Краткий FAQ по Python

Артамонов Ю.Н.

Международный университет
природы, общества и человека "Дубна"
филиал Котельники

15 марта 2016 г.

Содержание

О чем нужно получить представление, чтобы быстро освоить новый язык программирования?

- История возникновения языка программирования. Общие вопросы
- Основные структуры данных, переменные, присваивания, сравнения
- Вывод, ввод данных в языке программирования
- Реализация операторов ветвления
- Реализация операторов цикла
- Реализация функций и процедур
- Обзор поддерживаемых парадигм программирования
- Обзор существующих библиотек в языке программирования
- Практика создания консольного приложения
- Практика создания приложения с графическим интерфейсом (GUI)
- Практика создания web-приложения

История возникновения Python

Когда и кем создан:

Python ([пайтен], рус. Питон wiki) — скриптовый язык программирования, созданный голландским программистом Гвидо ван Россумом (ныне сотрудник Dropbox) в 1991 году. Назван так, по словам самого Гвидо, в честь шоу Monty Python. Наиболее знаменитая отличительная черта языка — использование отступов для выделения блоков кода и управляющих структур.
Официальный сайт

Основные особенности:

- Интерпретируемый
- Общего назначения (Сравнение языков программирования)
- Динамическая типизация
- Объектно-ориентированный
- Сборка мусор

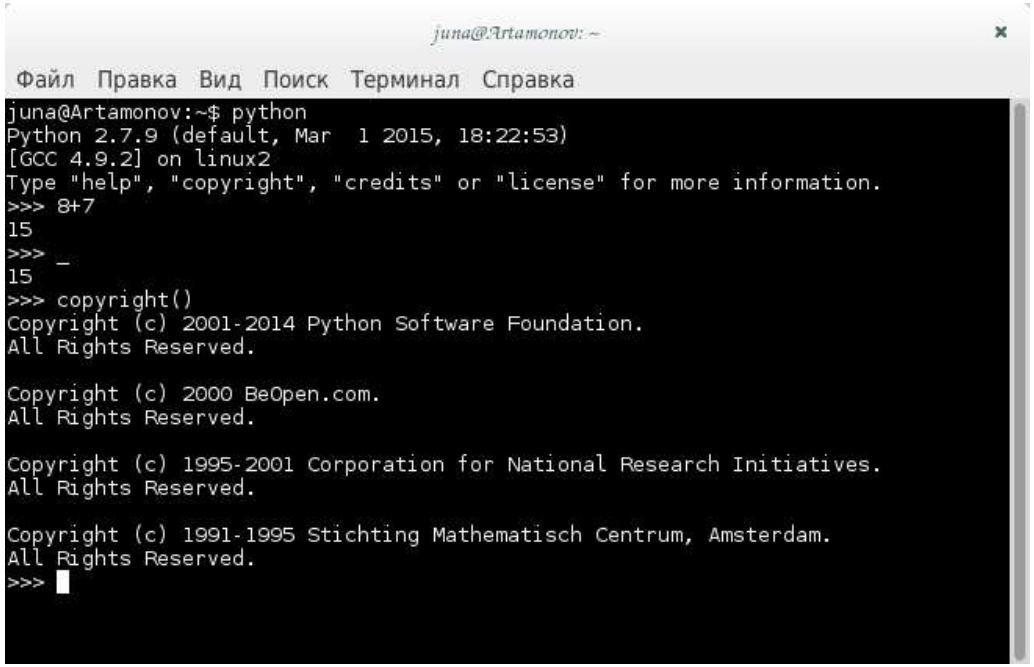
Несколько изображений

Логотип Python

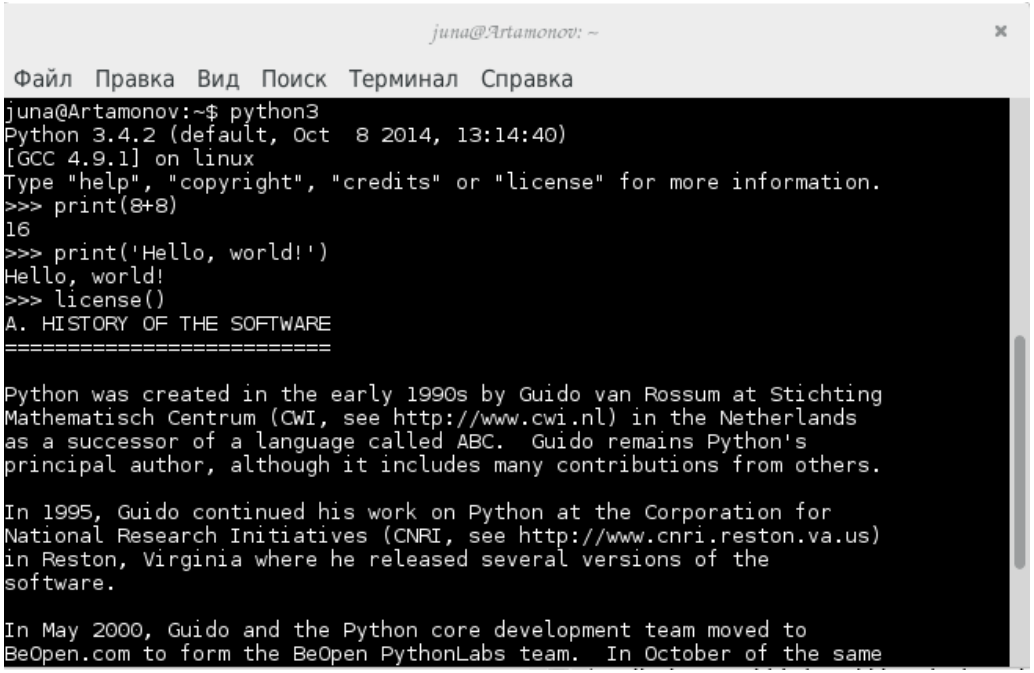
Гвидо ван Россум
создатель Python
(2006 год)

Работа с интерпретатором в консоле

Запуск в консоле python 2



Запуск в консоле python 3



Числа

Типы чисел:

- Целые числа (int,long) - положительные и отрицательные, а также нуль (например, 4, -8, 0, 7).
  Примеры: int('56')+int('543')=599; int(5.6)+int(9.8)=14;
  long('267867868767868576 44')=267867868767868576 44L (для python2)
- Вещественные числа (float) - положительные и отрицательные числа с плавающей точкой (например, 4.5, 3.0, 3.14).
  Примеры: float('78')=78.0; float('12.2')+float('3.5')=15.7;
  float('6e-2')*float('7e2')=42.0.

Числа (продолжение)

Типы чисел:

- Комплексные числа (complex) - числа вида a+b*i, где a,b - любые целые или вещественные числа, i - комплексная единица.
  Примеры: complex(6,7)=(6+7j); complex('3.2+1j')=(3.2+1j); (6+7j)/(4+3j)=(1.8+0.4j).
- Двоичные, восьмиричные, шеснадцатиричные числа.

Примеры: hex(890)=0x37a; bin(567)=0b1000110111; oct(567)=0o1067; «И»: 0b101 & 0b010=0; «Или»: 0b101 | 0b010=7; «Отрицание»: $\sim (0b010) = -3$.

Основные операции с числами

- Сложение: 6+8=14; 6.7+3.2=9.9; 8+7j+6+4j=(14+11j).
- Вычитание: 6-8=-2; 6.7-3.2=3.5; 8+7j-(6+4j)=(2+3j).
- Умножение: 6*8=48; 6.7*3.2=21.44; (8+7j)*(6+4j)=(20+74j).
- Деление: 7/2=3 в python2 (если делимое и делитель - целые числа, то результат тоже целое число); 7/2=3.5 в python3; 6.7/3.2=2.09375; (8+7j)/(6+4j)=(1.4615384615384615+0.19230769230769232j).
- Целочисленное деление: 12//5=2; 6.7//3.2=2.0.
- Остаток от целочисленного деления: 7%3=1; 7.1%5=2.0999999999999996.
- Возведение в степень: 5**2=25; 5.5**0.5=2.345207879911715.

Дополнительные операции по работе с числами

- Абсолютное значение: abs(-9)=9; abs(7)=7.
- Возведение в степень: pow(5,2)=25;pow(5,0.5)=2.23606797749979.
- Вычисление результата целочисленного деления и остатка: divmod(17,5)=(3, 2);divmod(17,5)[0]=3; divmod(17,5)[1]=2.

Дополнительные операции по работе с числами (продолжение)

- Остальные функции доступны при использовании пакета math:
  Пример:
  import math
  dir(math)
  ['doc', 'loader', 'name', 'package', 'spec', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'hypot', 'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'trunc']
  math.sin(5)
  -0.9589242746631385

Логический тип данных

- Число 1 имеет специальное обозначение True (истина).
- Число 0 имеет специальное обозначение False (ложь).

Примеры:
a=5; b=7
a==b : False
a<b : True
a<=b : True; a < 6 < b : True
a>b : False
a>=b : False
a!=b : True

Логический тип данных (продолжение)

- 1==True : True
  a<b or 0 != False : True
  a<b | 0 != False : True
  not(a<b | 0 != False) : False
  a<b and 0 != False : False
  a<b & 0 != False : False

Строки

Строки - неизменяемые последовательности символов, заключенные в кавычки (двойные или одинарные), любой длины.

Основные операции со строками

- Преобразование числа в строку: str(678) дает '678'
- Получение символа по коду, кода по символу: chr(67) дает 'C'; ord('г') дает 1075
- Длина строки: len('qwerty') дает 6. Конкатенация строк: 'Все' + ' ok!'
- Выбор элемента строки с заданным номером: 'qwerty'[0] дает 'q'; 'qwerty'[-1] дает 'y'

Строки (продолжение)

Основные операции со строками

- Получение среза s[i:j:k] - подстрока, содержащая символы строки s с номера от i до j с шагом k: 'qwerty'[1:3] дает 'we' (первый номер включается, последний нет); 'qwerty'[1:5:2] дает 'wr'
- Получение символа строки с наименьшим, наибольшим номером: min('qwerty') дает 'e'; max('qwerty') дает 'y'
- N-кратное повторение строки s * n (n * s): 'qwerty'*2 дает 'qwertyqwerty'

Кортежи

Кортежы (tuple) - неизменяемый упорядоченный набор объектов разных типов.

Основные операции с кортежами

- Задание кортежа: t=(12, 'a', 3.14, 'Python'); y=(); z=(t,)
- Присваивание переменным значений кортежа (a,b,c,d)=t
- Длина строки: len(t) дает 4. Объединение кортежей: t+y дает (12, 'a', 3.14, 'Python', (12, 'a', 3.14, 'Python'))
- Выбор элемента кортежа с заданным номером: t[0] дает 12, t[-1] дает 'Python'

Кортежи (продолжение)

Основные операции с кортежами

- Получение среза s[i:j:k] - кортеж, содержащий элементы с номера от i до j с шагом k
- Преобразование строки в кортеж: s='Python'; t=tuple(s) дает ('P', 'y', 't', 'h', 'o', 'n')
- N-кратное повторение кортежа s * n (n * s): t[0:5:2]*2 дает ('P', 't', 'o', 'P', 't', 'o')

Списки

Списки - изменяемый упорядоченный набор объектов разных типов.

Основные операции со списками

- Задание списка: lst=[1,2,'t','qw']
- Преобразование объектов в список: list('qwerty') дает ['q', 'w', 'e', 'r', 't', 'y']
- Длина списка: len(lst) дает 4. Объединение списков: lst+lst дает [1,2,'t','qw',1,2,'t','qw']
- N-кратное повторение списка lst * n (n * lst): lst*2 дает [1,2,'t','qw',1,2,'t','qw']
- Выбор элемента списка с заданным номером: lst[0] дает 1, lst[-1] дает 'qw'
- Получение среза s[i:j:k] - список, содержащий элементы с номера от i до j с шагом k
- Замена элемента с заданным номером: lst[0]='first' дает ['first', 2, 't', 'qw']

Списки (продолжение)

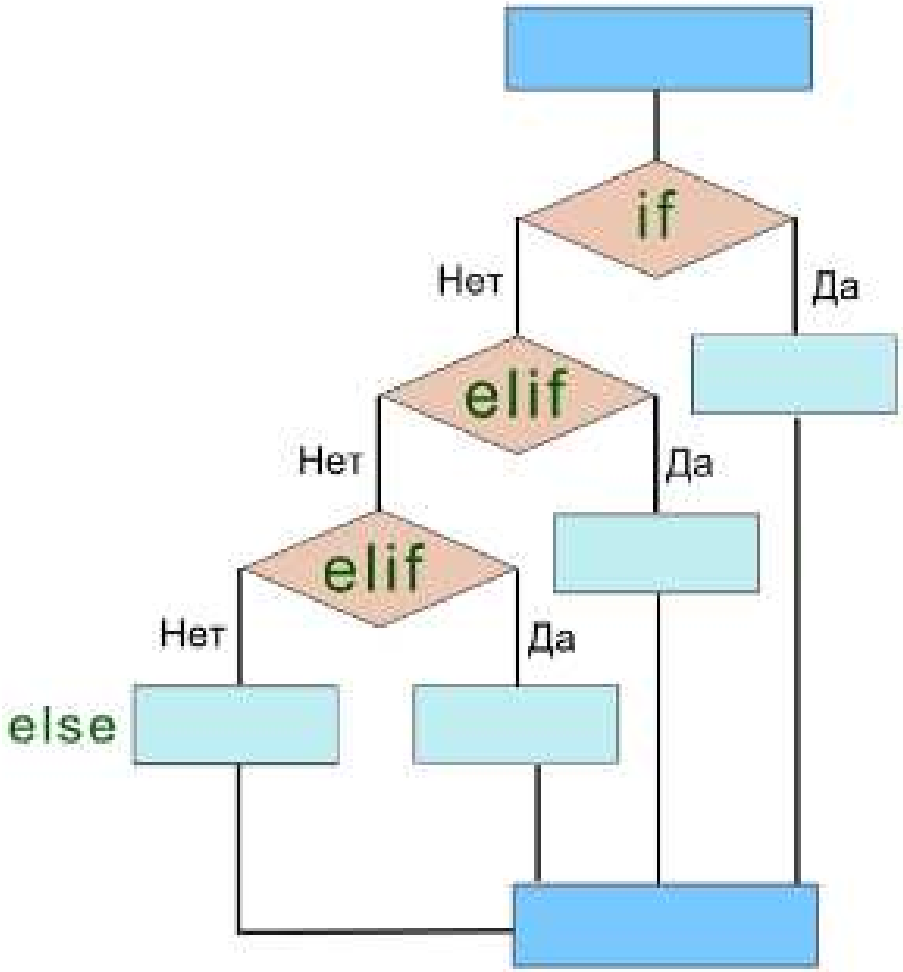
Основные операции со списками

- Удаление элемента с заданным номером: del lst[0] дает [2, 't', 'qw', '77']
- Добавление элемента в список s=[1,2,3]: s=s+[4] - дает [1,2,3,4]; s=[0]+s дает [0,1,2,3]
- Замена среза списка на элемент или список s=[1,2,3]: s[1:2]='qw' дает [1, 'q', 'w', 3], а s[1:3]='qw' дает [1, 'q', 'w']
- Удаление элементов, входящих в указанный срез s=[1,2,3,4]: del s[2:4] дает [1,2]

Ввод данных в Python

- Для ввода данных используется две команды input() - для ввода любых значений, raw_input() (только в python 2) - для ввода только строковых значений. В качестве аргумента этих команд предлагается использовать строку подсказку - приглашение для ввода.
  Примеры:
  a=input('a=')
  a,b,c=input('Введите три значения через запятую ') (работает только в python 2)
- Для вывода данных используется команда print для python 2, print() - для python 3
  Примеры:
  print 'Результат суммы', 5+3 для python 2 дает 'Результат суммы 8'
  print('Результат суммы равен ',5+3) - для python 3

Общая схема ветвления в Python



Примеры использования операторов ветвления

```
t=int(input('Введите температуру на улице'))
if t<10:
    s='Плохая погода'
else:
    s='Хорошая погода'
 print(s)
```

Примеры использования операторов ветвления

```
t=int(input('Введите ваш балл'))
if t==5:
    s='Молодец!'
elif t==4:
    s='Хорошо!'
else:
    s='Лентяй'
print(s)
```

Виды циклов

- Цикл for

```
for i in list:
    body\_cycle
```

Здесь переменная i последовательно принимает значения из списка list
Вместо списка можно строить диапазоны с помощью range:

```
for i in range(10):
    print('Hello')
```

В результате слово Hello! будет выведено 10 раз.

Виды циклов

- Цикл while

```
while condition:
    body_cycle
```

Например,

```
i=1
while i<10:
  print('Hello!')
  i=i+1
```

Процедура

- Процедура - это подпрограмма, которая выполняет свой участок кода и ничего не возвращает в основную программу.

  Пример:

```
def prt(n):
    for i in range(n):
        print(i)
print(prt(10))
```

Функция

- Функция - это подпрограмма, которая выполняет свой участок кода и возвращает некоторое значение в основную программу (для этих целей в python используется оператор return)

```
def summa(a,b):
    return a+b
print(summa(5,7))
```

Примерный перечень литературы

Теория

- Кормен Т. Х. и др. Алгоритмы: построение и анализ, 2 издание : Пер. с англ.- М.: Издательский дом «Вильямс», 2005. – 1296 с.
- Вирт Н. Алгоритмы + структуры данных = программа. – М.: Мир, 1989
- Ахо А.В., Хопкрофт Д.У., Джеффри Д. Структуры данных и алгоритмы.: Пер. с англ. : Уч. Пос. – М.: Издательский дом «Вильямс», 2000.- 384 с.

Практика

- http://pythontutor.ru/
- http://pythonworld.ru/
- http://itvdn.com/ru/video/python-starter