Co-funded by the Erasmus+ Programme of the European Union

Topics plan

Partner organization	University of Novi Sad	
Course	Programming 1	
Lesson title	Combinatorics: Variations without repetitions, combinations with and without repetitions	
Learning objectives	• Students will understand how to generate all the variations of the given set without repetitions. • Students will understand how to generate all the combinations with repetitions of the given set of numbers of given length $k \in \mathbb{N}$. • Students will understand how to generate all the combinations without repetitions of the given set of numbers of given length $k \in \mathbb{N}$. • 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.	**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
Aim of the lecture / Description of the practical problem	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.	
Previous knowledge assumed:	• Elementary programming skills in Python. • Basics of Combinatorics.	
Lecture	In the introduction, we give basic examples of the usage of recursion, repeat what was taught in the previous lesson, and repeat the connection between recursive formula in mathematics, with application to natural problems in biology and how we solve it	

through computers (STEAM).
We continue with recursive combinatorial problems that can be easily implemented in Python:

1. Variations without repetitions of length k of a given set with n elemenata, S_n, is any ordered k-tuple of distinct elements from that set. For the set we take $\{0, 1, \ldots, n-1\}$. To be able to generate the variations, we use the similar algorithm to the one for generating all the permutations, but we will exchange only the first k elements of the array with all the others, after we will have the variation on the first k positions in the array. We will use the separate function for writing the first k elements of the variation. In all calls, the parameters n and k will have their initial value.

```
import numpy as np


def ispisi(niz, k):
    for i in range(k):
        print(niz[i], end=" ")
    print()


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


def vbp(niz, n, k, m):
    if m == k:
        ispisi(niz, k)
    else:
        for i in range(m, n):
            zameni(niz, m, i)
            vbp(niz, n, k, m + 1)
            zameni(niz, m, i)



n = 4
k = 2
niz = np.arange(n)
vbp(niz, n, k, 0)
```

2. The combinations with repetitions of length k, of a given set with n elements, S_n, is any multiset with exactly k, not

Assessment of learning
X Test
☐ Quiz
☐Presentation
☐Project
☐Published work

necessarily distinct, elements of that set. For the set we use $\{0, 1, \ldots, n-1\}$. Having in mind that the order of the elements in the combinations is not important, and we want to place them in an array, we place them in nondecreasing order. To be able to go through all such combinations, we use again the recursion. If $n > 1$, all the combinations with repetitions of the set $\{0, 1, \ldots, n-1\}$ of length k are generated in a way that we first generate all the combinations with repetitions of the set $\{0, 1, \ldots, n-1\}$ of length $k-1$ and to all of them we add the element $n-1$, and then, we generate all the combinations with repetitions of the set $\{0, 1, \ldots, n-2\}$ of length k. If $n = 1$, only the first option is taken into consideration.

```
import numpy as np


def ksp(niz, n, k):
    if k == 0:
        print(niz)
    else:
        niz[k - 1] = n - 1
        ksp(niz, n, k - 1)
        if n > 1:
            ksp(niz, n - 1, k)


n = 3
k = 4
niz = np.empty(k, int)
ksp(niz, n, k)
```

3. Combinations without repetitions of a given set of n elements, S_n, of length k, is any subset of that set with exactly k elemenats. We take $\{0, 1, \ldots, n-1\}$ to be our set of interest. As with the combinations with repetitions, the order of the elements is not important, and we want to place them into an array, we place them in an increasing order (as no repetitions are allowed). To be able to

	do that, we use the recursion. If $n > k$, we generate all the combinations without repetitions $\{0, 1, \ldots, n-1\}$ of length k in such a way that we first generate all the combinations without repetitions of the set $\{0, 1, \ldots, n-2\}$ of length $k-1$ and then add element $n-1$ to all of them. After that we generate all the combinations without repetitions of set $\{0, 1, \ldots, n-2\}$ of length k. If $n = k$, only the first option can be applied, as no element can be skipped.
	
	`import numpy as np`
	
	`def kbp(niz, n, k):`
	`  if k == 0:`
	`  print(niz)`
	`  else:`
	`  niz[k - 1] = n - 1`
	`  kbp(niz, n - 1, k - 1)`
	`  if n > k:`
	`  kbp(niz, n - 1, k)`
	
	`n = 6`
	`k = 4`
	`niz = np.empty(k, int)`
	`kbp(niz, n, k)`	

Action	The demonstration of power of Python in solving the combinatorial problems and visualization.	
Materials / equipment / digital tools / software	Computer, electronic whiteboard, PyCharm software	
Reflections and next steps		

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	**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.
References	
In Apendix: **Photographs, Lists of students, Test, questionare**	