Очень краткое введение в python

Н. Ю. Золотых

ННГУ, ВМК, 17 декабря 2008

Содержание

1 Введение **4**

1.1 Сфера применения 7

2 python как калькулятор **9**

3 Типы данных **11**

3.1 Числа 11

3.2 Строки 12

3.3 Списки 13

3.3.1 Другие методы и функции 14

3.4 Кортежи 15

3.5 Словари 16

4 Управляющие конструкции **17**

4.1 while 17

4.2 for 18

4.3 if 19

5 Функции 20

6 Присваивания по ссылке и по значению 22

7 Работа с модулями 25

8 Ввод/вывод 26

9 Научные вычисления: `scipy` и `numpy` 27

9.1 `scipy` 29

10 Научная графика 31

11 Элементы ООП 32

12 Элементы функционального программирования 33

1. Введение

python (Питон, Пайтон) — высокоуровневый язык программирования общего назначения.

Поддерживает несколько парадигм программирования (структурное, объектно-ориентированное, функциональное и др.)

Основные черты:

- динамическая типизация (тип переменной определяется во время выполнения и может меняться в процессе работы программы),
- автоматическое управление памятью,
- интроспекция (возможность получения информации об объекте),
- обработка исключений,
- поддержка многопоточных вычислений,
- высокоуровневые структуры данных.

Код организуется в функции и классы, которые, в свою очередь, могут быть организованы в модули, а те — в пакеты.

Наиболее популярные реализации Питона — интерпретаторы, позволяющие использовать его в интерактивном режиме.

Другие важные черты:

- многоплатформенный (Microsoft Windows, все варианты UNIX, Mac OS, Symbian, ...),
- свободно распространяемый, с открытым кодом,
- богатая стандартная библиотека (работа с операционной системой, сетевыми протоколами и форматами интернета, XML, регулярными выражениями, мультимедийными форматами, криптогшрафическими протоколами, архивами, ...),
- *огромное* количество дополнительных библиотек (веб, базы данных, обработка изображений, обработка текста, численные методы, системные вызовы, графика, разработка игр, ...),
- программный интерфейс для написания собственных модулей на C и C++,
- возможность встраивания интерпретатора **python** в приложения.

Создан в 1990 г. (по другим источникам — 1991 г.) Гвидо ван Россумом (Guido van Rossum).

Последние релизы:

- **python** 3.0 (3 декабря 2008),
- **python** 2.6.1 (4 декабря 2008).

Ветки 2.x и 3.x не совместимы.

Имеется утилита «2to3».

Сайт: http://www.python.org

1.1. Сфера применения

- веб-программирование,
- графика,
- разработка ПО (системы по сборке проектов, управление версиями, среды визуальной разработки),
- научное ПО,
- игры,
- огромное число проектов использует **python** как скриптовый язык (язык сценариев),
- ...

python может использоваться

- как основной язык,
- для интеграции приложений,
- для написания прототипов.

python интенсивно используется в компаниях и организациях:

- Google,
- Яndex,
- Apple,
- NASA,
- CERN,
- Nokia,
- DreamWorks,
- ...

2. python как калькулятор

```
10./3.          # Деление чисел с плавающей запятой
10/3.           # То же
10/3            # Целочисленное деление
5*3             # Умножение
5**3            # Возведение в степень
12*(34-56)/7    # Арифметическое выражение
_**89           # _ - Результат последней операции
10%3            # Остаток от деления
(-10)/3         # Ух-ты!
(-10)%3         # Вот как!
x = 5           # Присваивание
y = x**3        # Результат не отображается
x, y
print x, y
```

Сравнение: ==, !=, <, >, <=, >=

```
1 + 2J          # Комплексное число
1 + 2j          # Так тоже можно, но не 1. + 2I или 1. + 2i
z = 1 + 2j
z.real          # Python - объектно-ориентированный язык
z.imag          # Python - объектно-ориентированный язык
z.conjugate()   # Вызов метода
2**100          # Большая арифметика (целые числа произвольной длины)!
```

3. Типы данных

Узнать тип можно с помощью вызова функции `type(a)`.

3.1. Числа

```
type(213)      # int   (4 байтовое знаковое целое)
type(2**100)   # long  (целое произвольной длины)
type(1.2)      # float (8-байтовое вещественное число с плавающей запятой)
type(1 + 2j)   # complex (16-байтовое комплексное число с плавающей запятой)
```

3.2. Строки

```
s1 = "Python"
type(s1)        # str
s2 = 'Питон'
s = "Язык программирования 'Питон'"
s1
s2
print s1, s2
s1 + s2         # Конкатенация
s[0]            # 1-й символ
s[1]            # 2-й символ
s[-1]           # Последний символ
s[-2]           # Предпоследний символ
len(s)          # Длина
s[0:4]          # Подстрока
s[0] = 'я'      # Нельзя
```

3.3. Списки

Список — изменяемый (mutable) контейнер. Обращение к элементу списка — по индексу.

```
x = [1, 2, 3, 'four']
type(x)                  # list
x[0]
x[1]
x[-1]
len(x)
range(5)                 # Список [0, 1, 2, 3, 4]
range(1, 5)              # Список [1, 2, 3, 4]
range(begin, end, step)
x[1:3]                   # Подсписок (slice)
x[:3]
x[1:]
x[3] = 4                 # Можно!
x[3, 4] = [4, 5, 6]      # Можно!
```

```
y1 = ['January', 'February']
y2 = ['March', 'April', 'May']
y1 + y2                          # Конкатенация
2*y1                             # Конкатенация
```

3.3.1. Другие методы и функции

```
lst.append(item)                 # Вставить в конец
lst.pop()                        # Удалить последний и возвратить значение
lst.insert(indx, item)           # Вставка в указанную позицию
lst.count(item)                  # Найти количество вхождений
lst.remove(item)                 # Удалить первое вхождение
lst.sort()                       # Сортировать
lst.reverse()                    # Записать в обратном порядке
```

3.4. Кортежи

Кортеж (tuple) — неизменяемый (immutable) контейнер. Обращение к элементу кортежа — по индексу.

```
x = (1, 2, 3, 'four')   # Скобки можно опустить
type(s)                 # tuple
x[0]
x[1]
x[-1]
len(x)
x[1:3]                  # Подкортеж
x[:3]
x[1:]
x[3] = 4                # Нельзя!
x[3, 4] = [4, 5, 6]     # Нельзя!
y1 + y2                 # Конкатенация
```

3.5. Словари

Словарь — изменяемый контейнер. Обращение к элементу словаря — по имени элемента.

```
person = {'FirstName': 'Isac', 'LastName': 'Newton'}
type(person)                # dict
person[0]                   # Нельзя!
person['FirstName']         # Индексация по имени
person['FirstName'] = 'Isaac'
person['Profession'] = 'Scientist'
```

4. Управляющие конструкции

4.1. while

```
a = 12
b = 8
while b != 0:
    tmp = a%b
    a = b
    b = tmp
print a
```

А лучше так:

```
a = 12
b = 8
while b != 0:
    a, b = b, a%b
print a
```

Имеются `continue`, `break`.

4.2. for

```
for k in range(1, 10):                  # или xrange(1, 10)
    print "%d  %d" % (k, k*k)

for k in "letters":
    print k
```

4.3. if

```
x = 1.5

if x < 0 or x > 10:
    y = 0.
elif x < 1:
    y = x
elif x < 2:
    y = x*x
else:
    y = 4

print y
```

5. Функции

Файл `fibo.py`

```
# Печатает числа Фибоначчи, не превосходящие B

def fibo(B = 100):
    a, b = 0, 1
    while b <= B:
        print b
        a, b = b, a + b
```

Файл gcd.py

```
# НОД чисел a, b

def gcd(a, b):
    a = abs(a)
    b = abs(b)
    while b != 0:
        a, b = b, a%b
    return a

print gcd(675222842, 709073246)
```

6. Присваивания по ссылке и по значению

```
a = b
```

- Атомные значения присваиваются по значению
- Контейнеры присваиваются по ссылке

```
a = 1
b = a
b = 2
print a, b                    # 1 2
```

```
a = [0, 1, 2]
b = a
b[2] = 222
print a                       # [0, 1, 222]
print b                       # [0, 1, 222]
```

Если объект нигде не используется, он удалится автоматически.

Простое и глубокое копирование

Файл xcopy.py

```
from copy import *
A = [0, 1, [2, 3, [4, 5]]]
B = A
print A                          # [0, 1, [2, 3, [4, 5]]]
print B                          # [0, 1, [2, 3, [4, 5]]]
B[1] = 111
B[2][1] = 333
print A                          # [0, 111, [2, 333, [4, 5]]]
print B                          # [0, 111, [2, 333, [4, 5]]]
C = copy(A)
C[1] = -1
C[2][2] = 222
print A                          # [0, 111, [2, 333, 222]]
print C                          # [0, -1, [2, 333, 222]]
D = deepcopy(A)
D[2][0] = 0
print A                          # [0, 111, [2, 333, 222]]
print D                          # [0, 111, [0, 333, 222]]
```

Пример: связные списки

Файл slist.py

```
Names = ["Lobachevski", "Gauss", "Newton", "Euler"]

# Создаем связный список
cur = 0
for name in Names:
    prev = cur;
    cur = {"data": name, "next": prev}
lst = cur

# Печатаем список
cur = lst
while cur:
    print cur["data"]
    cur = cur["next"]
```

7. Работа с модулями

```
import модуль
from модуль import что-то конкретное
from модуль import *
```

Пример:

```
import math
print math.pi, math.exp(1)
```

```
from math import pi, exp
print pi, exp(1)
```

```
from math import *
print pi, exp(1), sin(pi)
```

8. Ввод/вывод

```
print 'Pi is %5.3f (3 десятичные цифры)'% math.pi
print 'e**pi = %5.3f, pi**e = %5.3f'% (exp(pi), pi**(exp(1)))
x = raw_input('Введите число: ')
f = open('myfile', 'w')      # Открыть файл для записи
f.write('Hello, World!\n')   # Записать текст
f.write('e**pi = %5.3f, pi**e = %5.3f'% (exp(pi), pi**(exp(1))))
f.close()                    # Закрыть файл
f = open('myfile', 'r')      # Открыть файл для чтения
s = f.read()                 # Прочитать весь файл целиком
s = f.read(length)           # Прочитать length символов
s = f.readline()             # Прочитать одну строку
```

9. Научные вычисления: scipy и numpy

```
from numpy import *
A = array([[1, 2], [3, 4]]) # Матрица
B = array([[3, 1], [1, 3]])
b = array([1, 2])           # Вектор
dot(A, B)                   # Матричное произведение
dot(A, b)                   # Произведение матрицы на вектор-столбец
dot(b, A)                   # Произведение вектор-строки на матрицу
```

```
from numpy.linalg import *
det(A)                                      # Определитель
norm(A)                                     # Норма векторов и матриц
inv(A)                                      # Обратная матрица
solve(A, b)                                 # Решение системы линейных уравнений
x, residuals, rank, sv = lstsq(A, b)        # Задача наименьших квадратов
pinv(A)                                     # Псевдообратная матрица
evalues, evectors = eig(A)                  # Собственные числа и векторы
```

См. пример `xlinalg.py`

9.1. `scipy`

- Специальные функции
- Интегрирование
- О.Д.У.
- Оптимизация
- Интерполяция
- FFT
- ...

Файл xintegrate.py

```
from scipy import *

q, tol = integrate.quad(lambda x: sqrt(x), 0, 1)
print q

def fn(x):
    return x**2

s = integrate.quad(fn, 0, 1)
print s
```

10. Научная графика

pylab — 2d-графика a-la MATLAB

Файл xpylab.py

```
from math import *
from numpy import *
from pylab import *

x = linspace(-2*pi, 2*pi, 100)
y1 = sin(x)
y2 = cos(x)

plot(x, y1, x, y2)
show()
```

11. Элементы ООП

См. `rational.py`

12. Элементы функционального программирования

Файл primes.py

```
# Решето Эратосфена
# Простые числа, не превосходящие n

def primes(n):

    lst = range(2, n)
    i = 0
    while i < len(lst):
        x = lst[i]
        lst = filter(lambda y, x = x: (y == x or y%x != 0), lst)
        i += 1

    return lst

print primes(100)
```

Файл quicksort.py

```
def quicksort(a):
    if len(a) <= 1: return a
    v = a[0]
    return quicksort(filter(lambda w, v = v: w < v, a)) + \
           filter(lambda w, v = v: w == v, a) + \
           quicksort(filter(lambda w, v = v: w > v, a))

print quicksort([1, 32, 0, 0, 45, 12, 32, 11, 17, 0])
```