

Министерство образования и науки Украины
Донбасская государственная машиностроительная академия

СОВРЕМЕННЫЕ ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ

КОНСПЕКТ ЛЕКЦИЙ

(для студентов специальности
«Системы и методы принятия решений»)

Утверждено
на заседании кафедры ИСПР
Протокол № 2 от 9 сентября 2014г.

Краматорск 2014

УДК 681.3

Современные технологии программирования : конспект лекций (для студентов специальности «Системы и методы принятия решений» дневной форм обучения) / Сост. А.Ю. Мельников – Краматорск : ДГМА, 2014. – 16 с.

Приведены лекционные материалы к курсу.

Составитель	Мельников А.Ю., канд. техн. наук, доцент
-------------	--

Отв. за выпуск	Мельников А.Ю., канд. техн. наук, доцент
----------------	--

СОДЕРЖАНИЕ

Введение.....	4
1. Язык программирования Python.....	4
1.1 История создания.....	4
1.2 Основные сведения.....	4
1.3 Организация ветвления и циклов.....	6
1.4 Сложные и структурированные типы данных.....	8
1.5 Функции и модули.....	11
1.6 Специальные средства.....	11
2. Язык программирования Java.....	13
Список рекомендованной литературы.....	16

ВВЕДЕНИЕ

В настоящее время в области программирования наблюдаются такие тенденции:

- 1) получают распространение новые языки программирования, одновременно удовлетворяющие особым требованиям:
 - способность к решению различных классов задач (создание и обычных программ на отдельном компьютере, и web-приложений);
 - мультиплатформенность (независимость от операционной системы, наличие трансляторов для всех распространенных операционных систем);
 - свободное распространение (официальная бесплатность).
- 2) уменьшаются различия между различными средами программирования, создаются специальные приложения, унифицирующие процесс разработки программ в выбранной операционной системе (среда разработки Microsoft .Net позволяет создавать приложения для Windows на базе единой библиотеки классов с одновременным использованием Delhi, C# и других языков программирования).

К языкам программирования, удовлетворяющим вышеперечисленным требованиям, можно отнести Python и Java.

1 Язык программирования python

1.1 История создания

Разработан в конце 1989 года Гуидо ванн Россумом (Guido van Rossum), сотрудником исследовательской математической лаборатории (Амстердам, Голландия). Размещен в домене общего доступа в 1991 году. Название – по «Monty Python's Flying Circus».

1.2 Основные сведения

Основные характеристики:

- позволяет создавать как простые программы для математических расчетов, так и сложные системы с графическим интерфейсом;
- является объектно-ориентированным языком;

- может быть использован для интернет-программирования (с применением интерфейса CGI);
- может расширяться путем вставки блоков программ других пространственных языков программирования (например, Си).

Основные идеи:

- ничего лишнего (убраны даже операторные скобки, операторы группируются отступами);
- строгая структуризация программы, оператор GOTO вообще отсутствует.

Слабые места:

- обработка больших массивов строк производится не так эффективно, как, например, в Perl;
- непригодность для создания программ, работающих в режиме реального времени (следствие интерпретатора)

Программы создаются либо в любом текстовом редакторе, либо в специальной среде IDLE, написанной на этом же языке.

Синтаксис языка не базируется на синтаксисе какого-либо другого языка и является достаточно оригинальным, хотя основные операторы идентичны операторам известных языков программирования.

Программа не имеет специальных слов для обозначения начала и конца, хотя в ряде случаев некоторые указания должны быть размещены исключительно в начале программы (например, указание точного пути к интерпретатору в случае использования CGI).

Символы разделения операторов и описаний («;» или «:») в Питоне отсутствуют, каждый оператор (описание) занимает ровно одну строку программы. Комментарием считается текст от знака # до конца строки.

Подключение модулей может быть осуществлено двумя способами:

- `import sys`
- `from math import *`

В первом случае для обращения к модульным функциям потребуется явное указание имени модуля (`sys.exit(1)`, `sys.maxint`), во втором все функции импортируются в основную программу и не требуют указания имени модуля (`sin(x)`).

Числа могут быть целыми (`int`), длинными целыми (`long`), вещественными (`float`) и комплексными (`complex`). Длинное целое должно оканчиваться буквой L (`10000000000L`). В арифметическом выражении меньший операнд всегда автоматически приводится к типу большего (`2+10000000000L`); это

нужно обязательно учитывать при делении ($10/4=2$; $10/4.0=2.5$). Дополнительный арифметический символ – «%» («деление по модулю», т.е. остаток от деления: $19\%4=3$). Специальная функция `divmod(x,y)` возвращает два значения – x/y и $x\%y$. Явное преобразование может быть осуществлено записью `имя_типа(значение)` или `имя_типа(переменная)`, например: `float(4)=4.0`; `int(pi)=3`. Округление значения с сохранением исходного типа (вещественного) осуществляется функциями `floor()` и `ceil()`. Первая округляет число в меньшую сторону, вторая – в большую, однако при этом имеются особенности. Например: `floor(pi)=3.0`; `ceil(pi)=4.0`; `floor(-pi)=-4.0`; `ceil(-pi)=-3.0`. Возведение в степень – две звездочки ($x^2 \rightarrow x**2$).

Логические операции идентичны аналогам из Си, однако при этом осуществляется программный контроль над возможными ошибками (например, в Питоне невозможно написать оператор присвоения значения «=» вместо оператора проверки равенства «==»).

Переменные не нуждаются в предварительном описании; создаются в момент присваивания значений. Имена переменных не могут совпадать с зарезервированными словами языка (`if`, `for`), однако нет никаких препятствий для переопределения модульных функций (`sin`). Тип переменной определяется автоматически и может быть сколько угодно раз изменен в ходе работы программы.

Результаты работы программы выводятся командой `print` (в последних версиях языка `print` является не командой, а функцией). Вывод может быть записан как прямым перечислением выводимых переменных (по аналогии с Паскалем), так и по формату (по аналогии с Си):

```
print("x=",x," y=",y)
print("x=%5.2f, y=%5.2f\n" % (x,y))
```

(Формат: `d` – целый)

1.3 Организация ветвления и циклов

Оператор условия имеет вид:

```
if условие:
    выражения
elif условие:
    выражения
else:
    выражения
```

Пример:

```
if x > 10:
    y=x+x
    z=x*y
elif x>0 and x<=10:
    y=log(x)
    z=y
else:
    y=x
    z=2*x
```

Отступ после оператора играет роль операторных скобок. В случае вложенных блоков каждый должен иметь новый отступ, как минимум на одну позицию (точное число пробелов в отступе не регламентируется).

Питон имеет две разновидности циклов – с предусловием и с параметром. Цикл с предусловием ничем не отличается от своих аналогов в других языках:

```
while условие:
    выражения
```

Пример:

```
x=0
while x <= 5:
    y=exp(x)
    print("x=%5.2f, y=%5.2f\n" % (x,y))
    x=x+0.5
```

Цикл с параметром отличается от известных описаний:

```
for переменная in список:
    выражения
```

Пример:

```
for x in (1,2,3,4,5):
    y=exp(x)
    print("x=%5.2f, y=%5.2f\n" % (x,y))
```

В качестве «списка» может использоваться любая последовательность.

Для решения проблемы записи списка возможных значений переменной придумана специальная функция `range(n)`, которая позволяет создавать списки:

`range(10) -> [0,1,2,3,4,5,6,7,8,9]`

`range(-5,5) -> [-5,-4,-3,-2,-1,0,1,2,3,4]`

`range(-5,5,2) -> [-5,-3,-1,1,3]`

`range(10,-1,-1) -> [10,9,8,7,6,5,4,3,2,1,0]`

Общее правило: левая граница является первым элементом списка (по умолчанию – 0), последний элемент на единицу меньше правой границы.

Внутри цикла могут быть команды:

`break` – прервать цикл;

`continue` – перейти к следующей итерации цикла;

`pass` – пропустить действие (отсутствие действия).

1.4 Сложные и структурированные типы данных

К сложным и структурированным типам данным в Питоне относятся строки, массивы, наборы, списки и словари. Массивы практически не используются.

Строки

Строка символов заключается в любые кавычки и может содержать специальные символы, начинающиеся с обратной косой черты («\n» – перевод строки). Если перед строкой добавить букву `r`, то обратная косая черта будет рассматриваться как обычный символ (`path=r“c:\Temp”`). Для задания многострочного текста используются тройные кавычки (повторенные три раза).

Со строками возможны операции конкатенации (`s1+s2`), повторения (`s1*3`), проверки включения одной строки в другую (`s1 in s2`), извлечения элемента (`s[i]`, `s[-i]`) или последовательности элементов (`s[i:j]`; `s[0:3]` возвратит элементы с индексами 0,1,2), определения длины строки (`len(s)`), ее минимального (`min(s)`) и максимального (`max(s)`) элементов.

Все методы работы со строками расположены в модуле `string`. Часто используются:

`atoi(s)` – преобразовать строку в целое число;

`atol(s)` – преобразовать строку в длинное целое число;

`atof(s)` – преобразовать строку в вещественное число;

`split(s, separator, maxsplit)` – разбить строку на слова и вернуть список слов; по умолчанию `separator` – пробел; можно задать максимальное число слов, при этом все «неразбитое» сохранится в последнем элементе списка;

`join(list, separator)` – объединить все слова списка в одну строку; по умолчанию `separator` – пробел;

`find(s, sub, n1, n2)` – найти первое вхождение подстроки `sub` в строку `s`; можно задать ограничение на индекс поиска.

При запуске программы параметры передаются в строке `sys.argv`; первый параметр (`sys.argv[0]`) – имя самой программы.

Наборы

Наборы (`tuples`) данных позволяют работать с последовательностями значений любых типов, заключенных в круглые скобки:

```
t=(1,2,3,"abc","defg")
```

```
e=() # пустой набор
```

К элементам набора можно обращаться по индексу. Также, подобно Перлу, Питон позволяет «распаковывать» наборы:

```
import time
```

```
lt=time.localtime(time.time())
```

```
y,m,d,h,min,sec,wd,yd,dst=lt
```

```
print lt
```

```
print y,m,d
```

```
>>> (2009, 1, 25, 16, 38, 15, 6, 25, 0)
```

```
>>> 2009 1 25
```

Главный недостаток: нет отдельного модуля для работы с наборами. Следствие: невозможно добавить новый элемент к набору или удалить из набора существующий элемент. Необходимо, как и в случае со строками, создавать новый набор на базе имеющегося:

```
lt=(2008,)+lt[1:3]
```

```
>>> (2008, 1, 25)
```

Списки

Списки (`lists`) представляют собой последовательности значений любых типов, заключенных в квадратные скобки:

```
t=[1,2,3,"abc","defg"]
```

```
e=[] # пустой список
```

По сути списки являются массивами, содержащими разнотипные данные. К тому же списки снабжены рядом методов для работы с элементами:

`append(a)` – добавить элемент в конец списка (`t.append("tg32")`);
`count(a)` – подсчитать, сколько раз элемент встречается в списке (`t.count(2)`);
`index(a)` – определить индекс элемента (`t.index("abc")`);
`insert(i,a)` – добавить элемент в список в позицию `i` (`t.insert(2,"tg32")`);
`remove(a)` – удалить из списка первый найденный указанный элемент (`t.remove("abc")`);
`reverse()` – изменить порядок следования элементов на противоположный;
`sort(имя_функции)` – отсортировать список по возрастанию, функцию для сортировки можно задать свою (должна иметь два аргумента).

Словари

Словари подобны последовательностям, однако их элементы не хранятся в определенном порядке. В других языках (например, в Перле) словари называются хэшами (hashes) или ассоциативными массивами.

Данные хранятся в виде пар «ключ–значение»:

```
dict={"name": "James", "surname": "Bond", "code": 7}
```

```
d2={}          # пустой словарь
```

Извлечение элемента: `myname=dict["name"]`

Список ключей можно вывести в виде последовательности (списка):

```
k=dict.keys()
```

```
print k
```

```
>>> ['code', 'surname', 'name']
```

Словари снабжены рядом методов:

`clear()` – очистить весь словарь (`dict.clear()`);

`copy()` – создать новый словарь и скопировать в него указанный словарь (`newdict=dict.copy()`);

`has_key(k)` – проверить наличие указанного ключа в словаре (0 – нет, 1 – есть);

`items()` – вывести список пар «ключ–значение» `[('code', 7), ('surname', 'Bond'), ('name', 'James')]`

`keys()` – вывести список ключей;

`values()` – вывести список значений.

1.5 Функции и модули

Пример описания функции:

```
def maxchet(a):
    max=-maxint
    for i in range(len(a)):
        if a[i]>max and a[i]%2==0:
            max=a[i]
    if max==maxint:
        return 0
    return max
```

Модули подключаются «import sys» или «from math import *». Могут содержать как функции, так и исполняемый набор операторов. Необходимо проверять значение встроенной переменной `__name__` (главная программа будет иметь значение `__main__`):

```
print __name__
>>> __main__
```

Для избежания использования модуля как исполняемой программы нужно использовать проверку:

```
if __name__ == __main__:
    sys.exit()
```

1.6 Специальные средства

Питон содержит три оригинальные функции (встроенные) и одну инструкцию.

`map(функция, последовательность)` вызывает функцию для всех элементов последовательности; результат – измененная последовательность. В примере приводится перевод всех элементов списка в целые числа:

```
import string
s="1 2 3 4 5"
ls=string.split(s)
md=map(string.atoi,ls);
print md
>>> [1, 2, 3, 4, 5]
```

Можно короче:

```
import string
print map(string.atoi,string.split("1 2 3 4 5"));
```

`reduce(функция, последовательность)` предназначена для приведения последовательности значений к одному значению при помощи указанной функции, принимающей два аргумента и возвращающей один; результат – одно итоговое значение. В примере приводится нахождение числа 5!:

```
def mult(x,y):
    return x*y
n=range(1,6)
f=reduce(mult,n)
print f
>>> 120
```

`filter(функция, последовательность)` предназначена для формирования новой последовательности из исходной, при этом включение элемента определяется указанной функцией, принимающей значения 1 (включать) или 0 (не включать). В примере приводится формирование списка только из числовых элементов:

```
import string
def number(x):
    if x>=0 and x<=10**10:
        return 1
    else:
        return 0
s=[333,0.5,"abc",2,0,"fff",5.5]
f=filter(number,s)
print f
>>> [333, 0.5, 2, 0, 5.5]
```

2 ЯЗЫК ПРОГРАММИРОВАНИЯ JAVA

Язык Java разработан компанией Sun Microsystems (создатель Джеймс Гослинг – ныне вице-президент этой компании) и является свободно распространяемым. Трансляторы этого языка существуют для большинства существующих операционных систем; программа транслируется не в файл, выполняемый системой (например, .exe), а в специальный байт-код, выполняемый так называемой «виртуальной машиной Java» (JVM).

Если в C++ классы создавались внутри программы (главной функции main), то в Java основной программной единицей является именно класс (class), внутри которого, как правило, описывается компонентная функция main (если это необходимо). Расположение класса определяется средой разработки (как правило, классы сохраняются в т.н. пакетах – packages).

Исходный код класса сохраняется в файле с расширением java, класс после преобразования в байт-код JVM – в файле с расширением class.

Для задания идентификаторов Java позволяет использовать любые символы, например: `static final double π =3.14159;`

В отличие от C++, поля могут быть явно инициализированы. При отсутствии инициализаторов полям присваиваются значения по умолчанию (для чисел – 0, для строк – символ конца строки, для логических – false, для указателей – null).

Перед каждым идентификатором в java-программе может быть сколько угодно модификаторов, записываемых через пробел в любой последовательности, например: `public static void main(String[] args) { }`

Здесь описывается главная функция класса, которая ничего не возвращает (void), не меняется для разных экземпляров данного класса (static) и является общедоступной (public). Пример статических методов – функции класса Math (обращение идет не к экземпляру, а к самому классу: `y=Math.sqrt(x)`).

Возможны еще такие модификаторы:

`final` – описываемые данные являются константами (`static final int max=50`) или такой класс не допускает наследования;

`abstract` – такой класс не допускает создания экземпляров;

`private`, `package`, `protected`, `public` – уровень доступа;

`native` – определение метода, описанного на другом языке программирования (как правило, C++).

Наследование классов задается словом «extends»:

```
class Krug extends Point {}
```

Абстрактный базовый класс в Java называется интерфейсом, однако он может иметь атрибуты, даже снабженные модификаторами `static` и `final`. По умолчанию все компоненты интерфейса являются общедоступными (`public`).

Пример:

```
interface gr_obj {  
    int x,y;  
    void draw(); }
```

Здесь задаются координаты объекта и имя функции для рисования объекта; сама функция будет описана в классах – потомках.

Объекты создаются с использованием команды `new`:

```
Krug k1 = new Krug();
```

Каждому методу класса неявно передаются указатели на вызвавший его объект (`this`) и предок этого объекта (`super`). При этом допускается использование этих имен в качестве явных вызовов конструкторов: `this()`; `super()`;

Приведение типов осуществляется указанием имени типа в скобках перед именем переменной, например: `mref=(More)sref`; (здесь `mref` и `sref` – переменные, `More` – тип данных).

Проверка принадлежности объекта классу происходит при помощи оператора `instanceof`:

```
if (k2 instanceof Krug) Krug k3=(Krug)k2;
```

Все стандартные классы в Java хранятся в пакетах. При обращении к компонентам пакета нужно либо каждый раз полностью писать его имя:

```
java.util.Date now = new java.util.Date();
```

либо предварительно (перед началом описания класса) импортировать этот пакет:

```
import java.util.Date;  
Date now = new Date();
```

На вершине иерархии классов Java расположен класс `Object`, содержащий следующие методы:

`boolean equals (Object obj2)` – проверяет совпадение содержимого текущего объекта и объекта, заданного в качестве параметра `obj2`;

`Object clone()` – клонирует текущий объект, т.е. возвращает его копию;

`final Class getClass()` – возвращает информацию о классе текущего объекта;

`void finalize()` – деструктор;

`String toString()` – возвращает строковое представление объекта.

Последние два метода могут (и должны) быть переопределены в классах-потомках.

Java разрешает неявное преобразование типов: переменная более узкого типа может быть автоматически преобразована к более широкому:

```
int a=5;
```

```
System.out.println("k= "+a);
```

СПИСОК РЕКОМЕНДОВАННОЙ ЛИТЕРАТУРЫ

1. Иван ван Лейнингем. Освой самостоятельно Python за 24 часа.: Пер с англ. – М.: Издательский дом «Вильямс», 2001. – 448 с.
2. Кен Арнолд, Джеймс Гослинг, Дэвид Холмс. Язык программирования Java. 3-е изд.: Пер. с англ. – М.: Издательский дом «Вильямс», 2001.–624 с.
3. Дэвид М. Бизли. Python. Подробный справочник, 4-е издание. – М.: Символ, 2010. – 864 с.
4. Прохоренок Н.А. Python: Самое необходимое. – СПб.: BHV, 2010. – 416 с.
5. Марк Лутц. Изучаем Python. – М.: Символ, 2011. – 848 с.
6. Марк Саммерфилд. Программирование на Python 3. Подробное руководство. – М.: Символ, 2009. – 608 с.
7. Аккуратов Е.Е. Знакомьтесь: Java. Самоучитель. – М.: Вильямс, 2006. – 256 с.
8. Кей С. Хорстманн, Гари Корнелл. Java 2. Библиотека профессионала. Том 1. Основы. – М.: Вильямс, 2006. – 896 с.