TITLE: Structures and data types in Python

LEARNING SCENARIO		
School:	***Duration (minutes):***	**45**
Teacher:	***Students age:***	**13**

Essential Idea:	**Let's meet structures and data types in Python**

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:

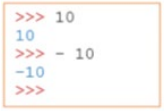
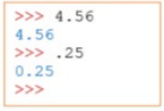
- Pupils create a more complex game, application, or mobile application that solves a particular problem from specific subject or topic.
- Pupils learn how to outline the operation of a more complex program into various patterns and generalizations.

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 about variables. While solving problems on our computer we create programs that use different input variables. Thinking about input variables is one of the first steps we take while designing a solution. Let's introduce ourselves to different data types used in Python. In our first computer programs in Python we have used numerical and text data. Text data was written using semiquotes, and we called that type of data strings. Data types used in computer solutions are important in applying suitable commands for input and output of data, as well as different functions which additionally shape our data. In Python there are three different types of simple data: - Numerical (numbers). - Characters (strings). - Logical (boolean).
MAIN PART **Numerical data type** While beginning to learn how to code and make new computer solutions in Python we have met different simple data types such as whole number, characters, and logical values. We have also met commands for input and output of those values using a computer program. But still, numbers in our everyday life don't need to be whole numbers – for example they can be decimal, so let's take a look at how we could use decimal numbers in Python. Decimal numbers are written just like in math, as two whole number separated with a decimal dot, for example 4.56.         What operations can be used with decimal numbers?

The ability to separate decimal and whole part of the decimal number is very interesting in programming and because of that many programming languages have a built-in function enableing just that.
How to convert a decimal to a whole number?
Let's look at some examples.

```
>>> int(4.2)
4
>>> int(4.8)
4
>>>
```

```
>>> float(5)
5.0
>>>
```

Function **int()** converts a decimal into a whole number by disregarding the decimal part of the number and leavng only the whole part.
Function **float()** converts a whole into a decimal number by adding a zero after the dot.

Logical data type

Logical data type is commonly used in commands where we're looking at a value of some logical conditions. A logical condition is a logical construct in whit we're checking the value of something. It can usually have only two values: True or False. In branching commands (if, if-else, elif-else) and conditional repeating (while) we have learned that, depending on the logical condition, the program is making a decision of it's continued functioning. If we want to store a logical value in a variable, we can assign a logical value to it.

```
>>> a=4<5
>>> print(a)
True
>>>
```

```
>>> b=5<2
>>> print(b)
False
>>>
```

```
>>> b=False
>>> print(b)
False
>>>
```

Strings

Data strings are a basic type of data for storing text values. They represent words or sentences enclosed in quotes or semi-qoutes.

```
>>> a='python'
>>> print(a)
python
>>>
```

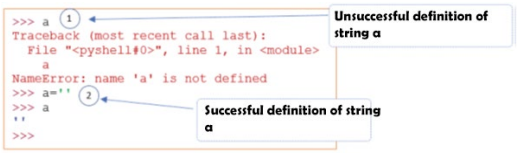
```
>>> b="dobar dan"
>>> print(b)
dobar dan
>>>
```

We have also learned how to apply mathematical operators over a string, for example addition or multiplication.

```
>>> x='abc'
>>> y='def'
>>> x + y
'abcdef'
>>>
```

```
>>> x * 4
'abcabcabcabc'
>>>
```

Adding two character strings creates a new string containing both of the original ones. Multiplying a string with a number simply multiplyes the string. A string can't be multiplyed with another string – you have to use a whole number.
A string can also be defined s empty, we do this by not writing any characters between the semi-quotes.



Indexing a string

A string can be viewed as multiple characters sorted in a certain order. Assigning numbers to the positions of the characters is called indexing. That is the procedure in whic every character is assigned with and index number, or in other words a number describing where the character is in a string.
In Pythong the first character in a string is always assigned the number zero (0), and then each next character is assigned numbers 1, 2, 3, 4…

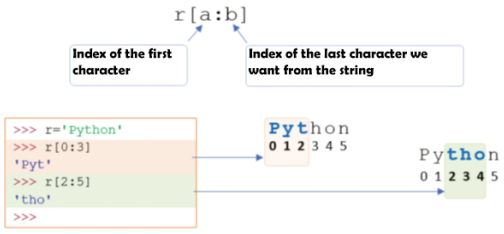
EXCERCISE 1

By using interactive user interface grab the first and the last character in the string „Dubrovnik“.

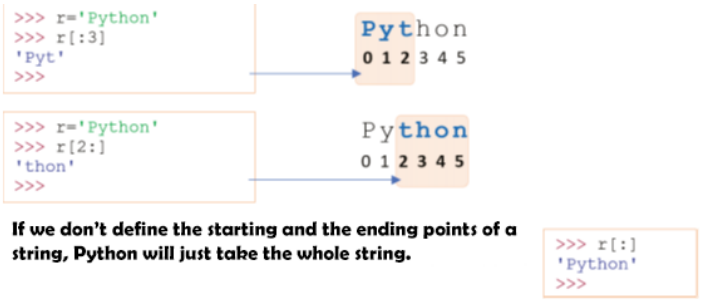
```
>>> r='Dubrovnik'
>>> r[0], r[8]
('D','k')
```

Grabbing multiple characters at the same time – a part of the string

If we want to grab multiple parts of a string, inside brackets of a string we can write down the indeks of the first and the last part of the string we want the grab, and separate it with two dots (:).



There are several ways of grabbing specific parts of strings depending on if we want those parts to include the first or the last character in the string. A part of a string that includes the first character can be grabbed without the indeks 0, for example instead of r[0:3] we can write r[:3]. the same goes for the last character in the string, for example we can write r[2:].



EXCERCISE 2

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