Python 3

Краткое введение

Хамовники, 2011

Язык Python

Python — мощный и простой для изучения язык программирования. Интерпретатор Python и разрастающаяся стандартная библиотека находятся в свободном доступе в виде исходников и бинарных файлов для всех основных платформ на официальном сайте Python http://www.python.org. Там же можно обнаружить подробное руководство по использованию Python.

Особенности Python

1. Интерпретируемый (нет процесса компиляции, интерпретатор выполняет код строка за строкой).
2. Динамические переменные (не надо объявлять переменную, достаточно присвоить ей значение).
3. Блоки с помощью отступов (вместо begin...end в паскале и {...} в C/C++ ту же роль в питоне выполняют отступы).

Посмотрим на несколько примеров...

Пример 1: Hello, world!

Написать "Hello world!" на Python очень просто. Для этого используем функцию print. Эта функция принимает на вход данные и выводит их на экран.

Примечание: кавычки могут быть как одинарные, так и двойные.

```
print("Hello, world!")
```

Пример 2. 2+3

Заведем переменную a и присвоим ей число 2. Теперь a - переменная типа int (целочисленного). Ту же операцию проделаем с b. Затем выведем их сумму.

```
a = 2
b = 3
print(a + b)
```

Пример 3. echo

Функция input() считывает одну строчку с клавиатуры и возвращает ее. А мы ее положим в s. Теперь s - переменная типа str (строкового). Выведем ее.

```
s = input()
print(s)
```

Пример 4: +1

Если нашей программе на вход подать число 15, функция input() считает его как строчку '15'. Если мы хотим работать с числом, нам надо привести строку к числу с помощью функции int. Операция a+=1 аналогична операции a=a+1.

```
a = int(input())
a += 1
print(a)
```

Пример 5: A+B

Если в строчке написано '2 3', а мы хотим добыть из нее два числа, нам надо воспользоваться методом split, который разобьет строчку на несколько по пробельным символам. Таким образом '2 3'.split() вернет список ['2', '3']. Python разрешает присваивания вида a, b = ['2', '3'], но об этом чуть позже.

```
a, b = input().split()
print(int(a) + int(b))
```

Пример 6: Сумма n чисел

```
print(sum(map(int, input().split())))
```

Разберем программу по шагам:

1. Считываем строчку из входного файла.
2. Разбиваем ее на список строк, в каждой по одному числу.
3. Функция map применяет функцию int к каждому элементу списка и возвращает список результатов.
4. Функция sum берет список чисел, складывает их и возвращает результат
5. Функция print выводит результат.

Пример 7. Проверка на четность

```
a = int(input())
if a % 2 == 0:
    print("Even")
    a_is_even = True
else:
    print("Odd")
    a_is_even = False
print("a_is even contains:", a_is_even)
```

Условный оператор. Какие инструкции принадлежат блоку определяется отступом (4 пробела). Пока отступ есть - инструкция в блоке. Если функции print передать несколько параметров, она выведет их через пробел.

Пример 8. Цикл for

```
lst = map(int, input().split())
count_of_even = 0
for elem in lst:
    if elem % 2 == 0:
        count_of_even += 1
print(count_of_even)
```

Пусть нам надо подсчитать количество четных чисел в списке. Цикл for пробегается по всем числам из списка lst и выполняет тело для каждого элемента. При этом значение текущего элемента записывается в elem.

Основные типы

Название	Краткое описание	Пример использования
int	Целые числа произвольной длины	`a = 2` `b = a * (5 - a)`
float	Дробные числа (15-16 значащих цифр)	`a = 2.5` `b = a * (5 - a)`
bool	Логический тип (True или False)	`a = True` `b = (2 < 3) and not a`
str	Строки произвольной длины (Символ - строка длины 1)	`s = "abacaba"` `t = s[1:3] + s[-1]`
list	Список (аналог массива)	`l1 = [1, 2, 3, [4, 5]]` `l2 = l1[:3] + l1[3]`
tuple	Кортеж (для упаковки в одну переменную)	`a = 1, 2, 3` `b, c, d = a`

Тип int (целое число)

1. Хранит целое число произвольной длины
2. Можно выполнять стандартные арифметические операции (+, -, *, / (5/2==2.5), // (5//2==2), % (5% 2==1), ** (2**5==32)).
3. Битовые операции (&, |, ^, <<, >>)
4. Операции присваивания (=, +=, -=, *=, и т.д.)
5. int(a) - приведение a к типу int

Тип float (вещественное число)

1. Та же арифметические операции, что и в int (включая // и %).
2. В модуле math много функций для дробных чисел.
    1. sin, cos, ... - тригонометрические функции.
    2. pi, e - стандартные константы.
    3. floor, ceil, trunc - округление.
3. floor(a) - превращает a в дробное.

Тип bool (логическое значение)

1. Хранит одно из двух значений (True, False).
2. Логические операции (and, or, not).
3. Операции сравнения (==, !=, >, >=, <, <=). Можно сравнивать числа.
4. Можно использовать в условных конструкциях.

Тип str (строка)

1. Хранит строку произвольной длины.
2. Неизменяемый (нельзя написать s[2]='a').
3. "abc" + "def" == "abcdef", "abc" * 3 == "abcabcabc".
4. len(s) - возвращает длину строки.
5. Индексы нумеруются слева с 0 и справа с -1. Таким образом s[-3] == s[len(s) - 3]
6. "abacaba".find("bac") ищет подстроку в строке. Возвращает индекс первого символа найденной подстроки, или -1, если не нашел.
7. Можно делать срезы вида s[1:3]. Если s = "abcde", то s[1:3] = "bc".
8. s.split() - разбивает строку на несколько по пробельным символам.
9. s.strip() - удаляет из строки пробелы в начале и в конце.

Тип list (список)

1. Хранит список данных. Может хранить данные разных типов.
2. a = [] или a = list() создает пустой список.
3. Сравнение выполняется поэлементно. [1,1,100] < [1,2,3].
4. a.append(d) приписывает элемент d в конец списка a.
5. a.insert(2, x) вставит x *перед* вторым элементом a.
6. a.sort() отсортирует список (sorted(a) вернет отсортированный массив не изменив a).
7. a.reverse() перевернет список (reversed(a) тоже есть).
8. a.pop(x) удалит элемент с номером x из массива и вернет его значение (a.pop() удалит и вернёт последний элемент).
9. Если мы присвоим один список другому, они будут работать с одной областью памяти и все, что мы будем делать с одним списком будет происходить со вторым. Чтобы этого избежать, надо копировать так: a = b[:]

Тип tuple (кортеж)

1. Кортежи — это фиксированные списки. Однажды созданный кортеж не может быть изменён никаким способом.
2. a = (1, 2, 3) или a = 1, 2, 3 - упаковка кортежа.
3. x, y, z = a - распаковка кортежа. В последний элемент запишется кортеж из того, что осталось в a в конце.
4. Кортежи нельзя менять, поэтому придется распаковать кортеж, изменить и упаковать обратно.
5. Кортежи сравниваются, как списки.

Условный оператор

Сначала проверяем condition1. Если верно, то выполняем первый блок. Иначе проверим condition2. Если верно, то выполняем второй блок. Иначе выполняем блок else. Если не надо ничего делать - используйте ключевое слово pass.

```
if (condition1):
    operation1
    operation2
    ...
elif (condition2):
    operation3
    operation4
    ...
else:
    pass
```

Цикл while

Тут все, как в других языках. Блок else выполняется только если при вычислении condition получили false.
Оператор break прерывает выполнение цикла не заходя в else. Оператор continue завершает текущую итерацию цикла и переходит к следующей.

```
while (condition):
    operator1
    operator2
else:
    operator3
```

Цикл for

Цикл for в питоне умеет только проходить по всем элементам чего-нибудь, в чем много элементов (списки, кортежи, строки). Чтобы перебирать индексы используется функция range.
range(r) - список от 0 до r-1
range(l,r) - список от l до r-1
range(l,r,step) - список от l до r, не включая r с шагом step (step может быть отрицательным).

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

for elem in list:
    operator1
    operator2
```

Срезы (slices)

1. Срезы работают для списков, кортежей и строк.
2. Срез получает получить новый массив, выбрав элементы из данного.
3. a[l:r] срез из элементов от l до r-1.
4. a[l:] - по умолчанию справа находится len(a).
5. a[:r] - по умолчанию слева находится 0
6. a[:] - копия массива.
7. a[l:r:step] - срез с шагом step.

import

Чтобы подключить библиотеку, можно использовать:

1. import math - подключает библиотеку, можно обращаться к элементам, как к math.somefunc()
2. from math import somefunc, somefunc2 - достать из math две функции и разрешить их использовать в коде прямо так: somefunc1()
3. from math import * - достать из math все (не рекомендуется).

help(), dir()

1. Для справки по языку можно использовать эти две функции.
2. Первая выводит краткую инструкцию по использованию того, что ей передали.
3. Вторая выводит содержимое данного обьекта или модуля. Например, dir(math).
4. Учите английский :).