TITLE: Built-in functions and methods of working with lists

LEARNING SCENARIO		
School:	***Duration (minutes):***	**90**
Teacher:	***Students age:***	**13**

Essential Idea:	**Let's meet built-in functions and methods of working with lists**

Topics:

- Pupils deepen their understanding of the use of various software and policies.

Aims:

- Pupils will be able to design and create programs that utilize subroutines, appropriate structures and data types, expressions, variables and iterative and conditional commands.
- General programming languages are used to create programs.
- Pupils understands the different ways to use simulations and step-by-step organization algorithms to solve problems.

Outcomes:

- Pupils create a more complex game, application, or mobile application that solves a particular problem from specific subject or topic.
- Pupils learn how to outline the operation of a more complex program into various patterns and generalizations.

Work forms:

- individual work
- work in pairs
- group work

Methods:

- presentation
- discussion
- interactive exercise

ARTICULATION
Course of action (duration, minutes)

INTRODUCTION

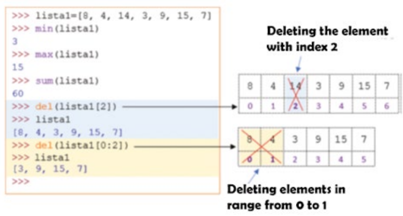
Teacher explains and starts discussion with pupils:
We have learned how to store data with common attributes (strings) in the same variable and how to access single elements of that variable. Let's meet some of the functions and methods that can make our life with lists easier.

MAIN PART

By using strings to store different data types with common attribtes we have made it more simple to access and view input data.
We often need to determine the length (number of elements) in the string, find the smallest or the largest value, or even delete an element.
Many programming languages offer different functions that do just that. Some of the functions built into Python are shown in the table below.

Function	Description
len(a)	Returns the length of the list
min(a)	Returns the smallest element of the list
max(a)	Returns the biggest element of the list
sum(a)	Returns the sum of all the elements
del(a[i])	Removes the element with the index **i**
del(a[i:j])	Removes all the elements form **i** to **j-1**

Applying these funcions depends on the type of data of the elements in the string.



EXERCISE 1

Create a list that's made of the heights of the students in a class. By using functions find and print the smallest and the biggest height.

```
>>> lista=[156,160,145,147,160,157]
>>> print('The height of the shortest student is',min(lista))
The height of the shortest student is 145
>>> print('The height of the highest student is',max(lista))
The height of the highest student is 160
>>> |
```

EXERCISE 2

Create a list made of students' grades in a test. By using functions print out the average grade.

```
>>> lista=[1,2,3,4,5,1,2,3,4,5]
>>> print(sum(lista)/len(lista))
3.0
>>>
```

```
>>> prazna=[]
>>> print(prazna)
[]
>>>
```

Just like with strings, a list can be empty. We can create an empty list by assigning empty brackets to a variable, for example a=[].

Built-in methods for working with lists

Besides operations and functions, while working with lists we can use methods. Methods are always assigned to a certain object (a variable or a list) and they are shown after the object they are assigned to. Some of the methods we can use in Python are:

Name of the method	The way to use it	Description
append()	a.append(b)	Method adds element b to the end of the list a
insert()	a.insert(i,b)	Method inserts element b before element i of the list a
remove()	a.remove(b)	Method deletes element b from the list. If there are multiple b-s, the one with the lowest indeks iz deleted.
reverse()	a.reverse()	Method makes an inverse order of the elements in the list a.
sort()	a.sort()	Method sorts the list a from the smallest to the biggest value.
count()	a.count()	Method returns the number of repetitions of object b in the list a as a result
index()	a.index(b)	Mehod finds the indeks of element b in the list.

Let's show these methods in some examples.

```
>>> a=[9,4,6,12,16,3]
>>> a.append(13)
>>> a
[9, 4, 6, 12, 16, 3, 13]
>>>
```

Adds a new element with value 13 and puts it at the end of the list

```
>>> a.insert(3,17)
>>> a
[9, 4, 6, 17, 12, 16, 3, 13]
>>>
 0  1  2  3   4   5   6  7
```

Adds a new element with the value 17 and puts it at index 3

```
>>> a.remove(16)
>>> a
[9, 4, 6, 17, 12, 3, 13]
>>>
```

Deletes the element with the value 16 from list a. The list is shorter by 1 element.

```
>>> a.reverse()
>>> a
[13, 3, 12, 17, 6, 4, 9]
>>>
```

Elements in list a are reverersed.

```
>>> a.sort()
>>> a
[3, 4, 6, 9, 12, 13, 17]
>>>
```

Elements in list a are sorted starting from the smallest to the highest.

```
>>> a.append(6)
>>> a
[3, 4, 6, 9, 12, 13, 17, 6]
>>> a.count(6)
2
>>>
```

We add one element with the values 6.
We count how many repetitions the value 6 has in the list a.

```
>>> a.index(12)
4
>>> a
[3, 4, 6, 9, 12, 13, 17, 6]
```

We print the index of an element with the value 12.

EXERCISE 3

Write a program that will take a list of n natural numbers as an input. Print the list of the numbers in a single row.

```
n=int(input('Enter the number of elements in the list:'))
a=[]
for i in range(n):
    m=int(input('Enter a number: '))
    a.append(m)
print(a)
```

Explanation:
in the task we can see that the length of the list is specified – it has n elements. We will write the elements using the function for which will write a value n times and the add it with the method append(), each time at the end of list a. We had to initialise the list at the beginning of the program. At the end we print the list by simple using the print command.

Making a list from input values
In the last task we have shown how to create a list from given values – characters or numbers. Still, we often don't know the number of elements in the list in advance. In cases like that we can't use the for loop with an assigned number of repetitions. In a case like that we need to use the split() method. That's the method with which we can

separate words with a space inside a string and therefore create a new list of words. Elements of a list made with split() are always characters.

```
>>> names=input('Enter the names of the students: ').split()
Enter the names of the students: Steve, John, Katie, Sarah
>>> print(names)
['Steve,', 'John,', 'Katie,', 'Sarah']
>>>
```

EXERCISE 4

Write a program in which we will input several words and then create a new list which will contain only the first letter of each word.

```
a=input('Enter a list of words: ').split()
b=[]
for i in range(len(b)):
    x=a[i]
    b.append(x(0))
print(b)
```

Explanation:

a=input('Enter a list of words: ').split()	Enters the words from the list a
b=[]	Creates an empty list b in which we will later store starting letters from the list a
for i in range(len(b)):	Defines the repetition by depending on the length of the list a
x=a[i]	Each word in a is temporarily stored in the variable x
b.append(x(0))	We're adding new elements to the list, the first letter from the word stored in x
print(b)	We print out the new list b

EXERCISE 5

Write a program that will input a list of letters inside a single row and then inverse the order of the elements in the list. Print the newly formed list.

Example of testing

```
Unesi slova: a b c d e f
['f', 'e', 'd', 'c', 'b', 'a']
>>>
```

Computer program

```
a=input('Unesi slova: ').split()
a.reverse()
print(a)
```

Can we use the last program to make a reverse list of numbers? If the numerical values are entered as characters (without int()) then we need to convert them into numbers.

```
>>> lista=input('Upiši brojeve: ').split()
Upiši brojeve: 7 6 3 12 4 15 8
>>> print(lista)
['7', '6', '3', '12', '4', '15', '8']
>>>
```

List of numbers stored as a list of characters.

Using the function int() converts the characters from the original list into numbers.

```
>>> lista=input('Upiši brojeve: ').split()
Upiši brojeve: 7 6 3 12 4 15 8
>>> nova=[]
>>> for i in range(len(lista)):
        nova.append(int(lista[i]))
>>> nova.reverse()
>>> print(nova)
[8, 15, 4, 12, 3, 6, 7]
>>>
```

EXERCISE 6

After each test the teachers correct the solutions and analise their students' work. Write a program that will input all of the grades students have achieved on a test. By using functions print out the number of students who have A as their grade, and the number of students who have not passed the test.

Example of testing

```
Unesi ocjene: 1 2 3 5 4 3 2 5 3 5 1
3 su učenika dobila ocjenu odličan.
2 su učenika dobila ocjenu nedovoljan.
>>>
```

Computer program

```
lista=input('Unesi ocjene: ').split()
nova=[]
for i in range(len(lista)):
    nova.append(int(lista[i]))
print(nova.count(5),'su učenika dobila ocjenu odličan.')
print(nova.count(1),'su učenika dobila ocjenu nedovoljan.')
```

EXERCISE 7

Write a program that will input an arbitraty list of numbers. The program needs to create and write a new list of numbers that will contain only the even numbers out of those that were inputed in the first place.

```
list=input('input the numbers: ').split()
new=[]
for in in range(len(list)):
    if int(list[i])%2==0:
        new.append(int(list[i]))
print(new)
```

Explanation:
Like in the previous task, we have to create a new list, but not from all of the input values. We're just using those that meet the condidion of being even. While solving this task we have to remind ourselves of the use of the operator % (remainder) to test whether a number is even. That's why the crucial command in our solution is:

if int(list[i])%2==0:
list[i] – Changes the indeks gets us through the list
int – turns the list element into a number
%2==0 – checks if the number is even

new.append – inserts the even number to the end of the new list

EXCERCISE 8

According to the previous example, pupils can design, create and test their own examples.

CONCLUSION

Pupils and teacher discuss and evaluate the presented solutions.

Methods		***Work forms***
presentation	*interview*	*individual work*
discussion	*demonstration*	*work in pairs*
work on the text	*role playing*	*group work*
graphic work		*frontal work*
interactive exercise /simulation on the computer		

Material:

-

Literature

PERSONAL OBSERVATIONS, COMMENTS AND NOTES