

Министерство образования и науки Российской Федерации
Ярославский государственный университет им. П. Г. Демидова
Кафедра компьютерных сетей

Информатика

Задания для лабораторных работ

Практикум

Ярославль
ЯрГУ
2017

УДК 004:378.147.88(076.5)

ББК 3973.2я73

И74

Рекомендовано

*Редакционно-издательским советом университета
в качестве учебного издания. План 2017 года.*

Рецензент

кафедра компьютерных сетей

Ярославского государственного университета им. П. Г. Демидова

Составитель

М. В. Краснов

**И74 Информатика : Задания для лабораторных работ :
практикум / сост. М. В. Краснов ; Яросл. гос. ун-т
им. П. Г. Демидова. — Ярославль : ЯрГУ, 2017. — 72 с.**

Основное использование вычислительной техники связано с обработкой информации. В настоящей работе рассматриваются вопросы построения программ на языке Python. Пособие предназначено для бакалавров и магистров, обучающихся по курсу «Информатика».

УДК 004:378.147.88(076.5)

ББК 3973.2я73

© ЯрГУ, 2017

Введение

Практикум содержит пять лабораторных работ, посвященных языку программирования Python, каждая из которых включает цель работы, краткие теоретические сведения, необходимые для выполнения задания и индивидуальные задачи.

Python – это интерпретируемый язык (код не компилируется в машинные команды, а выполняется специальной программой – интерпретатором). На сегодняшний день поддерживается две версии языка Python 2.x и Python 3.x (*обратите внимание*: версии языка имеют не полную совместимость). Стандартный интерпретатор запускается из командной строки командой *python*.

Программа на языке Python может состоять из одного или нескольких модулей. Каждый модуль представляет собой текстовый файл в кодировке, совместимой с 7-битной кодировкой ASCII. Для кодировок, использующих старший бит, необходимо явно указывать название кодировки. Например, если есть желание использовать комментарии на русском языке, то файл в первой строке должен иметь следующую спецификацию:

-*- coding: koi8-r -*- или # -*- coding: cp1251 -*-

Студентам следует обратить внимание на то, что все данные в Python представлены объектами. С каждым объектом связывается тип, внутренние данные и ассоциированные с ними наборы операций. Имена (переменные) являются лишь ссылками на эти объекты и не несут нагрузки по декларации типа. Связывание данных с переменной выполняется с помощью оператора присваивания. Отметим, что в качестве переменных нельзя использовать ключевые слова:

Таблица 1

Зарезервированные слова Pythona

and	as	assert	break	class	continue	def	del	elif
else	except	exec	False	finally	for	from	global	if
import	in	is	lambda	None	nonlocal	not	or	pass
raise	return	True	try	while	with	yield		

Объекты в Python обычно имеют атрибуты – поля (другие объекты), хранящиеся «внутри» данного объекта, и методы (функции), которые ассоциируются с объектом, имеющие доступ к внутреннему состоянию объекта (obj).

Студентам следует обратить внимание, что для структурирования кода в программах важны отступы, поэтому все операторы, входящие в один блок действий, должны иметь один и тот же отступ.

Лабораторная работа № 1

Тема: основы языка Python.

Цель работы:

- 1) познакомиться со встроенными типами языка Python, для каждого типа перечислить основные функции;
- 2) познакомиться с основными конструкциями языка Python.

Краткие теоретические сведения

Встроенные типы данных

- специальные типы:
 - None – это специальная константа в Python. Она обозначает пустое значение. None имеет свой собственный тип (NoneType);
- числа:
 - целые
 - обычное целое int
 - целое произвольной точности long
 - логический bool
 - число с плавающей точкой float;
- последовательности:
 - неизменяемые:
 - строка str
 - Unicode-строка unicode
 - кортеж tuple

- изменяемые:
 - список list;
- отображения:
 - словарь dict;
- множества set;
- объекты, которые можно вызывать:
 - функции (пользовательские и встроенные)
 - функции-генераторы
 - методы (пользовательские и встроенные)
 - классы (новые и «классические»);
- файлы file;
- модули.

Узнать тип любого объекта можно с помощью функции `type()`. Особое внимание следует обратить на следующие инструкции:

`help(<object>)` — выдает справку по объекту;
`dir(<object>)` — показывает все методы и свойство объекта;
`<object>.__doc__` — показывает строку документации.

Числовые типы:

- *типы int, long, float*

Таблица 2

Операторы и функции числового типа

<i>Оператор, функция</i>	<i>1</i>	<i>2</i>	<i>3</i>	<i>Пример</i>	<i>4</i>	<i>5</i>	<i>Описание (в результате получим)</i>
<code>abs()</code>	+	+	+	<code>abs(-10)</code>	10	число	вычисляет абсолютное значение
<code>bin();</code> <code>hex();</code> <code>oct()</code>	+	+		<code>bin(10);</code> <code>hex(10);</code> <code>oct(10)</code>	'0b1010'; '0xa'; '012'	str	выдает двоичную, шестнадцатеричную или восьмеричную строку
<code>chr()</code>	+	+		<code>chr(70)</code>	'F'	str	символ
<code>divmod()</code>	+	+	+	<code>divmod(10,3)</code>	(3, 1)	tuple	деление по модулю

Оператор, функция	1	2	3	Пример	4	5	Описание (в результате получим)
float()				float(5)	5.0	float	число с плавающей запятой
int()	+	+	+	int(10.77)	10	int	приведение к типу int
pow(); **	+	+	+	pow(10,3)	1000	числовой	выполняется возведение в степень
round()			+	round(10.77) round(10.3)	11.0 10.0	float	округление с плавающей запятой
/	+	+	+	11/3.0 11/3	3.6666 3	числовой	классическое деление (обязательно, чтобы одно из чисел было float)
//	+	+	+	11//3.0	3.0	числовой	деление с округлением вниз
%	+	+	+	11%3.0	2.0	числовой	остаток после (целочисленного деления), по модулю
и ^; ~d и &	+	+		5 6 и 5^6; ~5 и 5&6	7 и 3; – 6 и 4	int, long	битовое или (OR); исключающее или (XOR); битовое отрицание (not); битовое и (AND)
<< и >>	+	+		5<<1 и 5>>1	10 и 2	int, long	сдвиг на n двоичных разрядов влево и сдвиг на n двоичных разрядов вправо

Обозначения: 1 - int, 2 - long, 3 - float, 4 - результат, 5 - тип результата, + - есть функция

- *логический тип*

Не следует забывать о логическом типе (тип *bool*). Элементы этого типа True, False.

Рассмотрим некоторые операции для логического типа¹

¹ Аналогичная таблица рассматривается в работе Swaroop Chitlur «A Byte of Python». Пусть x = True, y = False.

Таблица 3

Операторы логического типа

Операция	Результат или пояснения	Операция	Результат или пояснения
type(x)	<type 'bool'>	int(x)	1
x+x	2	x-y	1
not x	отрицание результат False	x and y	логическое и результат False
x or y	логическое или результат True	y/x	0

Таблица 4

Операторы сравнения

Операция	Результат или пояснения	Операция	Результат или пояснения
x > y	условие x больше	x < y	условие x меньше
x >= y	(больше или равно) y	x <= y	(меньше или равно) y
x == y	условие равенства	x != y	условие неравенства

Последовательности

Последовательности – упорядоченные коллекции других объектов. Они поддерживают порядок размещения элементов, которые содержат, слева направо: элементы сохраняются и извлекаются исходя из их позиций в последовательностях.

В Python последовательности подразделяются на неизменяемые (строки, кортежи tuple) и изменяемые (списки list).

Таблица 5

Операторы и функции для последовательностей

Операция	Результат или пояснения
len (A)	возвращает длину последовательности A
A[i]	возвращает i-й элемент последовательности A или len (A) + i-й, если i < 0; или исключение IndexError, если i несуществующий индекс

<i>Операция</i>	<i>Результат или пояснения</i>
$x \in A$	проверка принадлежности элемента последовательности A
$A + B$	конкатенация последовательностей A и B одного типа
$A[i : j : d]$	срез из последовательности A от i до j (с шагом d). Обратите внимание, что элемент, указанный в качестве конечного, не включается в результат
$\min(A)$	возвращает наименьший элемент последовательности A
$A.index(x)$	возвращает наименьшее i , такое что $A[i]$ равно x или исключение <code>ValueError</code> , если x не найден в последовательности A

- *Строки*

Строки в Python бывают обычные и `unicode`. Фактически строка – это последовательность символов. Строковый литерал записывается в одиночных (') или двойных (") кавычках.

Таблица 6

Операторы и функции для строк

<i>Функция, атрибут</i>	<i>Результат</i>	<i>Функция, атрибут</i>	<i>Результат</i>
<code>str1="Python"</code>	создается строка <code>str1</code>	<code>type(str1)</code>	<code><type 'str'></code>
<code>str2="python"*2</code>	получим <code>'pythonpython'</code>	<code>"python"+"!"</code>	получим <code>'python!'</code>
<code>str_u=u"python"</code>	создается строка <code>str_u</code>	<code>str2=str2[:3]+'!'+str2[4:]</code>	получим <code>'pyt!onpython'</code>
<code>str1.upper()</code>	получим <code>'PYTHON'</code> , но строка <code>str1</code> по-прежнему остается «Python»	<code>str1.lower()</code>	получим <code>'python'</code> , но строка <code>str1</code> по-прежнему остается «Python»
<code>str1.replace('Py','Da')</code>	замена, в результате получим <code>'Dathon'</code>	<code>str1.find('th')</code>	возвращает индекс первого вхождения в <code>str1</code> подстроки <code>'th'</code>
<code>str2.count('python')</code>	возвращает количество неперекрывающихся вхождений подстроки <code>'python'</code> в строке <code>str2</code> .	<code>str2.startswith('p')</code>	возвращает <code>True</code> , если строка <code>str2</code> начинается с указанного префикса, иначе <code>False</code> .

Достаточно часто необходимо выполнить форматирование строк или подставить в строку некоторые данные. Это выполняется с помощью метода `format()`. Рассмотрим несколько примеров:

Таблица 7

Работа с методом `format()`

<i>Пример</i>	<i>Результат</i>	<i>Пример</i>	<i>Результат</i>
'{} python'.format('Language')	'Language python'	print('{1} и {0}'.format('си', 'java'))	будет выведено java и си
print('{b1} и {b2}'.format(b1='си', b2='java'))	результат: си и java	name = ('java', 'си') print('{1} и {0}'.format(*name))	будет выведено си и java
name = ('java', 'си') print('{0:15} и {1:5}'.format(*name))	результат: java и си	print('{1:_>5} и {0:r>5}'.format('си', 'java'))	будет выведено _java и гггси

Многие объекты Python можно преобразовать в строку с помощью функции `str()`.

- ***Кортежи `tuple`***

Они применяются для представления константной последовательности (разнородных) объектов.

Таблица 8

Операторы и функции типа `tuple`

<i>Функция, атрибут</i>	<i>Результат или пояснения</i>	<i>Функция, атрибут</i>	<i>Результат или пояснения</i>
<code>b1=()</code>	создается пустой кортеж	<code>b=(5,)</code>	создается кортеж <code>b</code>
<code>a1=(1,2,3,'aaa')</code> <code>a2=(1.0,2.5)</code>	создается кортеж <code>a1</code> создается кортеж <code>a2</code>	<code>type(a1)</code>	<type 'tuple'>
<code>print a1[-1]</code>	'aaa'	<code>a1,a2=a2,a1</code>	<code>a1=(1.0,2.5)</code> <code>a2=(1,2,3,'aaa')</code>

- ***Списки `list`***

Список позволяет хранить разнородные данные, обращаться к которым можно через имя списка².

² Пусть даны списки `L = [1,2,1,2,3,4,5,6,7]` и `L1=[3,8]`

Таблица 9

Операторы и функции создания списков

<i>Функция, атрибут</i>	<i>Результат или пояснения</i>	<i>Функция, атрибут</i>	<i>Результат или пояснения</i>
<code>a=[1,2,3]</code>	создается список <code>a</code>	<code>c = [x**2 for x in range(4)]</code>	создается список <code>c=[0,1,4,9]</code>
<code>b=range(4)</code>	создается список <code>b=[0,1,2,3]</code>	<code>b=[]</code>	создание пустого списка
<code>d=c[:]</code> или <code>d=list(c)</code>	создание копии списка; Пусть <code>d[0]=5</code> , тогда <code>d = [5,1,4,9]</code> , <code>c = [0,1,4,9]</code> .	<code>d=c</code>	создание второй ссылки на список. Пусть <code>d[0]=5</code> , тогда <code>d= [5,1,4,9]</code> , <code>c=[5,1,4,9]</code> .
<code>for x in L:</code> <code>print x</code>	итерация прямая. Получим 1,2,1,2,3,4,5,6,7	<code>for x in reversed(L):</code> <code>print x</code>	итерация обратная. Получим: 7,6,5,4,3,2,1,2,1
<code>for x in set(L):</code> <code>print x</code>	уникальная итерация. Получим: 1234567	<code>S = L+L1</code>	оператор сложения (+) выполняет конкатенацию списков. Получим <code>S=[1,2,1,2,4,5,6,7,3,8]</code>

Таблица 10

Операторы и функции для списков³

<i>Функция, атрибут</i>	<i>Результат или пояснения</i>
<code>L.append(x)</code>	добавляет элемент <code>x</code> в конец последовательности
<code>L.count(x)</code>	считает количество элементов, равных <code>x</code>
<code>L.extend(x)</code>	добавляет к концу последовательности последовательность <code>x</code>
<code>L.pop(i)</code>	возвращает <i>i</i> -й элемент, удаляя его из последовательности
<code>L.reverse()</code>	меняет порядок элементов в последовательности на обратный

³ Предположим, что у нас есть список *L*.

<i>Функция, атрибут</i>	<i>Результат или пояснения</i>
<code>L.remove(x)</code>	удаляет элемент <i>x</i> из списка <i>L</i> . Если такого элемента нет, возникает сообщение об ошибке ⁴ .
<code>L.insert(n, "T")</code>	метод используется для вставки элемента в середину списка, где <i>n</i> – позиция вставки элемента
<code>L[i] = x</code>	<i>i</i> -й элемент списка <i>L</i> заменяется на <i>x</i>
<code>L[i : j : d] = t</code>	срез от <i>i</i> до <i>j</i> (с шагом <i>d</i>) заменяется на (список) <i>t</i>
<code>del L[i : j : d]</code>	удаление элементов среза из последовательности

Для взаимного преобразования строк и списков используются методы `split()`, `join()` и функция `list()`

- `split()` — Метод делит строку по заданному символу-разделителю и создаёт список из фрагментов строки. Предположим, что существует строка `s='11 151 dd'`, тогда, выполнив `a=s.split(' ')`, получим список `a=['11', '151', 'dd']`
- `join()` — Метод формирует строку из элементов списка, поставив между ними заданную строку. Предположим, что существует список `a=['11', '151', 'dd']`, тогда, выполнив `s=" ".join(a)`, получим строку `s='11 151 dd'`
- `list(s)` — Функция пытается преобразовать аргумент *s* в список. Предположим, что существует строка `s='python!'`, тогда, выполнив `a=list(s)`, получим список `a=['p', 'y', 't', 'h', 'o', 'n', '!']`

Студентам не следует забывать, что в качестве элементов списка могут быть объекты любого типа, например списки: `lst = [['python',1],['c++',2],['java',3]]`. Обращение к элементу списка происходит через указания индекса, а к элементу вложенного списка через указание двух индексов:

`lst[0]` получим `['python', 1]` `lst[1][0]` получим `'c++'`

⁴ Если элементов *x* в списке *L* несколько, то удаляется первый попавшийся элемент слева.

Пример. Реализуйте с помощью списков решето Эратосфена

```
#-*- coding: cp1251 -*-
def eratosthenes (n): # создаем функцию
    primes = []      # создаем пустой список
    multiples = []
    for i in xrange(2, n + 1):
        if i not in multiples:
            primes.append(i)
    # добавляет элемент в конец последовательности
    multiples.extend(xrange(i * i, n + 1, i))
    # добавляет к концу последовательности другую последовательность
    return (primes, multiples) # возврат кортежа
n=100
print(eratosthenes (n)) #вызов функции
```

Отображения

- *Словарь dict*

Словарь (хэш, ассоциативный массив) — это изменчивая структура данных для хранения пар ключ-значение, где значение однозначно определяется ключом. Отображения не подразумевают какого-либо упорядочения элементов по их позиции. В качестве ключа может выступать неизменяемый тип данных. Доступ к элементам словаря производится по ключу.

Таблица 11

Операторы и функции для типа dict

Функция, атрибут	Результат или пояснение	Функция, атрибут	Результат или пояснения
a={1:'10',2:'20',4:'dd'}	создается словарь a	print a[4]	берет значение по ключу и выдает dd
a[1]='ddd'	присваивается значение по ключу a={1:'ddd',2:'20',4:'dd'}	del a[4]	удаляется пара ключ-значение с данным ключом

<i>Функция, атрибут</i>	<i>Результат или пояснение</i>	<i>Функция, атрибут</i>	<i>Результат или пояснения</i>
<code>d=dict(zip('ab',[1,2]))</code>	создается словарь {'a': 1, 'b': 2}	<code>a={} или a=dict()</code>	создается пу- стой словарь
<code>d.keys()</code>	возвращает все ключи из словаря d: ['a', 'b']	<code>d.values()</code>	возвращает все значения из сло- варя d: [1, 2]
<code>d.items()</code>	возвращает кор- тежные пары (ключ, значение) словаря d: [('a', 1), ('b', 2)]	<code>d.update(c)</code>	объединяет все записи в сло- варь d. Пусть c={'a': 5, 'c': 2}, тогда d: {'a': 5, 'c': 2, 'b': 2}
оператор <code>in</code>	поиск по множе- ству ключей: 'a' in d возвращает True	<code>d={}.fromkeys(['a', 'b'])</code>	создает словарь {'a': None, 'b': None}

Пример. Подсчитайте частоту повторов уникальных слов в текстовом файле.

```

#-*- coding: cp1251 -*-
import sys
u = open("file_name")
words = {} # связываем имя words с пустым словарём
    or line in u.readlines(): # читаем и по строкам
        line = line.strip("\n")
# отбрасываем начальные и конечные пробелы
    for word in line.split(" "):
# режем каждую строку на слова, ограниченные пробелами
        try: # блок обработки исключений
            words[word] += 1
# пытаемся увеличить words[word] на единицу
        except KeyError: #если не получилось (раньше words[word] не было)
            #Исключение KeyError возникает, если ключа в отображении нет.
            words[word] = 1 # создаем в словаре пару[word:1]
    for word in words: # распечатываем word, words[word]
        print(word, words[word])

```

Множества

Множества – это неупорядоченная коллекция неизменяемых, уникальных элементов. В отличие от кортежей, множества не предусматривают возможность доступа к элементам по числовому индексу. Заметим, что элементы множества никогда не повторяются.

Таблица 12

Операторы и функции для множеств

Функция, атрибут	Результат или пояснение	Функция, атрибут	Результат или пояснения
$A = \{'a', 'c', 4, 'c'\}$ $B = \text{set}('abca')$	создали множество $\text{set}(['a', 'c', 4])$ создали множество $\text{set}(['a', 'c', 'b'])$	$C = \text{set}([])$	создали пустое множество
$x \text{ in } A$	возвращает значение True, если множество A содержит x	$C = B - A$	множество C содержит элементы, которые входят в множество B и не входят в множество A: $\text{set}(['b'])$
$C = A B$	объединение множеств: $\text{set}(['a', 'c', 'b', 4])$	$C = A \& B$	пересечение множеств: $\text{set}(['a', 'c'])$
$C = A \leq B$	возвращает True, если каждый элемент из A присутствует в множестве B	$C = A \wedge B$	симметричная разность множеств: $\text{set}(['b', 4])$
$\text{len}(A)$	возвращает длину множества A	$\text{type}(A)$	<code><type 'set'></code>
$A.\text{remove}(x)$	удаление элемента из множества A по значению(x)	$A.\text{add}(x)$	пытается добавить элемент x в множество A
$A.\text{update}([1,3])$	добавляет несколько элементов в множество A	$A.\text{copy}()$	возвращает копию множества
$A.\text{discard}(x)$	удаляет элемент x, если он находится в множестве.	$A.\text{clear}()$	очищает множество

Пример. *Определите, сколько различных слов содержится в текстовом файле.*

```
.....
#-*- coding: cp1251 -*-
try:
    u = open("myfile")
    words = set([])
    # связываем имя words с пустым множеством
    for line in u.readlines(): # читаем и по строкам
        line = line.strip(" \n")
    # отбрасываем начальные и конечные пробелы
    for word in line.split(" "):
    # режем каждую строку на слова, ограниченные пробелами
        words.add(word)
    # пытаемся добавить слово в множество
    print(len(words))
    # выводим количество различных слов в тексте
except:
    print «Error»
.....
```

Объекты, которые можно вызвать:

- *Функции.* Для того, чтобы избежать повторного написания одного и того же кода в языках программирования, используются функции. Определение функции:

```
.....
def name_function(arguments): # определение функции
    #code                     # код функции
    return values             # команда: вернуть результат.
                              Она может отсутствовать. Тогда
                              выполняются все команды тела
                              функции и возвращается значение
                              None.
.....
```

Имя функции должно быть уникальным идентификатором. Для вызова функции нужно просто указать ее имя, за которым должны следовать ее параметры, заключенные в круглые скобки.

Когда внутри функции создаются новые переменные, они имеют локальную область видимости. Чтобы иметь возможность изменять глобальные переменные внутри функции,

эти переменные следует определить в теле функции с помощью инструкции `global`:

Рассмотрим, как передать параметры в функцию		
<code>def rema(a, b):</code>	<code>def rema(a,b=5,c=4):</code>	<code>def rema(a=1,b=5,c=4):</code>
<code>q = a // b</code>	<code>q = a // b</code>	<code>q = a // b</code>
<code>r = a % b</code>	<code>r = a % b; r=r+c</code>	<code>r = a % b; r=r+c</code>
<code>return r</code>	<code>return r</code>	<code>return r</code>
<code>print rema(5,10)</code>	<code>print rema(9,c=2)</code>	<code>print rema(a=9,c=2,b=7)</code>
результат 5	результат 6	результат 4

Может понадобиться, чтобы определяемая функция принимала любое число параметров. Будем использовать в качестве параметров кортеж или словарь.

Пример	Пример
<code>def list_sum(*args):</code>	<code>def list_sum(a,**param):</code>
<code>smm = 0</code>	<code>smm = 0</code>
<code>for arg in args:</code>	<code>for key in param:</code>
<code>smm += arg</code>	<code>smm += param[key]</code>
<code>return smm</code>	<code>return smm</code>
<code>list_sum(9,7,2)</code>	<code>list_sum(9,module=7,noise=2)</code>
Когда объявляем параметр со звёздочкой (например, <code>*args</code>), все наши позиционные параметры «упаковываются», начиная с этой позиции и до конца в кортеж <code>args</code> в соответствии с их «порядковым номером» при передаче.	Функция может принимать произвольное число параметров, заданных по ключу. Когда объявляем параметр с двумя звёздочками (например, <code>**param</code>), все ключевые аргументы, начиная с этой позиции и до конца, будут собраны в словарь под именем <code>param</code> .
Результат 18	Результат 9

Отметим, некоторые интересные моменты, связанные с функциями:

– одна функция может принимать в качестве аргументов разные типы данных. Например, рассмотрим следующую функцию:

```
def f(x):
    return x * 5
```

Вызов функции может быть таким `f(5)` или таким `f([1,2,5,4])`;

– Python позволяет передавать результаты вызова функций в качестве аргументов других функций без использования дополнительных переменных, например `str(pow(10,2))`. Имена функций в Python являются переменными, содержащими адрес объекта типа функция, поэтому этот адрес можно присвоить другой переменной и вызвать функцию с другим именем, например: `s=pow;str(s(10,2))`.

- *Функции – генераторы.* При использовании инструкции `yield` функция может генерировать последовательность результатов. При вызове функции-генератора создается объект с методом `next()`, который заставляет функцию выполняться, пока не будет достигнута следующая инструкция `yield`.

Рассмотрим работу функции-генератора:

Пример	Пояснение или результат	
<code>def coun(t,n,d=5):</code>	Будет выведено	
<code>while n < d:</code>	1	2
<code>t=2*t</code>	2	4
<code>yield t</code> # Генерирует значение (t)	3	8
<code>n += 1</code>	4	16
<code>c=coun(1,0)</code>		
<code>for i in range(1,5):</code>		
<code>print i, ' ', c.next()</code>		

- *Классы.* Оператор `class` применяется для определения объектов новых типов и для объектно-ориентированного программирования. Простейшая модель определения класса выглядит следующим образом:

```
class имя_класса:
    инструкция_1
    инструкция_n
```

Атрибуты класса бывают двух видов: атрибуты данных; атрибуты-методы. В классе метод определяется с помощью оператора `def`. Первый параметр каждого метода всегда ссылается на сам объект (этот параметр имеет имя `self`). Все операции, затрагивающие атрибуты объекта, должны явно ссылаться на переменную `self`.

Tun file

Объекты этого типа предназначены для работы с внешними данными. В простом случае – это файл на диске. Для работы с файлом прежде всего нужно создать специальный объект, а потом использовать методы этого объекта для чтения и записи данных.

Таблица 13

Операторы и функции типа file

<i>Функция, атрибут</i>	<i>Описание</i>
<code>open(path [, mode='r'])</code> <code>f=open("data.txt")</code>	возвращает файловый объект. Параметр <code>mode</code> указывает вариант доступа. Если <code>mode</code> равен <code>'r'</code> , <code>'rb'</code> , <code>'w'</code> , <code>'wb'</code> , то указатель стоит в начале файла; если <code>mode= 'a'</code> , то указатель стоит в конце файла
<code>f.fileno()</code>	возвращает целочисленный дескриптор файла
<code>f.close()</code>	закрывает файл
<code>f.closed</code>	аргумент выдает <code>True</code> , если файл закрыт; в противном случае – <code>False</code>
<code>f.mode</code>	аргумент выдает режим доступа, с которым был открыт файл
<code>f.name</code>	аргумент выдает имя файла
<code>f.read(n)</code>	считываем <code>n</code> байтов файла, если нет аргументов, тогда читается файл целиком
<code>f.readline()</code>	читаем строку файла целиком
<code>f.readlines()</code>	функция выполняет чтение файла целиком построчно, в результате прочтения файла получается список строк.
<code>f.write(s)</code>	пишем строку <code>s</code>
<code>f.writelines(L)</code>	пишем все строки из списка <code>L</code>
<code>f.tell()</code>	функция возвращает текущую позицию в файле

<i>Функция, атрибут</i>	<i>Описание</i>
<code>f.seek(offset [,whence])</code>	функция используется для произвольного доступа к файлу. Аргументы: <code>offset</code> – смещение в байтах относительно <code>whence</code> ; <code>whence</code> – параметр начального смещения: 0 – смещение от начала файла, 1 – смещение от текущей позиции, 2 – смещение от конца файла. По умолчанию <code>whence</code> равен 0.

Пример. Прочитать текстовый файл и записать его содержимое в другой файл

```

try:
    f = open('my_file','r')      # открываем файл для чтения
    lines = f.readlines()
# чтение файла целиком построчно в список lines
    f.close()                  # закрываем файл
except:
    lines[1] = "Python \n"      # изменяем 1-ю строку
except:
    lines[0] = "Python \n"
f = open('my_file2','w') # открываем файл для записи
f.writelines(lines)      # пишем все строки из списка lines
f.close()                # закрываем файл
except:
    print("Error")

```

Модули

При модульном подходе к программированию большая задача разбивается на несколько более мелких задач, каждую из которых (в идеале) решает отдельный модуль. Программа на языке Python организована как один главный файл, к которому могут подключаться дополнительные файлы (модули). Студентам следует обратить внимание, что модули желательно располагать в текущем каталоге (где расположен главный файл) или в каталоге, упомянутом в переменной окружения PYTHONPATH. Чтобы использовать функцию из модуля, его

надо подключить. Существует несколько способов подключения модулей:

- подключить модуль целиком

```
import math  
print math.sqrt(4)
```
- подключить только функцию

```
from math import sqrt  
print sqrt(4)
```
- подключить все функции модуля (применение этого способа не всегда удобно, подключаемые модули могут иметь функции с одинаковыми именами)

```
from math import*  
print sqrt(4)
```

Если потребуется подключить модуль под другим именем, достаточно добавить в инструкцию `import` дополнительный параметр `as`:

```
import math as ma  
print ma.sqrt(4)
```

Некоторые инструкции языка Python

Таблица 14

Инструкции языка Python

<i>Инструкция</i>	<i>Пример</i>	<i>Пояснение или результат</i>
<code>print</code>	<code>print 'пример на Пайтоне'</code>	вывод на консоль
<code>raw_input(); input()</code>	<code>quest = raw_input("input--- ")</code>	читает пользовательский ввод из потока стандартного ввода и возвращает строку без завершающего символа новой строки
<code>range()</code>	<code>a=range(25)</code>	создается последовательность (диапазон) чисел. В качестве аргументов функция принимает: начальное значение диапазона (по умолчанию 0), конечное значение (не включая его) и шаг (по умолчанию 1).

xrange()	a=xrange(2,6)	создает псевдосписок — объект, для которого мы можем получить значения "элементов", но не можем изменить их или порядок их следования. Функция xrange() очень похожа на функцию range(). Преимущества использования xrange() вместо range() заметны лишь при работе с огромным количеством элементов.
if /elif/ else	<p>Формат оператора:</p> <pre> if <<выражение1>>: <<набор>> elif <<выражение2>>: <<набор>>] else: <<набор>>] text=['python','java','c++'] if 'python' in text: print 'python' </pre>	<p>условный оператор – выбирает одно из нескольких действий (блоков операторов).</p> <p>Ветвь elif может как совсем отсутствовать, так и присутствовать несколько раз; наличие ветви else необязательно. Ключевое слово elif является короткой формой для записи else if</p>
while/else	<p>В общем виде цикл while выглядит следующим образом:</p> <pre> while "выражение": "набор" else: "набор"] a=0;b=1 while b < 5: print b a, b = b, a+b else: print 10 </pre>	<p>оператор цикла. В цикле while первый набор выполняется до тех пор, пока "выражение" является истинным. В качестве "выражения" можно рассматривать условие, строку, список и т. д. Ветвь else будет выполнена, если цикл завершится при нарушении условия без выполнения оператора break в первом наборе.</p> <p>В результате получим: 1 1 2 3 10</p>
for/else	<p>В общем виде цикл for выглядит следующим образом:</p>	<p>оператор цикла. Инstrukция for в языке Python перебирает элементы произвольной</p>

	<p>for "п-ная" in "посл-ть": "набор" [else: "набор"]</p> <p>tlist=xrange(2,4) for x in tlist: print x else: print 10</p>	<p>последовательности. Ветвь else будет выполнена при завершении последовательности. В результате на консоль будет выведено: 2 3 10</p>																		
try/except/ finally/else	<p>Формат оператора: try: "набор" except: "набор" [else: "набор"] [finally: "набор">>]</p> <p>Язык Python поддерживает много встроенных исключений, например:</p> <table><tr><td>ValueError</td><td>OSError</td></tr><tr><td>StandardError</td><td>EOFError</td></tr><tr><td>BaseException</td><td>SystemExit</td></tr><tr><td>KeyboardInterrupt</td><td>Exception</td></tr><tr><td>StopIteration</td><td>IndexError</td></tr><tr><td>OverflowError</td><td>ImportError</td></tr><tr><td>LookupError</td><td>KeyError</td></tr><tr><td>MemoryError</td><td>IOError</td></tr><tr><td>UnicodeError</td><td>и т.д.</td></tr></table>	ValueError	OSError	StandardError	EOFError	BaseException	SystemExit	KeyboardInterrupt	Exception	StopIteration	IndexError	OverflowError	ImportError	LookupError	KeyError	MemoryError	IOError	UnicodeError	и т.д.	<p>обработка исключений. Исключения возникают, когда в программе при исполнении возникает некоторая исключительная ситуация (например, деление на ноль). Когда возникает исключение, программа прекращает выполнение инструкций в блоке try и отыскивает блок except, соответствующий типу исключения. В случае успеха управление передается первой инструкции в найденном блоке except. После выполнения блока except выполнение программы продолжается с первой инструкции, следующей за блоком try-except. Этот программный код в свою очередь также может быть заключен в блок try-except. Выражение except может обрабатывать как одиночное исключение, так и список исключений. Если исключение не возникает, код в блоке except игнорируется.</p>
ValueError	OSError																			
StandardError	EOFError																			
BaseException	SystemExit																			
KeyboardInterrupt	Exception																			
StopIteration	IndexError																			
OverflowError	ImportError																			
LookupError	KeyError																			
MemoryError	IOError																			
UnicodeError	и т.д.																			

break, continue	<pre>while True: s=raw_input('input: ') print s if s == 'выход': break print('output: ', len(s)) print('Завершение')</pre>	<p>переходы в теле цикла.</p> <p>break – выходим из внутренне-го вложенного цикла. Важно отметить, что если циклы for или while прервать оператором break, соответствующие им блоки else выполняться не будут.</p> <p>continue – используется для указания Python, что необходимо пропустить все оставшиеся команды в текущем блоке цикла и продолжить со следующей итерации цикла.</p>
def, return, yield	<pre>def remainder(a, b): q = a // b r = a % b return r</pre> <p>Если потребуется вернуть из функции несколько значений, можно использовать кортеж</p> <pre>def remainder(a, b): q = a // b r = a % b return (q,r)</pre>	<p>создание функций</p> <p>def – представляет определение функции.</p> <p>return – инструкция выхода из функции и возврата из неё значения.</p>
pass	<pre>def remainder(a, b): pass</pre>	<p>pass – это «пустой оператор».</p> <p>Отметим, что в языке запрещены процедуры с пустым телом.</p>
global	<pre>x = 50 def pros(): global x print 'x равно', x x = 2 print "новое значение x=", x pros() print 'значение x=', x</pre>	<p>указывает на область видимости переменной. Это необходимо, чтобы присвоить некоторое значение переменной, определённой на высшем уровне программы.</p> <p>В результате получим: x равно 50; новое значение x = 2; значение x = 2</p>

import, from	<p>Предположим у нас есть файл <code>module.py</code> со следующим содержанием</p> <pre>def prin(x): print x</pre> <p>Теперь мы можем использовать</p> <pre>import module module.prin('Hello')</pre>	<p>используются для работы с модулями.</p> <p><code>import</code> позволяет получать модуль целиком.</p> <p><code>from</code> позволяет получать определенные имена из модуля</p> <p>Заметим, что Python в момент начала работы загружает в память модуль <code>__builtins__</code></p>
lambda	<p>В общем виде анонимные функции выглядят следующим образом:</p> <p>lambda args : expression</p> <pre>a=lambda x,y: x**y r=a(3,2)</pre>	<p><code>args</code> – это список аргументов, разделенных запятыми;</p> <p><code>expression</code> – выражение, использующее эти аргументы.</p> <p>Внутри инструкции <code>lambda</code> нельзя использовать несколько инструкций или использовать другие инструкции, не являющиеся выражениями, такие как <code>for</code> или <code>while</code>.</p> <p>В результате получим: <code>r = 9</code></p>
zip(s,t)	<pre>zip('ab','12')</pre> <p>в результате получим:</p> <pre>[('a', '1'), ('b', '2')]</pre> <p>или</p> <pre>a='ab';b='cd';c='ef';d='12' zip(a,b,c,d)</pre> <p>в результате получим:</p> <pre>[('a', 'c', 'e', '1'), ('b', 'd', 'f', '2')]</pre>	<p>извлекает все элементы из последовательностей <code>s</code> и <code>t</code> и создает список кортежей <code>[(s[0],t[0]),(s[1],t[1])]</code>.</p> <p>Последовательности должны быть одинаковой длины.</p>

map(f, s, ...)	<pre>def f(x): return x*x a=(1,2); b=(6,7) map(f,a) в результате получим: [1, 4]</pre> <p>или</p> <pre>map(lambda x: x*x, a)</pre> <p>или</p> <pre>map(lambda x,y: x*y, a,b) в результате получим: [6, 14]</pre>	<p>применяет функцию f к каждому элементу последовательности s и возвращает список результатов.</p> <p>Если функции map() передается несколько последовательностей, это предполагает, что количество аргументов, принимаемых функцией f, совпадает с количеством последовательностей, то есть каждый аргумент функции берется из другой последовательности.</p>
----------------	--	--

Индивидуальные задачи

1. Реализовать алгоритм шифрования Цезаря.
2. Реализовать алгоритм шифрования RSA.
3. Реализовать алгоритм шифрования Эль-Гамала.
4. Число из десятичной системы счисления, введенное по запросу с клавиатуры, в случае правильной записи, преобразовать в двоичную систему или в восьмеричную систему и результат вывести на экран.
5. Двоичное число, введенное по запросу с клавиатуры, в случае правильной записи, преобразовать в десятичное число и результат вывести на экран.
6. Число в восьмеричной системе счисления, введенное по запросу с клавиатуры, в случае правильной записи, преобразовать в десятичное число.
7. Подсчитайте частоту букв в текстовом файле.
8. Реализовать алгоритм сжатия информации – алгоритм Хаффмана.

Лабораторная работа № 2

Тема: пакет numpy.

Цель работы: освоение работы с основными функциями пакета numpy.

Краткие теоретические сведения

Основным объектом в numpy являются одномерные и многомерные массивы (в numpy они называются numpy.ndarray). Пакет состоит из нескольких модулей: random (случайные числа), linalg (линейная алгебра) и т. д. Следует обратить внимание, что для подключения пакета обычно используют `import numpy as np`.

- *Создание массивов*

Таблица 15

Операторы и функции для создания массивов

<i>Команда</i>	<i>Пример</i>	<i>Пояснение или результат</i>
np.array(<список компонентов>, [тип ⁵])	a = np.array([1,2,5,3],np.int8) ⁶	создали массив вида [1 2 3] причем dtype=int8
	a=np.array([[1],[a'],'b'],[2]])	создали массив вида [[1] [a', 'b'] [2]] и с dtype=object
np.zeros((размерность), [тип])	a=np.zeros((5))	в результате получим array([0., 0., 0., 0., 0.])
np.ones((размерность), [тип])	a=np.ones((4))	array([1., 1., 1., 1.])
	a=np.ones((3,2),np.int8)	array([[1, 1], [1, 1], [1, 1]], dtype=int8)
np.identity ((размер), [тип])	a=np.identity ((3),np.int8)	array([[1, 0, 0], [0, 1, 0], [0, 0, 1]], dtype=int8)

⁵ Тип(dtype) – это тип данных. Приведем примеры типов данных: целые без знака(uint8, uint16, uint32, uint64), целые со знаком (int8, int16, int32, int64), действительные (float32, float64, float96), комплексные (complex64, complex128, complex196), object, bool, str, unicode. Если не определено явно, то функция самостоятельно старается определить подходящий тип данных для создаваемого массива.

⁶ Функция type(a) возвращает <type 'numpy.ndarray'>

np.diag(<список элементов>)	np.diag((1,2,3))	array([[1,0,0],[0,2,0],[0,0,3]])
np.full((размерность), значение, [тип])	a=np.full((3,2),5)	array([[5., 5.],[5., 5.],[5., 5.]])
np.arange([start, stop,[step,] [тип])	c=np.arange(3)	array([0, 1, 2])
	c=np.arange(3,7)	array([3, 4, 5, 6])
	c=np.arange(3,7,2)	array([3, 5]) возвращает равномерно распределенные значения на заданном интервале (точки будут браться с шагом step).
np.linspace(start,stop,[n, [...]])	c=np.linspace(2.0,3.0,3)	array([2. , 2.5, 3.])
	c=np.linspace(2.0,3.0,3,0)	array([2.,2.33333333, 2.66666667])
	c=np.linspace(2.0,3.0,3, retstep=1)	(array([2. , 2.5, 3.]), 0.5) возвращает равномерно распределенные значения на заданном интервале (важно получить нужный размер массива) ⁷
np.logspace(start,stop,n, [...]) ⁸	c= np.logspace(2.0,3.0,3, base=2)	array([4., 5.65685425, 8.]) возвращает числа, расположенные равномерно на логарифмической шкале
np.tile(массив, размерность)	a = np.array([1,2,3], np.int8) np.tile(a, (2,3))	array([[1, 2, 3, 1, 2, 3, 1, 2, 3],[1, 2, 3, 1, 2, 3, 1, 2, 3]], dtype=int8)

⁷ Перечислим еще несколько аргументов: n – кол-во элементов; endpoint – это флаг, который указывает, входит последняя точка в интервал или нет; retstep – это флаг, который указывает, выводить шаг между точками или нет. По умолчанию размер=50, endpoint= *True*, retstep=*False*

⁸ Перечислим еще несколько аргументов: n –кол-во элементов; endpoint – это флаг, который указывает, входит последняя точка в интервал или нет; base – это основание логарифма. По умолчанию размер=50, endpoint= *True*, base=10.0.

Создавать массивы можно и из случайных элементов. Для этого нужно воспользоваться функциями из модуля `numpy.random` (подключить модуль можно с помощью `import numpy.random as rand`).

Таблица 16

Создание случайных массивов

Команда	Пример	Пояснение или результат
<code>np.random.sample()</code>	<code>a=np.random.sample()</code>	<p>рассмотрим аргументы:</p> <ul style="list-style-type: none"> - без аргументов возвращает просто число в промежутке $[0, 1)$; - с одним целым числом – одномерный массив из k элементов; - с кортежем – массив с размерами, указанными в кортеже⁹.
	<code>a=np.random.sample(k)</code>	
	<code>a=np.random.sample((m,n))</code>	
<code>np.random.randint(low [, high,size])</code>	<code>a= np.random.randint(1,5,7)</code>	<p>в результате получим массив из целых чисел. Числа находятся в диапазоне $[low,high)$. <code>size</code> – размеры массива</p>
<code>np.random.shuffle()</code>	<code>z=[1,2,3,4,5]</code>	случайным образом перемешивает последовательность
	<code>np.random.shuffle(z)</code>	
<code>np.random.choice()</code>	<code>z=[1,2,3,4,5]</code>	в результате получим случайный элемент из последовательности
	<code>np.random.choice(z)</code>	
<code>np.random.rand()</code>	<code>np.random.rand(3,2)</code>	создает массив данной формы и заполненный случайными выборками из равномерного распределения $[0, 1)$. В результате выполнения примера можем получить:

⁹ Все числа будут из промежутка $[0, 1)$

		array([[0.13250592, 0.99177714], [0.35252655, 0.00167288], [0.06701446, 0.78996734]])
np.random.normal(loc=0.0, scale=1.0, size=None)	np.random.normal()	случайная выборка из нормального распределения. Заметим, что <i>loc</i> – среднее значение, <i>scale</i> – стандартное отклонение.
np.random.uniform(low=0.0, high=1.0, size=None)	np.random.uniform()	Равномерное распределение. Возвращает числа в диапазоне [<i>low</i> , <i>high</i>).

Следует обратить внимание, что в массивах `numpy` хранится дополнительная информация (перечислим некоторые элементы этой информации: указатель на данные, тип данных, кортеж, описывающий форму массива).

- *Некоторые методы и атрибуты, связанные с характеристиками и структурой массивов*¹⁰.

Таблица 17

Функции, связанные с характеристиками массивов

Метод или атрибут	Пример	Пояснение или результат
<code>.reshape()</code> изменение формы	<code>c=a.reshape((3,2))</code>	<code>array([[1, 2],[3, 4],[5, 6]])</code> преобразовали массив <code>a</code> в двумерный массив ¹¹

¹⁰ Предположим, что нами были созданы массивы `a=np.array([1,2,3,4,5,6])`, `b=np.array([[1,3],[2,5],[4,6]])` и `mas3d=np.array([[[1,2,3],[4,5,6]], [[7,8,9],[10,11,12]])`.

¹¹ Следует обратить внимание на то, что `c` и `a` ссылаются на один массив. В этом легко убедиться, если выполнить следующие операции: `a[2]=77; print a; print c`. Если же копирование необходимо, то это можно организовать следующим образом: `c=a.reshape((3,2)).copy()`. Теперь массивы `c` и `a` совершенно независимы.

<i>Метод или атрибут</i>	<i>Пример</i>	<i>Пояснение или результат</i>
<code>.ravel()</code> изменение формы	<code>c=b.ravel()</code>	<code>array([1, 3, 2, 5, 4, 6])</code>
<code>.shape</code> получим кортеж размеров массива, его форму. Для матрицы из n строк и m столбцов результат будет (n, m) .	<code>a.shape</code>	<code>(6,)</code>
	<code>b.shape</code>	<code>(3, 2)</code>
<code>.ndim</code> найдем количество измерений массива	<code>b.ndim</code>	<code>2</code>
<code>.size</code> найдем количество элементов в массиве	<code>b.size</code>	<code>6</code>
<code>.dtype</code> получим тип элементов массива	<code>b.dtype</code>	<code>dtype('int32')</code>
<code>.itemsize</code> найдем размер каждого элемента массива в байтах	<code>b.itemsize</code>	<code>4</code>
<code>.transpose()</code>	<code>d=b.transpose()</code>	<code>array([[1, 2, 4],[3, 5, 6]])</code> матрица <code>d</code> - это транспонированная матрица <code>b</code>
<code>.transpose(<кортеж осей>)</code> указывается кортеж номеров осей, описывающий их перестановку	<code>mas3d.transpose((2,1,0))</code>	<code>array([[[1, 7],[4, 10]], [[2, 8],[5, 11]], [[3, 9],[6, 12]])</code>

- *Операции с отдельными компонентами*¹².

Таблица 18

Операторы и функции для массивов

<i>Операция</i>	<i>Пример</i>	<i>Пояснение или результат</i>
арифметическая операция элементов массивов со скалярами	<code>c=a*2</code>	<code>array([2, 4, 6, 8, 10, 12])</code>
	<code>c=a+5</code>	<code>array([6, 7, 8, 9, 10, 11])</code>
	<code>c=a/5</code>	<code>array([0, 0, 0, 0, 1, 1])</code>
	<code>c=a%5</code>	<code>array([1, 2, 3, 4, 0, 1])</code>
	<code>c=a**2</code>	<code>array([1, 4, 9, 16, 25, 36])</code>
арифметическая операция двух массивов одинакового размера (каждый элемент на каждый) ¹³ Любая арифметическая операция над массивами одинакового размера применяется к соответствующим элементам.	<code>c=a*b</code>	<code>array([5, 12, 21, 32, 45, 60])</code>
	<code>c=a**b</code>	<code>array([1, 64, 2187, 65536, 1953125, 60466176])</code>
	<code>c=a+b</code>	<code>array([6, 8, 10, 12, 14, 16])</code>
	<code>c=a/b</code>	<code>array([0, 0, 0, 0, 0, 0])</code>
	<code>c=a%b</code>	<code>array([1, 2, 3, 4, 5, 6])</code>
арифметическая операция двух массивов разного размера (каждый элемент на каждый) ¹⁴	<code>c=at+bt</code>	<code>array([[6,7,8,9], [7,8,9,10]])</code>
	<code>c=at**bt</code>	<code>array([[1, 32, 243, 1024], [1, 64, 729, 4096]])</code>
	<code>c=at*bt</code>	<code>array([[5, 10, 15, 20], [6, 12, 18, 24]])</code>
	<code>c=at/bt</code>	<code>array([[0, 0, 0, 0], [0, 0, 0, 0]])</code>
	<code>c=at%bt</code>	<code>array([[1, 2, 3, 4], [1, 2, 3, 4]])</code>
	<code>c = bt%at</code>	<code>array([[0, 1, 2, 1], [0, 0, 0, 2]])</code>
логические условия также распространяются на отдельные элементы	<code>c=a>3</code>	<code>array([False, False, False, True, True], dtype=bool)</code>
	<code>c=a[a>3]</code>	<code>array([4, 5, 6])</code>

¹² Предположим, что нами были созданы массивы `a=np.array([1,2,3,4,5,6])` и `b=np.array([5,6,7,8,9,10])`.

¹³ Массивы должны быть одинаковой формы.

¹⁴ Предположим, что `at=np.array([1,2,3,4])` и `bt=np.array([5],[6])`.

Следует помнить, что `numpy` предоставляет много функций для обработки массивов. Некоторые из этих функций аналогичны функциям из модуля `math` – только аргументами являются массивы. Также следует помнить, что в результате выполнения функции может возвращаться один или несколько массивов¹⁵.

Таблица 19

Операторы и функции для массивов

<i>Операция</i>	<i>Пример</i>	<i>Пояснение или результат</i>
<code>np.exp()</code> вычисляет экспоненту e для каждого элемента	<code>a=np.array([1,2,3,4,5,6])</code> <code>np.exp(a)</code>	<code>array([2.71828183, 7.3890561 , 20.08553692, 54.59815003, 148.4131591 , 403.42879349])</code>
<code>np.add()</code> складывает соответствующие элементы массивов	<code>np.add(a,b)</code>	<code>array([6, 8, 10, 12, 14, 16])</code>
<code>np.modf()</code> возвращает дробные и целые части массива в виде отдельных массивов.	<code>c=np.array([1.5,2.3,3.0,4.7])</code> <code>a,b=np.modf(c)</code>	в результате получим два вектора: <code>array([0.5, 0.3, 0. , 0.7])</code> и <code>array([1., 2., 3., 4.])</code>
<code>np.fromfunction(f,sh,...)</code>	<code>def f(i,j):</code> <code>return 2*(i+1)+j</code> <code>np.fromfunction(f, (2,2))</code> или <code>np.fromfunction(lambda i,j: 2*(i+1)+j, (2,2))</code>	<code>f</code> – это функция-генератор, которая в качестве аргументов принимает индексы элементов; <code>sh</code> – это кортеж размеров массива; в результате получим <code>array([[2., 3.], [4., 5.]])</code>
<code>np.meshgrid()</code>	<code>x = np.array([1,5,3])</code> <code>y = np.array([-1,1])</code>	в результате получим два массива:

¹⁵ Предположим, что нами были созданы массивы `a=np.array([1.,2,3,4,5,6])`, `b=np.array([5,6,7,8,9,10])` и `mas2d=np.array([[1,2,3],[4,5,6],[7,8,9]])`.

<i>Операция</i>	<i>Пример</i>	<i>Пояснение или результат</i>
создает структурированный массив с координатами x, y на сетке	<code>X,Y=np.meshgrid(x,y)</code>	массив X: <code>array([[1, 5, 3], [1, 5, 3]])</code> , он получается из массива x, количество строк равняется <code>len(y)</code> ; массив Y: <code>array([[-1, -1, -1], [1, 1, 1]])</code> , он получается из массива y, количество столбцов равняется <code>len(x)</code> .
<code>sum(a, axis=None, dtype=None)</code> ¹⁶ сумма элементов. Аргументы: <i>a</i> - массив, <i>axis</i> - ось	<code>np.sum(a)</code>	21.0 – это сумма элементов массива
	<code>np.sum(a, dtype=int)</code>	21
	<code>np.sum(mas2d,axis=0)</code>	<code>array([12, 15, 18])</code> – суммы по оси 0
	<code>np.sum(mas2d,axis=1)</code>	<code>array([6, 15, 24])</code> – суммы по оси 1
	<code>np.sum(mas2d)</code>	45
<code>prod(a, axis=None, dtype=None)</code> возвращает произведение элементов. Аргументы аналогичны тем, которые используются в методе <code>.sum()</code>	<code>np.prod(a)</code>	720.0 – это произведение элементов массива
	<code>np.prod(a, dtype=int)</code>	720
	<code>np.prod(mas2d,axis=0)</code>	<code>array([28, 80, 162])</code> – произведение по оси 0
<code>mean(a, axis=None, dtype=None)</code> вычисляет среднее арифметическое	<code>np.mean(a)</code>	3.5
	<code>np.mean(a, dtype=int)</code>	3
	<code>np.mean(mas2d,axis=1)</code>	<code>array([2., 5., 8.])</code> – значения вычислены по оси 1

¹⁶ *a* – исходный массив; *axis* – ось или кортеж осей, по заданной оси выполняется суммирование; *dtype* – тип возвращаемого числа или массива.

<i>Операция</i>	<i>Пример</i>	<i>Пояснение или результат</i>
std(a, axis=None, dtype=None, ddof=0) вычисляет стандартное отклонение, в основном аргументы аналогичны .sum(), ddof – для изменении степеней свободы.	np.std(a)	1.707825127659933
	np.std(mas2d,axis=1)	1
	np.std(a,ddof=1)	1.8708286933869707
	np.std(mas2d,axis=1)	array([0.81649658, 0.81649658, 0.81649658]) вычисляем значение по оси 1

- *Индексирование и вырезание*¹⁷

Таблица 20

Операторы и функции для массивов (индексирование)

<i>Адресация</i>	<i>Пояснение или результат</i>
mas2d[0]	array([1, 2, 3])
mas2d[0,1] или mas2d[0][1]	2
mas2d[:2]	array([[1, 2, 3],[4, 5, 6]]) вырезание производится вдоль оси 0.
mas2d[1:]	array([[4, 5, 6], [7, 8, 9]]) вырезание производится вдоль оси 0.
mas2d[:,2]	array([[1, 2],[4, 5],[7, 8]]) вырезание производится вдоль оси 1 ¹⁸ .
mas2d[:,1:]	array([[2, 3],[5, 6],[8, 9]]) вырезание производится вдоль оси 1.
mas3d[1]	array([7, 8, 9, [10, 11, 12]]) берем 1 элемент по 0
mas3d[1,1]	array([10, 11, 12])
mas3d[:,1]	array([[4, 5, 6], [10, 11, 12]]) берем все элементы по оси 0 и первый элемент по оси 1
mas3d[1,1,:]	array([10, 11, 12])
mas3d[1,1,0:2]	array([10, 11])

¹⁷ Пусть mas2d=np.array([[1,2,3],[4,5,6],[7,8,9]]) и mas3d=np.array([[[1,2,3], [4,5,6]], [[7,8,9],[10,11,12]]])

¹⁸ двоеточие без указания числа означает, что нужно взять всю ось целиком

- Упорядочивание элементов¹⁹

Таблица 21

**Операторы и функции для массивов
(упорядочивание элементов)**

Операция	Пример	Пояснение или результат
.max([ось]) находим максимальный элемент	mas2d.max() mas2d.max(0) mas2d.max(1)	9 array([7, 9, 6]) array([3, 9, 8])
.sort([<ось>=-1]) сортировка	mas2d.sort(0)	array([[1, 2, 3], [4, 8, 5], [7, 9, 6]])

- Матрицы²⁰

Таблица 22

Операторы и функции для матриц

Операция	Пример	Пояснение или результат
matrix(данные) создание матрицы	a1 = np.matrix(a) b1 = np.matrix(b) mas = np.matrix(mas2d)	matrix([[1, 2, 3]]) matrix([[5, 6, 7]]) matrix([[1, 2, 3], [4, 9, 6], [7, 8, 5]])
арифметическая операция двух матриц	a1*b1.transpose() a1*mas	matrix([[38]]) matrix([[30, 44, 30]])
.I обратная матрица	mas.I	matrix([[0.05769231, -0.26923077, 0.28846154], [-0.42307692, 0.30769231, -0.11538462], [0.59615385, -0.11538462, -0.01923077]])
.T транспонированная матрица	mas.T	matrix([[1, 4, 7], [2, 9, 8], [3, 6, 5]])

¹⁹ Предположим, что нами был создан массив mas2d=np.array([[1,2,3],[4,9,6],[7,8,5]])

²⁰ Пусть a=np.array([1,2,3]), b=np.array([5,6,7]) и mas2d=np.array([[1,2,3],[4,9,6],[7,8,5]])

Индивидуальные задачи

1. Создайте вектор со значениями от 5 до 10 и разверните вектор (первый становится последним).
2. Создайте массив 3×3 со случайными значениями, найдите минимум, максимум и среднее значение.
3. Создайте две матрицы 5×3 и 3×2 и перемножьте их.
4. Создайте массив 3×3 со случайными значениями от 1 до 6 и скажите, сколько элементов матрицы меньше 3.
5. Напечатайте все значения в массиве.
6. Найдите диагональные элементы произведения двух матриц.
7. Создайте массив 3×3 и выберите координату. Напишите функцию, выделяющую часть массива фиксированного размера с центром в данной координате.
8. Найдите наиболее часто встречаемое значение в массиве.

Лабораторная работа № 3

Тема: пакет Matplotlib.

Цель работы: освоение работы с основными функциями пакета matplotlib.

Краткие теоретические сведения

Пакет предназначен для визуализации числовой информации с помощью 2D и 3D графиков и диаграмм. Пакет состоит из множества модулей.

- *Модуль matplotlib.pyplot (двумерные графики)*

Чтобы импортировать модуль matplotlib.pyplot обычно используют `import matplotlib.pyplot as plt`.

Объектом самого высокого уровня при работе с matplotlib является рисунок (Figure). На нем располагаются одна или несколько областей рисования (Axes) и элементы оформления рисунка. Назначение Axes состоит в том, что на него наносится графика.

Рассмотрим некоторые методы экземпляра класса matplotlib.figure.Figure.

Таблица 23

Операторы и функции для модуля matplotlib.pyplot

<i>Функция</i>	<i>Пояснение или результат</i>
<code>.plot(x,y,[...])</code>	создает график. При создании графика можно указать тип линии, цвет линии, маркер точек и т. д. Напомним возможные аргументы: <ul style="list-style-type: none">• <code>c=<цвет линии></code>, возможна одна из букв <code>rgbymcwk</code>• <code>ls=<тип линии></code>, возможны: <code>-- - -</code>.• <code>lw=<толщина линии></code>• <code>marker=<тип маркера></code>, возможны: <code>o^v<>s+xDd1234hHp, _</code>• <code>mec=<цвет границы маркера></code>• <code>mfc=<цвет маркера></code>• <code>mew=<толщина границы маркера></code>• <code>ms=<размер маркера></code>

<code>.hist(values[, bin=10,...])</code>	создает гистограмму для входящих данных. Аргумент <code>bin</code> – количество отрезков, указывающих, на сколько делятся данные
<code>.polar(angle, radius,...])</code>	создает график в полярной системе координат
<code>.savefig(file_path[, dpi])</code>	сохраняет график в файл
<code>.show()</code>	отображает окно с графиком
<code>.xlim([new_xlim])</code>	возвращает или устанавливает предельные значения по оси
<code>.subplot(mnp[, ...])</code>	данная команда выполняется перед обращением к функциям построения графиков для одновременной выдачи нескольких графиков в различных частях графического окна. Аргументы <code>mnp</code> – 3 цифры, которые показывают, как будет происходить разбивка графического окна на несколько подокон. <code>m</code> указывает, на сколько частей разбивается окно по горизонтали, <code>n</code> указывает, на сколько частей разбивается окно по вертикали, <code>p</code> – номер подокна, куда будет выводиться очередной график
<code>.title(label)</code>	команда создает общие заголовки к графику и подграфикам
<code>.text(x,y,текст,...])</code>	выводит в области <code>Axes</code> текст в точке <code>(x,y)</code>
<code>.legend(...])</code>	команда отображает легенду в графической области активного окна. Напомним возможные аргументы: <code>borderaxespad</code> – величина зазора между осями и легендой; <code>legend_names</code> –название легенды; <code>loc</code> – местоположение вывода данных легенды (0 – лучший вариант, 1 – вверху справа, 2 – вверху слева, 3 – внизу слева, 4 – внизу справа, 5 – справа, 6 – посередине слева, 7 – посередине справа, 8 –посередине внизу, 9 – посередине вверху, 10 – посередине)

<code>.xlabel(label), .ylabel(label)</code>	устанавливает подпись для оси x и оси y
<code>.grid ([...])</code>	включает отображение сетки по значениям осей
<code>.axis([new_axis])</code>	команда устанавливает предельные координаты по осям: $[xmin, xmax, ymin, ymax]$
<code>.figure([...])</code>	создает экземпляр класса <code>matplotlib.figure.Figure</code> . Перечислим некоторые атрибуты этого класса: <code>figsize</code> – кортеж размера фигуры; <code>dpi</code> – количество точек на дюйм; <code>facecolor</code> – цвет фона фигуры; <code>edgecolor</code> – цвет границ фигуры; <code>linewidth</code> – ширина линии фигуры. Отметим, что экземпляр может автоматически создаваться при первом выполнении какой-либо графической функции
<code>.imshow (z,[...])</code>	выводит изображение (пиксельные картинки). Картинка задаётся массивом z : $z[i,j]$ – это цвет пикселя i,j , массив из 3 элементов (rgb, числа от 0 до 1). Перечислим несколько аргументов: <code>cm</code> – цветовая гамма; <code>norm</code> – нормирование данных при выводе; <code>aspect</code> – соотношение сторон; <code>interpolation</code> – метод интерполирования изображения
<code>.axes([left, bottom, width, height])</code>	используется, если вы хотите расположить оси (подокна) вручную, например не на прямоугольной сетке, а как-нибудь ёлочкой. Все значения аргументов изменяются от 0 до 1.

Рассмотрим несколько примеров.

Пример. Построим функции $y = \cos(10x)$ и $y = \frac{x}{2}$ на промежутке $x \in [-1,1]$.

код

результат выполнения кода смотри на рис. 1

```
#-*- coding: cp1251 -*-  
import numpy as np #импортируем модуль  
import matplotlib.pyplot as plt #импортируем модуль  
X = np.linspace(-1,1,50 )  
#возвращает массив с равномерно распределенные значениями  
#на интервале  
C,S = np.cos(10*X), 0.5*X #создаем значение функций на интер-  
#вале  
plt.figure(figsize=(6,6)) #зададим размер фигуры  
plt.title('Matplotlib_1') #заголовок графика  
plt.xlabel('x') #устанавливают подпись для оси x  
plt.ylabel('f(x)') #устанавливают подпись для оси y  
plt.text(0.25,0.75,'Text') #выводим текст  
plt.grid(c='#0000ff',ls='--',lw=1.5) #отображаем сетку  
plt.plot(X,C,c='r',ls='-',ms=10,marker='o') #создаем график  
plt.plot(X,S,c='g',ls='--') #создаем график  
plt.axis([-1,1,-2,2])  
plt.show() #отображает окно с графиком
```

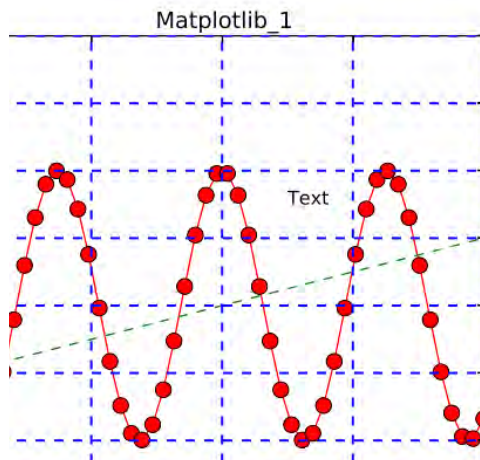


Рис. 1. Результат выполнения кода

Пример. Построим различные графики, каждый на своем подокне. Рассматриваются функции:

- $y = \sin(x)$ на промежутке $x \in [-10; 10]$.
- в полярной системе координат $y = \sin(\varphi)$ и $y = \cos(\varphi)$, где $\varphi \in [0, 2\pi]$.
- гистограмма из случайных значений
- цветную картинку

```

                                код
результат выполнения кода смотри на рис. 2
# -*- coding: cp1251 -*-
import numpy as np #импортируем модуль
import matplotlib.pyplot as plt
x=np.arange(-10,10.01,0.01)# создаем массив
#subplot 1
plt.subplot(221)
#будем выводить сразу четыре графика, создаем первый гра-
    фик
plt.plot(x,np.sin(x)) # строим первый график
plt.title('plot') #делаем заголовок для первого графика
#subplot 2
plt.subplot(222,polar="True")
#создаем второй график в полярной системе координат
plt.title('polar')
phi=np.linspace(0,2*np.pi,100) # создаем одномерный массив
plt.polar(phi,np.sin(phi)**2) #создаем график
plt.polar(phi,np.cos(phi)**2) #создаем график
#subplot 3
plt.subplot(223)
plt.title('Histogram')
a= np.random.uniform(size=100)# создаем массив
plt.hist(a,bins=20) # создаем 20 столбцов
#subplot 4
plt.subplot(224)
n = 256
u=np.linspace(0,1,n) # создаем одномерный массив
```

```
plt.title('image')
x,y=np.meshgrid(u,u); z=np.zeros((n,n,3))
z[:, :,0]=x
z[:, :,2]=y
plt.imshow(z)# создаем график
plt.show()
```

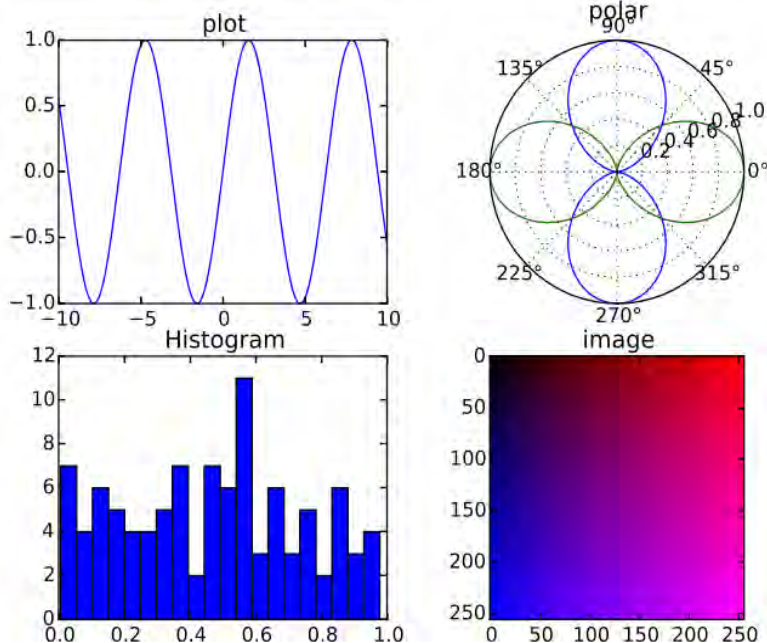


Рис. 2. Результат выполнения кода

- Модуль `mpl_toolkits.mplot3d` (трехмерные графики)

Для построения трехмерных графиков будем импортировать несколько пакетов:

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d.axes3d import Axes3D
import numpy as np
```

Операторы и функции для модуля *mpl_toolkits.mplot3d*

Функция	Пояснение или результат
<code>.Axes3D()</code>	создает трехмерные оси. Аргументом метода должен быть экземпляр класса Figure. Объект Axes3D можно также создать с помощью метода <code>add_subplot(111, projection = '3d')</code> экземпляра класса Figure
<code>.plot(x,y,z,[...])</code>	создает обычную кривую. Применяя метод экземпляра Axes3D, получим трехмерную кривую. Аргументы аналогичны <code>matplotlib.pyplot.plot</code> . Параметры x, y, z обычно задаются параметрически $x = x(t)$, $y = y(t)$ и $z = z(t)$
<code>.plot_surface(X,Y,Z,[...])</code>	строит поверхность, где X, Y, Z – двумерные массивы, содержащие x, y и z координаты узлов многогранной поверхности. Перечислим некоторые аргументы: <code>rstride=1</code> и <code>cstride=1</code> устанавливают шаг, используемый для выборки входных данных. Студентам следует помнить, что координаты узлов можно задавать параметрические: $x = x(u,v)$, $y = y(u,v)$, $z = z(u,v)$. Если мы хотим построить координаты в явном виде $z = z(x,y)$, то удобно создать массивы x, y и воспользоваться функцией <code>numpy.meshgrid</code>
<code>.plot_wireframe(X,Y,Z,[...])</code>	строит каркасную поверхность, где X, Y, Z – двумерные массивы, содержащие x, y и z координаты узлов многогранной поверхности. Аргументы аналогичны <code>plot_surface</code>
<code>.scatter(x, y, z,[...])</code>	строит график рассеяния. Функция рисует множество точек, абсциссы которых передаются в векторе x , ординаты – в векторе y , аппликаты – в векторе z .

Пример. Построим различные 3D графики, каждый на своем подокне. Рассматриваются функции: обычная кривая; кар-
касная поверхность; обычная поверхность; график рассеивания

```

.....
..... код
.....
..... результат выполнения кода смотри на рис. 3
.....
# -*- coding: cp1251 -*-
import numpy as np #импортируем модуль
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d.axes3d import Axes3D
fig = plt.figure()
#subplot 1
ax = fig.add_subplot(221, projection='3d') #создаем трехмерные
      оси
t=np.linspace(0,4*np.pi,100)      #задаем изменение параметра
x=np.sin(t); y=np.cos(t);z=t/(4*np.pi); #создаем x,y,z
ax.plot(x,y,z,c='r',ls='-',ms=10,marker='o') #создаем график
#subplot 2
ax = fig.add_subplot(222, projection='3d')
u=np.arange(-5,5,0.25) #задаем изменение параметра
x,y=np.meshgrid(u,u) #создаем x,y
z=np.sin(np.sqrt(x**2+y**2)) #вычисляем z
ax.plot_wireframe(x,y,z,rstride=1,cstride=1,color='green')
#строим поверхность
#subplot 3
ax = fig.add_subplot(223, projection='3d')
u = np.linspace(0, 2 * np.pi, 100)
v = np.linspace(0, np.pi, 100)
x = 10 * np.outer(np.cos(u), np.sin(v)) #создаем x
y = 10 * np.outer(np.sin(u), np.sin(v)) #создаем y
z = 10 * np.outer(np.ones(np.size(u)), np.cos(v)) #создаем z
ax.plot_surface(x, y, z, color='b') #строим поверхность
#subplot 4
ax = fig.add_subplot(224, projection='3d')
x = np.random.rand(500) * 20.0 - 10.0
.....

```

```

#np.random.rand(500) создает массив из 500 элементов
y = np.random.rand(len(x)) * 20.0 - 10.0
z = np.sin(x * 0.3) * np.cos(y * 0.75)
ax.scatter(x, y, z)
plt.show()

```

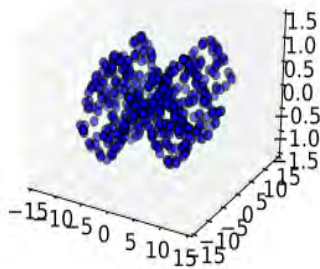
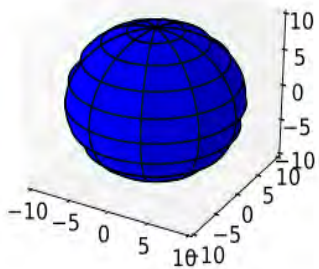
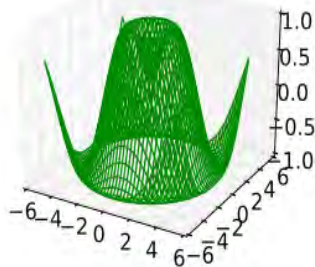
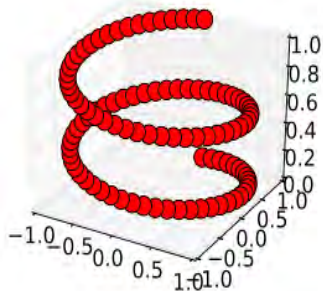


Рис. 3. Результат выполнения кода

Индивидуальные задачи

1. Посчитайте частоту букв в текстовом файле и постройте гистограмму.

2. Постройте график функции $y = \sin(5x + 1) + 2$ и $y = \frac{1}{x + 5}$ на промежутке $x \in [-1, 1]$.

3. Постройте график в полярной системе координат $y = 2\sin(\varphi)$ и $y = 3\cos(2\varphi)$, где $\varphi \in [-\frac{\pi}{2}, \frac{\pi}{2}]$.

4. Постройте 3D кривую $x=t/5$; $y=t^2$; $z=t+5$; $t \in [-\pi, \pi]$.

5. Постройте 3D кривую $x=t^2$; $y=t^2$; $z=t/5$ $t \in [-\pi, \pi]$.

6. Постройте 3D поверхность $z = \cos\sqrt{x^2 + y^2}$, где $x, y \in [-5, 5]$.

7. Постройте каркасную 3D поверхность $z = \cos\sqrt{x^2 + y^2}$, где $x, y \in [-3, 3]$.

7. Постройте 3D поверхность $z = x^2 + y^2$, где $x, y \in [-1, 1]$.

Лабораторная работа № 4

Тема: пакет SymPy.

Цель работы: освоение работы с основными функциями пакета SymPy.

Краткие теоретические сведения

SymPy представляет собой открытую библиотеку символьных вычислений на языке Python. Подключение пакета обычно выполняется с помощью инструкции: `from sympy import *`

Таблица 25

Создание символьных переменных

<i>Функция</i>	<i>Пример</i>	<i>Пояснение или результат</i>
<code>symbols()</code> <code>var()</code>	<code>x=symbols('x')</code> <code>var('Pyt')</code>	создает символьную переменную <code>x</code> или переменную с именем <code>Pyt</code>
<code>S(expr[,...])</code>	<code>c=S(1+3.5);</code> <code>print (type(c),c)</code> Для значения также можно указать точность <code>b=c.n(3); print (b)</code>	преобразует <code>expr</code> в тип, который можно использовать в выражениях SymPy. В результате получим: (<code><class 'sympy.core.numbers.Float'></code> , 4.500000000000000) 4.50
<code>Function()</code>	<code>b=Function('b')</code>	объявляет символьную функцию
<code>prod(выраж)</code>	<code>c= prod([(x-2*i) for i in range(1,6,2)]);</code> <code>print c;</code> <code>type (c)</code>	вычисляет произведение своих элементов. В результате получим $(x - 10)^*(x - 6)^*(x - 2)$ <code><class 'sympy.core.mul.Mul'></code>

Функция	Пример	Пояснение или результат
класс Rational	<code>c = Rational(1,2)</code> <code>b = Rational(1,4)</code> <code>print(c+b)</code>	класс Rational отвечает за работу с дробями. Любую пару целых чисел он определяет как числитель и знаменатель. Далее над дробями можно производить типичные действия. В результате выполнения примера получим: $3/4$
<code>.evalf(n=15,[...])</code> n - указывает точность	<code>c = Rational(1,2)</code> <code>b = Rational(1,4);</code> <code>print (c+b).evalf()</code>	возвращает десятичное представление любого символьного объекта. В результате выполнения примера получим: 0.7500000000000000
<code>init_printing([...])</code>	<code>init_printing()</code> <code>c,b=symbols('c b')</code> <code>f=c**2+2*b</code> <code>print f</code>	после этой инструкции будем получать красивые изображения формул. После выполнения примера получим: $c^2 + 2 * b$
символ ∞ ставится с помощью oo	<code>print (oo+4)</code>	доступен символ бесконечности. В результате выполнения примера получим: ∞
<code>Matrix()</code>	<code>init_printing()</code> <code>a,b,c,d=symbols('a b c d')</code> <code>M=Matrix</code> <code>([[a,b],[c,d]])</code>	с помощью этой функции создаются символьные матрицы. Доступ к элементам матрицы можно выполнить следующей записью <code>M[0,0]=a</code>

В модуле Sympy имеются особые константы, такие как E и π , которые ведут себя как переменные (то есть выражение $1 + \pi$ не преобразуется сразу в число), $E.evalf() = 2.71828182845905$ и $\pi.evalf() = 3.14159265358979$

Таблица 26

Операторы и функции для модуля sympy для матриц²¹

<i>Метод, атрибут</i>	<i>Пример</i>	<i>Пояснение или результат</i>
M.T	B=M.T	выполняет транспонирование матрицы. B= Matrix([[1,2],[4,3]])
M.det()	B= M.det()	вычисляет определитель. Результат – 5
M.inv()	B= M.inv()	вычисляет обратную матрицу B=Matrix([[–3/5, 4/5], [2/5, –1/5]])
M.eigenvals()	B= M.eigenvals()	функция выполняет вычисление собственных чисел. В результате получим словарь dict: значение кратность. Результат {–1: 1, 5: 1}
M.eigenvects()	B= M.eigenvects()	функция выполняет нахождение собственных векторов. Результат выполнения примера: [(–1, 1, [Matrix([[–2],[1]])]), (5, 1, [Matrix([[1],[1]])])]

- *Алгебраические выражения и вычисления*

Таблица 27

Алгебраические выражения и вычисления

<i>Метод, атрибут</i>	<i>Пример</i>	<i>Пояснение или результат</i>
factor(f [...])	x,y = symbols('x y') factor((x**2 - y**2)*x)	раскладывает алгебраическое выражение на множители. После выполнения примера получим $x*(x - y)*(x + y)$

²¹ Предположим, что есть символьная матрица $M=Matrix([[1,4],[2,3]])$

<i>Метод, атрибут</i>	<i>Пример</i>	<i>Пояснение или результат</i>
expand(expr [...])	x,y,z = symbols('x y z') b=(x+y-z)**2 b= expand(b) print b	раскрывает скобки алгебраического выражения. После выполнения примера получим: $x^{**2} + 2*x*y - 2*x*z + y^{**2} - 2*y*z + z^{**2}$
collect(expr,var)	x,y,z = symbols('x y z') collect(x**2 + 2*x*y - 2*x*z + y**2 - 2*y*z + z**2,x) или collect(x**2 + 2*x*y - 2*x*z + y**2 - 2*y*z + z**2,y)	функция собирает коэффициенты при одинаковых степенях переменной var. Результат $x^{**2} + x*(2*y - 2*z) + y^{**2} - 2*y*z + z^{**2}$ или $x^{**2} - 2*x*z + y^{**2} + y*(2*x - 2*z) + z^{**2}$
apart(expr [...])	init_printing() x=symbols('x') b=S(1)/((x+1)*(x+2)) apart(b)	функция выполняет разложение дробей. Было: $\frac{1}{(x + 2) * (x + 1)}$ В результате получим: $- \frac{1}{x + 2} + \frac{1}{x + 1}$
together(expr)	b=S(1)/(x+1) c=S(1)/(x+2) together(b-c)	функция приводит дроби к общему знаменателю. Было: $\frac{1}{(x + 2)} \text{ и } \frac{1}{(x + 1)}$ В результате получим: $\frac{1}{(x + 2) * (x + 1)}$
simplify(expr)	simplify ((2*x+ 3)* (10*x-5)/(2*x-1))	функция пытается упростить алгебраическое выражение. Результат 10*x + 15

<i>Метод, атрибут</i>	<i>Пример</i>	<i>Пояснение или результат</i>
метод .subs(old, new)	<pre>c,b=symbols('c b') f=c**2+2*b print (f.subs([(c,2),(b,3)]))</pre>	выполняет замену одних переменных на другие переменные, числа или выражения. Аргумент может быть последовательностью, которая должна содержать пары (old,new). В результате получим: 10

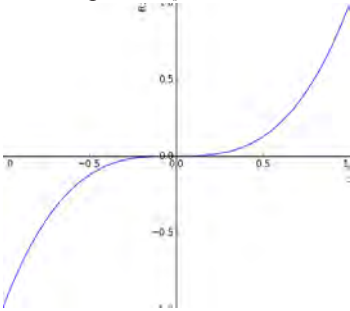
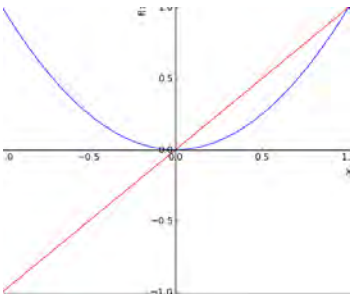
• *Графические возможности пакета(модуль модуль sympy.plotting).*

Графические функции SymPy используют библиотеку matplotlib, поэтому многие аргументы функций совпадают.

Таблица 28

Графические возможности пакета

<i>Метод, атрибут</i>	<i>Пример</i>	<i>Пояснение или результат</i>
plot(f(x),[range,...])	<pre>x=symbols('x') p=plot(x**3, (x,-1,1), ylim=(-1,1))</pre> Результат можно посмотреть на рис. 4	строится график функции $f(x)$, range – диапазон изменения аргумента $f(x)$. По умолчанию range=(-10,10). Перечислим некоторые аргументы: • line_color='цвет' – задает цвет линии • ylim=(a,b) предельные значения по оси y • title= 'Title' – задает заголовок • show = False – график будет строиться, но не будет отображаться. Отобразить график можно методом show(),

Метод, атрибут	Пример	Пояснение или результат
		то есть <code>p.show()</code>  <p>Рис. 4</p>
<code>plot((expr1, range), (expr2, range)[, ...])</code>	<pre>p=plot((x,(x,-1,1)), (x**2,(x,-1,1))) или p=plot(x,x**2, (x,-1,1), ylim=(-1,1), show=False) p[0].line_color='red'; p[1].line_color='blue' p.show()</pre> <p>Результат можно по- смотреть на рис. 5</p>	<p>функция строит несколько графиков с разными диапа- зонами</p>  <p>Рис. 5.</p>

- *Полезные функции*

Таблица 29

Вычислительные функции пакета

Решение уравнений		
<code>Eq(expr1, expr2)</code>	<pre>init_printing() x=symbols('x') Eq(x**2,x+4)</pre>	<p>функция используется для создания уравне- ния. Результат: $x^2 = x + 4$</p>

<code>solve(equation, v=None[,...])</code>	<p>Решим $x^2 - 2x + 10 = 4$</p> <code>solve(Eq(x**2-2*x+10,4),x);</code> или <code>solve(x**2-2*x+6,x)</code> <p>Решим систему</p> $\begin{cases} x - 3y = 0 \\ 3x + 2y - 7 = 0 \end{cases}$ <code>solve([x-3*y,3*x+2*y-7], x,y)</code>	<p>функция предназначена для символьного решения алгебраических уравнений, где equation может быть записано в форме Eq() или быть выражением, которое полагается равным нулю.</p> <p>v – имя искомой переменной</p> <p>Результаты: $1 - I * \sqrt{5}$ и $1 - I * \sqrt{5}$; $1 - I * \sqrt{5}$ и $1 - I * \sqrt{5}$;</p> <p>Результат решения системы:</p> $x : \frac{21}{11}, y : \frac{7}{11}.$
<code>dsolve(expr,[...])</code>	<p>решим $y'' + 9y = 0$</p> <code>x=symbols('x')</code> <code>y = Function('y')</code> <code>dsolve(Derivative(y(x),x,x) + 9*y(x), y(x))</code>	<p>функция пытается решать обыкновенные дифференциальные уравнения. Результат выполнения примера:</p> $y(x) = C1 * \sin(3*x) + C2 * \cos(3*x)$
<i>Вычисление пределов</i>		
<code>limit(f, v, p, dir= '+')</code>	<p>вычислим пределы</p> $\lim_{x \rightarrow 0} \frac{\sin x}{x}; \quad \lim_{x \rightarrow \infty} \frac{1}{x} \text{ и } \lim_{x \rightarrow -0} \frac{1}{x^3}$ <code>limit(sin(x)/x,x,0);</code> <code>limit(1/x,x,oo)</code> <code>limit(1/(x**3),x,0,'-')</code>	<p>вычисляем предел функции. Аргументы: f – функция, v – имя переменной, p – точка, dir используется для вычисления одностороннего предела. Результаты выполнения примеров 1 ; 0 и $-\infty$</p>

<i>Дифференцирование</i>		
<code>diff(f,v[,...])</code>	<p>дано $f_1(x) = \sin(x)$ и $g(x,y) = x^2y^2$ вычислим производные</p> $f_1'(x); f_1''(x); \frac{\partial g}{\partial x}; \frac{\partial^2 g}{\partial x \partial y}$ <p><code>x,y = symbol('x y')</code> <code>diff(sin(x),x); diff(sin(x),x,x);</code> <code>diff(x**2*y**2,x);</code> <code>diff(x**2*y**2,x,y)</code></p>	<p>функция выполняет символьное дифференцирование. Аргументы: <code>f</code> – функция, <code>v</code> – имя переменной, по которой вычисляется производная. Результаты выполнения примеров: <code>cos(x)</code>; <code>–sin(x)</code>; <code>2xy^2</code>; <code>4xy</code>.</p> <p>Следует помнить, что у символьных выражений есть метод <code>diff()</code>. Записи <code>diff(x**2*y**2,x,x,y)</code> и <code>g.diff(x,2,y)</code> эквивалентны. Напомним запись <code>x,y=symbols('x y');</code> <code>g=Function('g');</code> <code>g=x**2*y**2;</code> <code>g.diff(x,2,y)</code></p>
<code>Derivative(f,v[,...])</code>	<p><code>f= x**2*y**2</code> <code>f1=Derivative(f,x,y)</code> <code>f1.doit()</code></p>	<p>невывчисляемый эквивалент оператора <code>diff()</code>. Чтобы вычислить выражение, созданное этим оператором, нужно использовать метод <code>doit()</code>.</p>
<i>Разложение в ряд и сумма ряда</i>		
<code>series(expr [, x=None, x0=0, n=6., dir='+'])</code>	<p>разложим в ряд следующие функции: $\sin(x)$; и $\cos(x)$; $\sin^2(x) + \cos^2(x)$; $e^y \cdot \sin(x)$; в точке ноль.</p> <p><code>series(sin(x));</code> <code>series(cos(x));</code></p>	<p>функция используется для разложения символьного выражения <code>expr</code> в степенной ряд. Аналогичную операцию делает метод <code>.series()</code>. Разложение выполняется в окрестности точки <code>x=x0</code></p>

	<pre>series(sin(x)**2+cos(x)**2,n=8); series(E**y*sin(x),x);</pre> <p>Разложим в ряд: $\ln(x+1)$ в точке нуль и без остаточного члена $f = \ln(x+1)$ <code>f.series(n = 4).removeO()</code></p>	<p>до слагаемых $(x - x_0)^n$. Чтобы не выводить остаточный член $O(\dots)$, можно использовать метод <code>exrg</code>. <code>series(...).removeO()</code>.</p> <p>Приведем результаты выполнения примеров:</p> $x - \frac{x^3}{6} + \frac{x^5}{120} + O(x^6);$ $1 - \frac{x^2}{2} + \frac{x^4}{24} + O(x^6);$ $1 - O(x^8);$ $x * e^y - \frac{x^3 * e^y}{6} + \frac{x^5 * e^y}{120} + O(x^6);$ $\frac{x^3}{3} - \frac{x^2}{2} + x$
<code>summation(f, (i,a,b))</code>	<p>вычислим суммы</p> $\sum_{i=1}^{15} (2i-1) \text{ и } \sum_{i=2}^{\infty} \frac{1}{i^2 - 1}$ <p><code>summation(2*i-1, (i,1,15))</code> <code>summation(1/(i**2-1), (i,2,oo))</code></p>	<p>функция вычисляет сумму ряда $\sum_a^b f(i)$.</p> <p>Если найти результат суммирования в символьной форме не удастся, то будет возвращена невычисляемая форма этого оператора – <code>Sum()</code>. Результаты выполнения примеров: 225; 3/4</p>

<i>Интегрирование</i>		
<code>integrate(f,v [,...]).</code>	<p>вычислим следующие интегралы</p> <ul style="list-style-type: none"> $\int \cos(x) dx;$ $\int \frac{1}{x} dx$ $\int x \sin(x) dx$ $\int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} \cos(x) dx$ $\int_0^{\infty} e^{-x} dx$ $\iint 6x^2 y \, dx dy$ $\int_0^1 \int_0^1 6x^2 y \, dx dy$ <p> <code>integrate(cos(x),x);</code> <code>integrate(1/x,x);</code> <code>integrate(x*sin(x),x);</code> <code>integrate(cos(x),(x,-pi/2,pi/2));</code> <code>integrate(exp(-x),(x,0,oo));</code> <code>integrate(6*x**2 * y , x , y);</code> <code>integrate(6*x**2*y,(x,0,1), (y,0,1))</code> </p>	<p>функция предназначена для символьного интегрирования, поддерживает вычисление определенных и неопределенных интегралов. Аргументы: <code>f</code> – функция, которая будет интегрироваться, <code>v</code> – переменная интегрирования или кортеж вида (<code>v</code>, <code>ниж._предел</code>, <code>верх._предел</code>) Результаты выполнения примеров:</p> <ul style="list-style-type: none"> $\sin(x);$ $\log(x);$ $-x \cos(x) + \sin(x);$ $2;$ $1;$ $x^3 y^2;$ 1 <p>Если первообразную не удалось найти, то возвращается объект <code>Integral</code>.</p>
<code>Integral(f,v [,...])</code>	<p> <code>f= 6*x**2 * y</code> <code>f1=Integral(f, (x ,0,1),(y,0,1))</code> <code>f1.doit()</code> </p>	<p>невывчисляемый эквивалент оператора <code>integrate()</code>. Чтобы вычислить выражение, созданное этим оператором, нужно использовать метод <code>doit()</code>.</p>
<code>pprint()</code>		<p>пытается отобразить результат в хорошо читаемом виде.</p>

Рассмотрим применение пакета `sympy` в исследовании математических моделей²²

Пример *Рост колонии микроорганизмов.*

Дано: за время Δt прирост численности колонии равен: $\Delta t = R - S$; где R – число родившихся и S – число умерших за время Δt особей. Число родившихся и умерших особей можно задать формулами:

$$R(\Delta t, x) = R(x)\Delta t \text{ и } S(\Delta t, x) = S(x)\Delta t.$$

В дискретной форме: $\Delta x = (R(x) - S(x)) \Delta t$.

Разделив полученное на Δt , перейдем к пределу при $\Delta t \rightarrow 0$.

В результате получим $\frac{dx}{dt} = R(x) - S(x)$.

Будем считать, что рождаемость и смертность пропорциональны численности: $\frac{dx}{dt} = \alpha x - \beta x$ или $\frac{dx}{dt} = rx$. Предположим, что $x(0) = 5$, а $r = \pm 1$. Найти $x(t)$ и построить графики решений.

Рассмотрим символьное решение указанного дифференциального уравнения:

```

..... код
:
:      результат выполнения кода смотри на рис. 6
: #-*- coding: cp1251 -*-
:from sympy import* #подключили модуль
:x = Function('x') #создали логическую функцию
:C1,t,r,p = symbols('C1,t r,p') #создали логические переменные
:g = Eq(Derivative(x(t),t), r*x(t)) #создали уравнение
:d=dsolve(g, x(t)) #решили дифференциальное уравнение
:pprint(d) #выводим решение
:eq = d.rhs.subs(t,0)
: #значение функции в момент времени t=0
: #метод rhs говорит о том, что работаем
: #только с правой частью
:eq1=d.rhs.subs([(C1,5),(r,1)]) #устанавливаем C1=5,r=1
:eq2=d.rhs.subs([(C1,5),(r,-1)]) #устанавливаем C1=5,r=-1
:p=plot(eq1,eq2,(t,0,1),ylim=(0,10),show=False,legend=True)

```

²² Модели подробно рассмотрены в кн. Г. Ю. Ризниченко «Лекции по математическим моделям в биологии».

```

#создаем график
p[0].line_color='red';
p[1].line_color='blue'
p.show()

```

Решение $x(t) = C_1 e^{rt}$

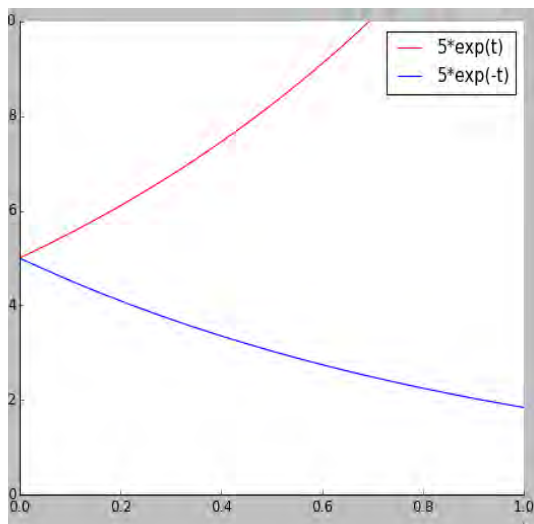


Рис. 6. Результат выполнения кода

Пример *Вещество переходит в раствор*

Дано: пусть количество вещества, переходящего в раствор, пропорционально интервалу времени и разности между максимально возможной концентрацией P и концентрацией x в данный момент времени, k – коэффициент пропорциональности: $\Delta x = k (P - x) \Delta t$

В форме дифференциального уравнения этот закон может быть представлен в следующем виде: $\frac{dx}{dt} = k(P - x)$. Предпо-

ложим, что $x(0) = 0$, $P = 200$ $k = 2$

Найти $x(t)$ и построить график решения.

код

результат выполнения кода смотри на рис. 7.

```
#-*- coding: cp1251 -*-  
from sympy import* # подключаем модуль  
x = Function('x') # создаем логическую функцию  
C1,t,k,P = symbols('C1,t k,P') # создаем логические переменные  
g = Eq(Derivative(x(t),t), k*(200-x(t))) # создаем уравнение  
d=dsolve(g, x(t)) # решаем дифференциальное уравнение  
pprint(d) # выводим решение  
eq = d.rhs.subs(t,0) # значение функции в момент времени t=0  
seq=solve(eq,C1) # решаем уравнение, чтобы найти C1  
eq1=d.rhs.subs([(C1,seq[0]),(k,2)]) # определяем в уравнении C1  
# и k  
pprint(eq1) # выводим окончательное решение  
p=plot(eq1,(t,0,5)) # создаем график
```

Решение $x(t) = C_1 e^{-kt} + 200$

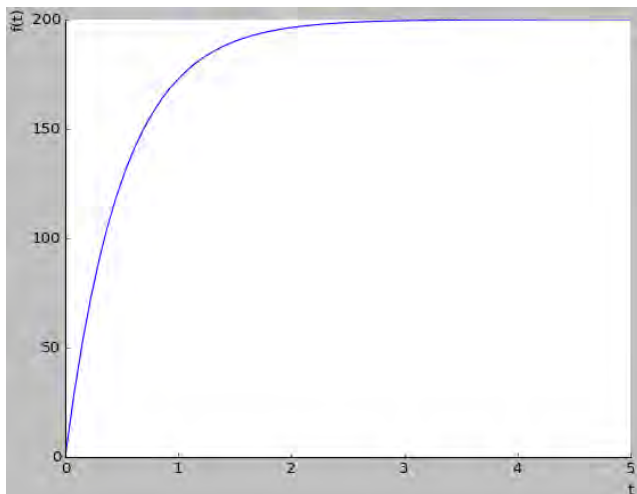


Рис. 7. Результат выполнения кода

Индивидуальные задачи

1. Решить уравнения:

а) $x^2 - 2x - 10 = 4$

б) $x^2 - 2x + 8 = 4$

в) $x^3 - 2x^2 + 8x = 4$

2. Решить системы:

а)
$$\begin{cases} 2x - 4y = 0 \\ 5x + 2y - 10 = 0 \end{cases}$$

б)
$$\begin{cases} 3x - 7y = 0 \\ 2x + 5y = 3 \end{cases}$$

3. Решить дифференциальные уравнения:

а) $y'' + 3y = \sin x$

б) $y'' + 3y = 4e^x$

3. Решить задачу Коши $y'' - y' - 2y = -4$ и $y'(0) = 1$; $y(0) = 0$

4. Вычислить интегралы

$$\int_{-1}^2 \int_0^1 6x^3 y^2 \, dx dy$$

и
$$\int_0^1 \int_0^{\frac{\pi}{2}} y \sin(5x) \, dx dy$$

и
$$\int_0^1 \int_0^1 6x^2 e^y \, dx dy$$

Лабораторная работа № 5

Тема: пакет SciPy.

Цель работы: освоение работы с основными функциями пакета SciPy.

Краткие теоретические сведения

Пакет SciPy является программным обеспечением с открытым исходным кодом для математики, естественных наук и инженерии. Он реализован как коллекция пакетов. Перечислим некоторые из них.

- `scipy.integrate` – предоставляет функции для численного вычисления интегралов при функциональном и табличном задании подынтегральной функции. Подключение пакета обычно выполняется с помощью инструкции: `from scipy.integrate import*`.

- *Функции интегрирования*

Таблица 30

Функции для интегрирования

<i>Метод, атрибут</i>	<i>Пример</i>	<i>Пояснение или результат</i>
<code>quad(f,a,b,...)</code>	<p>приведем несколько примеров:</p> <p>вычислим $\int_{-1}^1 \frac{xdx}{\sqrt{5-4x}}$</p> <pre>import numpy as np f= lambda x: x/np.sqrt(5-4*x) y = quad(f,-1,1) —</pre>	<p>однократное численное интегрирование $\int_a^b f(x)dx$. Аргументы: <code>f</code> – функция; <code>a</code>, <code>b</code> – нижний и верхний пределы интегрирования.</p> <p>Результат выполнения первой части примера: (0.16666666666666669, 8.976303190529687e-11) первое число – значение, второе – точность вычисления</p>

Метод, атрибут	Пример	Пояснение или результат
	$\int_{-1}^1 (ax + b)dx$ <p>с параметрами $a = 1$ $b = 2$ <code>f2=lambda x,a,b: a*x+b</code> <code>y= quad(f2,-1,1,args = (1,2))</code> <code>print(y)</code></p>	<p>Можно использовать опцию args, которая передает в подынтегральную функцию дополнительные параметры. После выполнения второй части примера получим (4.0, 4.440892098500626e-14)</p>
	<p>вычислим</p> $\int_2^{+\infty} \frac{dx}{x^2 + x - 2}$ <p><code>import numpy as np</code> <code>f=lambda x: 1/(x**2+x-2)</code> <code>y = quad(f,2,np.inf)</code></p>	<p>Результат третьего примера (напомним, что <code>np.inf</code> 'это ∞): (0.4620981203732969, 8.305488729933592e-11)</p>
<code>dblquad(f(x,y), a,b,gfun,hfun[,...])</code>	<p>вычислим</p> $\int_0^1 \int_0^1 (x + y)dydx$ <p><code>f=lambda y,x:x+y</code> <code>g=lambda x: 0</code> <code>h=lambda x: 1</code> <code>print(dblquad(f,0,1,g,h))</code></p>	<p>вычисление двойного интеграла, то есть</p> $\int_a^b \int_{gfun}^{hfun} f(x, y)dydx$ <p>Следует помнить, что при создании функции имя внутренней переменной (то есть y) должно быть первым; кроме того, gfun и hfun являются именами функций, определяющих нижний и верхний пределы интегрирования.</p> <p>Результат выполнения примера: (1.0, 1.1102230246251565e-14)</p>

Метод, атрибут	Пример	Пояснение или результат
	$\text{вычислим } \int_0^1 \int_{x^2}^x xy^2 dy dx$ <pre> f=lambda y,x:x*y**2 g=lambda x: x**2 h=lambda x: x print(dblquad(f,0,1,g,h)) </pre>	<p>Результат выполнения примера: (0.024999999999999994, 2.775557561562891e-16)</p>
nquad(f,range[...])	$\text{вычислим } \int_0^1 \int_0^x xy^2 dy dx$ <pre> f=lambda y,x:x*y**2 g=lambda x: [x**2,x] h=lambda : [0,1] print(nquad(f,[g,h])) </pre> $\text{вычислим } \int_0^1 \int_0^{\sqrt{1-x^2}} \int_{\sqrt{x^2+y^2}}^{\sqrt{2-x^2-y^2}} z^2 dz dy dx$ <pre> f=lambda z,y,x:z**2 g=lambda y,x: [np. sqrt(x**2+y**2),np.sqrt(2- x**2-y**2)] h=lambda x: [0, np.sqrt(1-x**2)] z=lambda : [0,1] print(nquad(f,[g,h,z])) </pre>	<p>вычисление n-кратного интеграла. Рассмотрим аргументы функции на примере</p> <p>$\int_a^b \int_{gfun}^{hfun} f(x, y) dy dx$:</p> <ul style="list-style-type: none"> • f – это $f(x,y)$; • $range$ – это список из двух элементов; первый элемент списка – имя функции, устанавливающей нижний и верхний пределы интегрирования внутреннего интеграла; второй элемент списка – имя функции для внешнего интеграла. <p>Следует помнить, что при создании функций порядок аргументов соответствует порядку передачи пределов интегрирования.</p> <p>Результат выполнения первой части примера: (0.024999999999999994, 7.5053547128889e-16)</p> <p>Результат выполнения второй части примера: (0.38294488148557376, 5.182907169428006e-09)</p>

- функция для решения системы обыкновенных дифференциальных уравнений первого порядка

Таблица 31

**Функции для решения систем
обыкновенных дифференциальных уравнений**

Метод	Пример или пояснение
odeint(f(y,t),y0,t[,...])	<p>решение системы обыкновенных дифференциальных уравнений первого порядка с начальными условиями в одной точке.</p> $\frac{dy_1}{dt} = f_1(y_1, \dots, y_n, t)$ \dots $\frac{dy_n}{dt} = f_n(y_1, \dots, y_n, t)$ <p>и $y_1(t_0) = \tilde{y}_1, \dots, y_n(t_0) = \tilde{y}_n$</p> <p>Рассмотрим аргументы функции</p> <p>$f(y,t)$, где $y = [y_1, \dots, y_n, t]$, функция должна возвращать n элементов; $y0 = [\tilde{y}_1, \dots, \tilde{y}_n]$ – массив начальных условий; $t = [t_0, \dots]$;</p> <p>full_output = False, если вы не хотите, чтобы функция возвращала словарь дополнительных параметров.</p> <p>Функция возвращает массив размера $lent(t) \times lent(y0)$. Он содержит значение y для каждого требуемого времени по t, с начальным значением $y0$ в первой строке. Другими словами, если система из двух уравнений, то первый столбец представляет значения функции $y_1(t)$, а второй – значения функции $y_2(t)$ в точках вектора t.</p>

Пример. Численно решим дифференциальное уравнение $y'(x) = -y$ с начальным условием $y(0) = 5$. Указанное уравнение имеет точное решение $y(x) = 5e^{-x}$. Построим графики для обоих решений

код

результат выполнения кода смотри на рис. 8

```
# -*- coding: cp1251 -*-  
import numpy as np  
from scipy import *  
import matplotlib.pyplot as plt  
from scipy.integrate import *  
f=lambda y,x: -y  
x=np.linspace(0,1,30) #  
y0=5.0  
y = odeint(f,y0,x) # решение уравнения численно  
y1=5*np.exp(-x) # точное решение  
y = np.array(y).ravel() # преобразование массива  
plt.subplot(121)  
plt.plot(x,y1,'-or')  
plt.subplot(122)  
plt.plot(x,y,'-sb')  
plt.show()
```

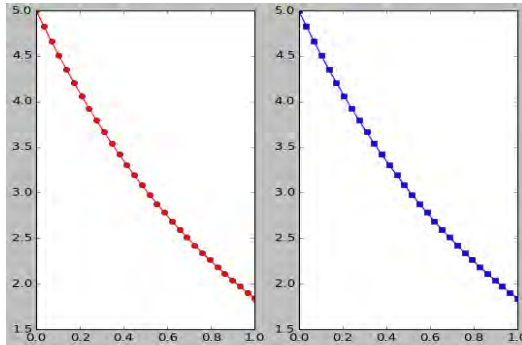


Рис. 8. Результат выполнения кода

Пример. Численно решим дифференциальное уравнение системы обыкновенных дифференциальных уравнений первого порядка:

$$\begin{cases} \frac{dy_1}{dt} = 2y_1 - 2.5y_1y_2 \\ \frac{dy_2}{dt} = -3y_2 + 2y_1y_2 \end{cases} \quad \text{и } y_1(0) = y_2(0) = [3.0, 3.1, 3.2, \dots, 3.9]$$

КОД

результат выполнения кода смотри на рис. 9

```
# -*- coding: cp1251 -*-  
from scipy.integrate import*  
import numpy as np  
import matplotlib.pyplot as plt  
def f(y,t):  
    y1,y2=y  
    return [2*y1-2.5*y2*y1,-3*y2+2*y1*y2]  
t=np.linspace(0,20,300)  
c=np.arange(3.0,4.0,0.1)  
for r in c:  
    y0=[r,r]  
    [y1,y2]=odeint(f,y0,t,full_output=False).T  
    plt.plot(y1,y2,linewidth=2)  
plt.ylabel("y2")  
plt.xlabel("y1")  
plt.grid(True)  
plt.show()
```

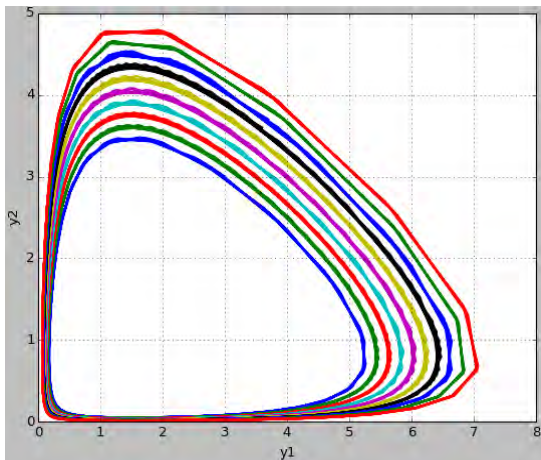


Рис. 9. Результат выполнения кода

- `scipy.linalg` – предоставляет функции, с помощью которых решаются основные задачи линейной алгебры.

Подключение пакета можно выполнить с помощью инструкции:

```
import scipy.linalg as lin.
```

Рассмотрим несколько функций модуля²³.

Таблица 32

Операторы и функции модуля `scipy.linalg`

<i>Метод, функция, атрибут</i>	<i>Пример</i>	<i>Пояснение или результат</i>
<code>lin.det(A)</code>	<code>import numpy as np A = np.mat([[1,3],[2,4]]) d = lin.det(A)</code>	вычисление определителя матрицы A. В результате выполнения примера получим -2
<code>lin.inv(A)</code>	<code>c=lin.inv(A)</code>	вычисление обратной матрицы A. В результате выполнения примера получим <code>array([[-2. , 1.5], [1. , -0.5]])</code>
<code>lin.solve(A,b)</code>	<code>c=lin.solve(A, b.T)</code>	решение системы $Ay=b$
<code>lin.block_diag()</code>	<code>a=[[1,0],[0,1]]; c=[[7]]; k=lin.block_diag(a,c)</code>	создает блок – диагональную матрицу. В результате выполнения примера получим <code>array([[1, 0, 0], [0, 1, 0], [0, 0, 7]])</code>

²³ Дано матрица $A = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}$ и вектор $b = (5,6)$.

lin.dft(n)	<pre>import numpy as np x=np.array([1,2,3,0,3,2,1,0]) m=lin.dft(8) m.dot(x)</pre>	создается дискретная матрица преобразования Фурье. Аргумент n – это размер матрицы. В результате выполнения примера получим array([1.20e+01 +0.00e+00j, -2.00e+00 -2.00e+00j, 8.88e-16 -4.00e+00j, -2.00e+00 +2.00e+00j, 4.00e+00 +1.46e-15j, -2.00e+00 -2.00e+00j, -2.33e-15 +4.00e+00j, -2.00e+00 +2.00e+00j])
------------	---	---

Индивидуальные задачи

1. Решите систему обыкновенных дифференциальных уравнений:

$$a) \begin{cases} \frac{dy_1}{dt} = 3y_1 - 5y_1y_2 \\ \frac{dy_2}{dt} = -3y_2 + 2y_1y_2 \end{cases} \quad \text{и } y_1(0) = y_2(0) = [3.0, 3.1, 3.2, \dots, 3.9]$$

$$б) \begin{cases} \frac{dy_1}{dt} = 4y_1 - 2y_1y_2 \\ \frac{dy_2}{dt} = -2y_2 + 5y_1y_2 \end{cases} \quad \text{и } y_1(0) = y_2(0) = [3.0, 3.1, 3.2, \dots, 3.9]$$

2. Вычислите интегралы:

а) $\int_0^1 \int_0^1 6x^2 e^y \, dy dx$

б) $\int_0^1 \int_0^1 \cos(2x) e^y \, dy dx$

в) $\int_2^{+\infty} \frac{dx}{x^2 + x - 2}$

3. Решите систему $Ax = b$:

а) $A = \begin{pmatrix} 1 & 3 & 5 \\ 7 & 2 & 8 \\ 4 & 9 & 4 \end{pmatrix}_И \quad b = \begin{pmatrix} 7 \\ 8 \\ 9 \end{pmatrix}$

б) $A = \begin{pmatrix} 4 & 3 & 5 \\ 7 & 3 & 4 \\ 4 & 10 & 2 \end{pmatrix}_И \quad b = \begin{pmatrix} 1 \\ 4 \\ 7 \end{pmatrix}$

в) $A = \begin{pmatrix} 1 & 5 \\ 3 & 4 \end{pmatrix}_И \quad b = \begin{pmatrix} 5 \\ 2 \end{pmatrix}$

Литература

1. Буйначев, С. К. Основы программирования на языке PYTHON / С. К. Буйначев, Н. Ю. Боклаг. – Екатеринбург, 2014. – 94 с.
2. Swaroop, С. Н. A Byte of Python / С. Swaroop. – URL: <https://wombat.org.ua/AByteOfPython> (дата обращения: 09.08.2017).
3. Федоров, Д. Ю. Основы программирования на примере языка PYTHON : учебное пособие / Д. Ю. Федоров. – СПб., 2016. – 155 с.
4. Бизли, Д. Python : Подробный справочник : пер. с англ. / Д. Бизли. – СПб., 2010. – 858 с.
5. Лутц, М. Изучаем Python : пер. с англ. / М. Лутц. – СПб., 2011. – 1280 с.
6. Ризниченко, Г. Ю. Лекции по математическим моделям в биологии / Г. Ю. Ризниченко . – URL: <http://chembaby.com/wp-content/uploads/2016/02/> (дата обращения: 09.08.2017).

Оглавление

Введение.....	3
Лабораторная работа № 1.....	4
Краткие теоретические сведения.....	4
Встроенные типы данных.....	4
Числовые типы.....	5
Последовательности.....	7
Отображения.....	12
Множества.....	14
Объекты, которые можно вызвать.....	15
Тип file.....	18
Модули.....	19
Некоторые инструкции языка Python.....	20
Индивидуальные задачи.....	25
Лабораторная работа № 2.....	26
Краткие теоретические сведения.....	26
Индивидуальные задачи.....	36
Лабораторная работа № 3.....	37
Краткие теоретические сведения.....	37
Индивидуальные задачи.....	46
Лабораторная работа № 4.....	47
Краткие теоретические сведения.....	47
Индивидуальные задачи.....	60
Лабораторная работа № 5.....	61
Краткие теоретические сведения.....	61
Индивидуальные задачи.....	68
Литература.....	70

Учебное издание

Информатика

Задания для лабораторных работ

Практикум

Составитель
Краснов Михаил Владимирович

Верстка Е. Б. Половкова
Редактор, корректор Л. Н. Селиванова

Подписано в печать 22.11.17. Формат 60×84 ¹/₁₆.

Усл. печ. л. 4,18. Уч.-изд. л. 2,0.

Тираж 5 экз. Заказ

Оригинал-макет подготовлен
в редакционно-издательском отделе ЯрГУ.

Ярославский государственный университет
им. П. Г. Демидова.
150003, Ярославль, ул. Советская, 14.