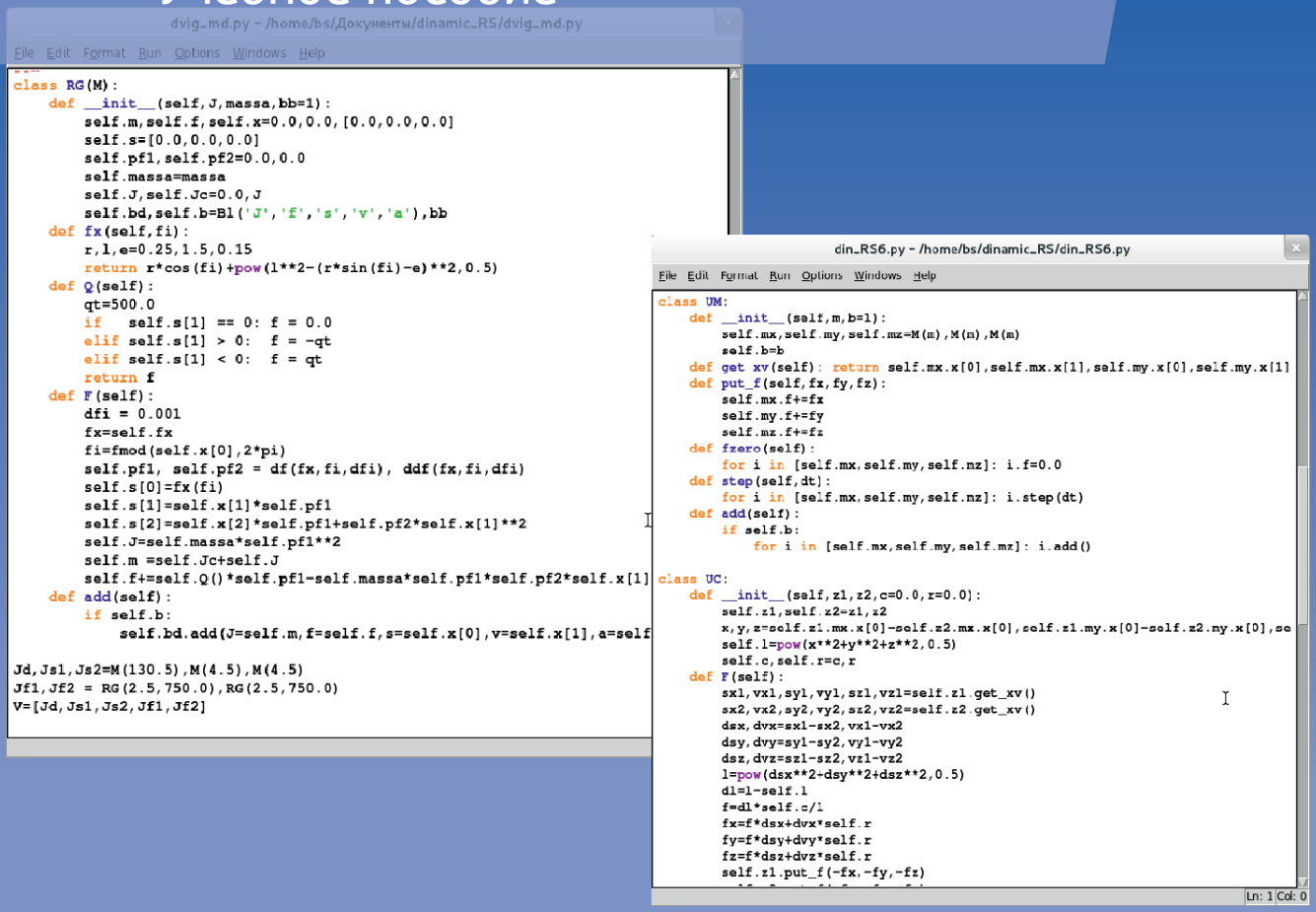
Уральский федеральный университет
имени первого Президента России Б.Н.Ельцина
Механико-машиностроительный институт

С. К. БУЙНАЧЕВ
Н. Ю. БОКЛАГ

ОСНОВЫ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ PYTHON

Учебное пособие



Министерство образования и науки Российской Федерации

Уральский федеральный университет
имени первого Президента России Б. Н. Ельцина

С. К. Буйначев, Н. Ю. Боклаг

ОСНОВЫ ПРОГРАММИРОВАНИЯ НА ЯЗЫКЕ PYTHON

Рекомендовано методическим советом УрФУ
*в качестве **учебного пособия** для студентов,*
обучающихся по направлениям подготовки
151000 «Технологические машины и оборудование»,
190100 «Наземные транспортно-технологические комплексы»,
190600 «Эксплуатация транспортно-технологических машин
и комплексов»

Екатеринбург
Издательство Уральского университета
2014

УДК 004.432Python(075.8)
ББК 32.973.26-018.1Pythonя73
Б90

Рецензенты: доц., д-р техн. наук Е. Е. Баженов (Уральский государственный экономический университет);

доц., канд. техн. наук В. П. Подогов (Российский государственный профессионально-педагогический университет)

Научный редактор – доц., канд. техн. наук Ю. В. Песин

Буйначев, С. К.

Б90 Основы программирования на языке Python : учебное пособие / С. К. Буйначев, Н. Ю. Боклаг. – Екатеринбург : Изд-во Урал. ун-та, 2014. – 91, [1] с.

ISBN 978-5-7996-1198-9

Пособие содержит начальные сведения о программировании на языке Python и является основой для изучения курса «Численные методы и оптимизация». Собраны сведения из книг таких известных авторов, как Г. Россум, М. Лутц, Р. Сузи, Д. Бизли, А. Лесса. Предложен новый подход к использованию баз данных для накопления результатов расчета с дальнейшим анализом и визуализацией решений.

Может быть рекомендовано студентам различных специальностей технических вузов, занимающихся программированием, математическим моделированием и численными методами, а также может служить справочным материалом при выполнении курсовых и дипломных работ, связанных с расчетами на компьютере.

Библиогр.: 10 назв.

УДК 004.432Python(075.8)
ББК 32.973.26-018.1Pythonя73

ISBN 978-5-7996-1198-9

ОГЛАВЛЕНИЕ

1. ЗАПУСК ОБОЛОЧКИ ПРОГРАММ И ИНСТРУКЦИИ ЯЗЫКА PYTHON

Программы Python выполняются интерпретатором. На компьютерах с системами Unix и Linux интерпретатор можно вызвать, набрав команду **python.** В системах Windows и Macintosh интерпретатор можно запустить как приложение (либо из меню Start, либо двойным щелчком на пиктограмме интерпретатора). После запуска интерпретатора появляется подсказка, в которой можно начать отладку операторов программы в простом цикле чтения/выполнения. Например, в приведенном ниже выводе интерпретатор отображает сообщение об авторских правах и предоставляет пользователю подсказку >>>, в которой пользователь набирает знакомую команду "Hello World":

```
Python 1.5.2 (#0, Jun I 1999, 20:22:04)
Copyright 1991-1995 Stichting Mathematisch Centrum, Amsterdam
>>> print "Hello World"
Hello World
>>>
```

Программы можно также помещать в файл:

```
# helloworld.py
print "Hello World"
```

Исходные файлы Python имеют расширение *.py. Символ # в предыдущем примере обозначает комментарий, который продолжается до конца строки.

В системе Windows программы Python можно запускать двойным щелчком на файле с расширением .py. При этом происходит запуск интерпретатора и выполнение программы в окне терминала. В таком случае окно терминала немедленно исчезает после того, как программа завершает свое выполнение (чаще всего прежде, чем удается прочитать ее вывод). Чтобы избежать этого, можно воспользоваться средой интегрированной разработки (Idle или Pythonwin). Альтернативным методом является запуск программы с использованием пакетного файла с расширением *.bat, python -i helloworld.py, содержащего оператор типа python -i helloworld.py, который указывает интерпретатору, чтобы он перешел в интерактивный режим после

выполнения программы. Можно также изменить расширение файла на *.pyw, что в Windows означает запуск как исполняемого файла (без использования консоли).

В системе Macintosh программы можно выполнять из встроенной среды интегрированной разработки. Кроме того, утилита BuildApplet (включенная в дистрибутив) позволяет преобразовать программу Python в документ, который автоматически запускается интерпретатором при его открытии.

В интерпретаторе программу можно выполнить с помощью функции execfile(), как показано в следующем примере:

```
>>> execfile("helloworld.py")
Hello World
```

В системе Unix можно также вызывать Python с использованием символов **#!** в сценарии командного интерпретатора:

```
#!/usr/local/bin/python
print "Hello World"
```

Интерпретатор продолжает работу до тех пор, пока не достигнет конца входного файла. При интерактивном выполнении можно выйти из него, введя символ EOF (end of file – конец файла) или выбрав Exit из выпадающего меню (если оно имеется). В Unix в качестве символа EOF служит <Ctrl+D>; в Windows – <Ctrl+Z>.

Из программы можно также выйти, вызвав функцию sys.exit() или активизировав исключение **SystemExit** (это эквивалентно). Например:

```
>>> import sys
>>> sys.exit()
```

или

```
>>> raise SystemExit
```

В программе Python выделяются следующие ступени иерархии:

- программы делятся на модули;
- модули содержат инструкции;
- инструкции состоят из выражений;
- выражения создают и обрабатывают объекты.

Инструкции в языке Python приведены ниже.

Инструкция	Роль	Пример
1	2	3
Присваивание	Создание ссылок	a,b,c='ножницы','бумага', 'камень'
Вызовы	Запуск функций	f.write('Пролог\n')
print	Вывод на консоль	print 'Знание – сила'
if/elif/else	Операция выбора	if 'python' in text: print text
for/else	Обработка последовательности в цикле	for x in thelist: print x
while/else	Цикл общего назначения	while x>y: y+=1
pass	Пустая инструкция	if a: pass
break, continue	Переходы в теле цикла	while 1: if not in line: break
try/except/ finally	Обработка исключений	try: action() except: print 'action error'
raise	Возбуждение исключений	raise endSearch, location
import, from	Доступ к модулям	import sys from sys import stdin

def, return, yield	Создание функции	def f(a,b,c=1,*d):
		return a+b+c+d[0]
		def gen(n):
		for i in n,
		yield i*2

class	Описание класса	class subclass(Superclass): staticData=[]
global	Пространство имен	def function(): global x,y x='new'

Окончание таблицы

1	2	3
del	Удаление ссылок	del data[k] del data[i:j] del obj.attr del variable
exec	Запуск фрагментов программного кода	exec 'import '+modName exec code in gdict, ldict
assert	Отладочные проверки	assert x > y
with/as	Менеджеры контекста	with open('data') as myfile: process(myfile)

2. ВСТРОЕННЫЕ ТИПЫ ДАННЫХ

Python – это язык с динамическим контролем типа, в котором имена во время выполнения программы могут представлять значения различных типов. И действительно, имена, используемые в программе, – это только метки для различных величин и объектов. Оператор присваивания просто создает связь между именем и значением. В этом состоит одно из отличий данного языка, например, от C, в котором имена представлены объектами с постоянным размером и размещением в памяти, где находятся результаты.

Все данные в Python представлены объектами. Имена являются лишь ссылками на эти объекты и не несут нагрузки по декларации типа. Значения встроенных типов имеют специальную поддержку в синтаксисе языка: можно записать литерал строки, числа, списка, кортежа, словаря (и их разновидностей). Синтаксическую же поддержку операций над встроенными типами можно легко сделать доступной и для объектов определяемых пользователями классов.

Следует также отметить, что объекты могут быть *неизменчивыми* и *изменчивыми.* Например, строки в Python являются неизменчивыми, поэтому операции над строками создают новые строки.

Карта встроенных типов (с именами функций для приведения к нужному типу и именами классов для наследования от этих типов):

1. Специальные типы: None, NotImplemented и Ellipsis;
2. Числа: