

## Physics in Python Course at Zagreb University of Applied Sciences

Alemka Knapp<sup>\*1</sup>, Dubravko Horvat<sup>2</sup>, Diana Šaponja-Milutinović<sup>3</sup>, Domagoj Kuć<sup>4</sup>

<sup>1</sup>Zagreb University of Applied Sciences, 10000 Zagreb, Croatia, <https://orcid.org/0009-0008-3631-6222>

<sup>2</sup>Department of Physics, Faculty of Electrical Engineering and Computing, 10000 Zagreb, Croatia, <https://orcid.org/0009-0003-2104-4010>

<sup>3</sup>Zagreb University of Applied Sciences, 10000 Zagreb, Croatia, <https://orcid.org/0009-0001-7578-2997>

<sup>4</sup>Zagreb University of Applied Sciences, 10000 Zagreb, Croatia, <https://orcid.org/0000-0001-6399-7600>

**ABSTRACT:** Python programming language is an increasingly widespread tool used for solving general computational problems in a variety of applications. In 2018, at the Zagreb University of Applied Sciences in Croatia, we introduced laboratory-computer exercises to a physics course in which Python was used to solve problems and to process the results of measurements. Students of Information Technology who attended our course had already completed a course in Programming basics in the previous semester. Our objective was to encourage IT students to apply the programming skills they had acquired to solve problems in other subjects. Before introducing Python, we used to solve problems by explaining the physical framework of the problem, making a sketch, writing down formulae and solving them on the blackboard. With the new approach, we additionally analyze problems using corresponding Python programs which were designed to provide further insights into the problem through graphical representation and additional visualization.

This proved to be particularly beneficial during the 2020 lockdown, when we transitioned to online classes.

The purpose of this paper is to demonstrate several examples of how we solved problems using the Python programming language and to analyse the reactions of students to this new way of teaching physics.

**KEYWORDS:** physics course, Python, education.

---

### INTRODUCTION

The curriculum of students majoring in Information Technology at the Zagreb University of Applied Sciences in the first year of undergraduate study includes lectures in physics as well as auditory and laboratory exercises. The exercises were conducted using a classic approach where tasks were solved on the blackboard and simple experiments were carried out and measurements were numerically analysed.

Some IT students occasionally expressed the view that physics was irrelevant to their studies and that they would not need to know physics for their future jobs. Our objective was to enhance the integration of physics content with other subjects on the IT curriculum, with the aim of fostering greater interest and enthusiasm for physics among students. Consequently, in 2018, we implemented a new approach to our curriculum, integrating programming into our auditory and laboratory exercises to enhance the relevance and appeal of the course to IT students. Students in the first year of IT, who attended our course in the second semester, learned the basics of programming in Python in the first semester as part of the Programming basics course (mostly text oriented). Also, we wanted to explore if using Python in physics would help students to better understand and visualise the problems we were working on.

Our new approach included, in addition to solving problem in a usual manner, analysing the problem using the corresponding Python program. When writing the program for each problem, we attempted to provide a more comprehensive solution than the standard one, incorporating additional visualisation and insight into the problem.

### OTHER IMPLEMENTATIONS OF PYTHON IN THE PHYSICS COURSES

Although we have prepared our Physics with Python course independently from other sources, we have found that different types of physics courses using Python have been introduced at other universities; we will mention some of those which we found the most similar to ours.

Bäcker (2007) described his experience in establishing a course using Python as a programming language for physics students in 2002 in Dresden, Germany with the aim to enable the students to solve problems in physics with the help of numerical computations. He found that the experiment of using Python for teaching computational physics was very successful and that some students continued to use Python for various tasks after they finished the course.

## Physics in Python Course at Zagreb University of Applied Sciences

Malthe-Sorensen developed a textbook for the course "Introduction to Mechanics" at the University of Oslo's Department of Physics, Norway using Python for solving physical problems. In the course are used both analytical and numerical methods with the aim to teach students to think like a physicist (Malthe-Sorensen, 2016). He believes that the most effective way to learn physics is to apply theoretical concepts to practical examples and exercises. Short Python programs to solve numerical problems are a part of the book.

O'Donnell, Rasmussen and Rodriguez (2021) from Utah State University, USA claim that computer-based curriculum for undergraduate physics students at their university before 2019 struggled to stay current and applicable. They describe how they created a Python toolkit for Computer Methods course. After employing the Python toolkit student knowledge on coding and coding physics significantly improved.

Mandanici, Mandaglio, Conti Nibali and Fiumara (2022) from the University of Messina, Italy assume that skills in computer science can have great value in studying, doing and communicating physics. They wanted to offer a new approach to physics, so they developed study materials for an introductory course in physics using Python with open source software. These materials were in the form of workbooks for courses in general physics for majors in Computer Science, Information Technology, and Data Science. Tufino, Oss and Alemani (2024) assert that introducing computation into high school or undergraduate courses has recently become an important part of physics education research. They described how they incorporated data analysis in Python in a first-year laboratory course for physics major students at the University of Potsdam, Germany.

Pols and Dekkers (2024) describe how they redesigned a first year physics laboratory course at the Delft University of Technology, Netherlands. They state that introducing students to experimental physics research is an important part of an undergraduate physics education. Programming and data and error analysis in Python is the first introductory part of their course (first 8 weeks) before starting with experiments.

Our Physics in Python course was introduced for students who are not pursuing a major in physics. These student study physics as a general education subject in one semester only and have less interest and prior knowledge of the subject. Accordingly, our examples are typically simpler and at a more basic level than those mentioned above. We hope that our examples will be useful to lecturers who teach physics at a similar level at universities, as well as to those in secondary schools with STEM orientation.

### WHY WE CHOSE PYTHON

In the field of computer programming, the Python language is becoming increasingly prevalent, both in terms of popularity and frequency of use (Tadlaoui & Chekou, 2021).

According to the *Tiobe index* (2024) Python is among the three most popular programming languages in recent years (along with Java, C and C++). The Tiobe index has been monitoring the popularity of programming languages for over twenty years, it uses popular search engines to determine which languages are most used among professional programmers, programming courses and third-party vendors. As of November 2024 Python is on the top of the list (as well as during the last couple of months). Python is ranked the top programming language both in the 2024 *IEEE Spectrum annual ranking* (2024) of the top programming languages, and in the *PYPL* (Popularity of Programming Language)(2024) index since November 2018 till November 2024.

Among other things, Python owes its popularity to its simple, clear syntax and to the fact that the basics of the language can be mastered quickly, which is why it also has great pedagogical value (Budin, Brođanac, Markučić & Perić, 2015 ). One of the advantages of Python (perhaps the most significant) compared to other popular programming languages, is that it has, in addition to its standard libraries, a large number of libraries for many areas of application (Stojanović & Kovačević, 2022).

Our reasons for selecting Python as a programming language were similar to those presented by Bäckér (2007). These included: Python is an accessible language that is straightforward to learn and intuitive, even for those with no prior programming experience. Its compact code allows for a relatively short development time, which frees up time to concentrate on the physics problem at hand. It is widely used by scientists, and it is freely available. Additionally, it comes with a plethora of libraries that enable programmers to build a wide range of applications. Its interactive plotting capabilities are also a valuable asset. Our students had some prior knowledge from the Programming basics course, which contributed to their ability to grasp the material.

### PYTHON LIBRARIES NUMPY AND MATPLOTLIB

For the needs of our classes, we upgraded standard Python, the basics of which students learned in the previous semester, with NumPy and Matplotlib libraries. Both are open-source softwares. These have been particularly useful for application in physics classes as they give us simpler code for easier calculating of numerical tasks and the possibility of graphical presentation of different physical quantities.

NumPy is a Python library used for working with arrays and matrices. NumPy provides efficient data structures and high-level mathematical functions that operate on these arrays and matrices. Matplotlib is a plotting library for Python and its numerical mathematics extension NumPy. We used Matplotlib to make simple plots, such as line graphs and scatter plots. We cover basic of these two libraries (which students do not learn in Programming basics course) in the first two weeks of the course using simple examples, before we start dealing with physical problems.

## Physics in Python Course at Zagreb University of Applied Sciences

Let's mention some characteristics of these libraries that we use in our examples in case that reader is not familiar with them (Pine, 2019; Idris, 2015; Ayars, 2013 ). *List* and *array* are data structures used in Python to store ordered multiple elements. A list (a data structure used in general Python) can hold various data type, whilst array (a data structure from NumPy) contains elements of the same type. One of the important differences between them (that makes array ideal for mathematical operations on a large set of data) is that mathematical operations are performed at once on all elements of the array, which simplifies the code and avoids the use of loops. List does not have this property.

We can first create a list by entering values, separated by commas between square brackets and then convert it to an array using the *np.array* command from the NumPy library: by command (in this manuscript, the commands will be indicated by the symbol '>')

```
> L = [1, 2, 3] we define a list
```

```
> A = np.array(L) converts list L into array A = [1 2 3].
```

A simple mathematical operation, such as multiplying by a number, will give disparate results when applied to lists and arrays.

Using command:

```
> B = 2 * A we get another array B, all elements of which are multiplied by 2: B = [2 4 6].
```

Unlike an array, if we multiply list by 2:

```
> B = 2 * L we get duplicate list: B = [1, 2, 3, 1, 2, 3].
```

We used commands from the Python library matplotlib to create graphs. One coordinate of the point we draw on the graph can be defined as element of an array using command:

```
> X = np.linspace(start, stop, n), where n is the number of array elements which will be equally spaced in the interval [start, stop], and the other one through its functional dependence on the first coordinate:
```

```
> Y = f(X), (function f applied to each element of array X)
```

Using command *plt.plot(X, Y)*, we set the graph of the function, which takes *x*-coordinates of its points from the array *X* and the *y*-coordinates from the array *Y*.

### METHODOLOGY EMPLOYED IN THE PYTHON PHYSICS EXERCISES

Our Physics with Python exercises were a combination of solving numerical physical problems, performing simple experiments and writing and analyzing Python programs. In addressing the problems, we would first explain the physical framework of the problem, make a graphical sketch, print formulae, solve it on the blackboard and then we would analyze the corresponding Python program. Some of the exercises from the pre-Python course were retained, while new exercises were also included, which were more suitable for analysis with Python. When writing the program for each problem, we tried to get "something more" comparing to the standard solution of the problem with graphical representation, additional visualization and insight into the problem. Following the resolution of a certain type of problem by the teacher, students were typically tasked with either developing a program for a similar type of problem or adapting the program from the previous example to address a new problem. During the semester, the students also had several laboratory exercises (measuring the dimensions of small bodies with calipers and micrometers, measuring the period of oscillation of a mass on a spring, mathematical pendulum, torsional pendulum etc.) They processed the results of these measurements in a Python program (eg. storing data and reading the stored data for later processing, calculating mean values, standard deviation, drawing graphs and performing linear regression analysis). These laboratory exercises were largely derived from our preexisting laboratory course, supplemented with computer data processing.

The Zagreb University of Applied Sciences uses a customized learning management system (LMS) based on Moodle. All materials and programs were available to the students on the LMS during the class (and later), and the students, in addition to the projection on the screen, could follow the program on their computer; start it, try it out, and change the default parameters. We also prepared a textbook for this course with an introduction to Python libraries NumPy and Matplotlib, as well as a description and instructions for performing laboratory exercises (Horvat, Šaponja-Milutinović & Knapp, 2022).

The following two examples show in detail how we added Python processing to the usual physics problems.

#### Example 1

The first example (that we adjusted from our pre-Python course) concerns the analysis of projectile motion and free fall using the popular example of a zookeeper shooting a monkey with a dart. (Young & Freedman, 2016)(pg. 84-85).

A monkey escapes from the Zoo and climbs a tree 15 metres high. The zookeeper finds him and after unsuccessfully trying to convince him to come down, he fires a dart at him from a distance of 25 metres from the tree. Trying to escape the monkey lets go at the instant the dart leaves the gun and starts to fall. At what speed and at what angle does the zookeeper have to shoot the dart to hit the monkey?

We would first explain the physical framework of the problem in order to avoid that students have to focus on physics and programming code at the same time.

The position of the arrow (projectile motion) can be described by formulae:

$$x_1 = v_{0x}t = v_0 \cos \alpha t \quad (1)$$

$$y_1 = v_{0y}t - \frac{gt^2}{2} = v_0 \sin \alpha t - \frac{gt^2}{2} \quad (2)$$

The position of the monkey (free fall) can be described by formulae:

$$x_2 = d \quad (3)$$

$$y_2 = h - \frac{gt^2}{2} \quad (4)$$

At the moment when the dart hits the monkey their coordinates are equal:

$$x_1 = x_2 \quad y_1 = y_2 \quad (5)$$

$$v_0 \cos \alpha t = d \quad (6)$$

$$v_0 \sin \alpha t - \frac{gt^2}{2} = h - \frac{gt^2}{2} \quad (7)$$

We can express the time from the equation (6) and include it in (7):

From this equation we get:

$$\operatorname{tg} \alpha = \frac{h}{d} \quad (8)$$

and the angle at which the zookeeper should shoot the monkey  $\alpha = 31^\circ$  (in this paper, we present the specific numerical solutions of the problems that the reader can more easily compare the calculation results and the graphs produced by Python programs).

We can notice that both equations are satisfied for any speed  $v_0$ , which means that the monkey will be hit if the dart is fired at the given angle, regardless of its velocity. The speed at which the dart is fired must only be large enough so that the dart does not hit the ground before it reaches the tree where the monkey is. So, the initial speed must be large enough for the range of the shot  $D$  to be at least equal to the distance  $d$  from the zookeeper to the tree.

$$D = \frac{v_0^2 \sin(2\alpha)}{g} \geq d \quad (9)$$

$$v_0 \geq \frac{dg}{\sin(2\alpha)} = 16.7 \frac{\text{m}}{\text{s}} \quad (10)$$

We can draw the path of the dart and the monkey for several different initial dart speeds using a Python program. Python code for this example and the graph produced by it are shown in Figure 1 and Figure 2.

```

File Edit Format Run Options Window Help

import numpy as np
import matplotlib.pyplot as plt

h = 15.0    # m
d = 25.0    # m
g = 9.81    # m/s**2

# calculus
alfa = np.arctan(h/d)
alfa_dg = alfa*360/(2*np.pi)
print('alfa = {0:.3g} degrees'.format(alfa_dg))
print('Zookeeper will hit the monkey at the angle alpha'
      'with any initial velocity v0>= v0_min.')
v0_min = np.sqrt(h*g/np.sin(2*alfa))
print('v0_min = {1:.3g} m/s'.format(v0_min))

# graph
v0=[12.7, 16.7, 20.7, 24.7]

for i in range(0, 4):
    ts = d/(v0[i]*np.cos(alfa))
    t = np.linspace(0.0, ts, 50)
    x1 = v0[i]*np.cos(alfa)*t
    y1 = v0[i]*np.sin(alfa)*t - g*t**2/2
    x2 = d*np.ones(50)
    y2 = h - g*t**2/2
    plt.plot(x2+0.2*(i-1),y2,x1,y1)

x3 = [0,d]
y3 = [0,h]
plt.plot(x3,y3,'g--')
plt.xlabel('x / m')
plt.ylabel('y / m')
plt.ylim(0, 15+1)
plt.xlim(0, 25+1)
plt.legend(['monkey 1','dart 12.7 m/s','monkey 2',
            'dart 16.7 m/s','monkey 3','dart 20.7 m/s',
            'monkey 4','dart 24.7 m/s'])

plt.show()

```

Figure 1: Python code for Example 1 - a monkey and a dart

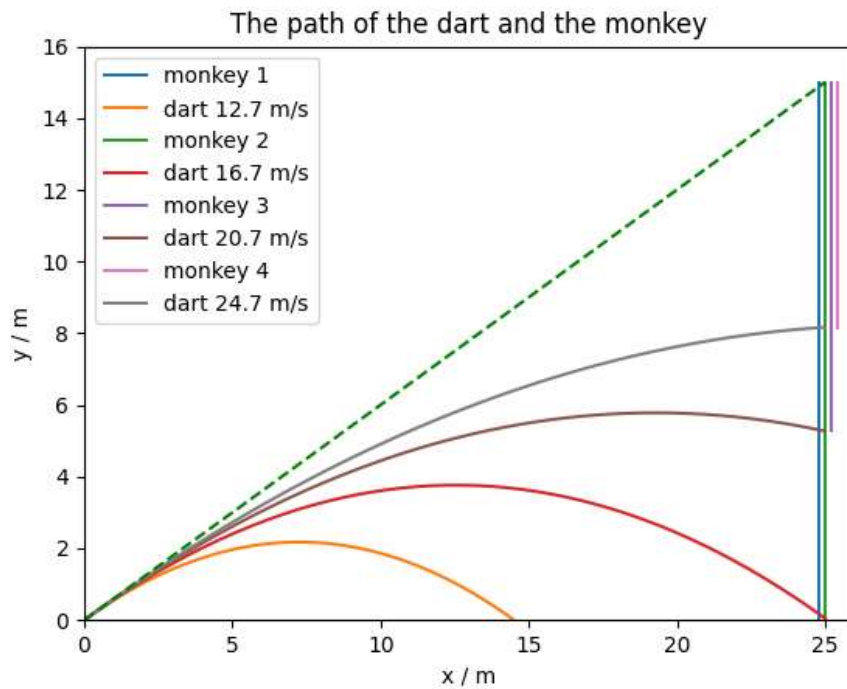


Figure 2: The path of the dart and the monkey for different initial dart speeds

The graph shows the motion of the dart and the monkey for four different initial speeds of the dart. We defined a list  $v_0$  which contains these speeds. With each pass through the loop, we define a graph (the coordinates of the dart and the monkey for 50 time instants) that shows the motion for a particular initial speed.

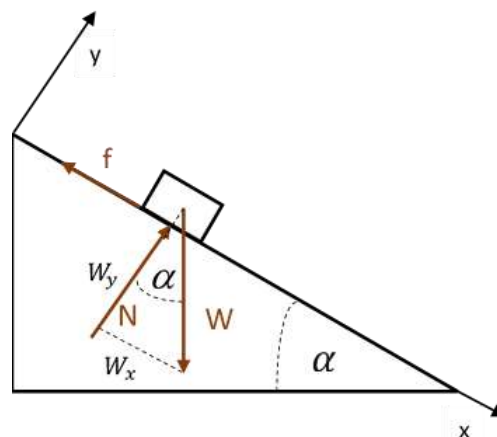
We spaced a little the vertical lines that show different monkey's paths that should all be at  $x = 25$  m so that they don't cover each other. In the original version of this program (the program was slightly adapted for the purposes of this text), the graph for the motion of the dart and the monkey was displayed for only one initial speed, which the students varied themselves to observe how it affects the motion and the position at which the hit will occur.

### Example 2

In this example we consider static friction, a concept that in our experience, some students do not understand properly (is there a frictional force when the body is at rest?).

What is the maximum inclination of the slope for which the body is still at rest on the slope? The coefficient of friction  $\mu = 0.4$ .

Figure 3: Forces on a body at rest on a slope



When the body is at rest on the surface (Figure 3), the frictional force (static friction) can have a value from 0 to  $\mu N$ . The frictional force is equal to the force that would initiate the motion of the body. While the body is at rest on the slope, the frictional force cancels the weight component acting down the slope  $W_x$ . As we increase the angle (of inclination) of the slope,  $W_x$  increases, and so does the frictional force until it reaches the value  $\mu N$ . For larger angles the body starts to slide down the slope and  $f = \mu N$  will be valid.



## Physics in Python Course at Zagreb University of Applied Sciences

We can write Newton's second law for the largest angle for which the body is still at rest on the slope (the total force is zero,  $f = \mu N$ ):

$$\vec{F}_{tot} = \sum \vec{F}_i = m\vec{a} = 0 \quad (11)$$

By splitting the equation (11) into components in the direction of the  $x$  and  $y$  axis, we get:

$$x: W\sin\alpha - f = 0 \quad (12)$$

$$y: -W\cos\alpha + N = 0 \quad (13)$$

From these equations we can express  $f$  and  $N$ :

$$\mu = \frac{f}{N} = \frac{G\sin\alpha}{G\cos\alpha} = \tan\alpha = 0,4 \quad (14)$$
$$\alpha = 21,8^\circ$$

```
File Edit Format Run Options Window Help

import numpy as np
import matplotlib.pyplot as plt

g = 9.81          #m/s**2
mi = 0.4
m = 2.00          #kg

alpha = np.arctan(mi)*360/(2*np.pi)
print('alpha = {0:.1f} st.'.format(alpha))

#graphs
angle = np.linspace(0,90,91)
Wx = m*g*np.sin(np.radians(angle))
N = m*g*np.cos(np.radians(angle))

plt.plot(angle,Wx,angle,N,angle,mi*N)

angle1 = np.linspace(0,alpha,50)
f1 = m*g*np.sin(np.radians(angle1))
plt.plot(angle1,f1,"k--")
angle2 = np.linspace(alpha,90,50)
f2= mi*m*g*np.cos(np.radians(angle2))
plt.plot(angle2,f2,"k--")

plt.xlabel('angle / degrees')
plt.ylabel('F / N')
plt.xlim(0,90)
plt.ylim(0,20)
plt.title('Dependence of the weight components and'
|         'the frictional force on the slope angle')
plt.legend(["Wx", "Wy","mi*N", "f1,f2"] )
plt.show()
```

**Figure 4: Python code for Example 2 - a body at rest on a slope**

Using the Python program (Figure 4), we can graphically show how the  $x$  and  $y$  components of the body's weight  $W_x$  and  $W_y (= N)$ , the product  $\mu N$  and the frictional force  $f$  depend on the angle of the slope. This graph (Figure 5) can help to understand how the frictional force changes as a function of the slope angle when the body is at rest and when it starts sliding down the slope (in this example we have assumed that the static and dynamic coefficients of friction are equal). If we zoom in on the graph, we can read the angle  $\alpha$  more precisely (lower graph in Figure 5). We ask the students to compare the results of the graph with those obtained by calculation. Students can easily change the default parameters and see how it affects the result.

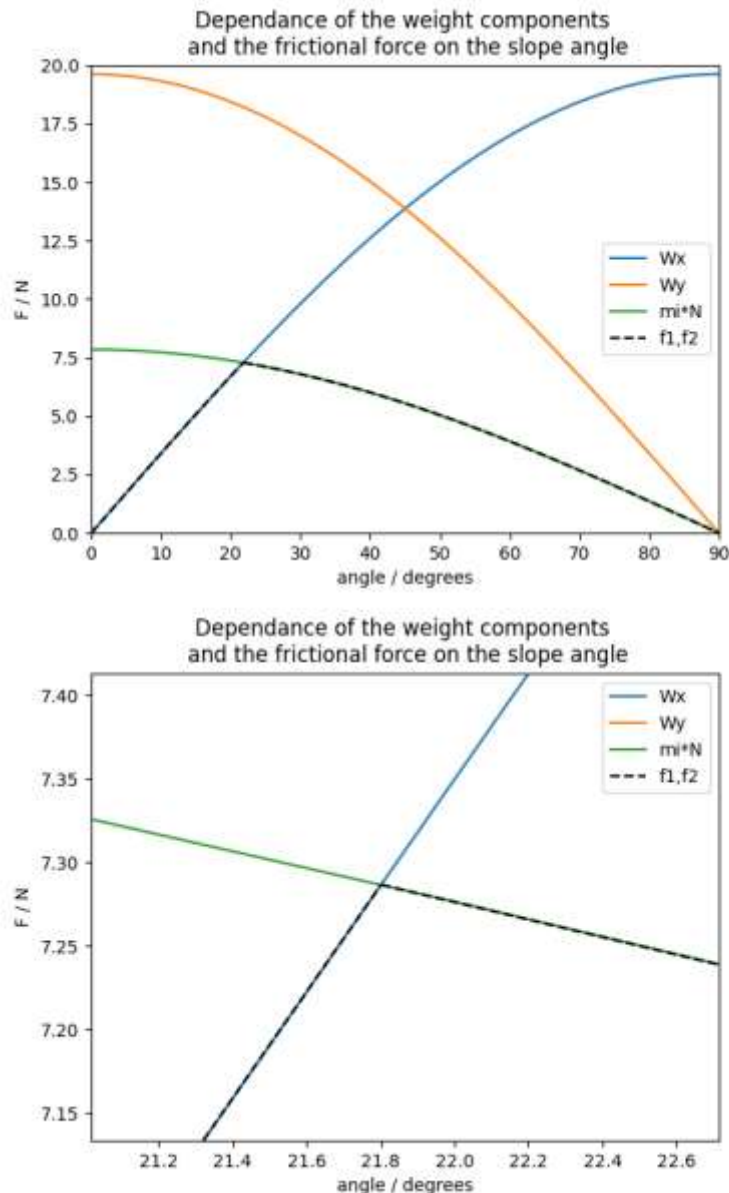


Figure 5: The body at rest on a slope: dependance of the components of weight and the frictional force on the slope angle

#### METHODS FOR THE EVALUATION OF ASSIGNMENTS AND EXAMINATIONS

During the semester, students were required to submit four homework assignments. These included short multiple-choice questions, the construction of graphs, the development of a Python program to address a specific physical problem, processing data from the measurements they performed in the laboratory part of these exercises (e.g. measuring the period of a mathematical pendulum in relation to the length of the thread and calculation of gravitational acceleration through linear regression). The final exam consisted of a series of multiple-choice questions and the creation of four Python programs to solve the given problems.

All homework assignments and examinations, both during the course of the semester and at its conclusion, were conducted via LMS. The tests and homework assignments were posted on the LMS using Moodle. Students accessed the examinations via computer and submitted their assignments on the LMS, where we also provided feedback and grades after a review of the assignments.

This kind of course organization which we started in 2018, required almost no additional adjustment when we switched to online classes in 2020 during the Covid lockdown. The course was practically ready for this way of teaching. The only thing we couldn't do online was performing experiments and measurements. These were replaced by recordings of laboratory exercises done live.

#### ANALYSIS OF STUDENTS' REACTIONS AND SUCCESS



## Physics in Python Course at Zagreb University of Applied Sciences

Students' reactions to the introduction of Python in physics varied from those who were enthusiastic and excited by the challenge of applying programming to solve problems in physics, to those who complained that introduction of Python made the physics course more difficult. The prior knowledge of our students regarding Python was also very varied: some were already experts in Python, some had mastered what they learned in the Programming basics course, and some had not yet passed that examination, so they struggled with the basics. We were faced with the challenge of how to choose the appropriate level of assignments for as many students as possible, especially during the first year or two. In terms of programming we kept our exercises on basic level.

In the *annual survey* for all courses conducted at the Zagreb University of Applied Sciences, students were presented with the questions listed in Table 1. They were asked to indicate their level of agreement with each statement on a scale of 1 to 5, with 1 representing the lowest level of agreement and 5 representing the highest. The number of students who participated in the survey and answered individual questions is slightly lower than the total number of students (which is listed later) because not all students participated in the survey and answered all questions.

**Table 1: Questions on students' survey conducted every year at TVZ**

1. The course was well organized.
2. My prior knowledge was sufficient to understand this subject.
3. The learning materials were of high quality.
4. The course objectives were achieved.
5. The pace of the lectures was appropriate.
6. The teaching was interesting and dynamic.
7. The atmosphere in the class was pleasant and stimulating.
8. Current trends and knowledge from the subject area are covered in the classes.
9. The content covered in the course will be useful for my future job.
10. Assessment at examinations and exercises provided useful feedback.
11. Assignments in lectures and exercises were of appropriate size and complexity.
12. Procedures at exercises were clear.
13. Assessment of your own level of interest in this subject.

Table 2 presents the mean scores for each of the questions posed to students over the period from 2016 to 2022. Additionally, it provides the mean scores for all questions combined, representing overall satisfaction, as well as the mean score for a single question across all years.

**Table 2: Ratings on student's survey 2016-2022**

	2016	2017	2018	2019	2020	2021	2022	average
1. course organisation	3.80	4.01	2.73	3.86	4.53	4.07	3.43	3.78
2. prior knowledge	3.31	3.72	3.49	4.12	4.02	3.90	3.55	3.73
3. learning materials	3.57	3.96	3.10	4.05	4.36	4.21	3.21	3.78
4. course objectives	3.57	4.00	2.82	4.16	4.44	4.21	3.22	3.77
5. pace of the lectures	3.51	3.87	3.32	4.19	4.46	4.12	3.39	3.84
6. interesting teaching	3.34	3.63	2.90	3.98	4.01	3.93	2.61	3.49
7. atmosphere	3.43	3.75	3.04	4.11	4.41	4.03	3.12	3.70
8. contemporary trends	3.51	3.86	3.08	4.17	4.39	4.07	3.36	3.78
9. usefulness for job	2.85	3.34	2.40	3.94	3.27	3.13	2.28	3.03
10. feedback	3.46	3.71	2.66	4.08	4.12	4.06	3.76	3.69
11. complexity of tasks	3.48	3.97	2.93	3.83	4.37	4.15	3.67	3.77
12. clarity on exercises	3.76	4.04	2.95	3.89	4.26	4.16	3.77	3.83
13. interest in subject	2.90	3.42	2.61	3.33	3.40	3.23	2.61	3.07
average	3.42	3.79	2.93	3.98	4.16	3.94	3.23	3.64

Here are some typical /more frequent comments from students on this survey: some were in favour and some were against the introduction of Python:

*"It was interesting in the physics labs, I really like that concept of physics in Python, I think it's a hit as an idea."*

*"Using Python as part of physics is a completely unnecessary complication and makes it difficult to master the subject. Knowledge of programming is required at a level that we did not reach in the previous semester, and I believe that physics is complex enough in itself and does not need additional complications. As far as I'm concerned, Python is redundant, physics is better in the classic form of exercises that follow the lessons, not drawing graphs in Python."*

## Physics in Python Course at Zagreb University of Applied Sciences

Table 3 presents the grades achieved by students in the physics examinations conducted between 2016 and 2022. The grades are on a scale of 1 to 5, with 5 representing the highest level of achievement and 1 indicating a grade insufficient for passing the examination. The data presented herewith refer to all examinations conducted in a given academic year. In 2016 and 2017, standard exercises were held; in 2018 and 2019, exercises incorporating the Python programming language were introduced; in 2020 and 2021, due to the global pandemic of 2020, all Python exercises were conducted online; and in 2022, live Python exercises were held once again.

**Table 3: Students' final grades at physics examination 2016-2022 (the number of students who achieved a certain in a particular year year)**

	classic exercises		Python exercises				
final grade	2016	2017	2018	2019	2020	2021	2022
5	24	19	7	5	33	16	2
4	28	33	18	18	51	63	19
3	53	35	50	45	55	75	60
2	36	40	65	57	39	28	60
total	141	127	140	125	178	182	141
enrolled	146	154	195	166	201	209	191
passing rate	96.6%	82.5%	71.8%	75.3%	88.6%	87.1%	73.8%
average grade	3.28	3.24	2.76	2.77	3.44	3.37	2.74

**Table 4: Student's grades at physics examination - summarized table for classic exercises and Python exercises**

final grade	classic (2016-2017)		Python (2018-2022)	
5	43	14.3%	63	6.5%
4	61	20.3%	169	17.6%
3	88	29.3%	285	29.6%
2	76	25.3%	249	25.9%
total	268	89.3%	766	79.6%
enrolled	300	100.0%	962	100.0%
passing rate	89.3%		79.6%	
average grade	3.26		3.06	

As Tables 3 and 4 illustrate, the average grade and passing rate at examinations of students showed a slight decline with the introduction of Python exercises, especially in the initial two years. Apart from that, students' satisfaction with the course (Table 2) did not decrease (except for the first year when the course was in its experimental phase): mean value of average score on survey before Python is 3.61 and 3.64 with Python. We do not have data on the results of the survey for individual students, but only averages, which are given in Table 2.

The result of Student's t-test comparing two samples (student's grades from Table 4, before Python and with Python) is  $t = 2.84$ ,  $p = .005$ . The grades of students who took the examination in the classic way ( $M = 3.26$ ,  $SD = 1.04$ ) are statistically significantly better than those who took the Python examination ( $M = 3.06$ ,  $SD = 0.93$ ).

The result of  $\chi^2$  - test comparing these two samples is:  $\chi^2 = 14.56$ ,  $p < .001$ , shows that the difference in grades is statistically significant.

Pearson coefficient of correlation between the average grade achieved on examinations from 2016 to 2022 (Table 3) and the overall satisfaction with the course (average grade from survey, Table 2) is  $r = 0.61$ , indicating a moderate positive correlation.

## CONCLUSION

A comparison of the average grades on the examinations before and after the introduction of Python, does not yield evidence that the introduction of the later resulted in improved outcomes. Nevertheless, we do not deem this comparison to be entirely pertinent for the assessment of students' actual knowledge, given that the exams held prior to the introduction of Python and those conducted subsequently were disparate (however, we still intend to perform some statistical analysis). The new examinations required students to demonstrate a greater level of proficiency than the previous ones. In addition to understanding the physical meaning of the tasks, students were also required to be able to write a Python script to produce the required results and graphs. This was a more demanding task and presented a greater challenge for students than the previous examinations which tested only their knowledge of physics. With regard to the students' satisfaction with the course; it is notable that while the course was well received by some students, others found it too challenging and unnecessary for their studies. Table 2 illustrates, that the lowest grades are consistently awarded for questions related to students' interest in physics and the perceived usefulness of the subject for their future job.

## Physics in Python Course at Zagreb University of Applied Sciences

The Informatics Technology programme offers three distinct majors: 1) Office Organization and Informatization, 2) Electronic Business and 3) IT Design, for which physics could be considered less important than for some other majors at TVZ. We have also noticed from working with these students that, on average, they were less interested in physics and they came with less prior knowledge than other students at our university. That is the reason that we now consider the course to be more suitable and likely to be more successful for some other majors.

In the meantime, in accordance with the decision of our authorities, physics is no longer an obligatory course for IT students (which is why the year 2022 is the last one in our tables). The question of whether physics is a necessary component of IT studies, which is classified under social studies, has been discussed on several occasions at our institution.

We are currently developing the introduction of Physics in Python as an optional course for all TVZ majors (Electrical Engineering, Computing Engineering, Information Technology, Mechanical Engineering, Mechatronics, Civil Engineering) in their later/advanced semesters. The course will be available to students for whom physics is a compulsory course as well as to those for whom it is optional. It is anticipated that this new arrangement will engage students with a greater interest in physics, and thus slightly more demanding tasks will be introduced, both in terms of physics and in terms of programming. In addition to the existing textbook, study materials are being developed which will include all the exercises and Python programs used in the current course, with the addition of some more advanced problems.

## REFERENCES

- 1) Ayars, E. (2013). Computational Physics With Python. California State University.
- 2) Bäcker, A. (2007). Computational Physics Education with Python. *Computing in Science & Engineering*. 9(3). 30-33. DOI:10.1109/MCSE.2007.48
- 3) Budin, L., Brođanac, P., Markučić, Z. & Perić, S. (2015). Rješavanje problema programiranjem u Pythonu (Problem Solving with Python Programming). Element.
- 4) Horvat, D., Šaponja-Milutinović, D. & Knapp, A. (2022). Laboratorijsko računalni praktikum iz fizike (Laboratory Computer Practicum in Physics). Zagreb University of Applied Sciences, ISBN 978-953-8444-03-6.
- 5) Idris, I. (2015). NumPy Beginners' Guide. Packt publishing.
- 6) IEEE Spectrum annual ranking 2023. Retrieved November 30,2024, from <https://spectrum.ieee.org/the-top-programming-languages-2023>
- 7) Malthe-Sorensen, A. (2016). Elementary Mechanics using Python. Springer.
- 8) ancini, A., Mandaglio, G., Pirrotta, G., Nibali, V.C. & Fiumara, G. (2022). Simple Physics With Python: A Workbook on Introductory Physics With Open-Source Software. *Computing in Science & Engineering*, 24 (2). 1-5. DOI: 10.1109/MCSE.2022.3160011.
- 9) Monahan J.F. (2011). Numerical Methods of Statistics. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press.
- 10) O'Donnell, E. & Rasmussen, M. (2021). Improving Skills in Computer Methods: Introductory Toolkit to Python for Undergraduate Physics Majors. *Physics Capstone Projects*. Paper 94.
- 11) Pine, D. J. (2019). Introduction to Python for Science and Engineering. CRC Press.
- 12) Pols, C.F.J & Dekkers, P.J.J.M. (2024). Redesigning a first year physics lab course on the basis of the procedural and conceptual knowledge in science model. *Physical Review Physics Education Research*. 20(1). 010117. DOI:10.1103/PhysRevPhysEducRes.20.010117
- 13) PYPL indeks. Retrieved November 30,2024 from <https://pypl.github.io/PYPL.html>
- 14) Stojanović, A. & Kovačević, Ž. (2022). Uvod u programski jezik Python (Introduction to Python Programming Language). Zagreb University of Applied Sciences, ISBN 978-953-7048-99-0.
- 15) Tadlaoui, M. A. & Chekou, M. (2021). A blended learning approach for teaching python programming language: towards a post pandemic pedagogy. *International Journal of Advanced Computer Research*. 11(52), 13-22. DOI:10.19101/IJACR.2020.1048120
- 16) Tiobe index. Retrieved November 30,2024 from <https://www.techrepublic.com/article/tiobe-index-language-rankings/>
- 17) Tufino, E., Oss, S. & Alemani, M. (2024). Integrating Python data analysis in an existing introductory laboratory course. *European Journal of Physics*. 45(4). 045707 DOI 10.1088/1361-6404/ad4fcc.
- 18) Young, H.D & Freedman, R.A. (2016). University Physics with modern Physics. Pearson Education Limited.