TITLE: Functions in Python

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

Essential Idea:	Let's meet functions 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:
When making solutions on a computer we usually analyse a problem and then try to divide it into smaller parts, after which we can look at the problem based on solutions of it's particular parts.
How can we write small program solutions inside a program with which we can solve parts of the problem, and which we can call multiple times inside a program?

MAIN PART

In computer programs every series of commands has a part to play. Some commands deal with input data, some with output data, some parts form solutions, and some of them deal with using the data according to instructions.
A series of commands that make sense to be looked at as a whole can be separated as an indepetendent part of the program, and that is called a function.
So far we have seen some inbuilt functions such as int(), input(), print(), len()… These functions are already built into Python and we just call them when we need them. Besides using built-in functions, we can also make are own functions for specific tasks. We usually need to make functions to separate independent parts of the program which tend to be repeated. When we do that our code becomes more elegant, and our programs work quicker.
In Python a new functions needs to be defined, meaning we need to use commands to describe what it does. Then we need to name it and give it a list of argument it will use. For example:

```
def funcion_name (paramethers):
    block_of_commands
    return value
```

After defining the function, we can activate it inside the program by simply calling its name. Considering that functions often use input data, we need to give that sort of data to the function if we want it to work properly. Functions can take different input paramethers that the user sends them and return them when and if they are needed. There are four different types of functions in Python.

Functions that don't return a value and:
- Don't have input paramethers

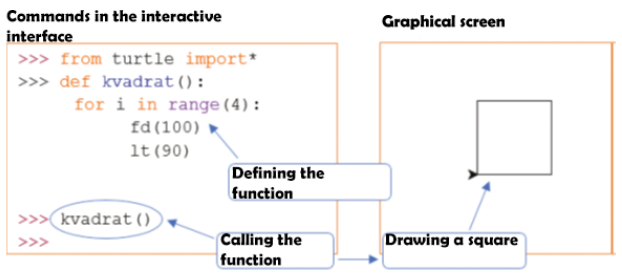
- Have input paramethers

Functions that return a value and:
- Don't have input paramethers
- Have input paramethers

A function that has no input values and returns no values after execution
This form of function has no input parameters, and only has values or variables inside the function itself. If the function doesn't return any values then we don't need to use the return command. The function is executed by calling its name without any paramethers in the brackets: function_name().

EXERCISE 1

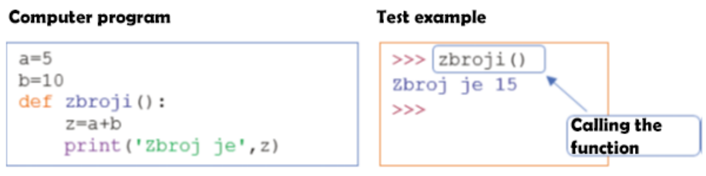
Let's apply a function to draw a square with a side 100 pixels long.



```
>>> from turtle import*
>>> def kvadrat():
        for i in range(4):
            fd(100)
            lt(90)

>>> kvadrat()
>>>
```

EXERCISE 2

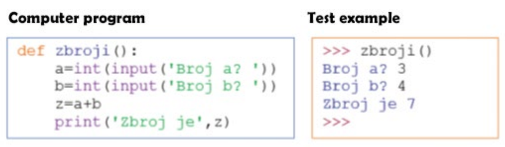
Write a program in which we will use a function in order to add two given numbers. The sizes of the numbers (variables a and b) will be defined outside of the function, in the program.



```
a=5
b=10
def zbroji():
    z=a+b
    print('Zbroj je',z)
```

```
>>> zbroji()
Zbroj je 15
>>>
```

In this task we have written a computer program that adds two previously defined values, and that is pretty limiting. A better way to do it would be to input variables we want to add.



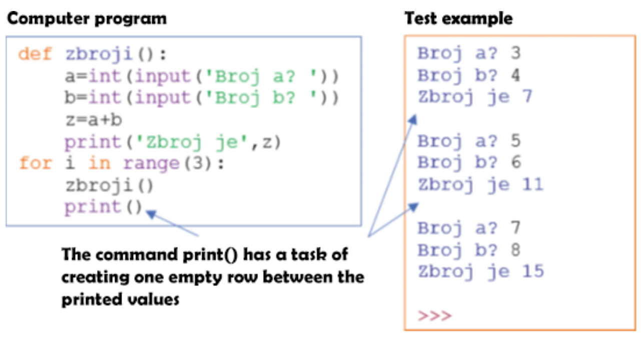
Computer program

```
def zbroji():
    a=int(input('Broj a? '))
    b=int(input('Broj b? '))
    z=a+b
    print('Zbroj je',z)
```

Test example

```
>>> zbroji()
Broj a? 3
Broj b? 4
Zbroj je 7
>>>
```

If we want to call the same adding function multiple times we can put it inside a for loop.



Computer program

```
def zbroji():
    a=int(input('Broj a? '))
    b=int(input('Broj b? '))
    z=a+b
    print('Zbroj je',z)
for i in range(3):
    zbroji()
    print()
```

Test example

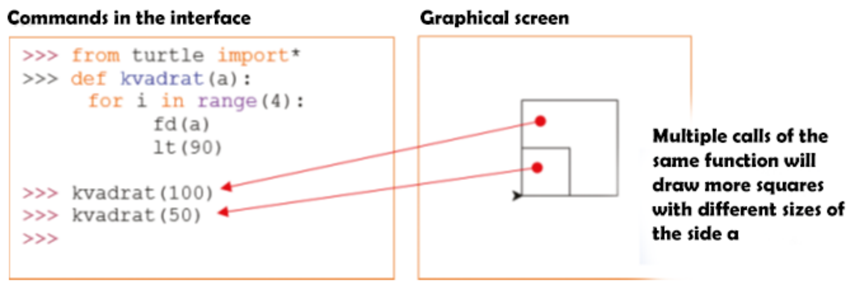
```
Broj a? 3
Broj b? 4
Zbroj je 7

Broj a? 5
Broj b? 6
Zbroj je 11

Broj a? 7
Broj b? 8
Zbroj je 15

>>>
```

The command print() has a task of creating one empty row between the printed values

A function that has input values and doesn't return a value after executing
This form of function has one or multiple input paramethers and doesn't rturn any values after executing. The fuction is executed by calling its name with a list of input paramethers in the brackets: function_name(list of paramethers).

EXERCISE 3

Let's make a function that will draw a square with a side a pixels long. The length of the side a is defined when calling the function as an input paramether.



Commands in the interface

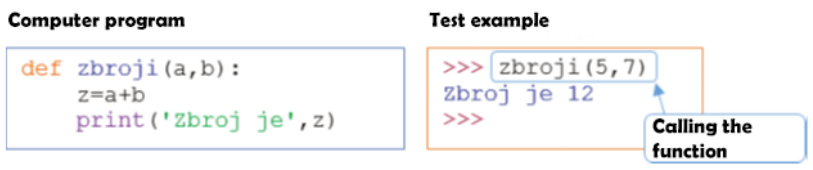
```
>>> from turtle import*
>>> def kvadrat(a):
      for i in range(4):
           fd(a)
           lt(90)

>>> kvadrat(100)
>>> kvadrat(50)
>>>
```

Graphical screen
Multiple calls of the same function will draw more squares with different sizes of the side a

The function square() from the previous task uses one input paramether weitten inside bracked after the name of the function. If we want to use input paramethers when calling the function, then those paramethers have to be defined: def square(a). A funcion can have many defined paramethers, which will be shown in the next ecercise.

EXERCISE 4

Let's write a program that will use a function to add two numbers.


Computer program

```
def zbroji(a,b):
    z=a+b
    print('Zbroj je',z)
```

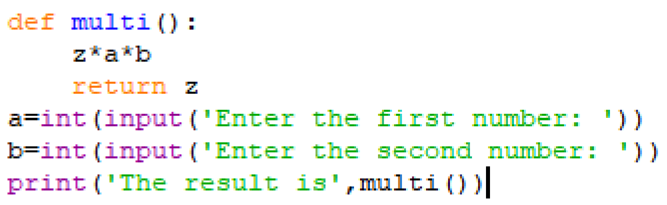
Test example

```
>>> zbroji(5,7)
Zbroj je 12
>>>
```

A function has no input parameters and returns no values after execution.
The input parameters are written when we change the built-in input() function, a the command „return“ returns the result of the caltulation. The value that will be returned to the program is written after the return command.

EXERCISE 5

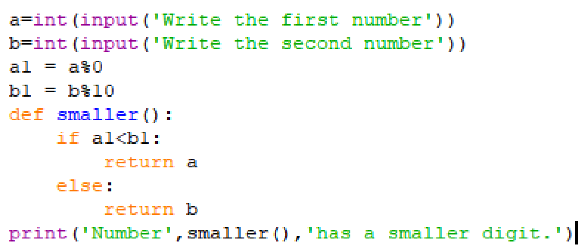
Let's write a program that will calculate and print a multiplication of two numbers. To calculate this w w will write a special function. The numbers will be written as input values to the main program.


```
def multi():
    z*a*b
    return z
a=int(input('Enter the first number: '))
b=int(input('Enter the second number: '))
print('The result is',multi())
```

Explanation
We have used input values to write the values of a and b and we have called the function multi() that multiplyes these two numbers. After executing the function multi(), the result is stored in the variable z as the result of multi() to the main program.

EXERCISE 6

Write a program that will input two three-digit number and print a number with digits smaller by 1. We'll write a function that will compare the digits and output the wanted number.



```
a=int(input('Write the first number'))
b=int(input('Write the second number'))
a1 = a%0
b1 = b%10
def smaller():
    if a1<b1:
        return a
    else:
        return b
print('Number',smaller(),'has a smaller digit.')
```

Explanation

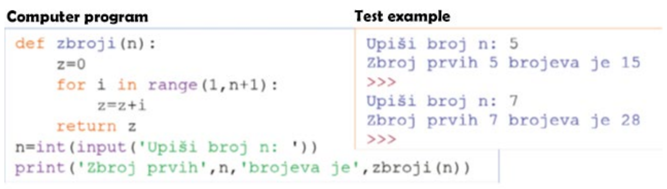
By using the operator % (remainder) we have separated digits of the elements written in a and b. We have then stored them in a1 and b1. By calling the function smaller() inside of print() we have compared the values of the variables a1 and b1 and returned the corresponding value into the program. That value was printed as a result.

A function that has input parameters and returns the value after execution

The most complicated form of a function is one that has input parameters and outputs a value after executing. We can see that such a function has to have a list of expected paramethers when we define it, and also has to have a value after the return command.

EXERCISE 7

Let's write a program that will calculate a sum of the first n natural numbers. Inside the program we will apply the function for calculating the addition of given numbers.



Computer program

```
def zbroji(n):
    z=0
    for i in range(1,n+1):
        z=z+i
    return z
n=int(input('Upiši broj n: '))
print('Zbroj prvih',n,'brojeva je',zbroji(n))
```

Test example

```
Upiši broj n: 5
Zbroj prvih 5 brojeva je 15
>>>
Upiši broj n: 7
Zbroj prvih 7 brojeva je 28
>>>
```

EXERCISE 8

In tasks for practicing math we have often seen prime numbers. For example, a student has had to check if a number is prime or not. Let's help a student and write a programe that will take any number as an input and see if it's a prime numbers. In our program we will use a function that will return a message saying whether the number is a prime or not.

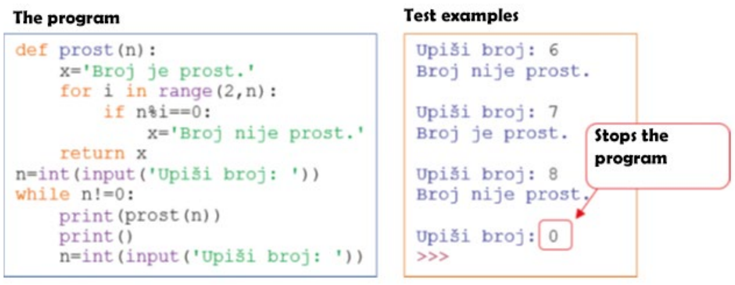
```
def prime(n):
    x='The number is a prime'
    for i in range(2,n):
        if n%i==0:
            x='The number is not a prime'
    return x
n=int(input('Write a number: '))
print(prime(n))
```

Explanation

We know that prime numbers can't be divided by anything but one (1) and themselves. So we only need to check whether n can be divided by other numbers from 2 to n-1. If there are no such number, n is a prime number.
The for loop makes sure that we go through all of the numbers. If we find such a number the value of x will be changed.

EXERCISE 9

Let's upgrade our last program so that it prints out numbers until it comes to zero. With every number it will also print out a message saying whether the number is a prime.



The program

```
def prost(n):
    x='Broj je prost.'
    for i in range(2,n):
        if n%i==0:
            x='Broj nije prost.'
    return x
n=int(input('Upiši broj: '))
while n!=0:
    print(prost(n))
    print()
    n=int(input('Upiši broj: '))
```

Test examples

```
Upiši broj: 6
Broj nije prost.

Upiši broj: 7
Broj je prost.

Upiši broj: 8
Broj nije prost.

Upiši broj: 0
>>>
```

EXCERCISE 10

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