**УДК 519.6**

**ИСПОЛЬЗОВАНИЕ ВОЗМОЖНОСТЕЙ СРЕДЫ ПРОГРАММИРОВАНИЯ PYTHON ПРИ ИЗУЧЕНИИ МАТЕМАТИЧЕСКИХ ДИСЦИПЛИН В ТЕХНИЧЕСКОМ ВУЗЕ**

Акишин Б.А. к.т.н., доцент,

Донской государственный

технический университет,

г. Ростов-на-Дону,

E-mail: [akiboralex@mail.ru](mailto:akiboralex@mail.ru)

*Аннотация.* Исследованы возможности и приведены примеры использования библиотек Python при решении типовых математических задач. Проанализированы особенности интерпретации полученных результатов

*Ключевые слова*: система компьютерной математики, язык программирования Python, библиотеки функций, символьные расчеты, линейная алгебра, математический анализ, дифференциальные уравнения.

**UDC 519.6**

**USING THE PYTHON POSSIBILITIES   IN STUDYING OF THE MATHEMATICAL SUBJECTS IN TECHNICAL HIGHER EDUCATION**

Akishin B. Ph.D., Associate Professor,

Don State Technical University,

Rostov-on-Don, Russia

E-mail: [akiboralex@mail.ru](mailto:akiboralex@mail.ru)

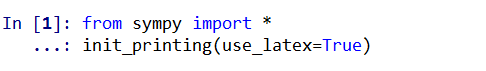
*Abstract.* Opportunities are investigated and examples are provided of using Python libraries at the solution of the typical mathematical tasks.Features of interpretation of the received results are analyzed.

*Keywords:* computer mathematics system, Python programming language, function libraries, symbolic calculations, linear algebra, mathematical analysis, differential equations

В работе [1] приводится обоснование целесообразности активного использования в процессе изучения математики в школе и вузе некоммерческих систем компьютерной математики (СКМ), в частности, программ *Maxima* и ***GeoGebra***. В то же время, студенты ряда специальностей в технических вузах, в том числе, в Донском государственном техническом университете, изучают язык программирования *Python*, поэтому, естественно, привлекать их и к решению математических задач именно в среде *Python*, тем более, что в его основных научных библиотеках *numpy, sympy, scipy, pandas, numpy*   
и других реализовано ,большинство известных аналитических и численных алгоритмов решения уравнений, задач линейной алгебры, вычисления пределов, производных и интегралов, аппроксимации, решения дифференциальных уравнений и их систем, задач теории вероятностей и математической статистики и т.д., а пакет *matplotlib* обладает хорошо развитыми возможностями визуализации двумерных и трехмерных данных.

Приведем несколько примеров аналитических (символьных) решений в *Python*  задач линейной алгебры и математического анализа.

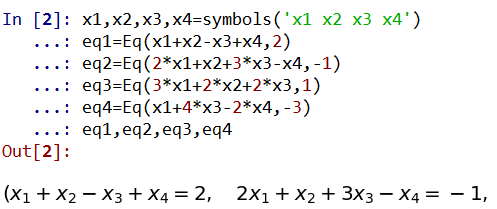
Используя интерактивную оболочку *IPython*, загрузим библиотеку символьных вычислений *sympy*  и установим режим графической печати Latex, который позволяет отображать исходные данные и результаты решения в привычном математическом виде:



**Пример 1.** Решить неопределенную систему линейных алгебраических уравнений (СЛАУ) [2]:

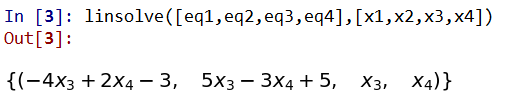
При решении подобных СЛАУ на практических занятиях по линейной алгебре обычно: а) проверяют условия выполнения теоремы Кронекера-Капелли; б) решают СЛАУ методом Гаусса (или одной из его модификаций).

Решим пример средствами *Python.* Для записи уравнений обычно применяют функцию *Eq(expr1,expr2)* из в библиотеке *sympy.*  Если выражения *expr1* и *expr2*, равны то функция *Eq()* возвращает значение True.





Для решения систем линейных уравнений предназначена, в первую очередь, функция *linsolve(...)*, в качестве аргументов которой задаются списки уравнений и неизвестных переменных:



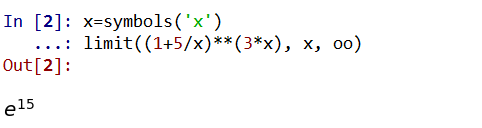
Общее решение этой неопределенной СЛАУ получено в привычном для студентов виде кортежа, где базисные переменные выражены через свободные.

Таким образом, аналитическое решение систем линейных алгебраических уравнений небольшой размерности, в том числе и неопределенных, в среде *Python* не представляет особых сложностей, не требует дополнительных проверок совместности и нагляднее, чем в некоторых СКМ, например, MathCAD.

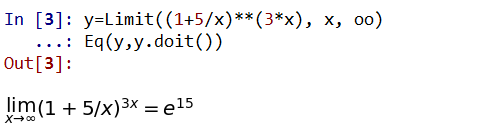
**Пример 2.**  Вычислить предел [2]:

Заметим, что решение данного примера на практике сводится ко второму специальному пределу.

Для аналитического вычисления пределов в *sympy* имеется функция *limit(...);*  для записи символа ∞ (бесконечность) используется запись oo (две буквы „o‟):



У функции *limit()* есть невычисляемый эквивалент – оператор *Limit()*, который возвращает символьный объект типа *'sympy.series.limits.Limit'* (невычисленный предел). Для вычисления символьного объекта, созданного невычисляемым оператором, нужно использовать метод *doit()*. Используя этот прием, а также упомянутую выше функцию *Eq(),* можно окончательный результатвычисления предела представить в наглядной форме:



Если функция *limit()* не может вычислить предел, например, он не существует, то, в зависимости от режима *init\_printing*, она возвращает на экран либо строку с невычисляемым эквивалентом *Limit(),* либо невычисленное исходное выражение..

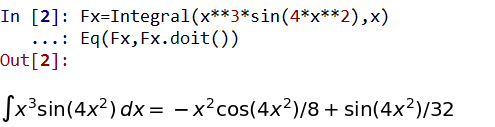
Таким образом, результаты вычисления в *Python* достаточно простых пределов весьма наглядны, однако в более сложных случаях нужно использовать возможности других СКМ: *Maxima,* *GeoGebra* или *MathCAD.*

**Пример 3.** Вычислить неопределенный интеграл [2]:

Отметим, что достаточно часто при вычислении такого типа интегралов методом «по частям» студенты неправильно выбирают части.

Для символьного интегрирования в *Python* предназначена функция *integrate(...)* из библиотеки *sympy.* С ее помощью можно вычислять как неопределенные, так и определенные интегралы. Первым аргументом функции должно быть символьное выражение, которое будет интегрироваться, вторым – переменная интегрирования или кортеж, состоящий из имени переменной и ее нижнего и верхнего пределов. Если второй аргумент – только имя, то вычисляется неопределенный интеграл, т.е. первообразная подынтегральной функции.

У функции *integrate()* также есть невычисляемый эквивалент – оператор *Integral().* Также как и в случае пределов, чтобы затем вычислить интеграл, нужно использовать метод *doit().* Воспользуемся этим фактом, чтобы сразу получить ответ в наглядной форме:

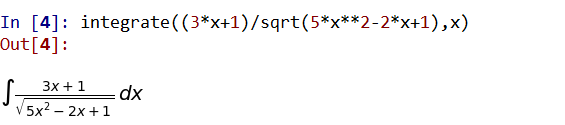


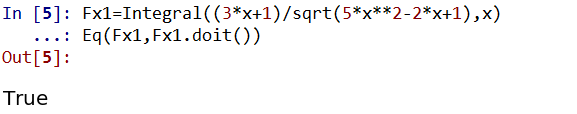
Заметим, что *sympy* не включает в результат произвольную постоянную интегрирования. Однако, константу можно добавить, если сформулировать задачу как решение соответствующего диффе-ренциального уравнения (см. пример 4)

Проверим результат дифференцированием (к объекту применяется метод *diff()* ):



Если первообразную не удалось найти, то, как было указано ранее, отображается невычисляемый объект *Integral()* в виде исходного выражения, а функция *Eq()* возвращает значение True, например [1],





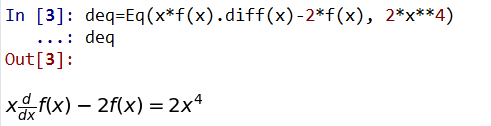
Считается, что вычисление первообразных является для студентов одной из наиболее сложных задач математического анализа, да и не все СКМ справляются с отдельными примерами. Как показывает практика, во многих случаях вычисление первообразных в среде *Python* предпочтительнее, так как просто и наглядно, но в сложных примерах нужно пробовать использовать другие программы, например, *Maxima, GeoGebra* и т.д.

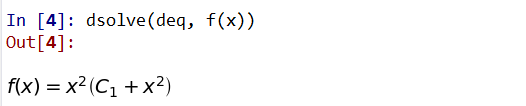
**Пример 4.** Найти общее решение обыкновенного дифференциального уравнения (ОДУ) [2]:

Данное ОДУ является линейным первого порядка

При решении ОДУ в *sympy* обычно используется следующая последовательность инструкций: а) объявляется символьная независимая переменная *x* и символьная функция *f*, которая будет представлять решение; б) создается объект, представляющий уравнение; в) решается дифференциальное уравнение с помощью функции *dsolve()*, у которой первым аргументом является объект уравнения, а вторым –искомая функция. Итак, получаем:







Обращаем внимание на то, что решение содержит произвольную переменную *.*

Функция *dsolve()* решает аналитически большинство известных типов ОДУ и систем, интегрируемых в квадратурах. Функция может использовать несколько различных методов решения ОДУ. Чтобы получить список методов, которые можно использовать для решения конкретного уравнения, следует использовать инструкцию *classify\_ode(уравнение, выражение/функция)*. Считается, что по умолчанию функция сама выбирает наиболее оптимальный метод и решения представляются, как правило, в наглядном виде. Однако стремление функции *dsolve()* выразить общее решение явно приводит к переусложненному виду, включающему неэлементарные функции.

**Пример 5.** Найти общее решение следующего ОДУ:



Данное ОДУ первого порядка является, в первую очередь, уравнением с разделяющимися переменными. При решении с параметрами по умолчания (как в примере 4), получаем:

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Видим, что общее решение этого ОДУ выражается через неэлементарную функцию LambertW() и является неприемлемым для студента. Посмотрим, какими еще методами можно решить это уравнение:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Поочередно перебирая методы, находим приемлемое решение, выдаваемое методом ***1st\_exact***. В крайнем случае, разделяем переменные (метод ***separable\_Integral***) и отдельно вычисляем интегралы (правда, без постоянной С1):

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Опция *ics* функции *dsolve()* позволяет задавать граничные условия и решать задачу Коши, однако в настоящий момент эта опция реализована только для отдельных типов уравнения и методов решения, так что получать аналитические решения задачи Коши студентам проще в других СКМ, например, в *Maxima*  [2]*.*

**Пример 5.** Исследовать функцию и построить график: 

Исследуемая функция является полиномом 5-го порядка. Она определена на всей вещественной оси, не имеет четности и никаких асимптот.

Загружаем необходимые библиотеки:

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание

Определяем функцию и списки для координат ее характерных точек:

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание

Находим точки пересечения графика с осями координат:

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Находим производные:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Исследование на экстремум:

Нахождение точек перегиба:

Строим график:

Изображение выглядит как текст, линия, График, диаграмма

Автоматически созданное описание

**Вывод.** Системы компьютерной математики весьма полезны при изучении вузовской математики. Опыт показывает, что в процессе общения с компьютером студент не только приобретает навыки работы с программами, которые пригодятся ему в дальнейшем, но и углубляет свои знания по математике, что зачастую приводит к освоению новых математических методов, заложенных в современные программы.

Заметим, что ответы, получаемые студентом на бумаге, зачастую отличаются от ответов, выдаваемых СКМ. В таких случаях студент должен провести углубленный анализ, разобраться в причинах несоответствия и довести решение до конца

Если студент знаком с основами программирования на Python, то при освоении разделов математики он может с успехом использовать его библиотеки и пакеты. В статье приведены примеры аналитического (символьного) решения лишь некоторых типовых математических задач и не рассматривались численные алгоритмы. В чем-то процесс решения и представление результатов на Python имеют преимущества перед другими СКМ, а в чем-то недостатки.

**Литература**

1. Акишин Б.А.Особенности использования систем компьютерной математики при изучении математических дисциплин в техническом вузе / Б.А. Акишин, В.А.Воронцова // Математическое образование в школе и вузе: инновации в информационном пространстве (MATHEDU' 2018): материалы VIII Международной научно-практической конференции (Казань, 17-21 октября 2018 г.). - Казань: Изд-во Казан. ун-та, 2018 , С.202-206
2. Акишин Б.А. Решение математических задач с помощью пакета Maxima.Учеб. пособие */* Б.А. Акишин. Л.В. Черкесова., А.В. Галабурдин*.* - Ростов н/Д: Издательский центр ДГТУ, 2015. - 100 с

**Регулярные выражения в Python**

1. Лучше пройти мини курс <https://regexone.com/>

2. Тестировать регулярки можно на  <https://regex101.com/#python>

3. Смотреть шпаргалки - <https://www.exlab.net/files/tools/sheets/regexp/regexp.pdf>

****

**Основные функции**

Изображение выглядит как текст, Шрифт, белый, чек

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

**Пример с решением**

1. Дана последовательность строк.

Выведите строки, содержащие две буквы "**z**﻿", между которыми ровно три символа.

**Sample Input:**

zabcz

zzz

zzxzz

zz

zxz

zzxzxxz

**Sample Output:**

zabcz

zzxzz

**Решение**

import re

SI = ['zabcz', 'zzz', 'zzxzz', 'zz', 'zxz', 'zzxzxxz']

pattern = r'z...z'

for words in SI:

if re.search(pattern, words):

print(words)

Ответ: zabcz zzxzz