TECHNICAL UNIVERSITY OF CLUJ-NAPOCA

ACTA TECHNICA NAPOCENSIS

Series: Applied Mathematics, Mechanics, and Engineering
Vol. 65, Issue Special IV, December, 2022

CODING IN PYTHON FOR MATHEMATICS, SCIENCE AND ENGINEERING

Iulian MALCOCI, Oleg CIOBANU, Radu CIOBANU

Abstract: *Python is a powerful programming language that provide the opportunity for easy understanding, learning and using the coding skills, combined with efficient and simple high-level structures data, obtained by effective approach to object-oriented programming. Therefore, we can state that dynamic coding within Python syntax, together with its simplistic but at the same time interpreted and powerful structures, make it an excellent language (computer programming) for effective application developments in several fields like mathematics, science and engineering. Why Python? As Python syntaxes are closer to human language, it is easier to learn, understand and write the code. Python has a large community support and also it has large open-source modules, libraries and packages. In this paper we will solve three different cases by coding in Python: a) Python is a powerful tool for mathematical calculations, we will create a code that will calculate mathematical expression; b) we will create a computer-generated harmonograf figure, a code that will translate the equations into a graph that emulates a harmonograf path; c) using Python coding we will simulate elastic-collision in one-dimensional Newtonian theory between two particles.*

Key words: *Python programming language, mathematical calculations, Harmonograf figure, elastic-collision, coding for mathematics, coding for science, coding for engineering.*

1. INTRODUCTION

Python is a programming languages which can claim to be both simple and powerful tools, both for beginners and experienced programmers. Regarding this aspect, Python's elegant syntax and dynamic programming, together with its interpreted nature, make it an excellent language for coding and effective application development in several fields (like math, science and engineering) on most platforms [1].

During the last 20 years Python has become more and more popular and used programming language, due to [2]:

- Python has a simple and flexible code structure and the code is easy to understand and read;
- Python is highly extendable due to its high number of free available Python Packaged and Libraries;
- Python can be used on all platforms (Windows, macOS and Linux);
- Python is multi-purpose and can be used for to program Web Applications, Data Science and Engineering Applications;
- Python is open source and free to use.

There are many different rankings regarding which programming language which is most popular. In most of these ranking, Python is in top 5. According the IEEE Spectrum`s (fig. 1) ranking of the top programming languages for 2021, Python is a leader [3]:

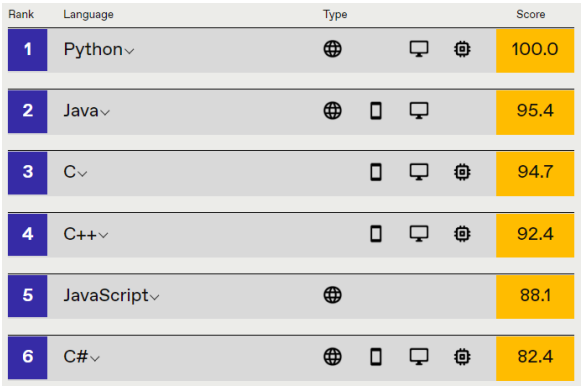

Fig. 1. IEEE Spectrum`s ranking.

2. INFORMATION ABOUT PYTHON

Python was first realized in 1991, at the moment latest version is 3.10.6. Python is a multi-purpose programming languages (due to its many extensions), examples are scientific computing and calculations, simulations, web development, GUI (Graphic User Interface), visualization and animation.

The programming language is maintained and available from (Python Software Foundation): https://www.python.org Here we can download the basic Python features in one package, which includes the Python programming language interpreter, and a basic code editor, or an integrated development environment, called IDLE.

But this is just the Python core, the interpreter a very basic editor, and the minimum needed to create basic Python programs. Typically, we will need more features for solving our tasks. Then we can install and use separate Python packages created by third parties. These packages need to be downloaded and installed separately (typically we use something called PIP), or we can choose to use a distribution package like Anaconda [1,2].

Advantages of using Python:

- As Python syntaxes are closer to human language, it is easier to learn, understand and write the code;

- Python has a large community support and also it has a large number of modules, libraries and packages;

- Python is a choice of language in data science, machine learning and artificial intelligence due to its wide variety of machine learning packages and libraries.

Disadvantages of using Python:

- There are no time optimizers in Python, hence it's slower than most of the languages like C, C++, and Java;

- The indentation-based coding makes it a little difficult for people changing their language from C, C++, or Java to Python.

Lots of Python packages exists, depending on what you are going to solve. We have Python packages for Desktop GUI Development, Database Development, Web Development, Software Development, Data-Science Development etc.

Python Packages for visualization, Science and Numerical Computations that we will use in this paper are:

- **NumPy** – the fundamental package for scientific computing with Python;

- **Matplotlib** – the fundamental package for 2D and 3D plotting in Python;

- **Tkinter** – the standard Python library designated to create Graphical User Interfaces (GUI);

- **math** – the standard Python module that provide access to the mathematical functions.

3. PYTHON FOR MATHEMATICS

In this study case we will create and solve mathematical expressions in Python. The mathematical expression will look like this:

$$c = 3a^2 + \sqrt{a^2 + b^2} - e^{\ln(a)}. \qquad (1)$$

We can create Python Script that will calculate c for different values for a and b.

Python Script:

```
1 import math as mt
2 a = 2
3 b = 2
4 def calcexpression(a,b):
5     c = 3*a**2+mt.sqrt(a**2+b**2) \
6     - mt.exp(mt.log(a))
7     return c
8 sol = calcexpression(a,b)
9 print('c =',sol)
```

After we Run this Script we will obtain solution for c when $a=2$ and $b=2$:

```
c = 12.82842712474619
```

To solve this mathematical expression we create a Python Script only from 8 rows (rows 5 and 6 is the same row), use *math* module to access mathematical function *mt.sqrt* *mt.exp* and *mt.log*. After I Run the script computation time is about 1 sec to obtain the solution.

In the other hand we can simply change values for a and b to obtain others results. For $a=2$ and $b=-2$ we will obtain the same result:

```
c = 12.82842712474619
```

For example, if $a=-1$ and $b=0.5$ the result become:

```
c = 3.118033988749895
```

Because *math* module cannot be used with complex numbers, we have to carefully choose the values for variables a and b.

4. PYTHON FOR ENGINEERING

In this study case we will create computer-generated harmonograph figures using Python coding.

A **harmonograph** is a mechanical equipment similar with pendulums, used to create drawings or geometric images, related to Lissajous curves or similar with drawings of higher complexity. The devices, which began to appear in the middle of19th century and peaked in popularity above 1890s, cannot be certainly attributed to a single person, although Hugh Blackburn, a professor of mathematics at the University of Glasgow, is commonly believed to be the official inventor [4].

In this paper we will create a code that have to simulate, so-called "lateral" harmonograph (two pendulums that control the path of a pencil relative to a work surface). Pendulums give the pencil two movements back and forth along one axis, and back and forth along a perpendicular axis. By modifying phase and frequency of the pendulums relative to one another, different drawings (figures) will be created, like ellipses, spirals, figure eights and other Lissajous figures. In figure 2 we can see a sample of real harmonograph [4].

Fig. 2. Harmonograph (Matemateca IME/USP/Rodrigo Tetsuo Argenton).

A harmonograph creates its figures using the movements of damped pendulums, where movement of a damped pendulum is described by the equation:

$$x(t) = A\sin\left(tf + p\right)e^{-dt}, \qquad (2)$$

in which: ***t*** represents time, ***A*** represents amplitude, ***f*** represents frequency, ***p*** represents phase and ***d*** represents damping.

A typical harmonograph has two pendulums that move in such a fashion, and a pen that is moved by two perpendicular rods connected to these pendulums. Therefore, the path of the harmonograph figure is described by the below parametric equations, which results from (2), that we will use to create our Python Script to emulate a harmonograph figure:

$$\begin{cases} x(t) = A_1\sin\left(tf_1 + p_1\right)e^{-d_1 t} + A_2\sin\left(tf_2 + p_2\right)e^{-d_2 t} \\ y(t) = A_3\sin\left(tf_3 + p_3\right)e^{-d_3 t} + A_4\sin\left(tf_4 + p_4\right)e^{-d_4 t} \end{cases}$$

For numerical computation and Harmonograph figures visualization we will use *NumPy* and *Matplotlib* libraries.

Python Script to create Harmonograph figures:

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  # do not change these variables
5  n = 10000
6  t = np.logspace(np.log10(10), \
7  np.log10(500),n)
8
9  # change these variables
10 A = [2,1,1.5,1.5]
11 d = [.004,.004,.002,.007]
12 f = [3,1,2,1]
13
14 # generate XY value pairs
15 x = A[0]*np.sin(t*f[0])* \
16 np.exp(-d[0]*t)+ \
17 A[1]*np.sin(t*f[1])* \
18 np.exp(-d[1]*t)
19
20 y = A[2]*np.sin(t*f[2])* \
21 np.exp(-d[2]*t)+ \
22 A[3]*np.sin(t*f[3])* \
23 np.exp(-d[3]*t)
24
25 # plot
26 plt.plot(x,y,'red',linewidth=.15)
27 plt.axis('off')
28 plt.show()
```

Now we can Run the Script above and we will obtain these equations into graph 1 that emulates a harmonograph (fig. 3).

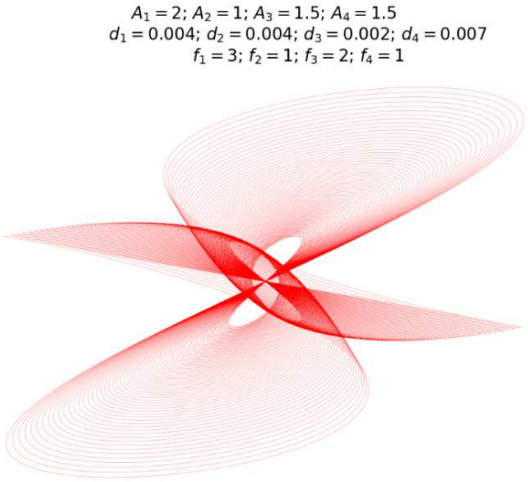

Fig. 3. Harmonograph, graph 1.

Modifying arguments for variables we can obtain a lot of Harmonograph shape. For the next 2 Harmonograph I will change the values for variables *A*, *d* and *f* to obtain graph 2 (fig. 4) and graph 3 (fig. 5).

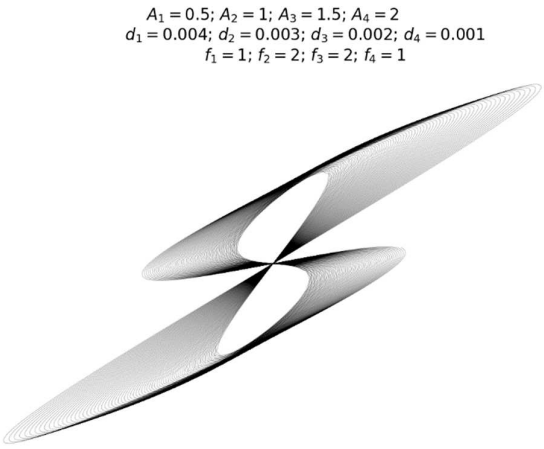

Fig. 4. Harmonograph, graph 2.

Also we can make another change to our Python Script to modify not only the variable *A*, *d* and *f*, but also we can modify color of Harmonograph pen or line width.

In figure 4 I change the color of our graph from `'red'` to `'black'`.

In figure 5 I change the color of our graph from `'black'` to `'blue'`, and `linewidth=.15` to `linewidth=.25`.

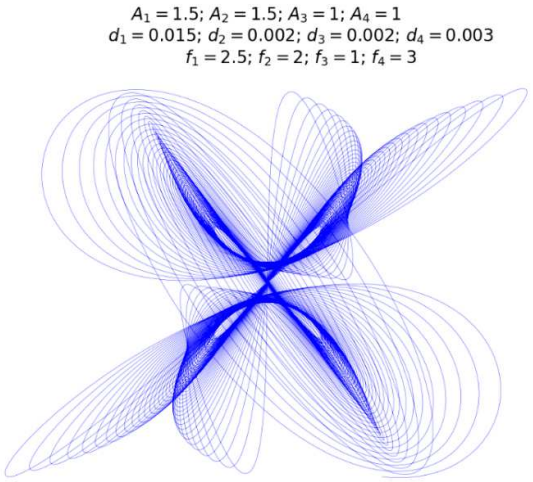

Fig. 5. Harmonograph, graph 3.

5. PYTHON FOR SCIENCE

In this study case we will compute and visualize elastic collision between two particles of unequal masses like it is shown in figure 6 [5].

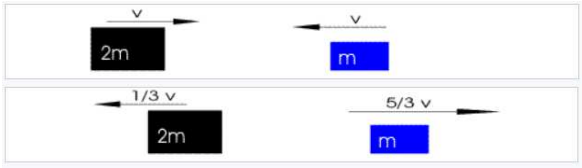

Fig. 6. Elastic collision of unequal masses.

In physics, an **elastic collision** is a such encounter collision between two particles (bodies) in which the total kinetic energy of the two bodies remains the same. For ideal, perfectly elastic collision, there is no net conversion of kinetic energy into other forms such as heat, noise, or potential energy.

After the collision of small bodies, kinetic energy is first converted to potential energy associated with a repulsive or attractive force between the particles (when the particles move against this force, the angle between the force and the relative velocity is obtuse), then this potential energy is converted back to kinetic energy (when the particles move with this force, the angle between the force and the relative velocity is acute). [5]

In an elastic collision, both momentum and kinetic energy are conserved. Consider particles 1 and 2 with masses m_1 an d m_2, and velocities u_1, u_2 before collision, v_1, v_2 after collision.

The conservation of the total momentum before and after collision is expressed by [5]:

$$m_1u_1 + m_2u_2 = m_1v_1 + m_2v_2 . \quad (3)$$

Likewise, the conservation of the total kinetic energy is expressed by:

$$\frac{1}{2}m_1u_1^2 + \frac{1}{2}m_2u_2^2 = \frac{1}{2}m_1v_1^2 + \frac{1}{2}m_2v_2^2 . \quad (4)$$

These equations may be solved directly to find v_1, v_2 when u_1, u_2 are known:

$$\begin{cases} v_1 = \dfrac{m_1 - m_2}{m_1 + m_2}u_1 + \dfrac{2m_2}{m_1 + m_2}u_2 \\ v_2 = \dfrac{2m_1}{m_1 + m_2}u_1 + \dfrac{m_2 - m_1}{m_1 + m_2}u_2 \end{cases} .$$

For numerical computation and elastic collision visualization we will use *Tkinter* and *math* modules.

Python Script to create Elastic-collision simulation:

```
1  from tkinter import *
2  import math as mp
3
4  # create simulation window
5  window = Tk()
6  window.title('Elastic-collision')
7
8  # create simulation space
9  x = 800
10 y = 400
11 canvas = Canvas(window, \
12 width=x, height=y)
13 canvas.grid(row=0, column=0)
14
15 # Initial masses and velocities
16 m1 = 8
17 m2 = 4
18 v1 = .2
19 v2 = -.2
20 # lengths to create the bodies
21 t1 = mp.sqrt(m1)*100
22 t2 = mp.sqrt(m2)*100
23
24 # Create the body1 and body2
25 body1 = canvas.create_rectangle( \
26 50,y-t1,50+t1,y, fill='purple')
27 body2 = canvas.create_rectangle( \
28 x-t2-50,y-t2,x-50,y, fill='gold')
29
30 # Create movements for bodies
31 for i in range(10000):
32     canvas.move(body1, v1, 0)
33     canvas.move(body2, v2, 0)
34     window.update()
35 # Create conditions for movements
36 # body1 touches the window left side
37     if canvas.coords(body1)[0] < 0:
38     v1 *= -1
39 # body2 touches the right side
40     if canvas.coords(body2)[2] > x:
41     v2 *= -1
42 # when body1 and body2 collide
43     if canvas.coords(body1)[2] >= \
44     canvas.coords(body2)[0]:
45     v = v1
46     v1 = (m1-m2)/(m1+m2)*v1 + \
47     (2*m2)/(m1+m2)*v2
48     v2 = (2*m1)/(m1+m2)*v + \
49     (m2-m1)/(m1+m2)*v2
```

To visualize the bodies coordinates of moving we can introduce a new row after row 34:

```
34     window.update()
35     print(canvas.coords(body1))
```

In this case when we will Run the Script we will obtain the moving coordinates for Body1 on *x* and *y* axis (fig. 7):

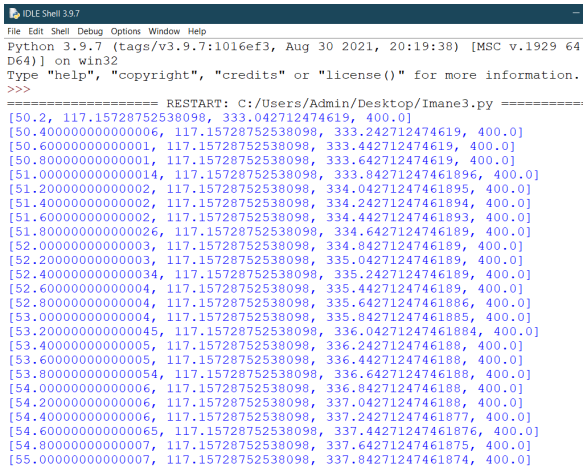

Fig. 7. Body1 *x* and *y* coordinates.

Due to the fact that our bodies during the elastic-collision simulation move (fig. 7) along *x* axis, we can observe that y_1=*117.15* and y_2=*400.0* stay constant, but x_1 change constantly his values from *50.2*, and x_2 change constantly his values from 333.24 (in pixels). In (fig. 8) we can see the simulation between bodies.

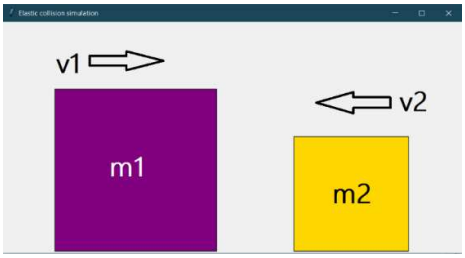

Fig. 8. Elastic-collision simulation window

6. CONCLUSION

To create and run scripts in this paper I used Python programming language interpreter, an integrated development environment, called IDLE. An Editor is a program where you create your code (and where you can run and test it). For simple Python programs you can use the IDLE Editor, but for more advanced programs a better editor is recommended, like *Spyder, PyCharm, Atom, Jupyter Notebook* etc.

Many people don't know that Python is a really powerful tool for learning math, perform algebra and trigonometry calculus, matrix analysis, numerical calculations, scientific and technical computing.

Numerical calculation and scientific computing was performed in this paper. Python limits don`t stop here, in this programming language we can perform Desktop GUI development, Database development, Web development, Software development, optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing and other tasks common in science and engineering.

For the next research I will focus to scientific animation and create it in Python environment to make impressive animation and visualize scientifically data like a movie, for example flying airplanes, 3 joint robot movement, free falling object etc.

7. REFERENCES

[1] Swaroop, C.H., *A Byte of Python*, 2005, https://homepages.uc.edu/~becktl/byte_of_python.pdf
[2] Halvorsen, H.P., *Python for Science and Engineering*, ©Hans-Petter Halvorsen, ISBN: 978-82-691106-5-4, 2019, https://www.halvorsen.blog
[3] Top Programming Languages 2021: https://spectrum.ieee.org/top-programming-languages/
[4] Wells, D. *The Penguin Dictionary of Curious and Interesting Geometry*, London: Penguin, ISBN 978-0140118131, 1992.
[5] https://www.vedantu.com/formula/elastic-collision-formula

PROGRAMARE ÎN PYTHON, APLICAȚII PENTRU ȘTIINȚĂ, MATEMATICĂ ȘI INGINERIE

Python este un limbaj de programare puternic, versatil dar în același timp ușor de învățat și înțeles. Crearea codurilor în acest limbaj de programare este relativ simplă și eficientă, alături de sintaxa foarte elegantă și interactivă. Astfel Python este limbaj ideal de programare pentru crearea de coduri și dezvoltarea rapidă a aplicațiilor în multe domenii precum matematică, știință și inginerie. De ce Python? Deoarece sintaxa în Python este foarte asemănător de limbajul uman, este ușor de învățat, de înțeles și de scris codul chiar și pentru neinițiați în programare. Python are suport și o comunitate foarte mare la nivel mondial, precum și module, biblioteci și pachete open-source. În această lucrare vom rezolva trei studii de caz prin programare în Python: a) Python este un instrument puternic pentru calcule matematice, vom crea un cod care va calcula o expresie matematică; b) vom crea o figură harmonograf generată de computer, un cod care va transpune ecuațiile într-un grafic; c) folosind Python vom simula coliziunea elastică în teoria newtoniană unidimensională între două particule.

Iulian MALCOCI, PhD, Associate Professor, Technical University of Moldova, Fundamentals of Machines Design Department, iulian.malcoci@bpm.utm.md, 00373-22-509988, Rep.Moldova, city Chișinău, șos. Hîncești 55/5, ap.5, MD-2028, 00373-69729957.

Oleg CIOBANU, PhD, Associate Professor, Technical University of Moldova, Fundamentals of Machines Design Department, oleg.ciobanu@bpm.utm.md, 00373-22-509988, Rep.Moldova, city Chișinău, str. Studenților 3/1, ap.510, MD-2045, 00373-69469460.

Radu CIOBANU, PhD, Associate Professor, Technical University of Moldova, Fundamentals of Machines Design Department, radu.ciobanu@bpm.utm.md, 00373-22-509988, Rep.Moldova, city Chișinău, str. Studenților 3/1, ap.406, MD-2045, 00373-68584665.