USING PYTHON FOR TEACHING CLASSIC MECHANICS IN UNIVERSITY STUDENTS

[1]MARIA FERNANDA HEREDIA MOYANO, [2]NATALY BONILLA GARCÍA, [3]MYRIAN CECILIA BORJA SAAVEDRA, [4]MIGUEL ÁNGEL SÁEZ PAGUAY, [5]ALEX FERNANDA ERAZO LUZURIAGA, [6]JHONNY MARLON BORJA BORJA, [7]SANDRA FABIOLA HEREDIA MOYANO,

1mariaf.heredia@espoch.edu.ec, fernandah63@hotmail.com, https://orcid.org/0000-0002-0145-2098, Facultad de Ciencias, Escuela Superior Politécnica de Chimborazo ESPOCH matriz

[2]nbonilla@espoch.edu.ec, naty_bg21@hotmail.com, https://orcid.org/0000-0001-6089-4876, Facultad de Ciencias, Escuela Superior Politécnica de Chimborazo ESPOCH matriz

[3]mc_borja@espoch.edu.ec, mc_borja@yahoo.es, https://orcid.org/0000-0002-8230-2891, Facultad de Ciencias, Escuela Superior Politécnica de Chimborazo ESPOCH matriz

[4]miguel.saez@espoch.edu.ec, https://orcid.org/0000-0003-3192-5084, Facultad de Recursos Naturales, Escuela Superior Politécnica de Chimborazo (ESPOCH), Orellana.

[5]alexerazo1407@hotmail.com, https://orcid.org/0000-0002-1089-383X, Ingeniero en Sistemas investigador independiente.

[6]johnborjab@hotmail.com,https://orcid.org/0000-0002-3708-0126, Máster en Ingeniería Química – investigador independiente.

[7]herediafhm@gmail.com, https://orcid.org/0000-0003-3668-1269, Máster en Química – estudiante PhD en la Universidad de Pardubice

ABSTRACT

This article contains codes in Python programming language as a tool to support the teaching of classical physics, presents a menu of options for calculating magnitudes within uniformly varied rectilinear motion, parabolic motion, free fall, and uniform circular motion. The written codes can be verified using Python online, and there is also a code that does require the installation of Python and additional libraries to be able to see the graph of position-time, speed-time and acceleration-time. It was verified that each code works in two different online Python links and for a better understanding of the user the calculation menu is explained in parts.

Keywords: Classic physics, Python, Teaching physics

INTRODUCTION

This research is based on the use of the Python programming language for teaching classical physics exercises in university students.

In higher education we have been faced with the challenge of using several digital tools, in this way in recent years several free services have been put online to support teaching and thus be a didactic way for learning. For this reason, it has been considered important to include learning tools through Python programming to classical physics topics.

Classical physics is considered a very important basic subject for most careers at a higher level, much more in students of the Physics career who require adequate understanding of the first topics of classical physics to continue advancing and delving into the other topics of study, then it is a priority to look for other educational means to reach the easy understanding of physics topics towards students. By including a new study methodology, it turns out to be of greater interest to the student who is in the learning stage, and thus seeks to venture into the Python programming language to solve exercises.

It is considered to address the topics of varied uniform rectilinear motion, parabolic motion, free fall and uniform circular motion which are the initial topics with which students could begin to solve exercises through an algorithm designed in Python programming language.

Python is an easy-to-understand programming language, easy to use for anyone who has basic programming knowledge and is also freely accessible, open source, this makes you find a lot of information on how to write the algorithms, how to download libraries and applications to work in a multidisciplinary way. In case the user has questions or doubts about the programming language in Python, there are many support forums that are useful and quick to dispel inconveniences.

It is intended that students to venture into two topics such as programming and solving classical physics exercises are enthusiastic, interested and concentrated at the time of

studying and solving the exercises, in this way two lines of study are consolidated, on the one hand programming and on the other hand classical physics, We believe that there are new methods and methodologies to teach and learn the basic sciences in order to attract the attention and interest of the student.

METHODOLOGY

In the first stage of this teaching is to understand the formulas that will be used in Python to solve the exercises, then the formulas that will be programmed to solve the exercises are explained.

Varied uniform rectilinear motion

This movement considers that a single system moves in a straight line that is to say in one dimension and that the velocity changes in the passage of time, so that it has an acceleration different from zero, the following formulas are considered:

- $d = V_o t + \frac{1}{2} a t^2$Distance
- Final speed $V_f = Vo + at$,
- Acceleration $a = \frac{(V_f - V_o)}{t}$,
- Time t= $\frac{(V_f - V_o)}{a}$.

Where: Vo is initial velocity, a is acceleration, t is travel time.

Parabolic motion

This is a two-dimensional motion in which the acceleration remains constant and is equal to the value of the acceleration of gravity $(9.8\, m/s^2)$, in this motion we can calculate the following quantities:

- Maximum distance,$R = \frac{v_0^2 sen\, 2\propto_0}{g}$
- Maximum height,$h = \frac{v_0^2 sen\, 2\propto_0}{2g}$
- Flight time,$t_v = 2\frac{v_o sen \propto_0}{g}$
- Upload time, $t_s = \frac{v_o sen \propto_0}{g}$
- Initial speed, , $v_{ox} = v_o cos \propto_0$, $v_{oy} = v_o sen \propto_0$ $v_o = \sqrt{v_{ox}^2 + v_{oy}^2}$

Where, v_o is the initial speed of the movement, is the angle $\propto_0$ with respect to the horizontal from where the trajectory of the movement starts, sometimes this angle is 0, g is the value of the acceleration of gravity.

Free Fall

This movement occurs when a single system is dropped from a certain height and through the action of the acceleration of gravity will touch the ground, for these calculations we will consider that if the movement is in favor of gravity then g is positive, with this, you can find exercises that ask to calculate the following unknowns:

- Height from which he was thrown, $y = \frac{1}{2} g t^2$
- Final speed,$v_y = gt$

Where, initial velocity in this movement we consider zero, g is the value of the acceleration of gravity 9.8 m / s^2 and t is the time in which the movement elapses.

Uniform circular motion

The uniform circular movement allows us to know when a single system has a movement no longer rectilinear but circular, in this case we can know the following magnitudes:

- Angle of the circular path, $\theta = \omega . t$
- Time $t = \theta / \omega$
- Centripetal acceleration, $a_c = v^2 / r$

Where, is the angular velocity, t is the time, is the ω linear velocity, v is the radius of the circular path where the motion occurs. r

Python programming code

To begin with, a menu of options was created so that the user can choose which movement he wants to use, and we used the import math library.

5 options are created to choose from in the main menu

```
import math
Options = ["A", "B", "C", "D", "E"]
while True:
   print('''
   Select the topic you want to work on:
to.  Varied uniform rectilinear motion
b. Parabolic Movement
c.  Free Fall
        d. Uniform circular motion
         and. Get out
        ''')
   option = input("Enter the topic you want to solve:")
   if not (opcion in opciones):
print("I do not select a  valid option")
       input("Press enter to continue")
       continue
```

Python programming code for Rectilinear orNiform Movement variado

For this move the following options have been placed in Python:

```
if option == "a":
       try:
          option = ["1", "2", "3", "4", "5"]
          while True:

             print("""
             ************************************************
             * UNIFORMLY VARIED RECTILINEAR MOTION *
             *                                                                                *
             ************************************************
          """)
             print('''
            Select the formula you want to solve:

            1. Tiempo -----> t=(Vf+Vo)/a
            2. Velocidad Final ---> Vf=Vo+a*t
3. Distancia ----> d=((Vo*t)+(0.5*a*t**2))
4. Aceleracion ----> a=(Vf-Vi)/t
            5. Sign Out

          ''')

             opcb = input("Enter the option you want")
             if not (opcb in opci):
print("I do not select a  valid option")
                input("Press enter to continue")
                continue
             if opcb == "1":
                try:
a = float(input("Enter the acceleration in (m/s^2) : "))
                   print()
                   vo = float(input("Enter the initial velocity in (m/s) : "))
                   print()
                   vf = float(
                      input("Enter the final velocity in (m/s) : "))
```

```
                    t = (vf+vo)/a
                    print("THE TIME IS: t = ", round(t, 3), " s ")
                except ValueError:
                    print("Incorrect Quantity")
                    continue
            if opcb == "2":
                try:
                    vo = float(input("Enter the initial velocity in (m/s) : "))
                    print()
a = float(input("Enter the acceleration in (m/s^2) : "))
                    print()
t = float(input("Enter time in (s) : "))
                    vf = vo+(a*t)
                    print("THE FINAL SPEED IS : vf = ",
                          round(vf, 3), " m/s ")
                except ValueError:
                    print("Incorrect Quantity")
                    continue
            if opcb == "3":
                try:
                    vo = float(
                        input("Enter the initial speed in (m/s) : "))
                    a = float(
                        input("Enter acceleration in (m/s^2) : "))
t = float(input("Enter time in (s) : "))
                    d = ((vo*t)+(0.5*a*(t**2)))
                    print("THE DISTANCE IS : d = ", round(d, 3), " m ")
                except ValueError:
                    print("Incorrect Quantity")
                    continue
            if opcb == "4":
                try:
                    vo = float(
                        input("Enter the initial speed in (m/s) : "))
                    print()
                    vf = float(
                        input("Enter the final velocity in (m/s) : "))
                    print()
t = float(input("Enter time in (s) : "))
                    print()
                    a = (vf-vo)/t
                    print("THE ACCELERATION IS : a = ",
                          round(a, 3), " m/s^2 ")
                except ValueError:
                    print("Incorrect Quantity")
                    continue
            if opcb == "5":
                print("End of program")
                break
        except:
            print("Incorrect quantity")
        continue
```

Python programming code for Parabolic Motion
For this movement was programmed to calculate 5 magnitudes that are as follows:

```
if opcion == "b":
        try:
            opcio = ("1", "2", "3", "4", "5", "6")
```

```
        while True:

            print("""
            ******************************
            * PARABOLIC MOVEMENT *
            *                                    *
            ******************************
        """)
            print('''
            Select the formula you want to solve:
1. Calculate the maximum distance.
2. Calculate the maximum height.
               3. Calculate the flight time.
               4. Calculate the upload time.
               5. Calculate the velocity and its components in x-y in a time t.
               6. Sign Out
               ''')
            opcc = input("Enter the case you want to solve: ")
            if not (opcc in opcio):
print("I do not select a  valid option")
               input("Press enter to continue")
               continue
            if opcc == "1":
               try:
                  Vo = float(input("Initial velocity (m/s):"))
                  B = float(input("Tilt angle (degrees):"))
                  g = float(input("Gravedad (m/s^2):"))
                  O = (B*(math.pi))/180
                  Dmax = ((Vo**2)*(math.sin(2*O)))/g
print("The maximum distance  is: ", round(Dmax, 2), "m")
                  input("click to continue")
               except:
                  print("Incorrect Quantity")
               continue
            if opcc == "2":
               try:
                  Vo = float(input("Initial velocity (m/s):"))
                  B = float(input("Tilt angle (degrees):"))
                  g = float(input("Gravedad (m/s^2):"))
                  O = (B*(math.pi))/180
                  I go = Vo*(math.sin(O))
                  Hmax = float((Voy**2)/(2*g))
print("The maximum height  is: ", round(Hmax, 2), "m")
                  input("click to continue")
               except:
                  print("Incorrect Quantity")
               continue
            if opcc == "3":
               try:
                  Vo = float(input("Initial velocity (m/s):"))
                  B = float(input("Tilt angle (degrees):"))
                  g = float(input("Gravedad (m/s^2):"))
                  O = (B*(math.pi))/180
                  I go = Vo*(math.sin(O))
                  tv = (2*I go)/g
                  print("Flight time is: ", round(tv, 2), "s")
                  input("click to continue")
               except:
                  print("Incorrect Quantity")
               continue
```

```
            if opcc == "4":
                try:
                    Vo = float(input("Initial velocity (m/s):"))
                    B = float(input("Tilt angle (degrees):"))
                    g = float(input("Gravedad (m/s^2):"))
                    O = (B*(math.pi))/180
                    I go = Vo*(math.sin(O))
                    ts = (I go)/g
                    print("Upload time is: ", round(ts, 2), "s")
                    input("click to continue")
                except:
                    print("Incorrect Quantity")
                continue
            if opcc == "5":
                try:
                    Vo = float(input("Initial velocity (m/s):"))
                    B = float(input("Tilt angle (degrees):"))
                    g = float(input("Gravedad (m/s^2):"))
                    O = (B*(math.pi))/180
                    I go = Vo*(math.sin(O))
                    Vox = Vo*(math.cos(O))
                    tv = (2*I go)/g
                    while True:
                        t = float(
                            input("Time at which you want to calculate the speed(s):"))
                        if t >= tv:
                            print(
                                "The time must be less than the flight time of", round(tv, 2), "s")
                            continue
                        if t < tv:
                            Vy = Voy-(g)*t
                            Vx = Vox
                            a = (math.pow(Vy, 2))
                            b = (math.pow(Vx, 2))
                            c = a+b
                            V = (math.pow(c, 1/2))
                        break
                    print("Velocidad en ", t, "s =", round(V, 2), "m/s")
                    print("Vx en ", t, "s =", round(Vx, 2), "m/s")
                    print("Vy en ", t, "s =", round(Vy, 2), "m/s")
                    input("click to continue")
                except:
                    print("Incorrect Quantity")
                continue
            if opcc == "6":
                print("End of program")
                break
        except:
            print("Incorrect Quantity")
        continue
```

Python Programming Code for Free Fall

In this movement you can enter the value of time and will result in the height of which the object was thrown and the final velocity.

```
if option == "c":
        print("""
            ***************************
            *FREE FALL*
            *                         *
```

```
            **************************
        """)
    try:
        g = float(9.8)
        print("Let's consider an object that is thrown at a vo=0")
        t = float(input("Enter the time s-> "))
        cl = float(0.5*g*(t**2))
        vf = float(g*t)
        print("The height the object was thrown:", round(cl, 3), " m")
        print("")
        print("The final velocity of the object:", round(vf, 3), " m/s")
        print("")
    except:
        print("Incorrect Quantity")
    continue
```

Python programming code for Uniform Circular Motion
For the circular movement we can calculate 3 magnitudes

```
if opcion == "d":
    try:
        opc = ["1", "2", "3", "4"]
        while True:
            print("""
            **********************************
            * UNIFORM CIRCULAR MOTION   *
            *                                                *
            **********************************
        """)
            print('''
            Select the formula you want to solve:
                1. Angulo:                  An=w*t
                2. Tiempo:                  t=An/w
                3. Acceleration centripeda    Ac=v^2/r
                4. Exit
                ''')
            opca = input("Enter what you want to find: ")
            if not (opca in opc):
print("I do not select a  valid option")
                input("Press enter to continue")
                continue
            if opca == "1":
                try:
value = float(input("Angular velocity (rad/s)-> "))
                    valorb = float(input("Tiempo (s)-> "))
                    valorc = value*valorb
print("The  angle of rotation is: ",
                        round(valorc, 2), "degrees")
                    input("Click to continue")
                except:
                    print("Incorrect quantity")
                    continue
            if opca == "2":
                try:
                    valora = float(input("Angulo (rad)-> "))
                    valorb = float(input("Velocidad angular (rad/s)-> "))
                    valorc = value/valorb
                    print("The time it takes is: ",
                        round(valorc, 2), "rad/s")
                except:
```

```
                    print("Incorrect quantity")
                    continue
            if opca == "3":
                try:
                    values = float(input("Tangential speed (m/s)-> "))
                    valorb = float(input("Radio (m)-> "))
                    valorc = value**2/valorb
print("The  centripoded acceleration is: ",
                        round(valorc, 2), "m/s^2")
                except:
                    print("Incorrect quantity")
                    continue
            if opca == "4":
                print("End of program")
                break
        else:
            print("End of program :)")
    except:
        print("Incorrect quantity")
    continue
```

All the boxes with the codes are placed in a single Python file and run the program with all the options.

In addition, a code was written to show the graph of the behavior of:

- Position – time
- Speed – time
- Acceleration – time

To do this you need to use the Python program on the desktop and install the libraries of "from matplotlib import pyplot" and "import matplotlib.pyplot as plt", the code is shown as follows:

```
from matplotlib import pyplot
import matplotlib.pyplot as plt

vo=float(input("Enter initial speed ==> "))
xo=float(input("Enter the starting position ==> "))
a=float(input("Enter acceleration ==> "))

def f1(t):
   return xo+vo*t+(1/2)*a*(t**2)
def f2(t):
   return vo+a*(t)
def f3(t):
   return a
t = range(0, 8)

plt.grid(True)
plt.title('''
        MRU-MRUV Charts
        Position-Time
        Speed-Time
        Acceleration-Time
      ''')

# Graph
pyplot.plot(t, [f1(i) for i in t])
pyplot.plot(t, [f2(i) for i in t])
pyplot.plot(t, [f3(i) for i in t])
```

```
# Set the color of the axes.
pyplot.axhline(0, color="black")
pyplot.axvline(0, color="black")

# Limit axis values.
pyplot.xlim(0, 7)
pyplot.ylim(0, 10)

# Save graphic as PNG image .
pyplot.savefig("output.png")
# Show it.
pyplot.show()
```

With these codes it is very useful for students to corroborate their handmade answers and even see the graphs for rectilinear movement.

RESULTS

We use the online Python link to be able to obtain the results so that users in this case students do not need to install the Python program on their computers on a mandatory basis.
Link to use Python online:
Online Python Compiler (Interpreter) (programiz.com)
https ://replit.com/languages/python3

This is how the initial program message is displayed

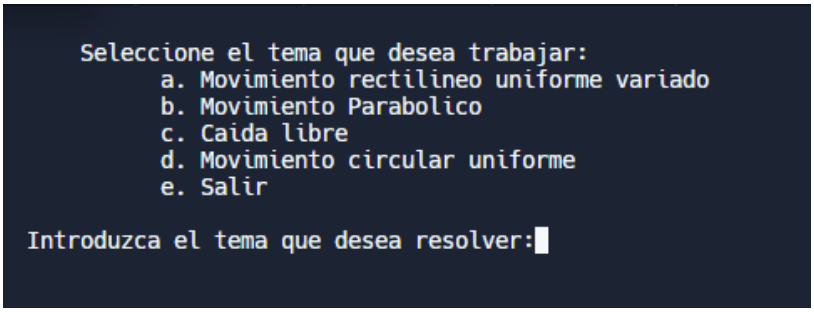

Fig 1. Initial message of the program to solve classical physics exercises

If we choose each of the options it will be displayed as shown in Fig. 2.

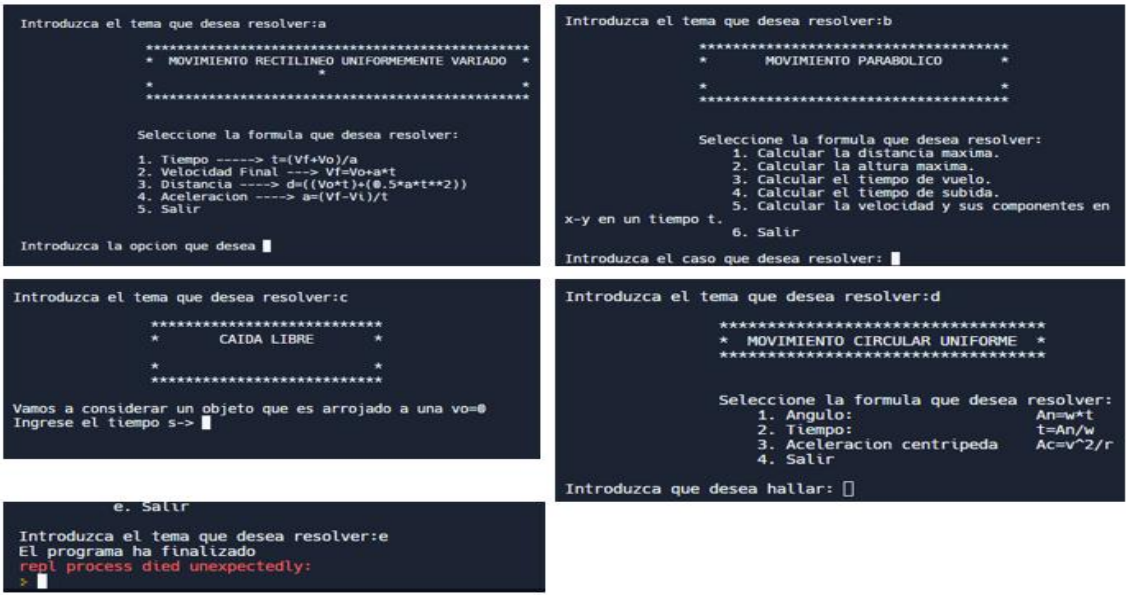

Fig. 2: Output of options a, b, c, d, e, from the main menu of the program.

It was verified that each of the formulas give the result and as an example is placed in Fig. 3, the output of the result for the distance in the uniformly varied rectilinear movement.

```
**************************************************
*  MOVIMIENTO RECTILINEO UNIFORMEMENTE VARIADO  *
                      *
*                                                *
**************************************************


Seleccione la formula que desea resolver:

1. Tiempo -----> t=(Vf+Vo)/a
2. Velocidad Final ---> Vf=Vo+a*t
3. Distancia ----> d=((Vo*t)+(0.5*a*t**2))
4. Aceleracion ----> a=(Vf-Vi)/t
5. Salir


Introduzca la opcion que desea 3
Introduzca la velocidad inicial en (m/s) : 20
Introduzca la aceleración en (m/s^2) : 5
Introduzca el tiempo en (s) : 15
LA DISTANCIA ES : d =  862.5  m
```

Fig. 3: Calculation of uniformly varied rectilinear distance

To corroborate the graph of the code that shows position-time, velocity-time and acceleration-time, the data of the movement of a gazelle that when traveling 2 meters reaches an initial speed of 4 m / s and after 8 seconds has an acceleration of 0.75 m / s^2 was considered. Fig. 4 shows how this movement is represented in the graph.

```
PS C:\Users\alexe\OneDrive\Escritorio>
PS C:\Users\alexe\OneDrive\Escritorio>  c:; cd 'c:\Users\alexe\OneDrive\Escritorio'; & 'C:\Use
rs\alexe\AppData\Local\Programs\Python\Python311\python.exe' 'c:\Users\alexe\.vscode\extension
s\ms-python.python-2022.20.2\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '6
0364' '--' 'c:\Users\alexe\OneDrive\Escritorio\grafica.py'
Ingrese velocidad inicial ==> 4
Ingrese la posición inicial ==> 2
Ingrese la aceleración ==> 0.75
```

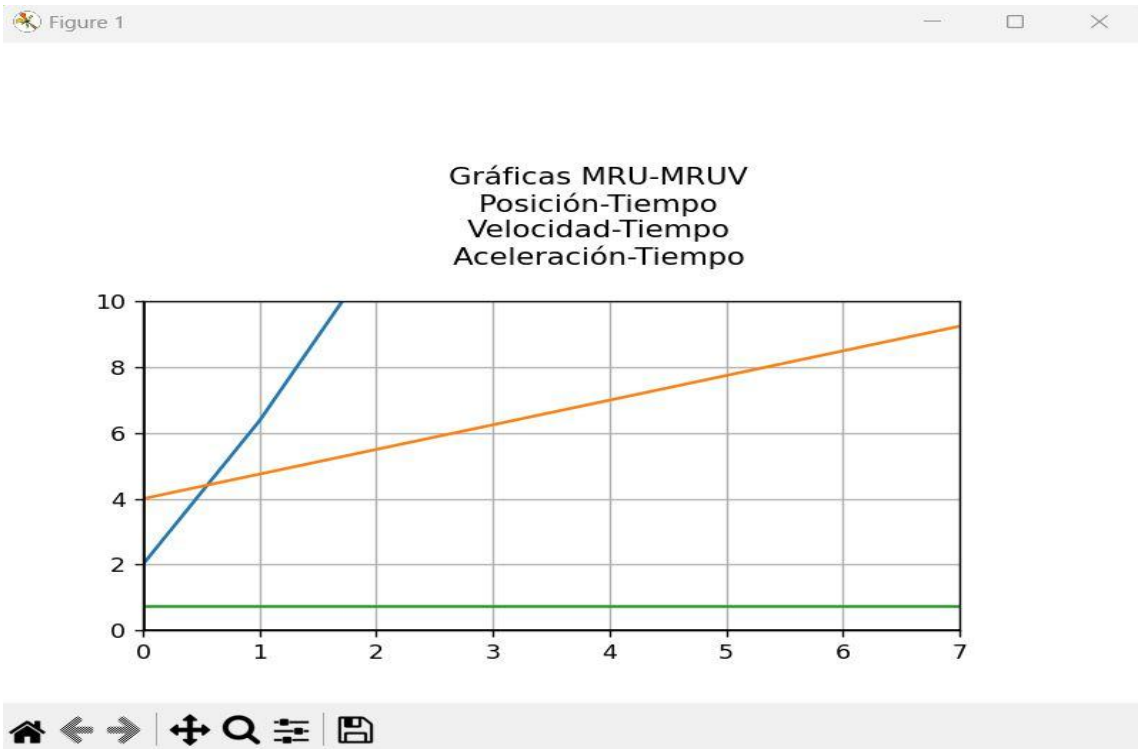

Fig. 4: Position-time blue line, velocity-time orange line, acceleration-time green line, representing the movement of a gazelle in 8 seconds.

CONCLUSIONS

The importance of consolidating the concepts of classical physics and programming in Python for the teaching of higher education as a useful study tool was verified. By knowing programming concepts you can use Python to learn and practice exercises in a didactic way uniform rectilinear movement, parabolic movement, free fall and uniform circular movement.

The behavior in time of position, velocity, and acceleration was graphed, which could be seen in Fig. 4 which represented the movement of a gazelle.

It was verified that each of the options of the main menu of the submenu that runs when choosing the options of the a-e, ran without errors and the data can be entered according to the exercise according to the user's need.

REFERENCES

[1] Atoeva Mehriniso Farhodovna, A. J. (2020). Innovative Pedogogical Technologies For Training The Course Of Physics. *The American Journal of Interdisciplinary Innovations and Research*, 82–91. doi:https://doi.org/10.37547/tajiir/Volume02Issue12-12

[2] Ayars. (2013). *Computational Physics with Python.* California State: Ayars E.

[3] Bogusevschi, D. M. (2020). Teaching and Learning Physics using 3D Virtual Learning Environment: A Case Study of Combined Virtual Reality and Virtual Laboratory in Secondary School. *Journal of Computers in Mathematics and Science Teaching, 39*(1), 5-18. Obtenido de https://www.learntechlib.org/primary/p/210965/.

[4] Borcherds, P. (2007). Python: a language for computational physics. *Elsiever, 177, Issues 1–2*, 199-201. doi:https://doi.org/10.1016/j.cpc.2007.02.019.

[5] Esther Cascarosa, C. S.-A. (2021). Model-based teaching of physics in higher education: a review of educational strategies and cognitive improvements. *Journal of Applied Research in Higher Education, 13*(1), 33-47. doi:https://doi.org/10.1108/JARHE-11-2019-0287

[6] Giancoli. (2008). *Physics for Science and Engineering* (4th ed., Vol. 1). Mexico: Pearson.

[7] Giancoli, D. (2014). *Physics principles with applications* (Vol. 1). Mexico: Pearson.

[8] Gisin, F. D. (2019). Physics without determinism: Alternative interpretations of classical physics. *Advancing Physics, 100*, 062-107. doi:https://doi.org/10.1103/PhysRevA.100.062107

[9] Resnick, H. a. (2007). *Fundamentals of Physics* (8 ed.). Cleveland State: Wiley.

[10] Sears, Z. (2009). *University Physics* (12 ed., Vol. 1). Mexico: Pearson.

[11] Serway, J. (2004). *Physics for Scientist and Engineers* (6 ed.). California State: Thomson Brooks.

[12] Tetiana Goncharenko, N. Y.-C. (2021). Experience in the Use of Mobile Technologies as a. *Kherson State University*. Obtenido de https://lib.iitta.gov.ua/727258/1/20201298.pdf