МРНТИ 14.25.07
УДК 37
DOI: https://doi.org/10.51889/2020-1.1728-7901.06

TEACHING HIGH SCHOOL STUDENTS TO SOLVE DIFFERENTIAL EQUATIONS USING PYTHON AT MATH CLASS

Alkhan K.B.[1], Shaimova Z.E.[1]

[1]University of International Business, Almaty, Kazakhstan

Abstract

The article discusses examples of the differential equation problem in Python with a graph for high school students. The basic characteristics of some tools for solving problems in mathematics using information technology are given. The difference between the two modern-known computer programs Python and Pascal is briefly explained. The article uses illustrative examples of built-in and manually entered functions that can be repeated by readers in Python to recreate graphs of trigonometric, differential, and other functions. In conclusion, it describes how the program can save time and energy for solving graphic problems in mathematics using the Python program. The article concludes that it is important for students to use logic to solve problems with improvised tools, where there are built-in functions, compared to remembering programming language algorithms for solving problems.

Keywords: Python, program language, importing, function, derivative, differential equation.

Аңдатпа

Қ.Б. Альхан[1], Ж.Е. Шаимова[1]

[1] Халықаралық Бизнес Университеті, Алматы, Қазақстан

ЖОҒАРҒЫ СЫНЫП ОҚУШЫЛАРЫН МАТЕМАТИКА САБАҒЫНДА ДИФФЕРЕНЦИАЛДЫҚ ТЕҢДЕУДІ PYTHON АРҚЫЛЫ ШЕШУДІ ҮЙРЕТУ

Мақалада орта мектепке арналған графигі бар Python-да дифференциалдық теңдеу есептерінің мысалдары қарастырылған. Ақпараттық технологияны қолдану арқылы математикада есептерді шығаруға арналған кейбір құралдардың негізгі сипаттамалары келтірілген. Екі Python және Pascal компьютерлік бағдарламаларының арасындағы айырмашылық қысқаша түсіндіріледі. Мақалада тригонометриялық, дифференциалдық және басқа функциялардың графиктерін қайта құру үшін Python-да оқырмандар қайталай алатын кіріктірілген және қолмен енгізілген функциялардың көрнекі мысалдары келтірілген. Қорытындылай келе, бұл программа Python бағдарламасын қолдана отырып, математикадағы графикалық есептерді шешуге уақыт пен күшті қалай үнемдейтінін сипаттайды. Мақала оқушыларға есептерді шешуге арналған бағдарламалау алгоритмдерін есте сақтаумен, кіріктірілген функциялар бар импровизацияланған құралдармен мәселелерді шешу үшін логиканы қолдану маңыздылығын тұжырымдайды.

Түйін сөздер: Python, программа тілі, импорттау, функция, туынды, дифференциалдық теңдеу.

Аннотация

К.Б. Альхан[1], Ж.Е. Шаимова[1]

[1]Университет Международного Бизнеса, Алматы, Казахстан

ОБУЧЕНИЕ УЧЕНИКОВ СТАРШИХ КЛАССОВ НА УРОКАХ МАТЕМАТИКИ РЕШЕНИЮ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ С ИСПОЛЬЗОВАНИЕМ PYTHON

В статье рассматриваются примеры задачи дифференциального уравнения в Python с графиком для учащихся средней школы. Приводятся основные характеристики некоторых инструментов для решения задач математики с помощью информационных технологий. Кратко поясняется разница между двумя компьютерными программами, на языках Python и Pascal. В статье используются наглядные примеры встроенных и вручную вводимых функций, которые могут быть повторены учащимися в Python, чтобы воссоздать графики тригонометрических, дифференциальных и других функций. В статье сделан вывод о важности использования логики для решения задач подручными средствами, где есть встроенные функции, по сравнению с запоминанием алгоритмов языка программирования для решения задач для учащихся.

Ключевые слова: Python, программный язык, импорт, функция, производная, дифференциальное уравнение.

Computer science is currently the most changing educational field. Discipline, both among schools, and among subjects studied at schools and high education universities. It should be noted that in most schools Pascal or Basic is currently being studied as a programming language, which raises the logical question of whether there is a need for future computer science teachers studying at the university to learn the Python

language, whether there is a need to master the basics of this language high school students [1]. Of course, schoolchildren learning Python, as the first programming language, may cause some legitimate fears: these will include, first, dynamic typing and high-level language. For example, replacing the concept of "array" with a high-level list does not give students a full-fledged opportunity to analyze the principles of the internal organization of the array. However, the undoubted advantages of learning Python as the first programming language at school. Programs in Python are much more concise than Pascal, which greatly simplifies the task of introducing the language to novice programmers, as finding errors and debugging requires significantly less time. For example, compare two elements of program code written in the Pascal and Python programming languages *(Picture 1)*:

```
a=[1]*1000 и
var a: array [1..1000] jf integer;
…
for i:=1 to 1000 do
a[i]:=1;
```

Picture 1. Difference between Python and Pascal program languages

Based on the above code, we can see that in two programming languages equivalent operations are written, as a result of which we will get an array of 1000 elements filled with units. However, in Python this code takes 1 line, whereas in Pascal 3. Perhaps, from a methodological point of view, when studying this section and solving the above problem, the student needs to explain that the array is a continuous fragment of the allocated memory, and when creating it, we should reserve a place for it in memory, declaring it, and then initialize it. However, the line a = [1] * 1000, in our opinion, reflects the meaning of the action performed by the student (you need an array of 1 repeated 1000 times) is more complete and, ultimately, easier to write [2]. Considering the issues of teaching computer science in grades 7-9, we can note that these are the initial ideas about programming, perhaps there is a threshold at which the child stops, having a general idea of arrays, their declaration and processing. At the next stage of training, in specialized classes, the teacher and student will have at their disposal a universal, modern programming language that is used for software development [3-4]. Of course, many high-level routines built into the Python language, the wide functionality of the language will lead to the temptation for the student to use these features, instead of really studying the algorithms and principles of operation of these functional elements. To look more closer, there are given some examples using Python. First, in order to code off we need to import some of the necessary libraries which we will be use throughout the code *(Picture 2)*.

```
In [1]:  import math
         import numpy as np
         import matplotlib.pyplot as plt
```

Picture 2. Importing libraries in Python

Then, we need want to take a derivative of some function f(x). To define what x values those are we create some array from 0 to 4 math.pi. we have to specify how many function evaluations we will be using. Before that we define what N is, we make it 10,000 points (number of function evaluation) *(Picture 3)*.

```
In [2]:  N=10000 #Number of function evaluation
         xvals=np.linspace(0,4*math.pi,N)
         h=(4*math.pi)/(n*1,0)
```

Picture 3. Number of function evaluation

Then to define function f(x) equals to Val equals math.sin(x) return Val. Next, define a function to evaluate what the derivative of f(x). Define fPrime(x) and we essentially use the same equation but without the limit as H goes zero because we use a specific H for choosing one that sufficiently close to zero [5] *(Picture 4)*.

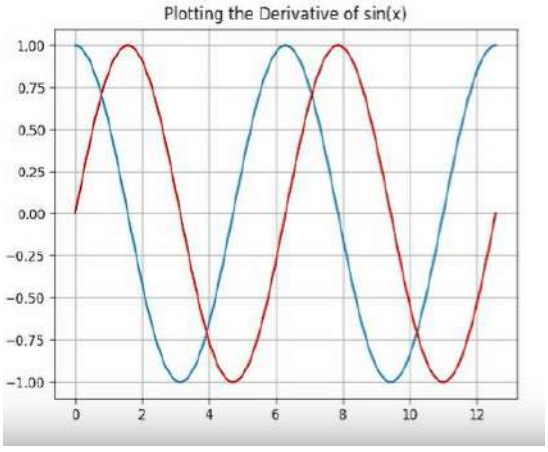
```
def f(x):
    val = math.sin(x)
    return val

def fPrime(x):
     new val = (f(x+h)-f(x))/h
    return new val

#xvals = {0,h,2h,3h,...,4 pi}

yvals = []

for i in xvals:
    edrivative = fPrime(i)
    yvals.append(drivative)

fvals = {}

for i in xvals:
    function = f(x)
    fvals.append(function)
    plt.plot(xvals,yvals) #plotting the derivative of sin(x)

    plt.plot(xvals,fvals,"r") #plotting sin(x)
    plt.grid()
    plt.title("plotting the derivative of sin(x)")
    plt.show()
```

Picture 4. Describing the process of function sin(x)

Val is equal to f(x+h)-f(x))/h. If we think about what our x values are we start from zero go to 4 pi 10,000 points. So the first point of that should be zero. The next point should be 0 + h and then the next point plus H of 2H, 3H so on until we get to 4 pi. There would be 10,000 points in that array [6].

Next we need to evaluate fPrime at all of these points. For I in xvals. We want to evaluate out fPrime at those X values we call it derivative is equal to Fprime evaluated at pi. So this is automatically is going to start evaluating the derivative. Create another list y vowels it will be an empty list. And we continuously add those evaluations to the list. yvals.append(derivative). So after each iteration of the loop finishes add what we have got to the list [7].

And we add another for loop. Fvals will correspond to the y values associated with just sin(x). We plot what all this looks like. So, plt.plot and add depended and independed x plt.plot(xvals,fvals, 'r') # plotting sin(x), plt.grid(), plt.title("plotting the derivative of sin(x)"). *(Picture 5)*.



Picture 5. Plotting the derivative of sin(x)

The next example, pic.6 shows differential equation solved in scipy.integrate package using ODIENT function, it requires inputs like:

y= odeint (model, y0, t)

1. model: function name which returns derivative value y and dydt=model (y,t)
2. y0= initial conditions of derivative states
3. t: time points at which the solution should be reported.

$$\frac{dy(t)}{dt} = -ky(t)$$

We come up with numerical solution to integrate this differential equation and will use odeint package as it shows above.

First, we define function, where function returns the derivative dy/dt, as we give different t values and y values. The initial condition is equal to 5, time point is equal to three arguments the fourth one we add args=k. We record them as y1, y2, y3.

At the end we plot those three lines. We add labels in order to know which one those are *(Picture 6)*.

```
import numpy as np
import scipy.integrate
import matplotlib.pyplot as plt

#function that returns dy/dx
def model(y,t,k):
    dydt = -k*y
    return dydt

#initial condition
y0 = 5

#time points
t= np.linspace(0,20)

#solve ODEs
k = 0,1
y1 = odeint(model. y0, t, args=(k,))
k = 0,2
y2 = odeint(model. y0, t, args=(k,))
k = 0,5
y3 = odeint(model. y0, t, args=(k,))

#plot results
plt.plot(t, y1, 'r-', linewidth=2, label='k=0,1')
plt.plot(t, y2, 'b--', linewidth=2, label='k=0,2')
plt.plot(t, y3, 'g:', linewidth=2, label='k=0,5')
plt.xlabel('time')
plt.ylabel('y(t)')
plt.legend()
plt.show
```

Picture 6. Differential equation in Python example #2

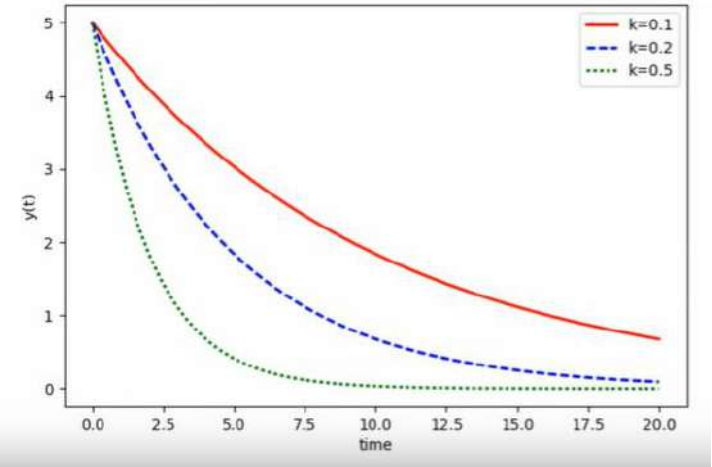
The solution of our odeint equation shows below. We can see that the higher the k value the more quickly it decays to the value of 0 [8]. Using this we can solve different and difficult differential equations like *(Picture 7)*:

1. $\frac{dy(t)}{dt} = -y(t) + 2; \quad y(0) = 0$
2. $4\frac{dy(t)}{dt} = -y(t) + u(t); \; y(0) = 1$
3. $\frac{dy(t)}{dt} = 3exp(-t); \quad y(0) = 0$

4. $\frac{dy(t)}{dt} = 3 - y(t); \;\; ; \;\; y(0) = 0$

5. $2\frac{dy(t)}{dt} = -z(t) + u(t);$

6. $5\frac{dy(t)}{dt} = -y(t) + x(t); \; u = 2S(t-5), \; x(0) = 0,$



Picture 7. The solution of differential equation in Python example #2

The next example pic.8 describes differential equation which is in latex format and used sympy library. To solve an order differential equation we need to define the variables that our function depends on as well as function itself. X is independent variable, x = sp.symbols('x').

To define differential equation diffeq = Eq(f.diff(x,x)-2*f,0) $-2f(x) + \frac{d^2}{dx^2}f(x) = 0.$

To solve it we write dsolve(diffeq,f)$f(x) = C_1 e^{-\sqrt{2}x} + C_2 e^{\sqrt{2}x}$ *(Picture 8).*

```
In [7]: from sympy.interactive import printing
        printing.init_printing(use_latex=True)
        from sympy import *
        import sympy as sym

        x = sym.symbols('x')

        f = sym.Function('f')(x)

        diffeq = Eq(f.diff(x,x)-2*f, 0)

        display(diffeq)
        dsolve(diffeq,f)
```

$$-2f(x) + \frac{d^2}{dx^2}f(x) = 0$$

Out[7]: $$f(x) = C_1 e^{-\sqrt{2}x} + C_2 e^{\sqrt{2}x}$$

Picture 8. Differential equation in Python example #3

Thus, the teacher can approach most of the tasks associated with sorting arrays, searching for elements, which ultimately at the profile stage of training will allow the student to solve a large number of various problems in a short time. Considering the intricacies of preparing for the exam in computer science, in the framework of the application of the functional elements of the language, in our opinion, students should not

have difficulties in preparing for the exam [9]. The student can successfully overcome the ban on the use of built-in functions by understanding the intricacies of the operation of the algorithms if language learning takes place taking into account the methodological features of the awareness of the regularity and algorithm of the element's functions. As a result, it can be noted that not only is there no need to abandon the idea of studying high-level programming languages at school, but, on the contrary, learning Python, with the right approach and taking into account methodological features, will open up new horizons and opportunities for the student, as modern programming languages, improving, becoming more universal, flexible and simple, convenient for perception and debugging. Such an approach to the study of high-level languages will make it possible to train beginner programmers with versatile experience in writing programs at the school level.

References:

1 Amit, S., (2015). Doing math with Python. San Francisco, SF: No Strach Press, Inc.// http://index-of.es/Varios-2/Doing%20Math%20with%20Python.pdf

2 Thompson, P. W., & Silverman, J. (2008). The concept of accumulation in calculus. In M. P. Carlson & C. Rasmussen (Eds.), Making the connection: Research and teaching in undergraduate mathematics (pp. 43-52). Washington, DC: Mathematical Association of America // https://www.researchgate.net/publication/264119290_Introducing_the_derivative_via_calculus_triangles

3 Туркин А.В. Автоматическое дифференцирование в Python// https://cyberleninka.ru /article/n/avtomaticheskoe-differentsirovanie-v-python/viewer

4 Евтушенко Ю. Г. Оптимизация и быстрое автоматическое дифференцирование //М.: Научное издание ВЦ РАН. – 2013. http://www.ccas.ru/personal/evtush/p/198.pdf

5 Mortensen M., Langtangen H. P. High performance Python for direct numerical simulations of turbulent flows //Computer Physics Communications. – 2016. – Т. 203. – С. 53-65.

6 B. Stadie, Z. Xie, P. Moritz, J. Schulman, J. Ho. Computational graph toolkit: a library for evaluation and differentiation of functions of multidimensional arrays. https://github.com/joschu/cgt

7 Andersson J. A general-purpose software framework for dynamic optimization //Arenberg Doctoral School, KU Leuven. –2013.

8 Community Portal for Automatic Differentiation. http://www.autodiff.org/

9 Weinstein M. J., Rao A. V. A source transformation via operator overloading method for the automatic differentiation of mathematical functions in MATLAB //ACM Transactions on Mathematical Software (TOMS). – 2016. – Т. 42. – №. 2. – С. 11.

МРНТИ 27.39.29
УДК 517(075.8)

DOI: https://doi.org/10.51889/2020-1.1728-7901.07

А.А. Джумабаева[1], А.Е. Жетписбаева[1]

[1]Евразийский национальный университет им.Л.Н. Гумилева, г.Нур-султан, Казахстан

ПОРЯДОК НАИЛУЧШЕГО ПРИБЛИЖЕНИЯ ФУНКЦИЙ В ПРОСТРАНСТВЕ ЛЕБЕГА

Аннотация

В статье рассматривается $L_p(T^2)$ пространство Лебега периодических функций двух переменных. Изучены вопросы приближения функций двух переменных тригонометрическими полиномами с “номерами” гармоник из ступенчатых гиперболических крестов. Величина $E_{Q_n^r}(f)_p = \inf_{t \in T(Q_n^r)} \| f - t \|_p, i \le p \le \infty$ -наилучшее приближение функции $f(x)$ тригонометрическими полиномами с "номерами" гармоник из ступенчатого гиперболического креста Q_n^r. Статья состоит из двух разделов. В первом разделе приведены некоторые известные утверждения, необходимые для доказательства основных результатов. Во втором разделе установлены точные по порядку оценки наилучших приближений некоторых функций. Эти оценки дают возможность оценить величины верхних граней наилучших приближений для определенных классов функций. Вопросы, рассмотренные в настоящей работе, относятся к кругу вопросов, изученных в работах К. И. Бабенко, С. А. Теляковского, Я . С. Бугрова, Н С. Никольской.

Ключевые слова: пространство Лебега, наилучшее приближение двумерным углом, ступенчатый крест, тригонометрический полином, производная Лиувилля-Вейля, общие монотонные последовательности.