Python

шпаргалка

Для чего используется Python 3

Python может использоваться для многих целей, например:

- Разработка прикладного ПО
- Разработка мобильных приложений
- Разработка web-приложений
- В качестве встраиваемого скриптового языка во многих играх, и программах
- В научных рассчетах

Переменные в python

Переменная — это именованная область памяти для хранения данных, которые могут изменяться в процессе исполнения программы.

Переменные

Язык Python чувствителен к регистру. Переменная Z и z – разные переменные. Python, в отличие от многих языков, не требует описания переменных.

Типы переменных:

- int { целая }
- float { вещественная }
- list { список, аналог массивов}
- str { символьная строка }
- bool { логическая }

Объявление переменных (выделение памяти):

```
int("88") результат 88
```

Ввод и вывод данных в программах на языке Python

Функция input

возвращает строковое значение, введённое пользователем с клавиатуры.

Функции преобразования в числовые типы:

- ✓ в целые числа – int (<данные>);
- ✓ в вещественные числа – float (<данные>).

Инструкция print

выводит данные из оперативной памяти компьютера на экран через разделители, которыми по умолчанию являются пробелы.

Функция format

формирует символьную строку заданного формата.

VIDEOUROKI.NET

Арифметические операторы

+	Сложение	15 + 5 = 20 20 + -3 = 17 13.4 + 7 = 20.4
–	Вычитание	15 – 5 = 10 20 – -3 = 23 13.4 – 7 = 6.4
*	Умножение	5 * 5 = 25 7 * 3.2 = 22.4 -3 * 12 = -36
/	Деление	5 / 2 = 2.5 (В Python 2.x версии результат будет 2) 5.0 / 2 = 2.5 (Хотя бы один операнд должен быть *float*)
%	Остаток	6 % 2 = 0 7 % 2 = 1 13.2 % 5 = ~3.2
**	Возведение в степень	5 ** 2 = 25 2 ** 3 = 8 -3 ** 2 = -9 **(степень приоритетнее знака)**
//	Целочисленное деление	12 // 5 = 2 4 // 3 = 1 25 // 6 = 4

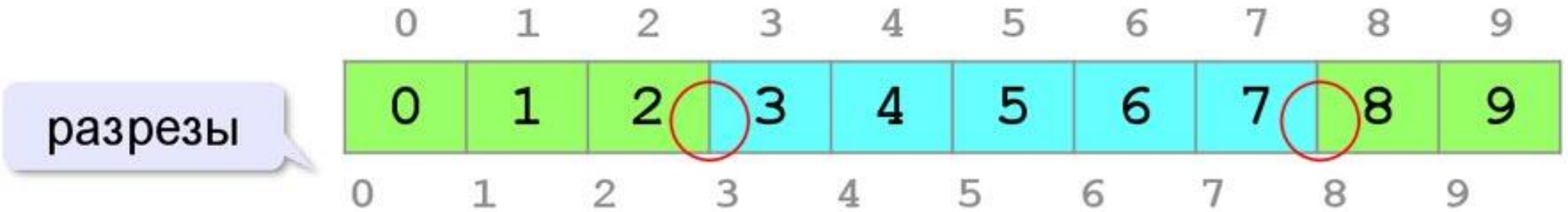
Объединение *(конкатенация)* :

```
s1 = "Привет"
s2 = "Вася"
s  = s1 + ", " + s2 + "!"
```

"Привет, Вася!"

Срезы: извлечение из данной строки одного символа или некоторого фрагмента подстроки или подпоследовательности

```
s = "0123456789"
s1 = s[3:8]                    # "34567"
```



Уловие в python

- Условный оператор **if/elif/else**, ветвление кода программы. Возможно, самый известный тип инструкций - это конструкция **if/elif/else**. Часто возникает необходимость, чтобы некоторый код выполнялся только при соблюдении определенного условия или код подлежащий выполнению, должен выбираться исходя из выполнения одного из нескольких взаимоисключающих условий.

Условный оператор

```
if <условие> :
  {что делать, если условие верно}

else :
  {что делать, если условие неверно}
```

Особенности:

- вторая часть (***else*** …) может отсутствовать (неполная форма)

Цикл в Python

это инструмент, который позволяет программе повторять определённое действие несколько раз.

Организация цикла в Python

```
>>> for i in range(10,0,-1):
      print(i)
10
9
8
7
6
5
4
3
2
1
>>> print('Поехали!')
Поехали!
```

для переменных цикла, как правило (но не всегда), используют буквы **i**, **j**, **k**, **n**

Цикл с условием

При известном количестве шагов:

```
k = 0
while k < 10:
  print ( "привет" )
  k += 1
```

Зацикливание:

```
k = 0
while  k < 10:
  print ( "привет" )
```

Списки и массивы

Списки (list). Функции и методы списков

Списки в Python - упорядоченные изменяемые коллекции объектов произвольных типов (почти как массив, но типы могут отличаться).

```
>>> list('spisok Alex')
['s', 'p', 'i', 's', 'o', 'k', ' ', 'A', 'l', 'e', 'x']
```

Список можно создать и при помощи литерала:

```
>>> s=[]
>>> l=['a','l',['ex'],2]
>>> s
[]
>>> l
['a', 'l', ['ex'], 2]
```

Массивы = Списки

- **В языке Python нет такой структуры данных, как массив.**
- **Для хранения группы однотипных объектов используют списки (тип данных `list`).**
- В отличие от массивов в других языках, у списков нет никаких ограничений на тип переменных, поэтому в них могут храниться объекты разного типа.

- **Списки** являются упорядоченными последовательностями, которые состоят из различных объектов (значений, данных), заключающихся в квадратные скобки [ ] и отделяющиеся друг от друга с помощью запятой.

- Пример:

```
list1 = ['физика', 'химия', 1997, 2000];
list2 = [1, 2, 3, 4, 5, 6, 7 ];
list3 = ["a", "b", "c", "d"]
```

Методы в python

Методы строк

S.split(символ)	Разбиение по разделителю
S.isdigit()	Состоит ли строка из цифр
S.isalpha()	Состоит ли строка из букв
S.isalnum()	Состоит ли строка из цифр или букв
S.islower()	Состоит ли строка из символов в нижнем регистре
S.isupper()	Состоит ли строка из символов в верхнем регистре
S.istitle()	Начинаются ли слова в строке с заглавной буквы
S.upper()	Преобразование строки к верхнему регистру
S.lower()	Преобразование строки к нижнему регистру
S.startswith(str)	Проверка начала строки

MyShared

Функции и методы списков

list.append(x)	Добавляет элемент в конец списка
list.extend(L)	Расширяет список list, добавляя в конец все элементы списка L
list.insert(i, x)	Вставляет на i-ый элемент значение x
list.remove(x)	Удаляет первый элемент в списке, имеющий значение x. ValueError, если такого элемента не существует
list.pop([i])	Удаляет i-ый элемент и возвращает его. Если индекс не указан, удаляется последний элемент
list.index(x, [start [, end]])	Возвращает положение первого элемента со значением x (при этом поиск ведется от start до end)
list.count(x)	Возвращает количество элементов со значением x
list.sort([key=функция])	Сортирует список на основе функции
list.reverse()	Разворачивает список
list.copy()	Поверхностная копия списка
list.clear()	Очищает список

Методы строк

Функция или метод	Назначение
S1 + S2	Конкатенация (сложение строк)
S1 * 3	Повторение строки
S[i]	Обращение по индексу
S[i:j:step]	Извлечение среза
len(S)	Длина строки
str in S	Проверка на вхождение подстроки в строку
S.find(str, [start],[end])	Поиск подстроки в строке. Возвращает номер первого вхождения или -1
S.rfind(str, [start],[end])	Поиск подстроки в строке. Возвращает номер последнего вхождения или -1
S.replace(шаблон, замена)	Замена

MyShared

словарь

Методы словаря

- *[... 'clear', 'copy', 'fromkeys', 'get', 'items', 'keys', 'pop', 'popitem', 'setdefault', 'update', 'values']*

clear()	Очищает словарь.
copy()	Возвращает копию словаря.
fromkeys(seq[, value])	Создает словарь с ключами из seq и значением value (по умолчанию ***None***).
get(key[, default])	Возвращает значение ключа, а такого нет, не генерирует исключение, а возвращает ***default*** (по умолчанию ***None***).
items()	Возвращает пары (ключ, значение).
keys()	Возвращает ключи в словаре.
pop(key[, default])	Удаляет ключ и возвращает значение. Если ключа нет, то возвращает ***default*** (по умолчанию бросает исключение).
popitem()	Удаляет и возвращает пару (ключ, значение). Если словарь пуст, бросает исключение KeyError. (примечание: словари неупорядочены)
setdefault(key[, default])	Возвращает значение ключа, но если его нет, не бросает исключение, а создает ключ с значением ***default*** (по умолчанию ***None***).
update([other])	Обновляет словарь, добавляя пары (ключ, значение) из other. Существующие ключи перезаписываются. Возвращает ***None*** (не новый словарь!).
values()	Возвращает значения в словаре.

Словари Python

```
# Пройтись по словарю (Способ №1)
for key in d1:
    print(key, d1[key])

# Пройтись по словарю (Способ №2)
for key in d1.keys():
    print(key, d1[key])

# Пройтись по словарю (Способ №3)
for key, value in d1.items():
    print(key, value)
```

Словари (dict) и работа с ними. Методы словарей

Словари в Python - неупорядоченные коллекции произвольных объектов с доступом по ключу. Их иногда ещё называют ассоциативными массивами или хеш-таблицами. С помощью литерала:

```
>>> d={}
>>> d
{}
>>> d={'odin':1, 'dva':2}
>>> d
{'dva': 2, 'odin': 1}
```

функции в python

Синтаксис Python

```
def fib(n):
    if n <= 2:
        return 1
    return fib(n - 1) + fib(n - 2)


n1 = fib(1)  # = 1
n10 = fib(10)  # = 55
```

- Объявление функций начинается с ключевого слова **def**.
- Т.к. объявления типов в Python нет, то и аргументы функций объявляются просто именами.
- Значение из функции возвращается с помощью **return**.
- Функция может вызывать сама себя (рекурсия).

- Вызвать функцию можно либо просто передав аргументы позиционно, либо по их именам

класс в python

Создание классов, методов и объектов

```
class ИМЯ КЛАССА ():
    ПЕРЕМЕННАЯ = ЗНАЧЕНИЕ
    . . .
    def ИМЯ МЕТОДА (self, . . .):
        self. ПЕРЕМЕННАЯ = ЗНАЧЕНИЕ
        . . .
    . . .
# Основная часть
ПЕРЕМЕННАЯ = ИМЯ КЛАССА ()
. . .

ОБЪЕКТ.ИМЯ МЕТОДА ()
. . .
```

Объявление класса

Объявление метода

Создание объекта

Вызов метода

Атрибуты класса — это имена переменных вне функций и имена функций. Наследуются всеми объектами, созданными на основе данного класса.

Создание классов, методов и объектов

```
Пример_3. # Создание класса, объекта и вызов метода
class Person():
    name = ""
    money = 0
    def out (self): # self - ссылка на экземпляр класса
        print(self.name,'has',self.money,'dollars.')
    def changemoney (self,newmoney):
        self.money = newmoney

obj1 = Person()
obj2 = Person()
obj1.name = 'Bob'
obj2.name = 'Masha'
obj1.out()
obj2.out()
obj1.changemoney(150)
obj1.out()
```

```
Вывод:

>>>
Bob has 0
dollars.
Masha has 0
dollars.
Bob has 150
```