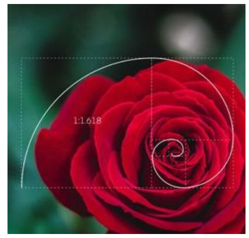
<table>
<tr><th colspan="3">Topics plan</th></tr>
<tr><td>Partner organization</td><td colspan="2">University of Novi Sad</td></tr>
<tr><td>Course</td><td colspan="2">Programming 1</td></tr>
<tr><td>Lesson title</td><td colspan="2">Combinatorics: Splitting the numbers into sum, variations with repetitions, permutations</td></tr>
<tr><td>Learning objectives</td><td>• Students will understand the methods of splitting the number into the sum of the k numbers, for given $k \in \mathbb{N}$.
• Students will understand how to generate all the variations of the given set with repetitions.
• Students will understand how to generate all the permutations of the given set.
• Students will understand how to implement the methods in Python programming language.
• Students will understand how to apply the algorithms in solving similar combinatorial problems.</td><td rowspan="4">Methodology
☐Modeling
X Collaborative learning
☐Project based learning
X Problem based learning

Strategies/Activities
☐Graphic Organizer
☐Think/Pair/Share
☐Discussion questions

Assessment for learning
X Observations
X Conversations
☐Work sample
☐Conference
☐Check list
☐Diagnostics

Assessment as learning
X Self-assessment
☐Peer-assessment
☐Presentation
☐Graphic Organizer
☐Homework</td></tr>
<tr><td>Aim of the lecture / Description of the practical problem</td><td>The aim of the lecture is to make students able to use Python in solving combinatorial problem, with visual solutions.

As a practical problem, the lecturer poses several questions related to the applications of combinatorial methods in real life situations.</td></tr>
<tr><td>Previous knowledge assumed:</td><td>• Elementary programming skills in Python.
• Basics of Combinatorics.</td></tr>
<tr><td>Lecture</td><td>In the introduction, we give basic examples of the usage of recursion, and the connection between recursive formula in mathematics, with application to natural problems in biology and how we solve it through computers (STEAM).
As the starting example, we use Fibonacci sequence, and relate it to the golden ratio, and give an example where it can be found.</td></tr>
</table>



We continue with recursive combinatorial problems that can be easily implemented in Python:

1. splitting the given number as the sum of non-negative integers, where we use the recursive description of the structure $x_0 + x_1 + \cdots + x_{k-1} = s$, where s and k are given, that for x_{k-1} we can use any of the numbers $0, 1, 2, \ldots, s$, and the rest forms the splitting of the number $s - x_{k-1}$ into $k - 1$ numbers.

   We write the following code in Python, after which we give some further examples, like not knowing the number of the summands in advance, but they have to be positive integers, and some others.

```
import numpy as np

def ispisi_sabirke(sabirci):
    print(sabirci[0], end="")
    for i in range(1, len(sabirci)):
        print("    +    {0}".format(sabirci[i]),
end="")
    print()


def razbij_u_zbir(sabirci, s, k):
    if k == 1:
        sabirci[0] = s
        ispisi_sabirke(sabirci)
    else:
        for i in range(s+1):
            sabirci[k - 1] = i
            razbij_u_zbir(sabirci,  s  -  i,  k  -
```

Assessment of learning

X Test

☐ Quiz

☐Presentation

☐Project

☐Published work

```
1)

k = 3
s = 4
sabirci = np.empty(k, int)
razbij_u_zbir(sabirci, s, k)
```

2. Next, we consider the variations with repetition of length k, of the given set of n elements, S_n, which is the ordered k-tuple of the elements of that set. For the set we take $\{0, 1, \dots, n-1\}$. To be able to generate all the variations with repetitions, we use the recursive description of the structure: the last element can be any element of the given set, and before this element we can put any variation with repetition of the set $\{0, 1, \dots, n-1\}$ of length $k-1$.
   In Python, the following code executes the aforementioned.

```
import numpy as np


def vsp(niz, n, k):
    if k == 0:
        print(niz)
    else:
        for i in range(n):
            niz[k - 1] = i
            vsp(niz, n, k - 1)


n = 2
k = 4
niz = np.empty(k, int)
vsp(niz, n, k)
```

3. Lastly, we consider all the permutations of the given set with n elements, S_n, is any $n-$tuple of different elements from that set. We start with the set $\{0, 1, \dots, n-1\}$. To be able to go through all the permutations, we use the recursive description of this structure: the first element can be any element of the given set, after which we can place any permutation of the remaining elements. In Python, this looks as follows.

```
import numpy as np
```

<table>
<tr><td></td><td>

```
def zameni(niz, a, b):
    niz[a], niz[b] = niz[b], niz[a]


def per(niz, n, m):
    if m == n:
        print(niz)
    else:
        for i in range(m, n):
            zameni(niz, m, i)
            per(niz, n, m + 1)
            zameni(niz, m, i)


n = 4
niz = np.arange(n)
per(niz, n, 0)
```

</td><td rowspan="3"></td></tr>
<tr><td>Action</td><td>The demonstration of power of Python in solving the combinatorial problems and visualization.</td></tr>
<tr><td>Materials / equipment / digital tools / software</td><td>Computer, electronic whiteboard, PyCharm software</td></tr>
<tr><td colspan="3">Reflections and next steps</td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="2">Reflections

The attendance was average due to the fact that we were working under certain epidemiological restrictions; the feedback was positive; the results of the tests show that the students have benefited from the materials and equipmebt used to deliver the lecture</td><td>Next steps

Since this approach was successfully implemented and was well received, the next steps include implementing other parts of the curriculum using the strategies devised for the pilot lecture.</td></tr>
<tr><td colspan="3">References</td></tr>
<tr><td colspan="3">In Appendix:
Photographs, Lists of students, Test, questionare</td></tr>
</table>