass – ничего не делает, пустая функция

```
>>> def func():
...     pass
```

Именные функции, инструкция def

```
>>> def newfunc(n):
        ... def myfunc(x):
                ... return x + n
        ... return myfunc ...
>>> new = newfunc(100) # new - это функция
>>> new(200)
300
```

Именные функции, инструкция def

Для всех параметров функций можно указывать значения по-умолчанию, это дает возможность вызвать функцию с меньшим числом параметров. Например, у нас есть функция для авторизации пользователя на сайте:

```
>>> def login(username="anonymous", password=None):
        ... """Тут какие-то действия"""
        ... pass
```

Вызвать эу функцию мы можем одним из нижеприведенных способов

```
>>> login("root", "ujdyzysqgfhjkm")
>>> login("guest")
>>> login()
```

Можем указать какой из параметров мы передаем, указав его имя в явном виде:

```
>>> login(password="nobody@mail.com")
```

Аргументы функции

Функция может принимать произвольное количество аргументов или не принимать их вовсе. Также распространены функции с произвольным числом аргументов, функции с позиционными и именованными аргументами, обязательными и необязательными.

```
>>> def func(a, b, c=2):
        ... return a + b + c
>>> func(1, 2)
5
```

Аргументы функции

```
>>> func(a=1, b=3)
6
>>> func(a=3, c=6)
Traceback (most recent call last):
        File "", line 1, in func(a=3, c=6)
TypeError: func() takes at least 2 arguments (2 given)
```

Аргументы функции

Функция также может принимать переменное количество позиционных аргументов, тогда перед именем ставится *:

```
>>> def func(*args):
        ... return args ...
>>> func(1, 2, 3, 'abc')
(1, 2, 3, 'abc')
>>> func() ()
>>> func(1)
(1,)
# args - это кортеж (неизменяемый список!)
```

Аргументы функции

Функция может принимать и произвольное число именованных аргументов, тогда перед именем ставится **:

```
>>> def func(**kwargs):
        ... return kwargs
>>> func(a=1, b=2, c=3)
{'a': 1, 'c': 3, 'b': 2}
>>> func()
{}
>>> func(a='python')
{'a': 'python'}
```

Аргументы функции

[Важно!] Функции в Python являются объектами, соответственно, их можно возвращать из другой функции или передавать в качестве аргумента. Также следует помнить, что функция в python может быть определена и внутри другой функции.

Анонимные функции, инструкция lambda

- Анонимные функции могут содержать лишь одно выражение
- Выполняются они быстрее.
- Создаются с помощью инструкции **lambda**.
- Не обязательно присваивать переменной, как делали мы инструкцией def func():
- lambda функции, в отличие от обычной, не нужен return

Анонимные функции, инструкция lambda

```
>>> func = lambda x, y: x + y
>>> func(1, 2) 3
>>> func('a', 'b')
'ab'
>>> (lambda x, y: x + y)(1, 2)
3
>>> (lambda x, y: x + y)('a', 'b')
'ab'
```

Анонимные функции, инструкция lambda

```
>>> func = lambda *args: args
>>> func(1, 2, 3, 4)
(1, 2, 3, 4)
```

Задача 1

Обьявить функцию, принимающую на вход список, дописать к этому списку его первые 3 элемента и вывести результат. Причем если список не задан, то по умолчанию он равен пустому.

Входные данные	Выходные данные
lister=[1, 2, 3]	[1, 2, 3, [1, 2, 3]]
lister=[-1, 2, 3, 4, 5, 6, 7]	[-1, 2, 3, 4, 5, 6, 7, [-1, 2, 3]]
lister=[-1, 4, -2 ,2, 1, 2]	[-1, 4, -2 ,2, 1, 2, [-1, 4, -2]]

Начало программы:

```
>> def myfunc(lister = []):
        pass
```

Задача 2

Обьявить функцию, суммирующую все аргументы на входе

Входные данные	Выходные данные
myfunc(1, 2, 3)	6
myfunc(1, 2, 3, 4)	10
myfunc(-1, 4, -2 ,2, 1, 2)	6

Начало программы:

```
>> def myfunc():
        pass
```

Понятие модулей

Каждую из задач(в идеале) решает отдельный модуль.

[*] При построении модульной структуры программы важнее составить такую композицию модулей, которая позволила бы свести к минимуму связи между ними.

Набор классов и функций, имеющий множество связей между своими элементами, было бы логично расположить в одном модуле.

Модули должно быть легче использовать, чем написать заново. Это значит, что модуль должен иметь удобный интерфейс: набор функций, классов и констант, который он предлагает своим пользователям.

Понятие модулей

В программе на Python модуль представлен объектом-модулем, атрибутами которого являются имена, определенные в модуле

```
>>> import datetime
>>> d1 = datetime.date(2004, 11, 20)
```

В результате работы оператора import в текущем пространстве имен появляется объект с именем datetime.

Модули Python

Модуль оформляется в виде отдельного файла с исходным кодом. Стандартные модули находятся в каталоге, где их может найти соответствующий интерпретатор языка. Пути к каталогам, в которых Python ищет модули, можно увидеть в значении переменной sys.path:

```
>>> sys.path
['', '/usr/local/lib/python23.zip', '/usr/local/lib/python2.3',
'/usr/local/lib/python2.3/plat-linux2', '/usr/local/lib/python2.3/lib-tk',
'/usr/local/lib/python2.3/lib-dynload', '/usr/local/lib/python2.3/site-
packages']
```

[*] В последних версиях Python модули можно помещать и в zip-архивы для более компактного хранения (по аналогии с jar-архивами в Java).

Модули Python

- При запуске программы поиск модулей также идет в текущем каталоге. (Нужно внимательно называть собственные модули, чтобы не было конфликта имен со стандартными или дополнительно установленными модулями.)
- Подключение модуля к программе на Python осуществляется с помощью оператора import. У него есть две формы: import и from-import:

```
import os

import pre as re

from sys import argv, environ

from string import *
```

Модули Python

Повторный импорт модуля происходит гораздо быстрее, так как модули кэшируются интерпретатором. Загруженный модуль можно загрузить еще раз (например, если модуль изменился на диске) с помощью функции reload():

```
import mymodule
...
reload(mymodule)
```

Встроенные функции

В среде Python без дополнительных операций импорта доступно более сотни встроенных объектов, в основном, функций и исключений. Для удобства функции условно разделены по категориям:

1. Функции преобразования типов и классы (см. 1-2 лекции): str, repr, int, list, tuple, long, float, complex, dict, super, file, bool, object

2. Числовые и строковые функции (см. 1-2 лекции): abs, divmod, ord, pow, len, chr, unichr, hex, oct, cmp, round, unicode

3. Функции обработки данных: apply, map, filter, reduce, zip, range, тут же есть и xrange, max, min, iter, enumerate, sum

4. Функции определения свойств: hash, id, callable, issubclass, isinstance, type

5. Функции для доступа к внутренним структурам: locals, globals, vars, intern, dir

6. Функции компиляции и исполнения: eval, execfile, reload, __import__, compile

Функции ввода-вывода: input, raw_input, open

Функции для работы с атрибутами: getattr, setattr, delattr, hasattr

Функции-"украшатели" методов классов: staticmethod, classmethod, property

Прочие функции: buffer, slice

Встроенные функции

Уточнить назначение функции, ее аргументов и результата можно в интерактивной сессии интерпретатора Python:

```
>>> help(len)
```

```
Help on built-in function len: len(...) len(object) -> integer
Return the number of items of a sequence or mapping.
```

Или так:

```
>>> print(len.__doc__)
len(object) -> integer
Return the number of items of a sequence or mapping.
```

Функции определения свойств

Эти функции обеспечивают доступ к некоторым встроенным атрибутам объектов и другим свойствам. Следующий пример показывает некоторые из этих функций:

```
>>> s = "abcde"
>>> s1 = "abcde"
>>> s2 = "ab" + "cde"
>>> print("hash:", hash(s), hash(s1), hash(s2))
```

hash: -1332677140 -1332677140 -1332677140

```
>>> print("id:", id(s), id(s1), id(s2))
```

id: 1076618592 1076618592 1076618656

Здесь, можно увидеть, что для одного и того же строкового литерала "abcde" получается один и тот же объект, тогда как для одинаковых по значению объектов вполне можно получить разные объекты.

Функции для доступа к внутренним структурам

- В современной реализации языка Python глобальные и локальные переменные доступны в виде словаря благодаря функциям globals() и locals(). Правда, записывать что-либо в эти словари не рекомендуется.
- Функция vars() возвращает таблицу локальных имен некоторого объекта (если параметр не задан, она возвращает то же, что и locals()). Обычно используется в качестве словаря для операции форматирования:

```
>>>a=1, b=2, c=3
>>>print ("%(a)s + %(b)s = %(c)s" % vars())
1 + 2 = 3
```

Функции компиляции и исполнения

Функция reload() уже рассматривалась, а из остальных функций этой категории особого внимания заслуживает eval()

Эта функция вычисляет переданное ей выражение

```
>>> a=2
>>> b=3
>>> for op in "+-*/%":
        ... e = "a " + op + " b"
        ... print(e, "->", eval(e))
...
a + b -> 5
a - b -> -1
a * b -> 6
a / b -> 0.6666666666666666
a % b -> 2
```

Функции компиляции и исполнения

- У функции eval() кроме подлежащего вычислению выражения есть еще два параметра - с их помощью можно задать глобальное и локальное пространства имен, из которых будут разрешаться имена выражения.

```
for op in "+-*/%":
        e = "a " + op + " b"
        print(e, "->", eval(e, {'a': 2, 'b': 3}))
```

Функцией eval() легко злоупотребить.

Нужно стараться использовать ее только тогда, когда без нее НЕ обойтись!!!

Из соображений безопасности не следует применять eval() для аргумента, в котором присутствует непроверенный ввод от пользователя.

Функции ввода-вывода

Функции input() и raw_input() используются для ввода со стандартного ввода. В серьезных программах их лучше не применять. Функция open() служит для открытия файла по имени для чтения, записи или изменения.

```
f = open("file.txt", "r", 1)
for line in f:
        print(line)
f.close()
```

Подробнее см. 1 лекцию

Встроенные функции

В след. лекциях продолжение следует:

Функции для работы с атрибутами: getattr, setattr, delattr, hasattr

Функции-"украшатели" методов классов: staticmethod, classmethod, property

Прочие функции: buffer, slice

Домашнее задание №3

1 Представьте матрицу 3x3 в виде списка, содержащего три других списка, по одному в ряд:

```
>>> mat = [
    ... [1, 2, 3],
    ... [4, 5, 6],
    ... [7, 8, 9],
    ... ]
```

2* Оформите эту программу с функцией, которая на вход принимает размер матрицы n и m, а заполнение матрицы происходит при помощи функции **random.randint**(1, 9) из модуля randon

Домашнее задание №3

3 Пользователь делает вклад в размере ’x’ рублей сроком на ‘years’ лет под 13% годовых (Это значит, что каждый год размер его вклада увеличивается на 13%. Эти деньги прибавляются к сумме вклада, и на них в следующем году тоже будут проценты).

Написать функцию storage, принимающая аргументы ‘x’ и ‘years’, и возвращающую сумму, которая будет на счету пользователя.

Домашнее задание №3

4* Написать программу- переводчик из RU-EN в EN-RU
На вход подается файл ru-en.txt с таким содержимым:
Яблоко - Apple
Апельсин - Orange
Оранжевый - Orange
Груша - Pear
Виноград – Grapes
Нужно получить файл словаря en-ru.txt