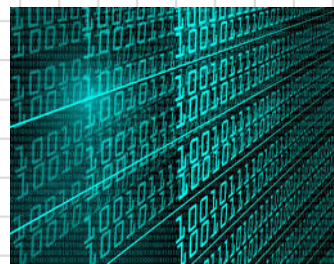




LESSON PLANS



PROGRAMMING AND CODING IN VOCATIONAL EDUCATION

2023-2-SK01-KA210-VET-000184496 01. 01. 2024 – 31. 12. 2025

CNC

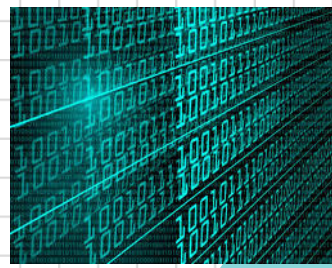
Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Education and Slovak Academic Association for International Cooperation (SAAIC). Neither the European Union nor SAAIC can be held responsible for them.



Funded by
the European Union



LESSON PLAN



SUBJECT: PROGRAMMING

TOPIC: PROGRAMMING IN PYTHON – DRAWING GEOMETRIC SHAPES.

OBJECTIVES: 1. ACQUIRE BASIC KNOWLEDGE OF PROGRAMMING IN PYTHON USING THE TKINTER GRAPHIC LIBRARY.

2. TO LEARN TO WORK WITH GRAPHIC CANVAS. DRAW SIMPLE SHAPES ON IT.

3. CREATE IMAGES AT A RANDOM LOCATION USING THE TKINTER LIBRARY.

TOOLS:

**A COMPUTER WITH PYTHON AND THE
TKINTER LIBRARY INSTALLED.
PYTHON SAMPLE TEXTBOOK
WORKSHEET WITH EXERCISES**

**INTRO: WE WILL SHOW HOW TO
CREATE AND RUN THE FIRST
PROGRAM, CREATE A GRAPHIC
CANVAS ON WHICH WE WILL DRAW
GRAPHIC SHAPES. WE'LL USE A
VARIABLE, GENERATE RANDOM
VALUES, AND DRAW IMAGES AT A
RANDOM LOCATION USING THE
TKINTER LIBRARY.**

LESSON STRUCTURE: ENGAGEMENT (4 MINUTES) – DRAWING AN ELEMENTARY PICTURE IN CARTESIAN COORDINATES INTERVIEW WITH STUDENTS ABOUT THE COORDINATE SYSTEM IN THE TKINTER MODULE (TASK 1)

- RESEARCH (12 MINUTES) – RESEARCHING A NEW COMMAND (TASK 2 TO 4)**
- PRESENTATION (15 MINUTES) – SUMMARY AND EXPLANATION OF KNOWLEDGE FROM THE SECTION RESEARCHING THE ISSUE OF CREATING PROGRAMS WITH A GRAPHICAL USER INTERFACE**
- DEVELOPMENT (9 MINUTES) – INDEPENDENT PROGRAMMING OF MORE DEMANDING TASKS (TASKS 5 TO 6)**
- ASSESSMENT (5 MINUTES) – SOLVING THE SELF-ASSESSMENT TEST, DISCUSSING THE ANSWERS.**

A BRIEF OVERVIEW OF PYTHON PROGRAMMING (FOR BEGINNERS).

INSTALLING THE TKINTER LIBRARY (IF NOT INSTALLED).

PRESENTATION OF THE TKINTER LIBRARY.

EXPLANATION OF THE PRINCIPLE OF DRAWING ON CANVAS.

WORKSHEET FOR THE STUDENT

WE WILL PROCEED ACCORDING TO OUR ELECTRONIC TEXTBOOK PROGRAMMING IN PYTHON, WHICH IS INTENDED FOR SECONDARY SCHOOLS. YOU CAN FIND A SAMPLE TEXTBOOK AT:

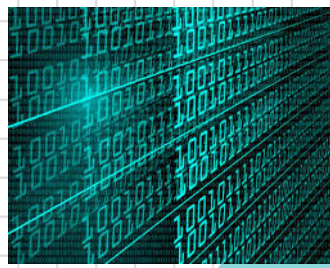
[HTTP://WWW.PROGRAMUJEMEVPYTHONE.SK/UKAZKA.PDF](http://www.programujemevpythone.sk/ukazka.pdf) THE SAMPLE CONTAINS SIMPLE EXAMPLES OF DRAWING:

1. DRAWING BASIC SHAPES:

DRAWING A CIRCLE, RECTANGLE, SQUARE AND TRIANGLE USING DIFFERENT COLORS AND LINE THICKNESSES CODE EXAMPLE FOR EACH SHAPE EXPOSURE PHASE – EXPOSURE OF A NEW OBJECT – I WILL BRIEFLY NOTE THE BASIC TERMS ON THE BOARD



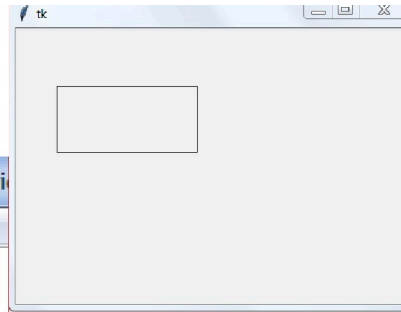
LESSON PLAN



WORKSHEET FOR THE STUDENT

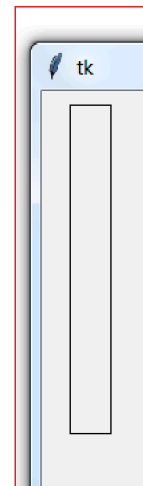
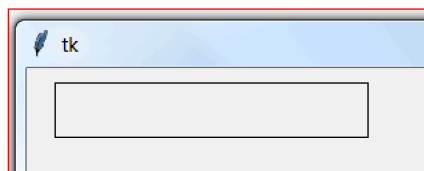
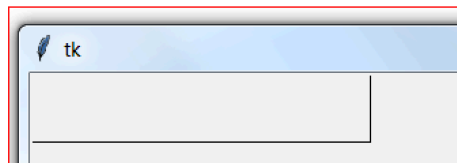
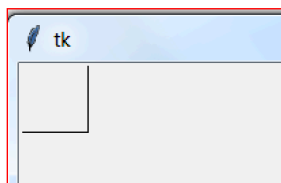
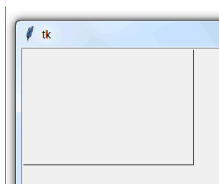
1. Task: Create the file 05lek_p04.py, the commands listed in it and run the program xxxxx:

```
05lek_p04.py - F:/14_Python(SŠ)/PythonSŠ_PracovnéListy/05_PL (ri  
File Edit Format Run Options Window Help  
import tkinter  
canvas = tkinter.Canvas()  
canvas.pack()  
  
canvas.create_rectangle(50, 70, 220, 150)
```



2. Task: There are 4 numbers in the brackets of the command `canvas.create_rectangle(...)`. Try changing them gradually in the program 05lek_p05.py (changes compared to the previous entry are highlighted in yellow). Run the program each time to see what it

- draws:
- a) `canvas.create_rectangle(0, 0, 220, 150)`
 - b) `canvas.create_rectangle(0, 0, 50, 50)`
 - c) `canvas.create_rectangle(0, 0, 250, 50)`
 - d) `canvas.create_rectangle(20, 10, 250, 50)`
 - e) `canvas.create_rectangle(20, 10, 50, 250)`





LESSON PLAN



1.

WORKSHEET FOR THE STUDENT

1. 3. Task: On the computer, there is a point with coordinates $[0, 0]$ in the upper left corner. The x-axis goes from left to right. The y-axis is reversed compared to mathematics - it goes from top to bottom (the bigger the number, the lower).

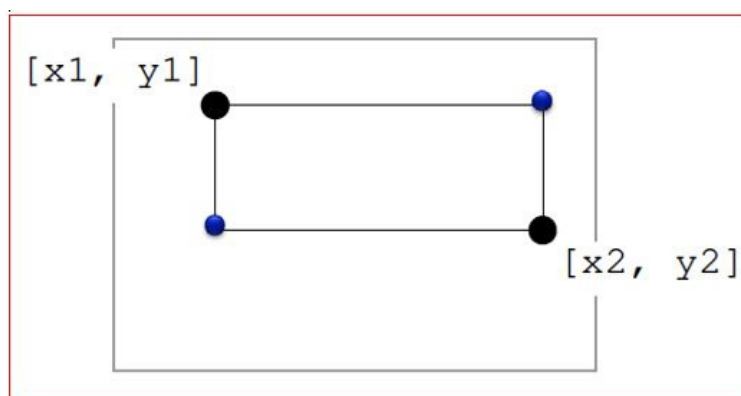


1.

4. Task: In the command `canvas.create_rectangle(x1, y1, x2, y2)`

2.

we write in the brackets the coordinates of the opposite vertices of the rectangle:



1. 5. Task: Change your program 05lek_p06.py to draw a rectangle that has the coordinates of opposite vertices $[50, 30]$ and $[300, 200]$.

```
*05lek_p05.py - F:/14_Python(SŠ)/PythonSŠ_PracovnéListy/05_PL (riešenia)/05lek_p05.py
File Edit Format Run Options Window Help
import tkinter
canvas = tkinter.Canvas()
canvas.pack()

canvas.create_rectangle(50, 30, 300, 200)
```

Šírka: $300 - 50 = 250$

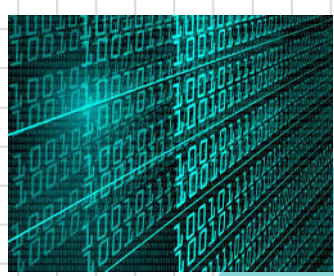
1.

6. Task: Calculate the width and height of the rectangle from the previous example with width: $300 - 50 = 250$ and height: $200 - 30 = 170$

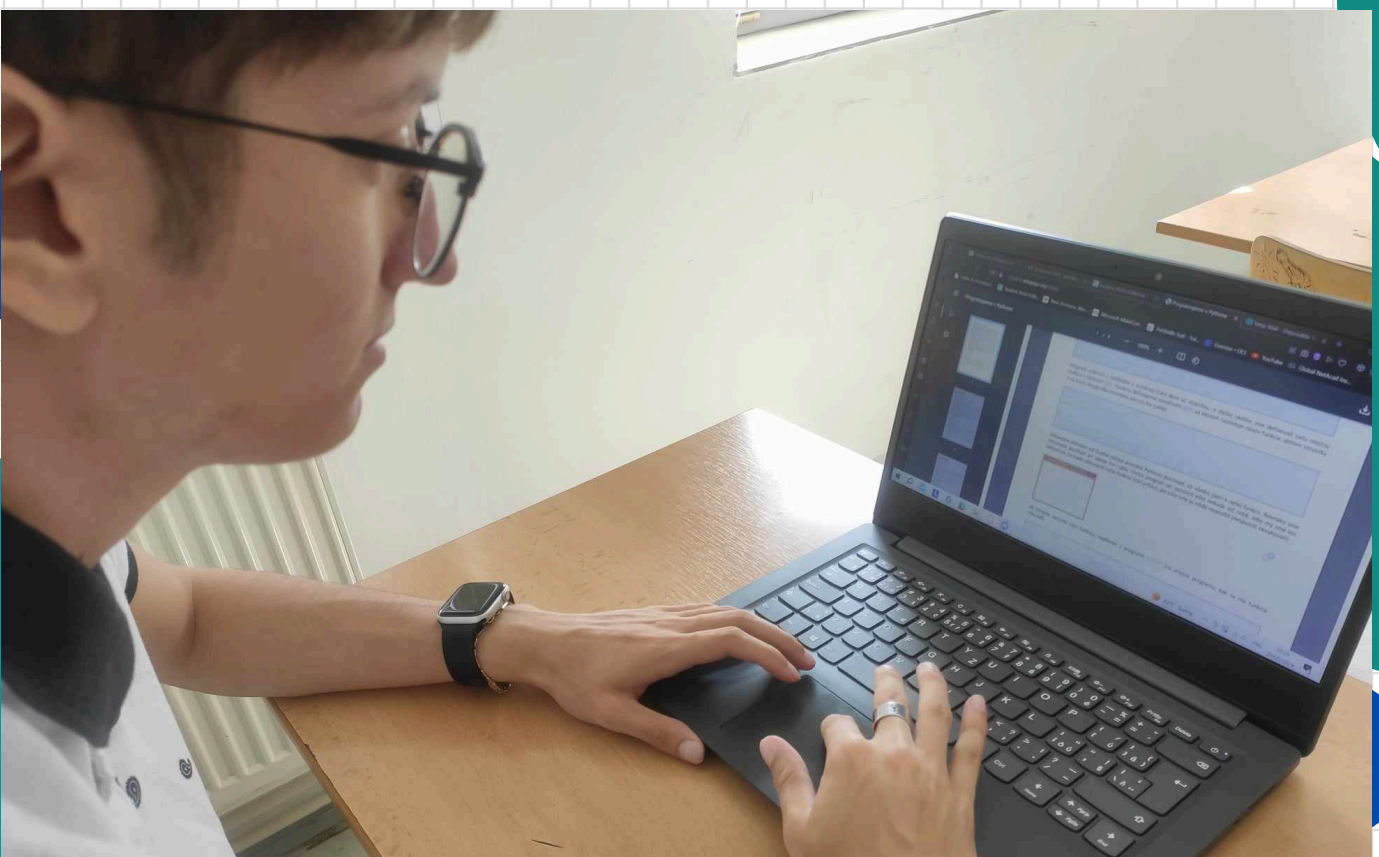
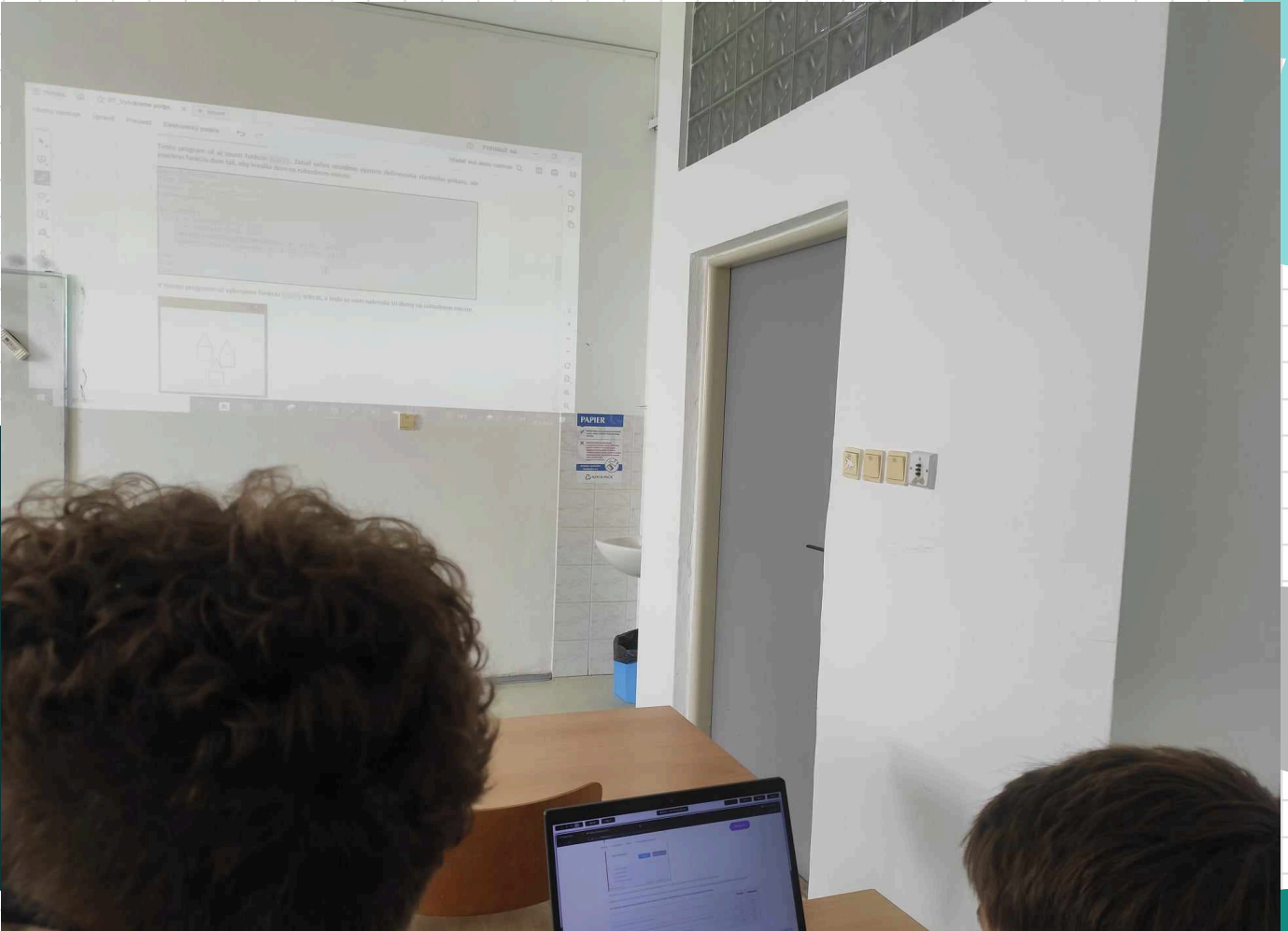
- 1.7. Task: Create a new program 05lek_p08.py and draw a rectangle that has one vertex at coordinates $[200, 100]$, its width is 60 and its height is 140.



ANNEX

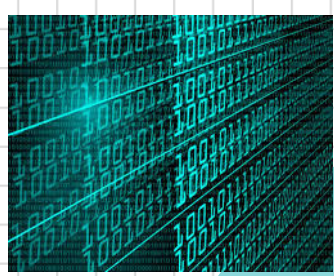


PHOTOS FROM THE LESSON IMPLEMENTATIONS

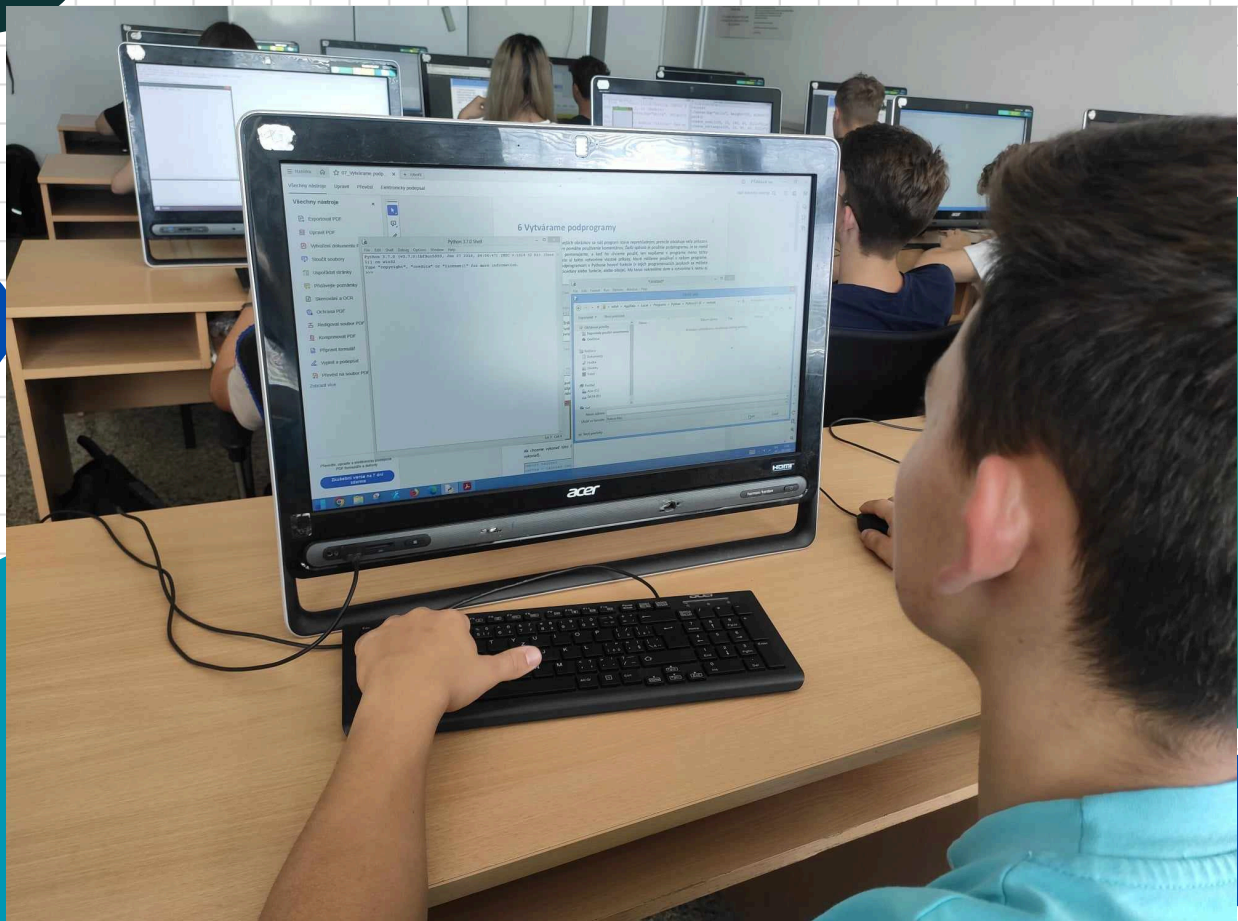
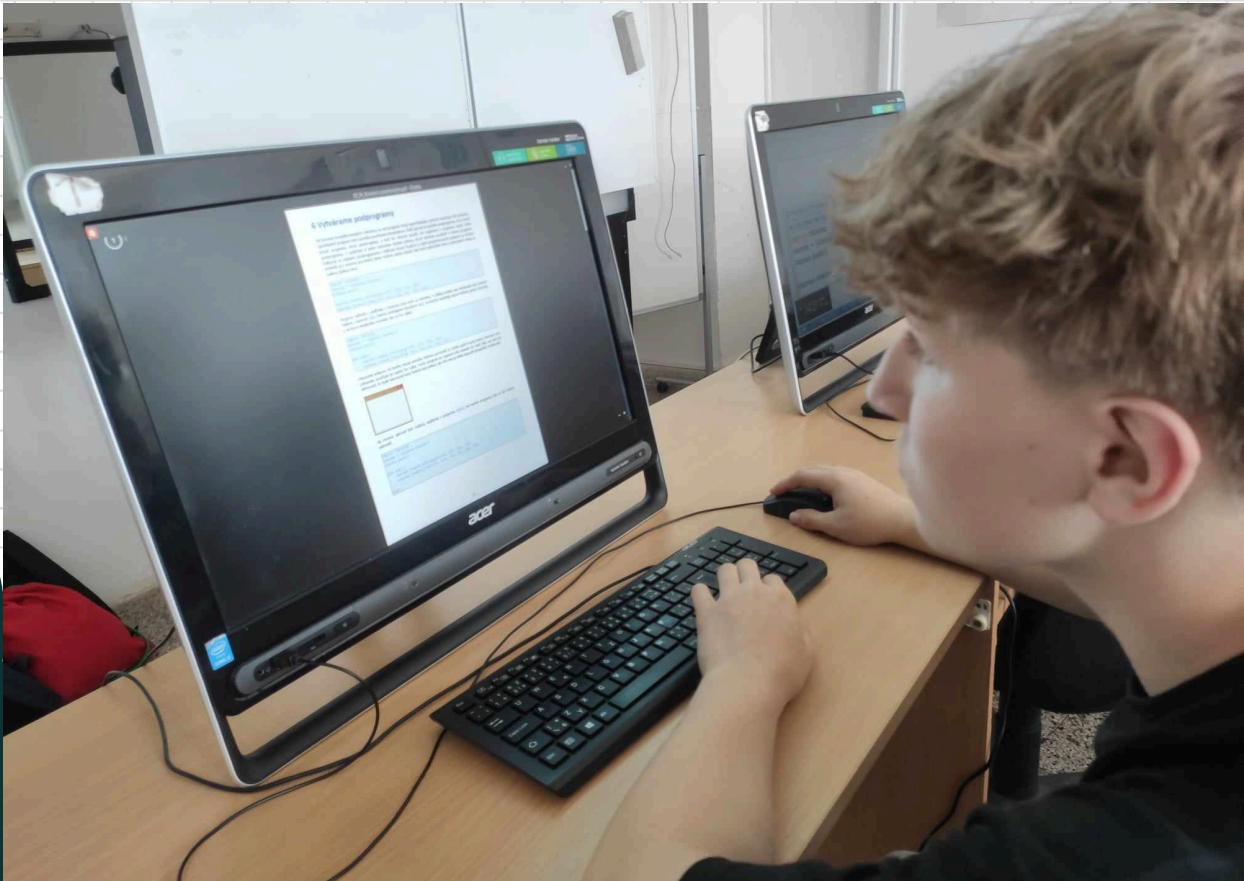




ANNEX



PHOTOS FROM THE LESSON IMPLEMENTATIONS

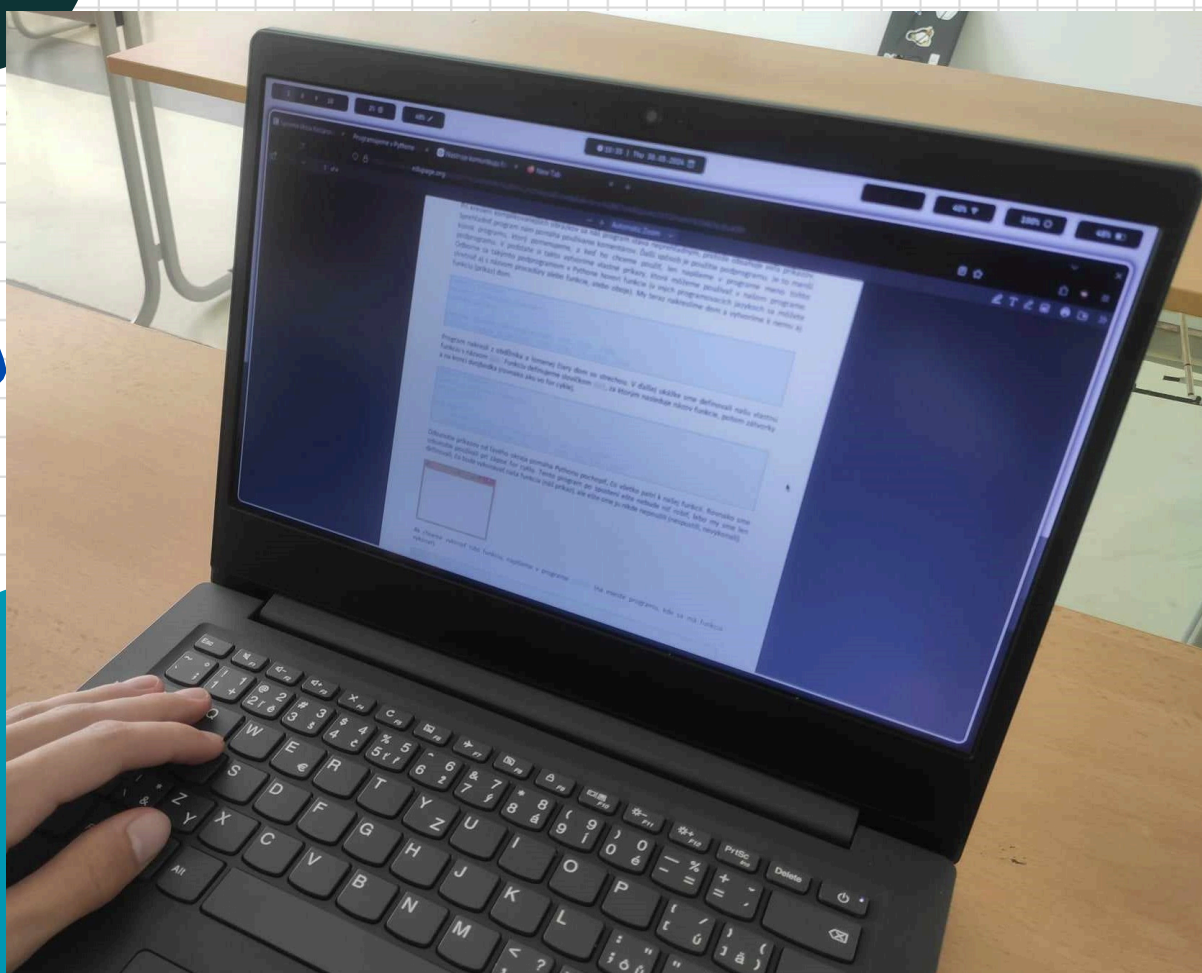
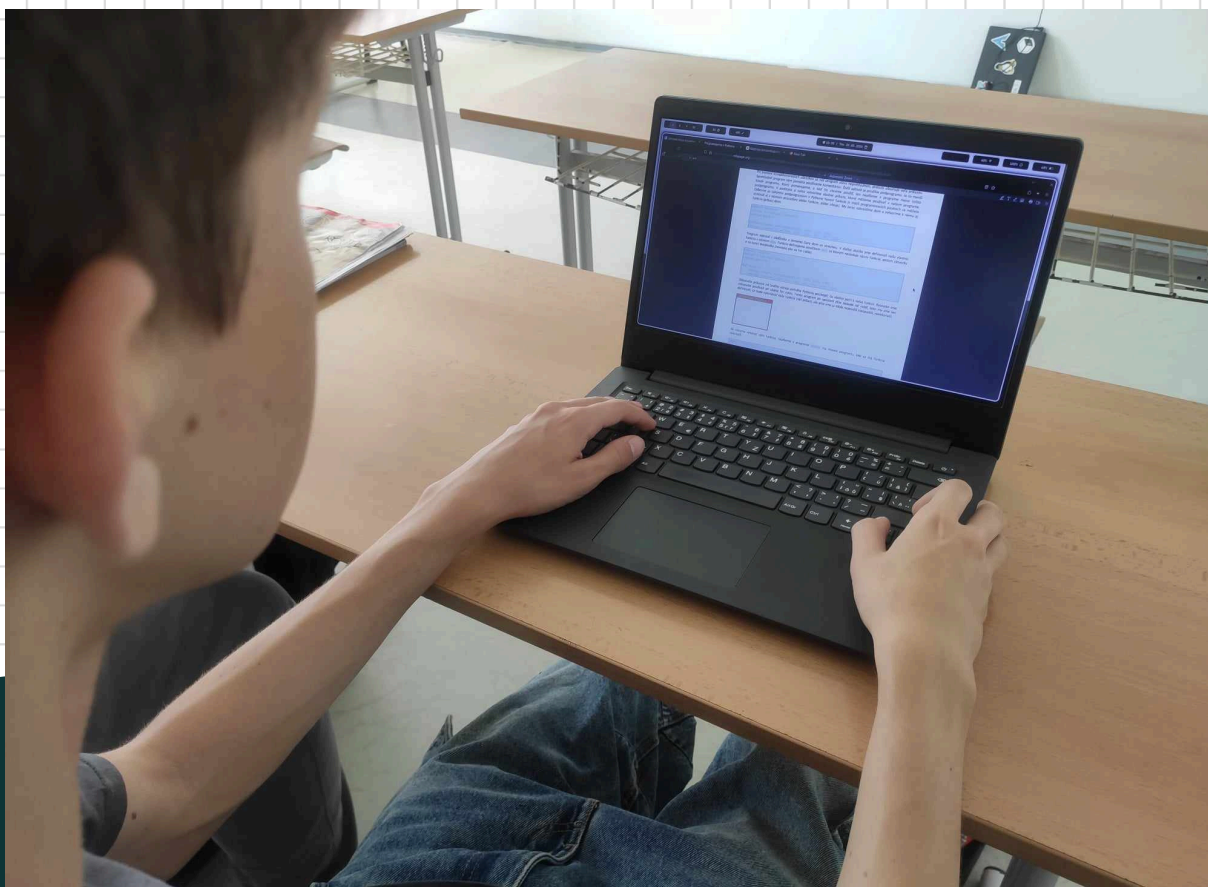




ANNEX

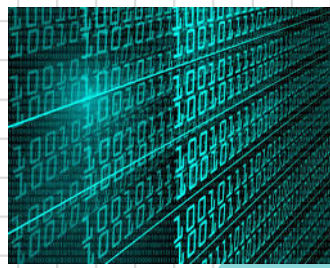


PHOTOS FROM THE LESSON IMPLEMENTATIONS





ANNEX

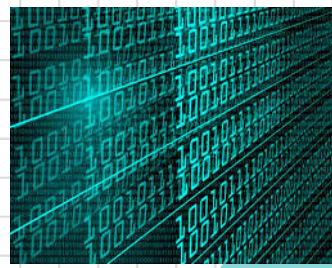


STUDENTS OUTPUT

```
FROM TKINTER IMPORT *  
CREATE A TKINTER WINDOW  
ROOT = TK()  
CREATE A CANVAS TO DRAW ON  
CANVAS = CANVAS(ROOT, WIDTH=300, HEIGHT=300) CANVAS.PACK()  
CREATE A SQUARE  
CANVAS.CREATE_RECTANGLE(50, 50, 80, 80, FILL="BLUE")  
CREATE A RECTANGLE  
CANVAS.CREATE_RECTANGLE(100, 100, 160, 140, FILL="RED")  
RUN THE TKINTER MAIN LOOP  
ROOT.MAINLOOP()
```




LESSON PLAN



SUBJECT: PROGRAMMING

TOPIC: PROGRAMMING IN PYTHON – DRAWING AND COLORS.

LEARNING OBJECTIVES:.

- **1. UNDERSTAND THE BASIC PRINCIPLES OF DRAWING IN PYTHON USING THE TKINTER LIBRARY**
- 2. LEARN TO USE DIFFERENT COLORS AND EFFECTS WHEN DRAWING IN PYTHON.**

TOOLS:

**A COMPUTER WITH PYTHON AND THE TKINTER LIBRARY INSTALLED.
PYTHON SAMPLE TEXTBOOK WORKSHEET
WITH EXERCISES**

**INTRO: USING DIFFERENT COLORS IN
DRAWING:**

**DEFINING COLOR USING RGB VALUES
USE OF DIFFERENT COLORS FOR INDIVIDUAL
DRAWING OBJECTS**

- **CREATING EFFECTS USING COLOR
COMBINATIONS**

**LESSON STRUCTURE: ENGAGEMENT (4 MINUTES) – CREATING A SIMPLE IMAGE USING DRAWING IN PYTHON
USING COLORS AND EFFECTS**

**CONVERSATION WITH STUDENTS ABOUT THE BASIC PRINCIPLES OF DRAWING AND WORKING WITH COLORS IN
PYTHON USING THE TKINTER LIBRARY**

- **RESEARCH (12 MINUTES) – EXPLORING THE NEW IMAGE FILL AND OUTLINE COMMANDS**
- **PRESENTATION (15 MINUTES) – SUMMARY AND EXPLANATION OF KNOWLEDGE FROM THE SECTION
EXPLORING THE ISSUE OF CREATING PROGRAMS FROM THE BASIC PRINCIPLES OF DRAWING AND WORKING
WITH COLORS IN PYTHON ELABORATION (9 MINUTES) – SUPPORTING STUDENTS IN FURTHER INDEPENDENT
DRAWING AND DISCOVERING THE POSSIBILITIES OF PROGRAMMING IN PYTHON USING COLORS**
- **ASSESSMENT (5 MINUTES) – SOLVING THE SELF-ASSESSMENT TEST, DISCUSSING THE ANSWERS.**

WORKSHEET FOR THE STUDENT

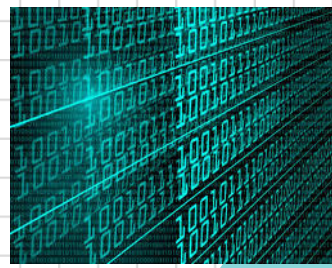
**1. WE WILL PROCEED ACCORDING TO OUR ELECTRONIC TEXTBOOK PROGRAMMING IN PYTHON, WHICH IS
INTENDED FOR SECONDARY SCHOOLS. YOU CAN FIND A SAMPLE TEXTBOOK AT:
[HTTP://WWW.PROGRAMUJEMEVPYTHONE.SK/UKAZKA.PDF](http://www.programujemevpythone.sk/ukazka.pdf) THE SAMPLE CONTAINS SIMPLE EXAMPLES OF
DRAWING IN COLOR:**

1. SHAPE FILLING:

**CIRCLE, RECTANGLE, SQUARE AND TRIANGLE OUTLINE OF DIFFERENT COLORS AND LINE THICKNESSES
CODE PREVIEW FOR FILL AND OUTLINE EXPOSURE PHASE – EXPOSURE OF NEW COMMANDS
– I WILL BRIEFLY NOTE THE BASIC TERMS ON THE BOARD**



LESSON PLAN



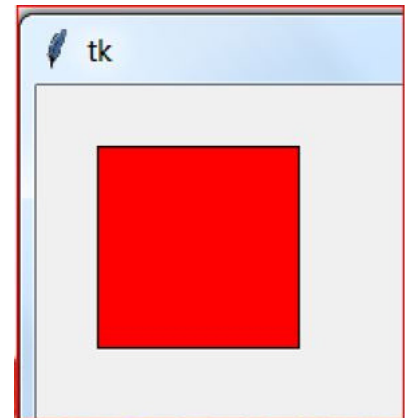
2. STUDENT WORKSHEET

1. Task: Create a new program 06lek_p02.py, draw a colored rectangle using the following code:

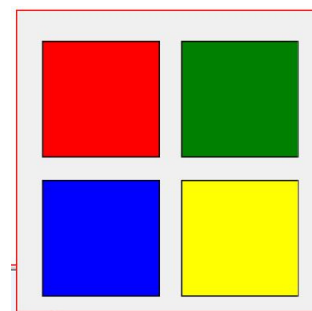
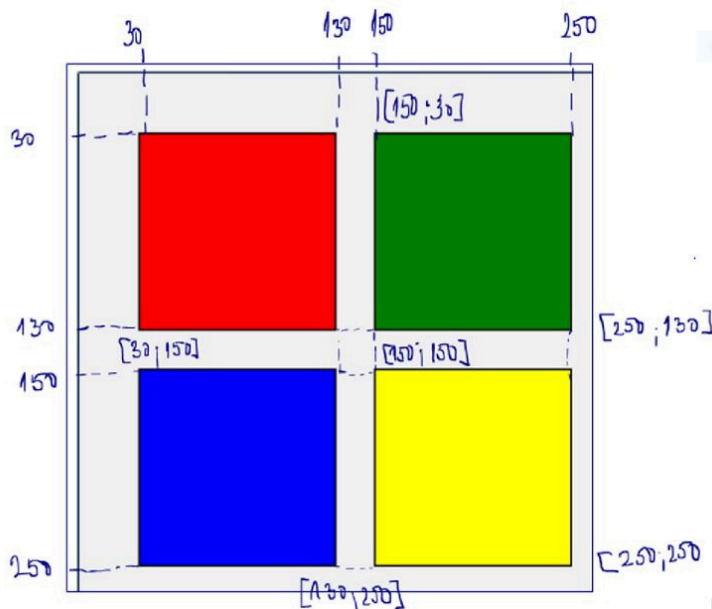
```
*06lek_p02.py - F:/14_Python(SŠ)/PythonSŠ_PracovnéListy/06_PL (riešenia)/06lek_p02.py (3.6.2rc1)*
File Edit Format Run Options Window Help
import tkinter
canvas = tkinter.Canvas()
canvas.pack()

canvas.create_rectangle(30, 30, 130, 130, fill='red')
```

The width (and height) of the square is 100

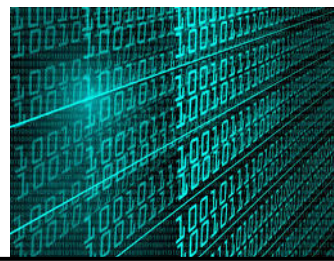


2. Task: Add 3 more commands for drawing rectangles to the program 06lek_p03.py to get the following image:
The width (and height) of the square is 100. The distance between the squares is 20





LESSON PLAN



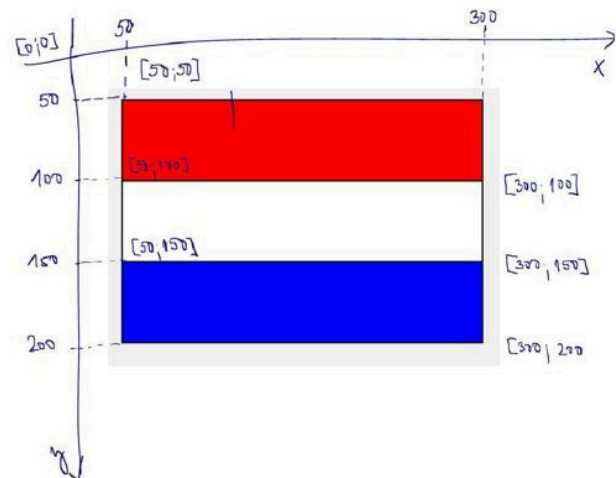
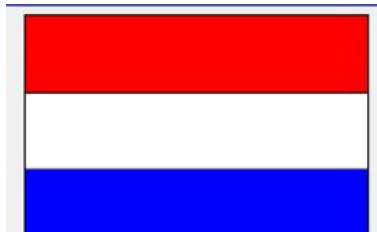
1

```
import tkinter
canvas = tkinter.Canvas()
canvas.pack()

canvas.create_rectangle(30, 30, 130, 130, fill='red')
canvas.create_rectangle(150, 30, 250, 130, fill='green')
canvas.create_rectangle(30, 150, 130, 250, fill='blue')
canvas.create_rectangle(150, 150, 250, 250, fill='yellow')
```

1.

3. Task: Create a new program 06lek_p04.py that draws the Dutch flag:



```
import tkinter
canvas = tkinter.Canvas()
canvas.pack()
```

```
canvas.create_rectangle(50, 50, 300, 100, fill='red')
canvas.create_rectangle(50, 100, 300, 150, fill='white')
canvas.create_rectangle(50, 150, 300, 200, fill='blue')
```

1.

4. Task: Create a program 06lek_p05.py that draws the Irish flag with the color 'orange':



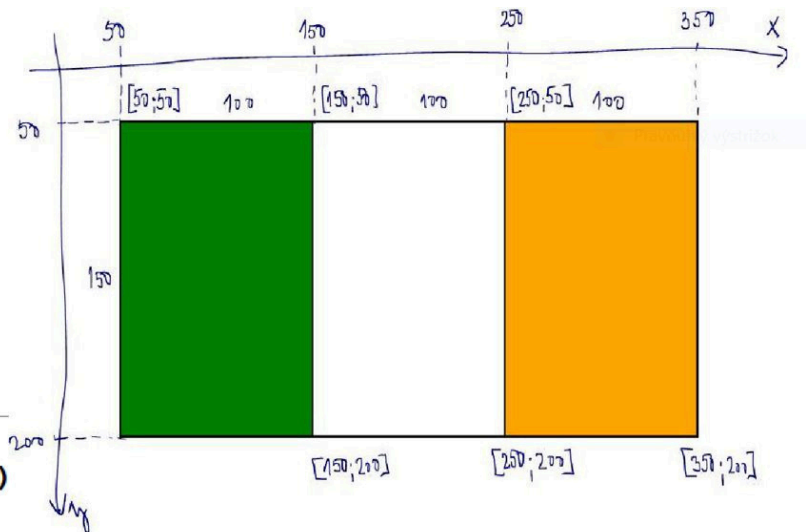


LESSON PLAN



1.

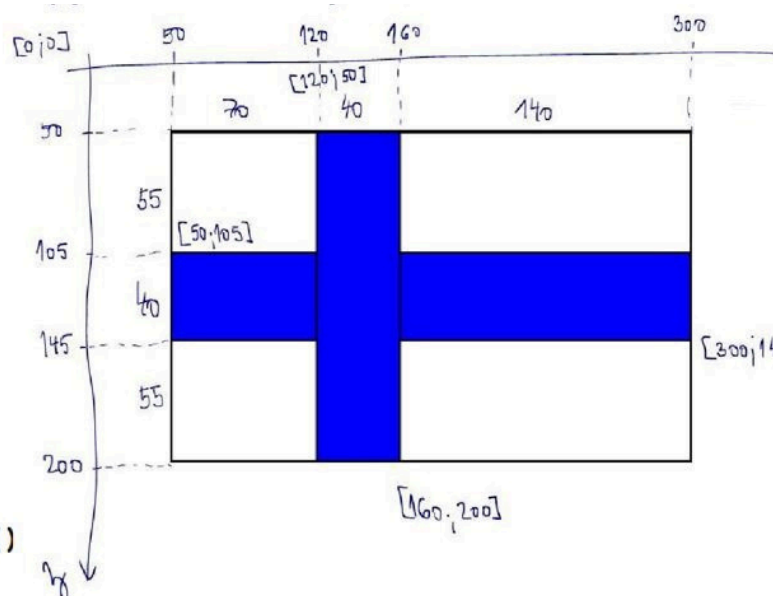
WORKSHEET FOR THE STUDENT



```
import tkinter
canvas = tkinter.Canvas()
canvas.pack()
```

```
canvas.create_rectangle(50, 50, 150, 200, fill='green')
canvas.create_rectangle(150, 50, 250, 200, fill='white')
canvas.create_rectangle(250, 50, 350, 200, fill='orange')
```

1. 5. Task: Create a new program 06lek_p06.py and draw a picture like this: Which country's flag does the picture resemble?

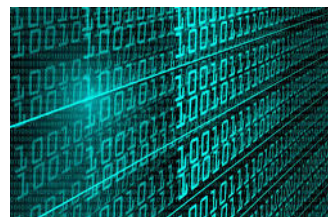


```
import tkinter
canvas = tkinter.Canvas()
canvas.pack()
```

```
canvas.create_rectangle(50, 50, 300, 200, fill='white')
canvas.create_rectangle(50, 105, 300, 145, fill='blue')
canvas.create_rectangle(120, 50, 160, 200, fill='blue')
```

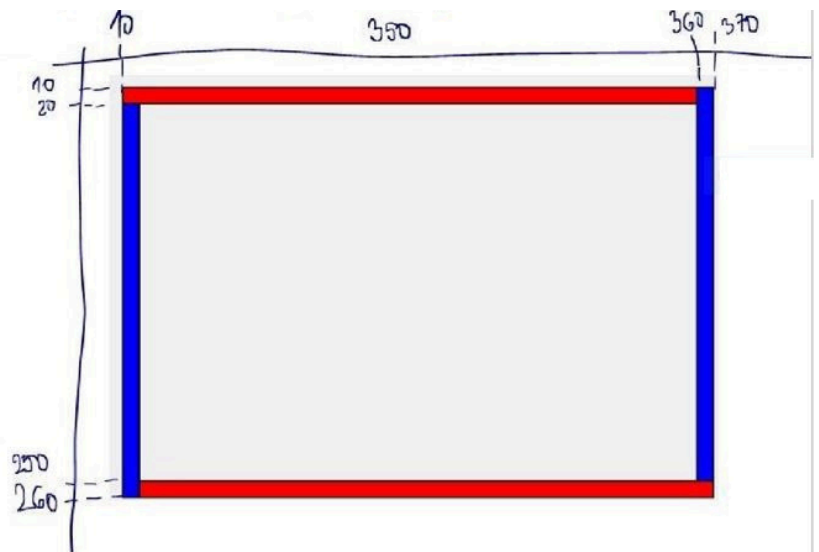


LESSON PLAN



1. WORKSHEET FOR THE STUDENT

1. 6. Task: • Create a new program 06lek_p07.py, in which draw such a frame from four narrow rectangles:



```
import tkinter
canvas = tkinter.Canvas()
canvas.pack()

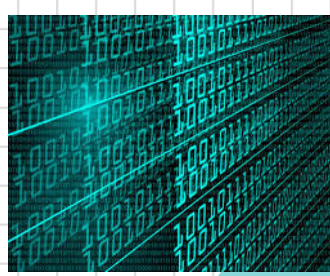
canvas.create_rectangle(10, 10, 360, 20, fill='red')
canvas.create_rectangle(10, 20, 20, 260, fill='blue')
canvas.create_rectangle(20, 250, 370, 260, fill='red')
canvas.create_rectangle(360, 10, 370, 250, fill='blue')
```

1. WHEN DRAWING IN PYTHON, WE HAVE THE OPPORTUNITY TO USE MANY COLORS. HERE IS A SELECTION OF SOME OF THEM:

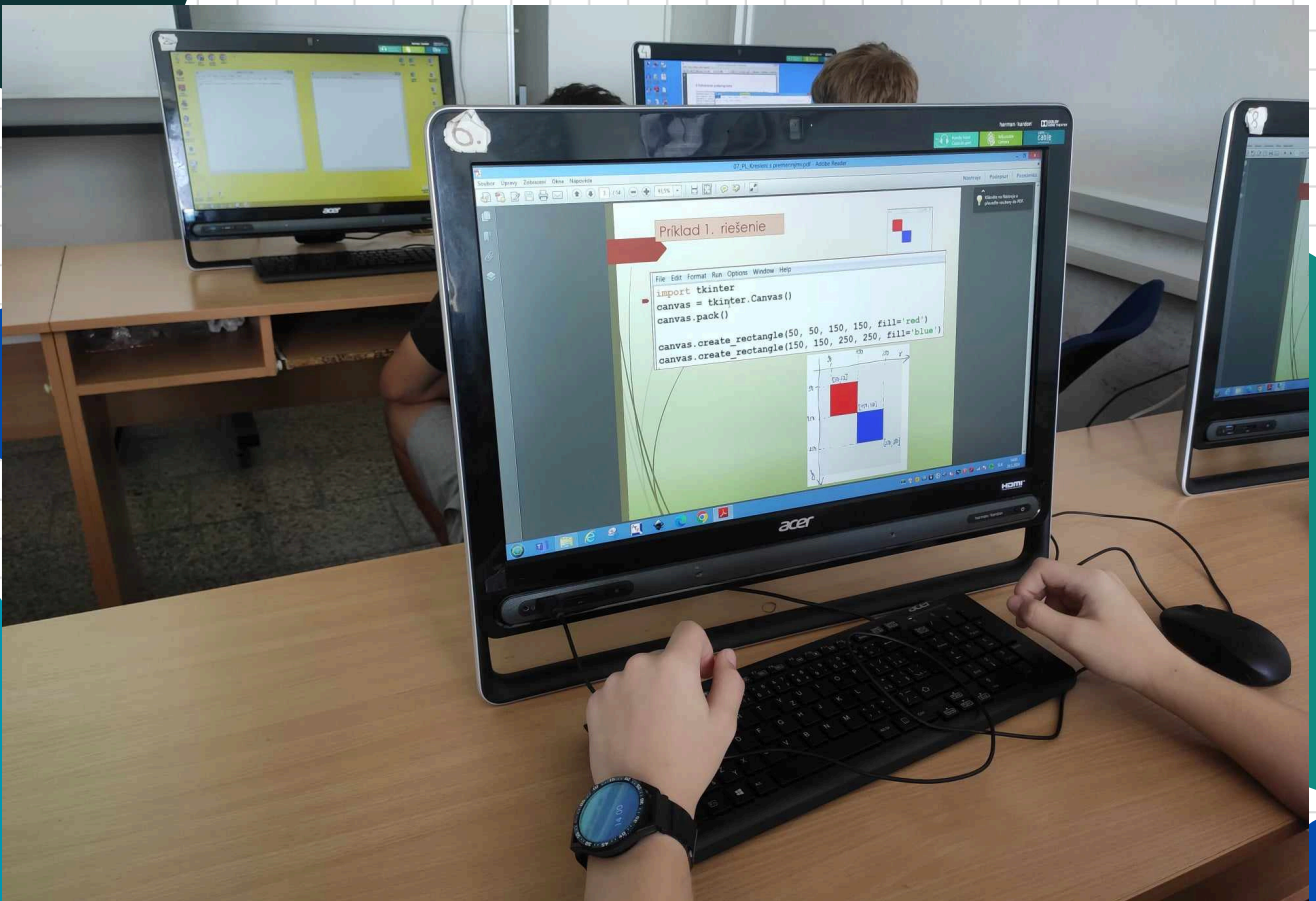
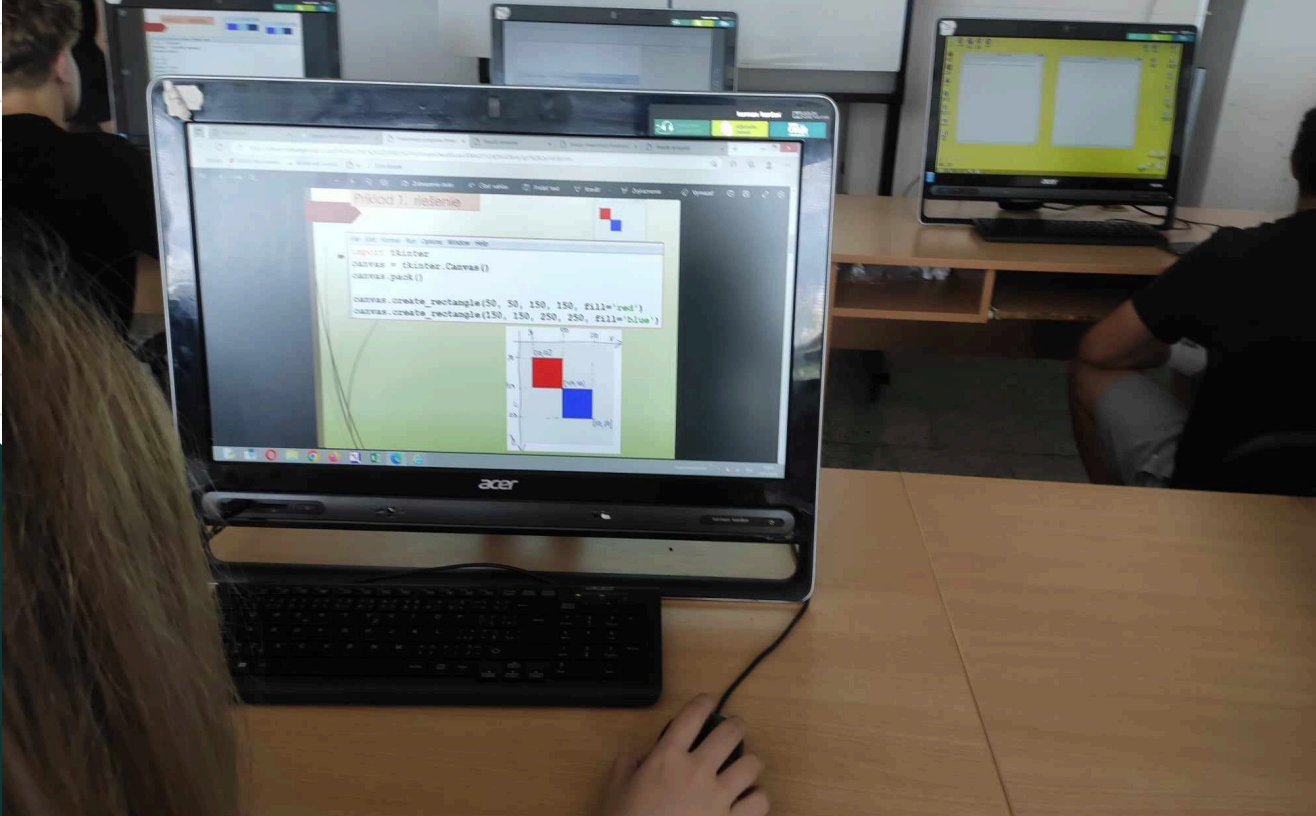
Blue	LightBlue	Cyan	SkyBlue	CornFlowerBlue	DeepSkyBlue	DodgerBlue
RoyalBlue	SlateBlue	SteelBlue	MediumBlue	Navy	Red	SandyBrown
Salmon	Coral	Tomato	Orange	DarkOrange	OrangeRed	IndianRed
Chocolate	Tan	Maroon	Sienna	Brown	SaddleBrown	Pink
Plum	Violet	Orchid	Magenta	Purple	DarkMagenta	Green
PaleGreen	YellowGreen	MediumSeaGreen	LawnGreen	LimeGreen	ForestGreen	DarkGreen
Yellow	Khaki	Gold	Gray	LightGray	Black	White



ANNEX

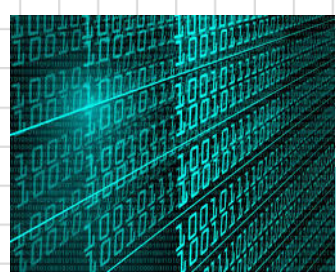


PHOTOS FROM THE LESSON IMPLEMENTATIONS

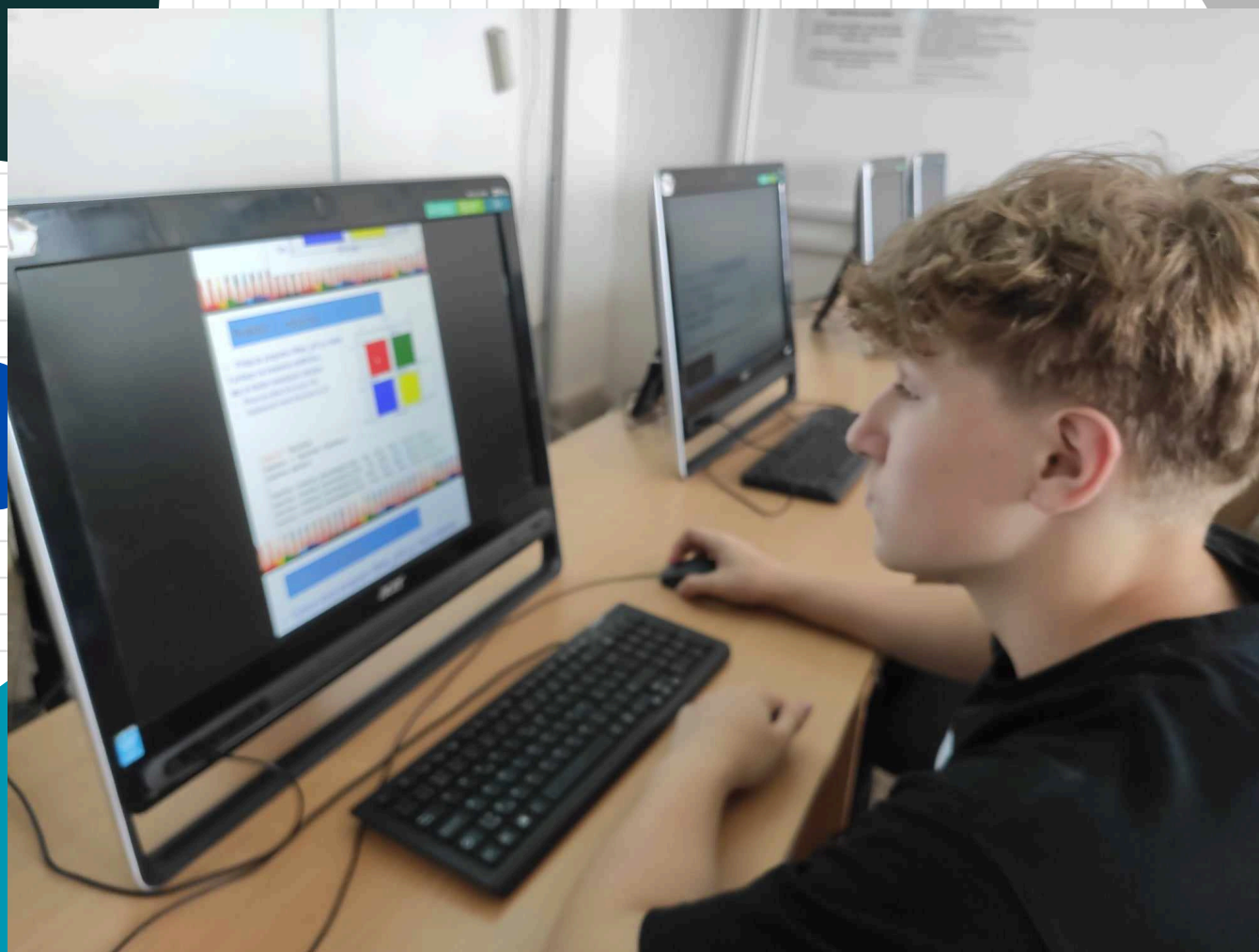
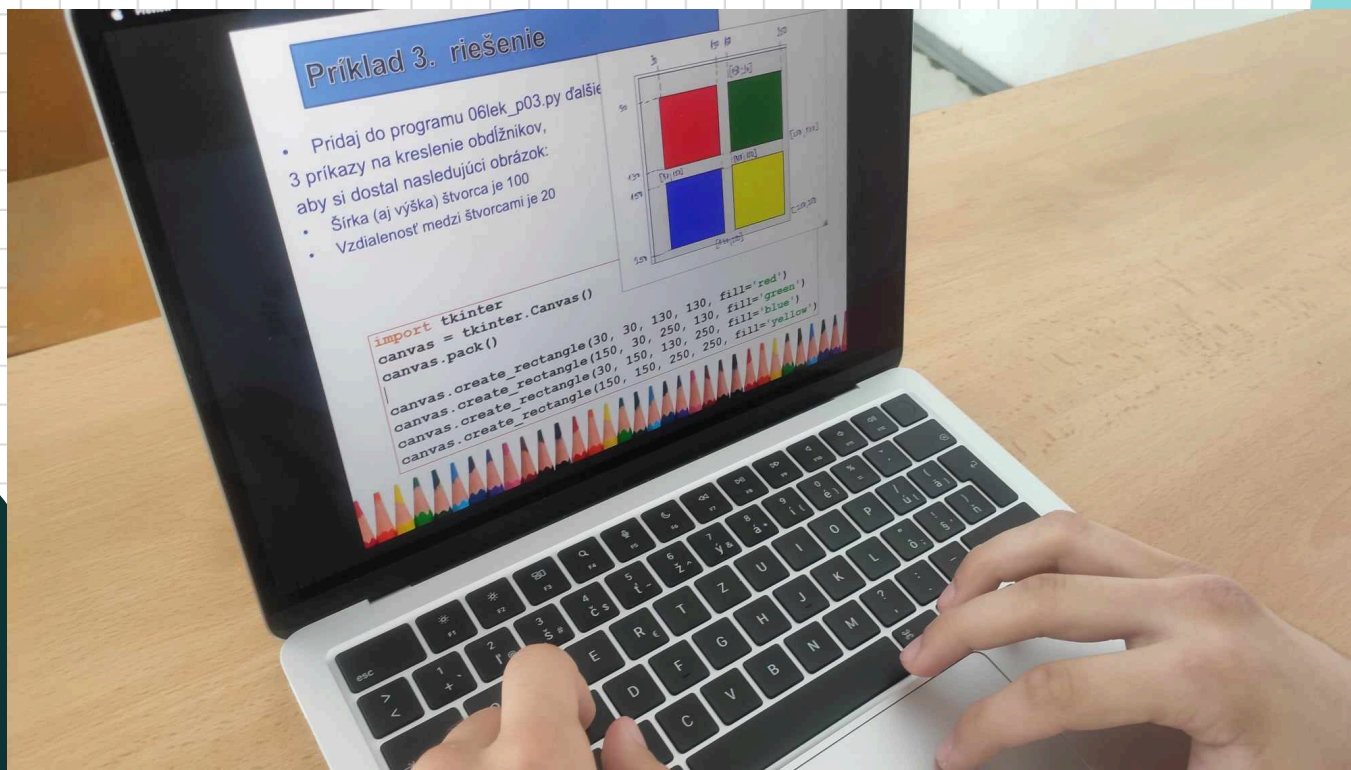




ANNEX

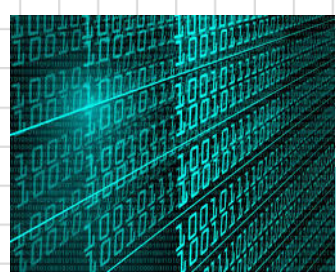


PHOTOS FROM THE LESSON IMPLEMENTATIONS

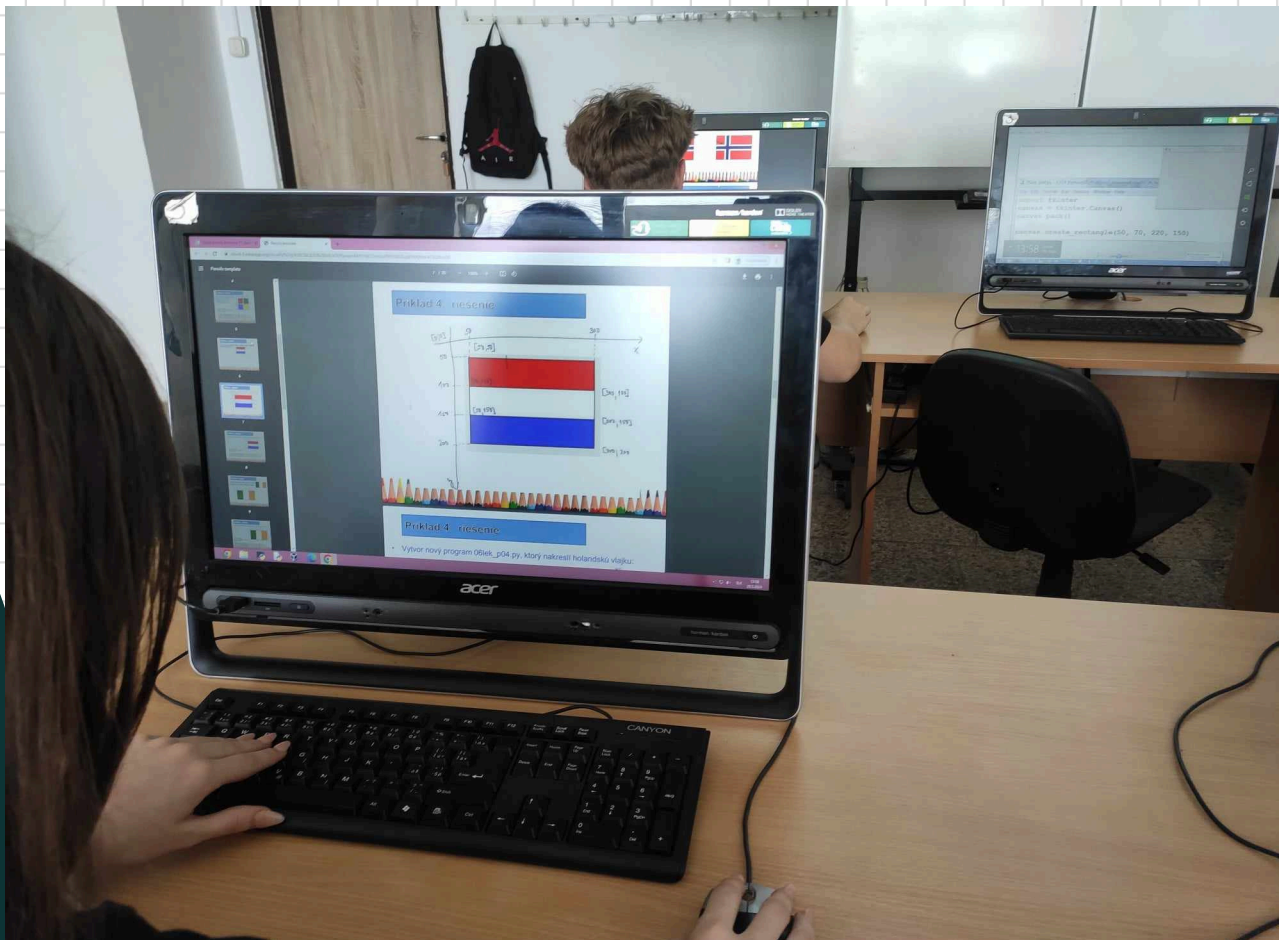




ANNEX

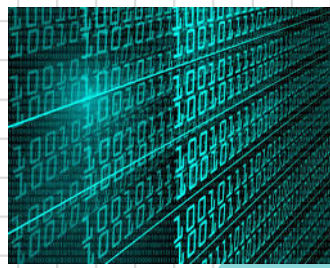


PHOTOS FROM THE LESSON IMPLEMENTATIONS





ANNEX



STUDENTS OUTPUT

```
IMPORT TKINTER
```

```
# WE WILL CREATE A WINDOW WINDOW = TKINTER.TK()  
WINDOW.TITLE("THREE RECTANGLES")
```

```
# WE SET THE WINDOW SIZE WIDTH = 400 HEIGHT = 200  
WINDOW.GEOMETRY(F"{WIDTH}X{HEIGHT}")
```

```
# WE WILL CREATE A CANVAS CANVAS = TKINTER.CANVAS(WINDOW,  
WIDTH=WIDTH, HEIGHT=HEIGHT) CANVAS.PACK()
```

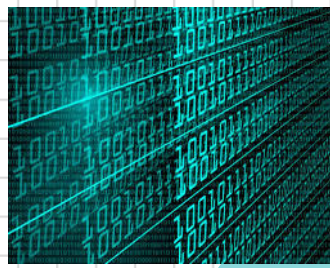
```
# FIRST RECTANGLE X1 = 50 Y1 = 20 X2 = 150 Y2 = 80 COLOR1 =  
"RED" CANVAS.CREATE_RECTANGLE(X1, Y1, X2, Y2, FILL=COLOR1)
```

```
# SECOND RECTANGLE X2 = 200 Y2 = 50 COLOR2 = "BLUE"  
CANVAS.CREATE_RECTANGLE(X1, Y1, X2, Y2, FILL=COLOR2)
```

```
# THIRD RECTANGLE X1 = 100 Y1 = 100 X2 = 250 Y2 = 150 COLOR3 =  
"GREEN" CANVAS.CREATE_RECTANGLE(X1, Y1, X2, Y2, FILL=COLOR3)
```




LESSON PLAN



SUBJECT: PROGRAMMING

TOPIC: PROGRAMMING IN PYTHON – DRAWING WITH VARIABLES.

LEARNING OBJECTIVES:.

- **1. UNDERSTAND THE BASIC PRINCIPLES OF DRAWING IN PYTHON USING VARIABLES.**
- **2. TO LEARN HOW TO USE VARIABLES WHEN DRAWING IN PYTHON.**

TOOLS:

**A COMPUTER WITH PYTHON AND THE TKINTER LIBRARY INSTALLED.
PYTHON SAMPLE TEXTBOOK WORKSHEET
WITH EXERCISES**

**INTRO: USING VARIABLES WHEN DRAWING:
DEFINING AND USING VARIABLES IN PYTHON
USE VARIABLES TO DRAW SHAPES.
SET THE COLOR AND SIZE OF THE SHAPES.**

LESSON STRUCTURE: ENGAGEMENT (4 MINUTES) – CREATING A PICTURE USING DRAWING

- **IN PYTHON USING A VARIABLE CONVERSATION WITH STUDENTS ABOUT THE BASIC PRINCIPLES OF DRAWING AND WORKING WITH COLORS IN PYTHON USING THE TKINTER LIBRARY**
- **RESEARCH (12 MINUTES) – EXPLORING NEW VARIABLE COMMANDS FOR IMAGE FILL AND OUTLINE**
- **INTERPRETATION (15 MINUTES) – SUMMARIZING AND WRITING A SIMPLE PROGRAM IN PYTHON THAT DRAWS DIFFERENT SHAPES WITH DIFFERENT COLORS AND SIZES ELABORATION (9 MINUTES) – SUPPORTING STUDENTS IN FURTHER INDEPENDENT DRAWING OF PYTHON PROGRAMMING POSSIBILITIES USING VARIABLES**
- **ASSESSMENT (5 MINUTES) – SOLVING THE SELF-ASSESSMENT TEST, DISCUSSING THE ANSWERS.**

WORKSHEET FOR THE STUDENT

**1. WE WILL PROCEED ACCORDING TO OUR ELECTRONIC TEXTBOOK PROGRAMMING IN PYTHON, WHICH IS INTENDED FOR SECONDARY SCHOOLS. YOU CAN FIND A SAMPLE TEXTBOOK AT:
[HTTP://WWW.PROGRAMUJEMEVPYTHONE.SK/UKAZKA.PDF](http://www.programujemevpythone.sk/ukazka.pdf) THE SAMPLE CONTAINS SIMPLE EXAMPLES OF DRAWING WITH A VARIABLE:**

- 1. A PROGRAM THAT DRAWS A SQUARE, RECTANGLE, TRIANGLE AND CIRCLE. THE COORDINATES OF THE UPPER LEFT CORNER OF THE OBJECT ARE STORED IN THE VARIABLES X, Y.**
 - 2. CODE SAMPLE WITH VARIABLES FOR FILL AND OUTLINE DIFFERENT COLORS AND LINE WEIGHTS EXPOSE PHASE – EXPOSURE OF NEW COMMANDS WITH VARIABLES**
- I WILL BRIEFLY NOTE THE TERM MAP OF BASIC CONCEPTS ON THE BOARD**

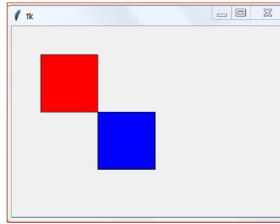


LESSON PLAN



3. STUDENT WORKSHEET

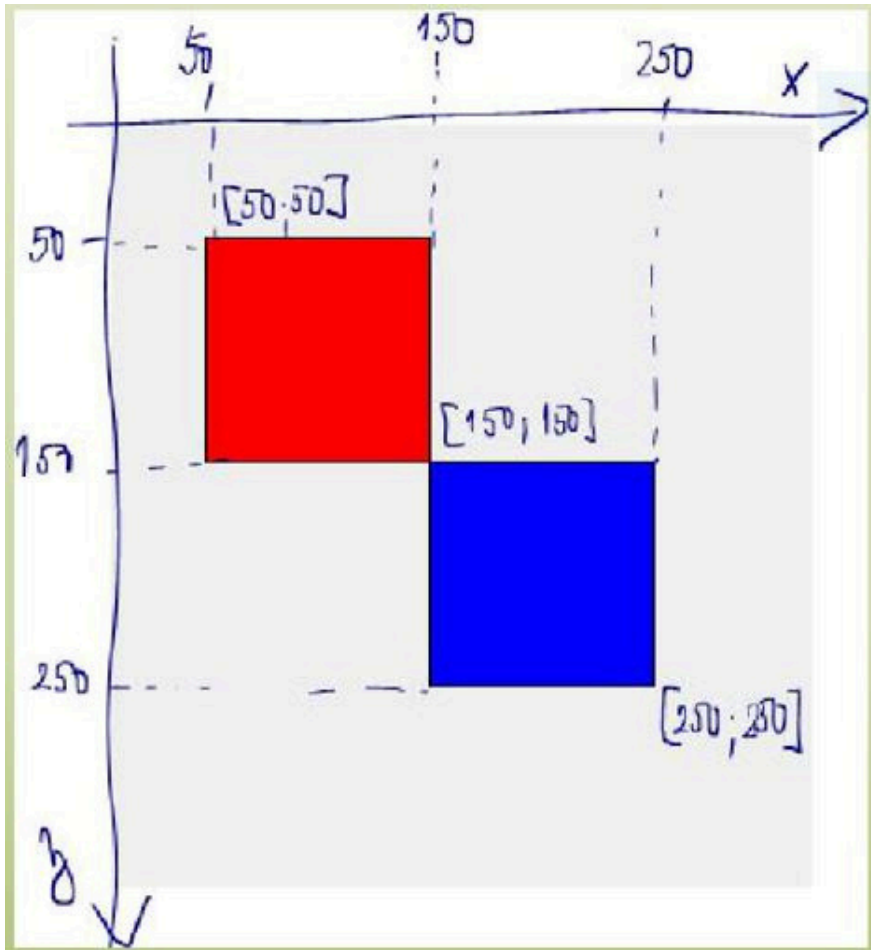
1.Task: Create a new program 07lek_p01.py that will draw two touching squares as shown in the picture. Both have a side length of 100, with the red one having its upper left corner at the point [50, 50].



File Edit Format Run Options Window Help

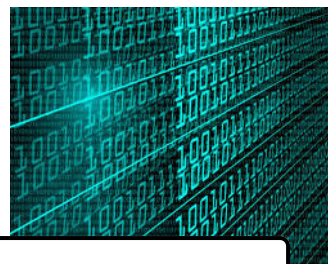
```
import tkinter
canvas = tkinter.Canvas()
canvas.pack()

canvas.create_rectangle(50, 50, 150, 150, fill='red')
canvas.create_rectangle(150, 150, 250, 250, fill='blue')
```





LESSON PLAN



1.

3. STUDENT WORKSHEET

1.

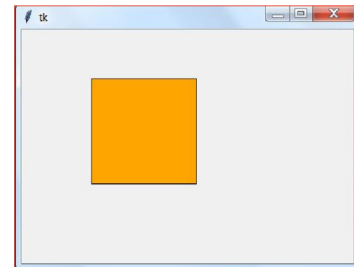
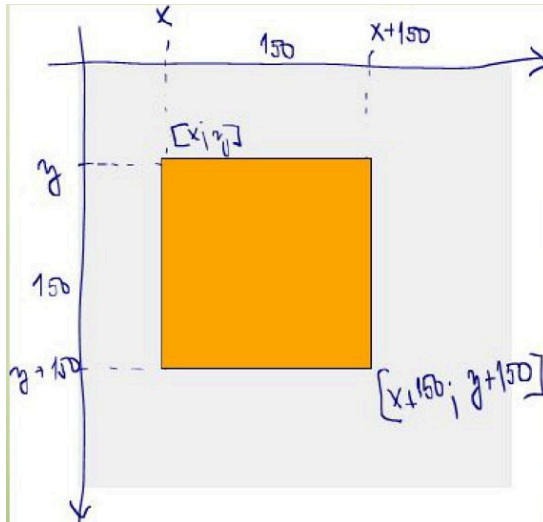
2. Task: Create a new program 07lek_p02.py that draws a square with side length 150.

2.

The variables x, y store the coordinates of the upper left corner of the square.

3.

The coordinates of the upper left corner $[x;y] = [100;70]$ are entered as variables.

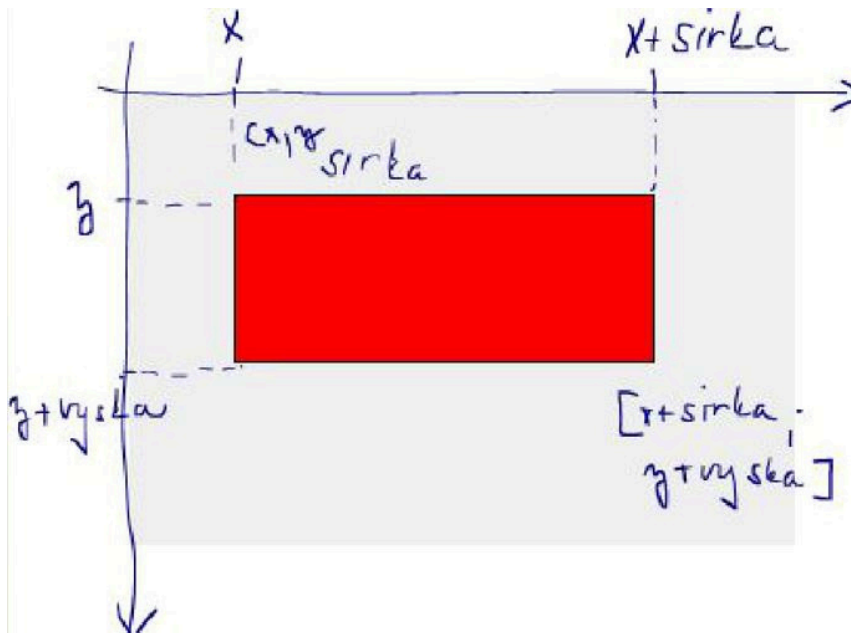


```
import tkinter
canvas = tkinter.Canvas()
canvas.pack()

x = 100
y = 70
canvas.create_rectangle(x, y, x + 150, y + 150, fill='orange')
```

1. 3. Task: Create a program 07lek_p03.py that draws a rectangle with the upper left corner on x, y coordinates, given width and height. Choose the color as you like.

2. ' For example, when the program contains: $x = 100, y = 70, \text{width} = 200, \text{height} = 80$, a rectangle is drawn as in the picture:





LESSON PLAN



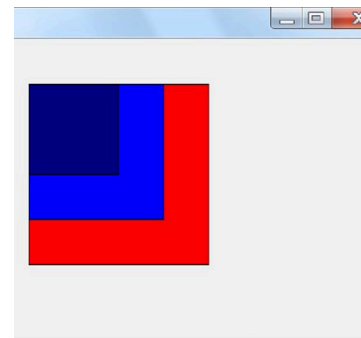
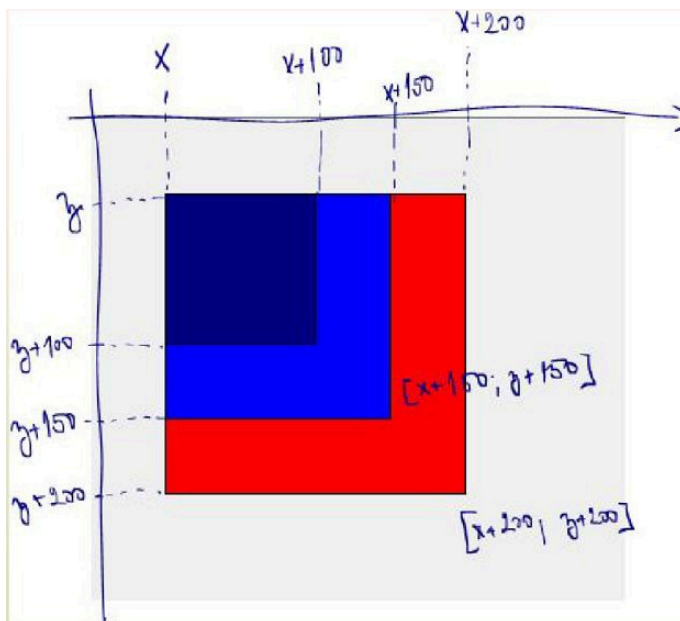
1. 3. STUDENT WORKSHEET

File Edit Format Run Options Window Help

```
import tkinter
canvas = tkinter.Canvas()
canvas.pack()

x = 100
y = 70
sirka = 200
vyska = 80
canvas.create_rectangle(x, y, x + sirka, y + vyska, fill='red')
```

4. Task: Create a program 07lek_p04.py that draws squares as shown in the picture. These squares have a common upper left corner, whose coordinates are in the variables x, y.
- The squares get progressively smaller so that red has a side length of 200, blue 150 and dark blue 100.
- The coordinates of the center can be, for example: [x;y] = [80;50]



File Edit Format Run Options Window Help

```
import tkinter
canvas = tkinter.Canvas()
canvas.pack()

x = 80
y = 50
canvas.create_rectangle(x, y, x+200, y+200, fill='red')
canvas.create_rectangle(x, y, x+150, y+150, fill='blue')
canvas.create_rectangle(x, y, x+100, y+100, fill='navy')
```



LESSON PLAN

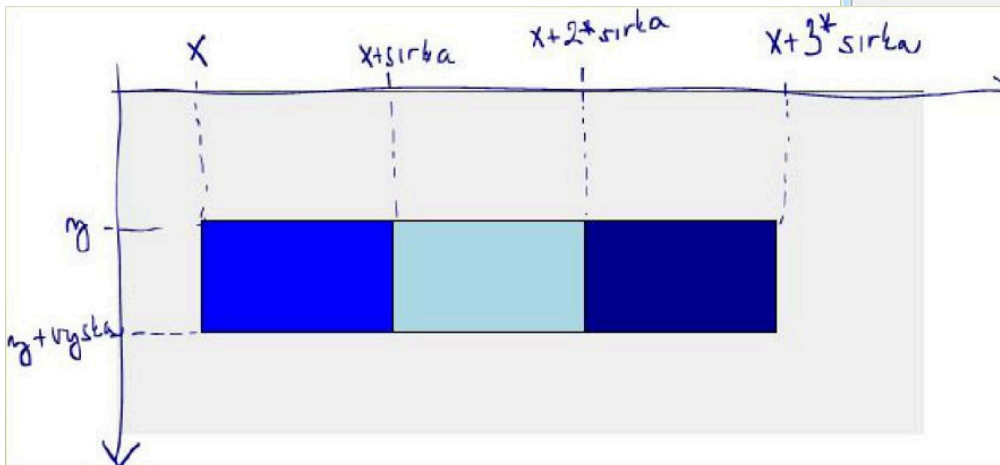
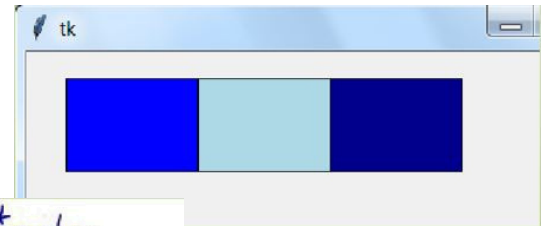


1. 3. STUDENT WORKSHEET

1. 5. Task: Create a program 07lek_p05.py that draws three rectangles touching each other:
2. The coordinates of the upper left corner of the first rectangle are stored in the variables x, y.

1. All three rectangles have the same width and height - these dimensions are stored in the width, height variables.

1. Will the program work correctly even if you increase the value of the variable width by 20 and the value of the variable y by 10? If not, the repair program.



File Edit Format Run Options Window Help

```
import tkinter
canvas = tkinter.Canvas()
canvas.pack()

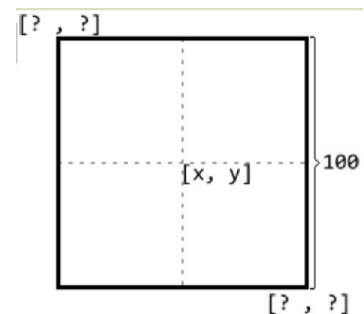
x = 30
y = 20
sirka = 100
vyska = 70
canvas.create_rectangle(x, y, x+sirka, y+vyska, fill='blue')
canvas.create_rectangle(x+sirka, y, x+2*sirka, y+vyska, fill='light blue')
canvas.create_rectangle(x+2*sirka, y, x+3*sirka, y+vyska, fill='dark blue')
```

Task: Create a new program 07-lek_p06 .py that draws a square whose center has coordinates x, y and sides have length 100.

Coordinates x, y are stored in variables of the same name.

In order to draw this square, you have to calculate the coordinates of its upper left and lower right corners.

In a new program, write code that draws a green square with center [x, y] and side length 100.





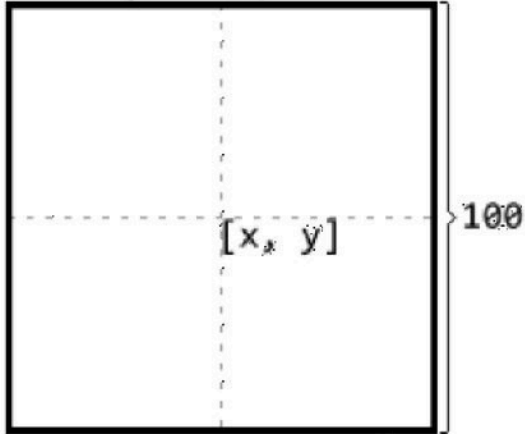
LESSON PLAN



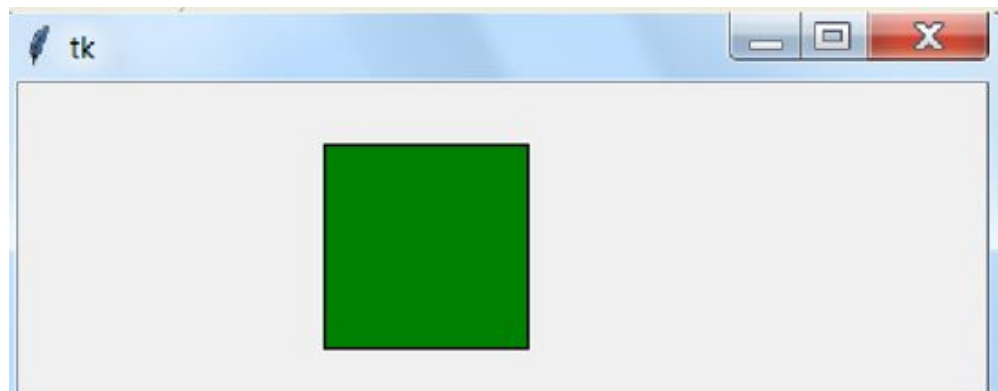
1.

3. STUDENT WORKSHEET

$[x - 50, y - 50]$



$[x + 50, y + 50]$



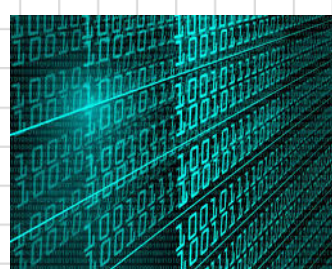
File Edit Format Run Options Window Help

```
import tkinter
canvas = tkinter.Canvas()
canvas.pack()

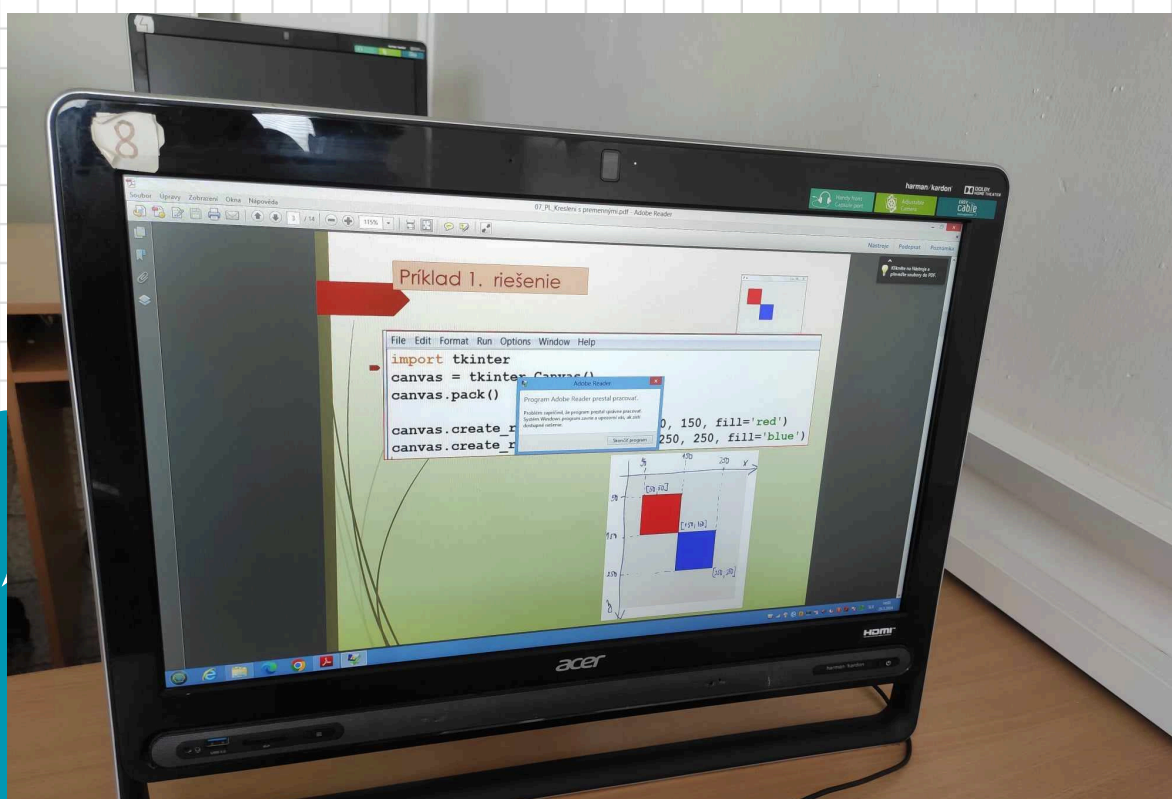
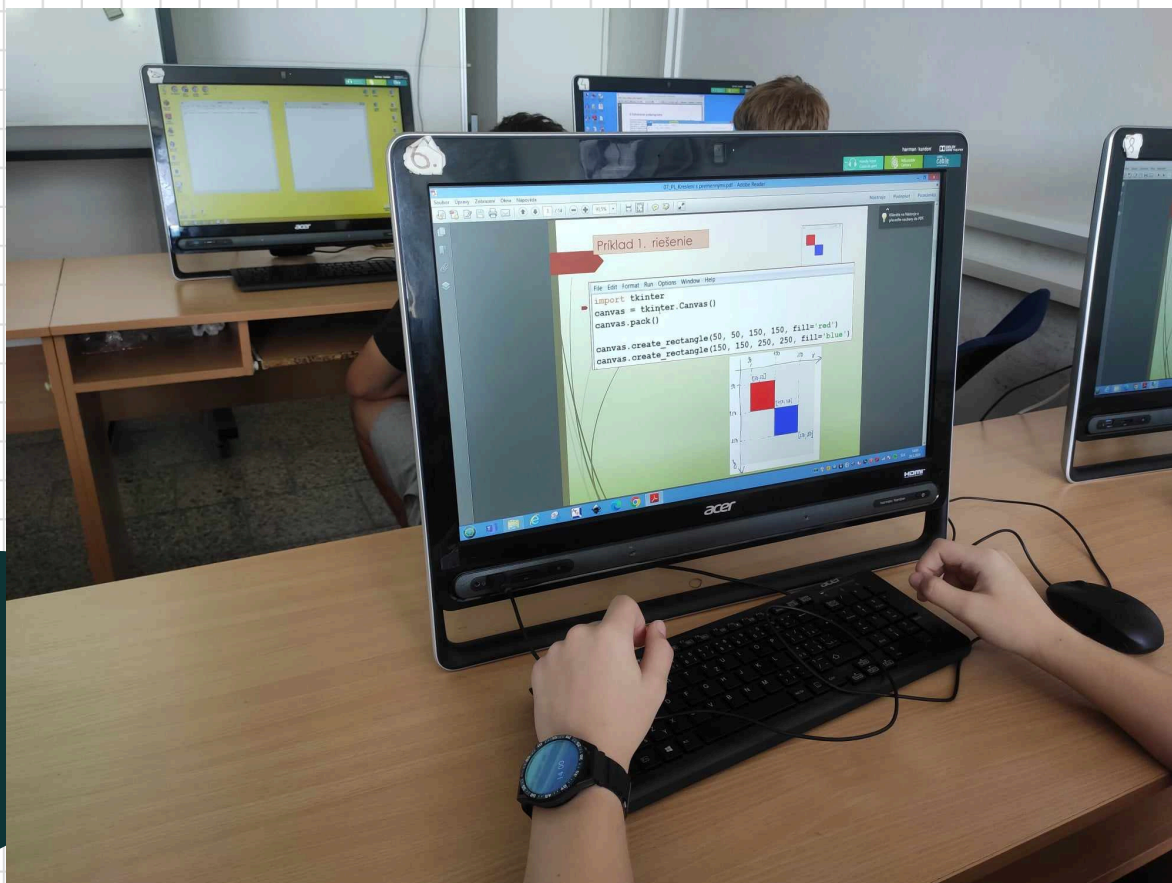
x = 200
y = 80
canvas.create_rectangle(x-50,y-50,x+50,y+50,fill='green')
```




ANNEX

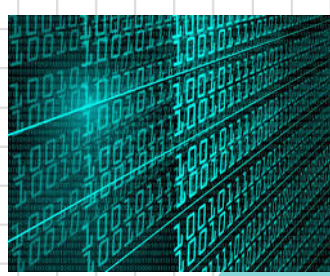


PHOTOS FROM THE LESSON IMPLEMENTATIONS

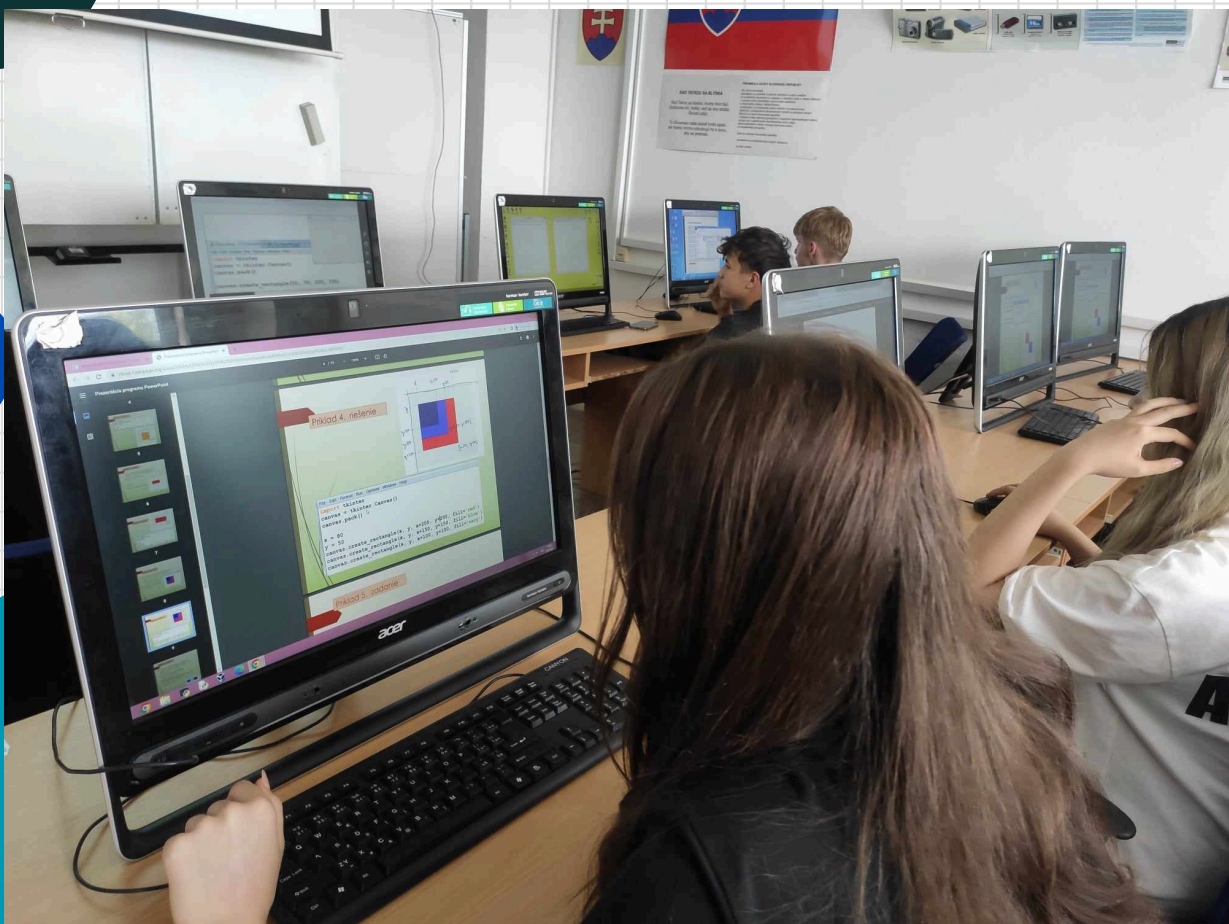
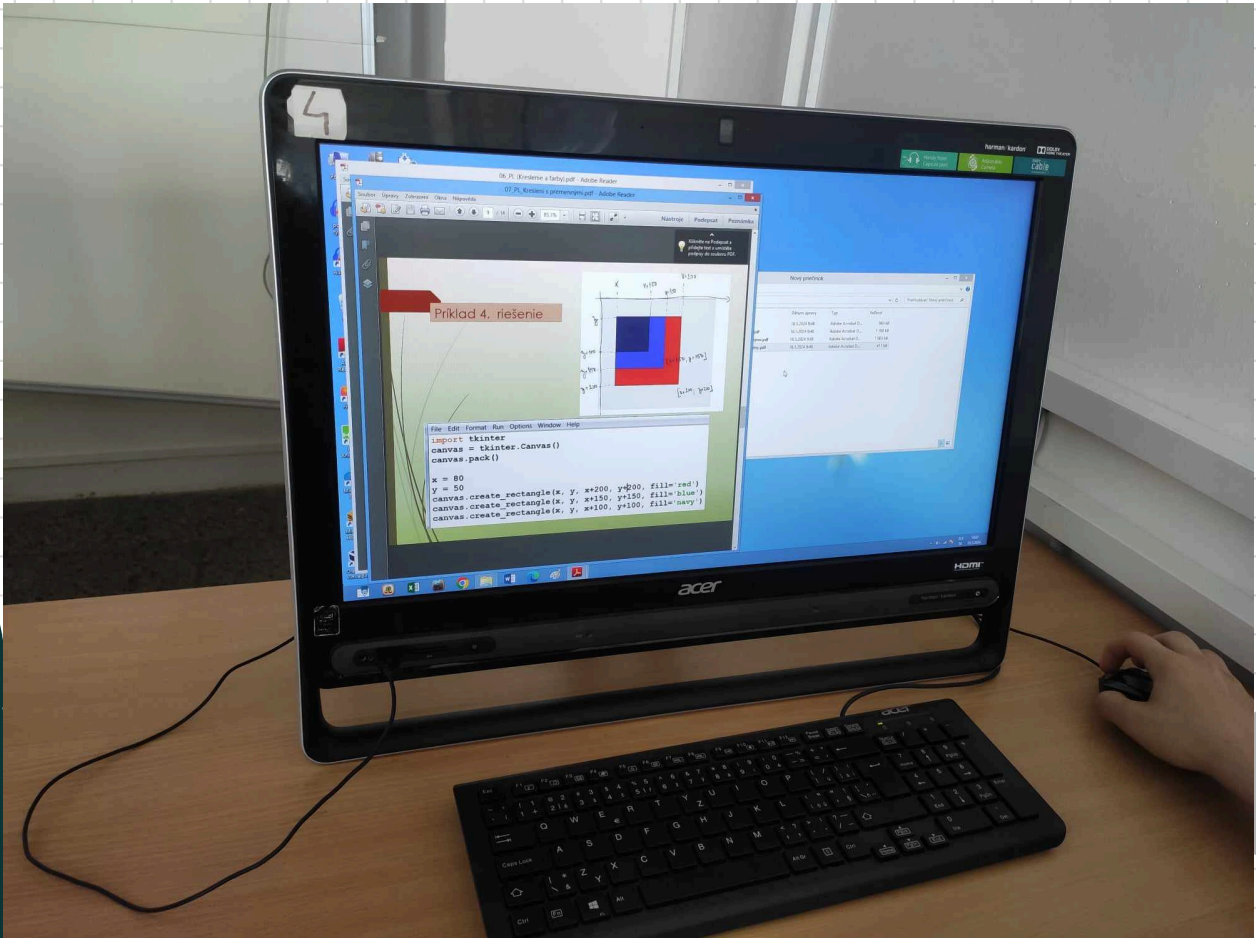




ANNEX

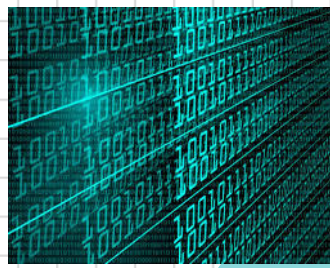


PHOTOS FROM THE LESSON IMPLEMENTATIONS





ANNEX



STUDENTS OUTPUT:

**WRITE A PROGRAM THAT READS THREE WORDS AND PRINTS ALL
6 DIFFERENT PERMUTATIONS. AFTER LAUNCH YOU CAN GET:**

ENTER THE 1ST WORD: WHITE

ENTER THE 2ND WORD: BLUE

ENTER THE 3RD WORD: RED

WHITE BLUE RED

WHITE RED BLUE

BLUE WHITE RED

BLUE RED WHITE

RED WHITE BLUE

RED BLUE WHITE

OR USE VARIABLES:

BEHIND 1ST GOODBYE: X

AT THE BACK 2ND GOODBYE: ***

BEHIND 3RD GOODBYE: :-)

X * :-)**

X :-) ***

***** X :-)**

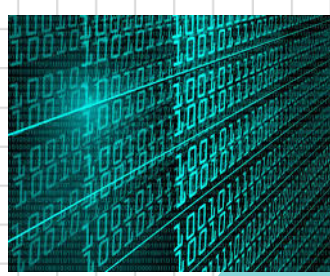
***** :-) X**

:-) X ***

:-) * X**



LESSON PLAN



SUBJECT: PROGRAMMING

TOPIC: PROGRAMMING IN PYTHON – SUBROUTINES.

LEARNING OBJECTIVES:.

- **1. USE SUBROUTINES TO CLARIFY DRAWING COMMANDS IN PYTHON**
- **2. LEARN TO DEFINE FUNCTIONS WHILE DRAWING IN PYTHON.**

TOOLS:

**A COMPUTER WITH PYTHON AND THE TKINTER LIBRARY INSTALLED.
PYTHON SAMPLE TEXTBOOK WORKSHEET
WITH EXERCISES**

**INTRO: USING DIFFERENT COLORS IN DRAWING:
DEFINING FUNCTIONS**

- **USING A SUBROUTINE FOR INDIVIDUAL DRAWING OBJECTS**
- **MODIFY THE FUNCTION SO THAT WE CAN SPECIFY A LOCATION WHEN USING IT**
- **REPEAT COMMANDS IN A LOOP**

LESSON STRUCTURE: CONNECTION (4 MINUTES) – A SUBROUTINE IS A SMALLER SECTION OF THE PROGRAM THAT WE NAME, AND WHEN WE WANT TO USE IT, WE JUST WRITE THE NAME OF THIS SUBROUTINE IN THE PROGRAM.

CONVERSATION WITH STUDENTS ABOUT THE ADVANTAGES OF USING SUBPROGRAMS IN PYTHON: – USE OF SUBPROGRAMS WHEN DRAWING MORE COMPLICATED PICTURES, IT IS SUITABLE SO THAT THE PROGRAM DOES NOT BECOME CONFUSING, BECAUSE IT CONTAINS MANY COMMANDS. THE USE OF COMMENTS HELPS US MAKE THE PROGRAM MORE TRANSPARENT. A BETTER WAY IS TO USE A SUBROUTINE.

- **RESEARCH (12 MINUTES) – EXPLORING NEW COMMANDS – A FUNCTION IS A NAMED BLOCK OF COMMANDS (SOMETIMES CALLED A SUBROUTINE). WE DESCRIBE (DEFINE) IT WITH A SPECIAL CONSTRUCTION**
- **PRESENTATION (15 MINUTES) – SUMMARY AND EXPLANATION OF KNOWLEDGE FROM THE SECTION**
- **INVESTIGATING THE ISSUE OF CREATING SUBPROGRAMS IN PYTHON DEVELOPMENT (9 MINUTES) – SUPPORTING STUDENTS IN FURTHER INDEPENDENT CREATION OF PYTHON SUBPROGRAMS**
- **ASSESSMENT (5 MINUTES) – SOLVING THE SELF-ASSESSMENT TEST, DISCUSSING THE ANSWERS.**

WORKSHEET FOR THE STUDENT

WE WILL PROCEED ACCORDING TO OUR ELECTRONIC TEXTBOOK PROGRAMMING IN PYTHON, WHICH IS INTENDED FOR SECONDARY SCHOOLS. YOU CAN FIND A SAMPLE TEXTBOOK AT:

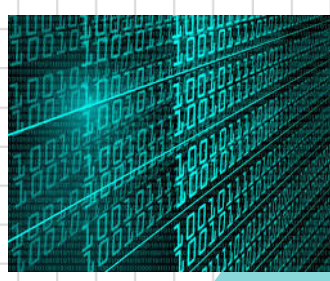
[HTTP://WWW.PROGRAMUJEMEVPYTHONE.SK/UKAZKA.PDF](http://www.programujemevpythone.sk/ukazka.pdf) THE SAMPLE CONTAINS SIMPLE EXAMPLES OF CREATING SUBPROGRAMS IN PYTHON:

1. WE WILL CREATE OUR OWN COMMANDS THAT WE CAN USE IN OUR PROGRAM. PROFESSIONALLY, SUCH SUBROUTINES IN PYTHON ARE CALLED FUNCTIONS (IN OTHER PROGRAMMING LANGUAGES YOU MAY ALSO ENCOUNTER THE NAME OF PROCEDURE OR FUNCTION, OR BOTH).

EXPOSE PHASE – PUSHING THE STATEMENTS AWAY FROM THE LEFT MARGIN HELPS PYTHON UNDERSTAND WHAT ALL GOES INTO OUR FUNCTION. – I WILL BRIEFLY NOTE THE BASIC TERMS ON THE BOARD



LESSON PLAN



4. STUDENT WORKSHEET

1. Task: Draw a house and create a function (command) house for it.

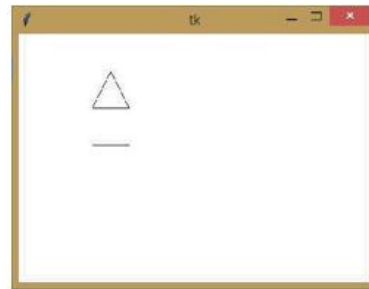
```
import tkinter
canvas = tkinter.Canvas()
canvas.pack()
canvas.create_rectangle(100, 100, 150, 150)
canvas.create_line(100, 100, 125, 50, 150, 100)
```

The program draws a house with a roof from a rectangle and a broken line. In the next example, we defined our own function called house. We define the function with the word `def`, followed by the name of the function, then parentheses and at the end a colon (just like in the for loop).

```
import tkinter
canvas = tkinter.Canvas()
canvas.pack()
def dom():
    canvas.create_rectangle(100, 100, 150, 150)
    canvas.create_line(100, 100, 125, 50, 150, 100)
```

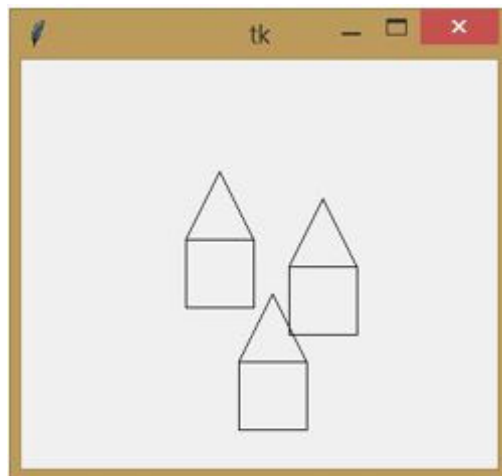
If we want to execute this function, we write `dom()` in the program (in the place of the program where the function is to be executed).

```
import tkinter
canvas = tkinter.Canvas()
canvas.pack()
def dom():
    canvas.create_rectangle(100, 100, 150, 150)
    canvas.create_line(100, 100, 125, 50, 150, 100)
dom()
```



This program also starts the `dom()` function. We can better see the importance of defining a custom command if we change the house function to draw a house at a random location.

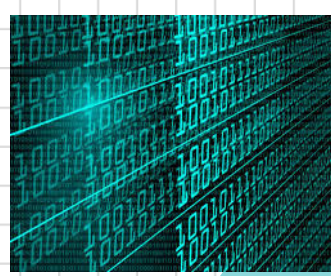
```
import tkinter
from random import *
canvas = tkinter.Canvas()
canvas.pack()
def dom():
    x = randint(100, 300)
    y = randint(100, 300)
    canvas.create_rectangle(x, y, x+50, y+50)
    canvas.create_line(x, y, x+25, y-50, x+50, y)
dom()
dom()
dom()
```



In this program, we will already execute the `house()` function three times, and thus three houses will be drawn at a random location.



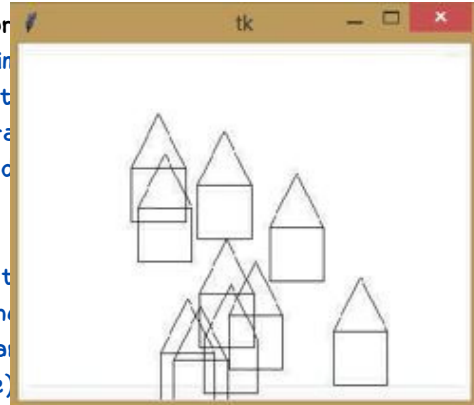
LESSON PLAN



4. STUDENT WORKSHEET

2.Task: Use the for loop, so you can easily draw 10 houses.

```
import random
from random import *
canvas = tkinter.Canvas()
x = randint(100, 300) y = randint(100, 300)
canvas.create_rectangle(x, y, x+50, y+50)
for i in range(10):
    canvas.create_line(x, y, x+25, y-50, x+50, y)
```

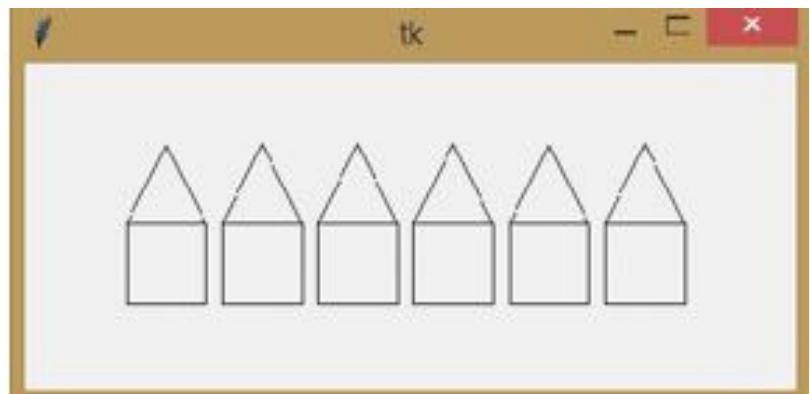
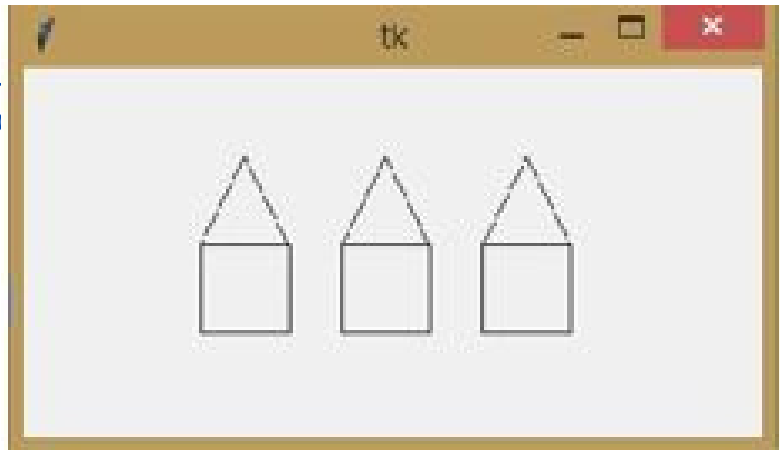


The function can also be modified so that when using it, we can enter the parameters. When defining a function with a parameter in parentheses, separated by a comma (if we want to have more than one), some kind of variables that can only be used in this function) and they have the value that we entered when using the function.

Similar to using parameters in a function (like in data) that affect the function. Basically they are variables that can only be used in this function and they have the value that we entered when using the function.

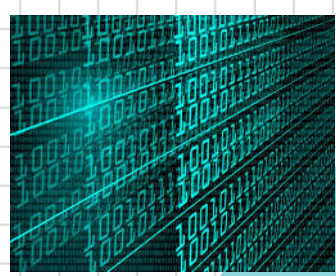
```
import tkinter
from random import *
canvas = tkinter.Canvas()
canvas.pack()
def dom(x, y):
    canvas.create_rectangle(x, y, x+50, y+50)
    canvas.create_line(x, y, x+25, y-50, x+50, y)
dom(100, 100)
dom(180, 100)
dom(260, 100)
```

```
import tkinter
from random import *
canvas = tkinter.Canvas()
canvas.pack()
def dom(x, y):
    canvas.create_rectangle(x, y, x+50, y+50)
    canvas.create_line(x, y, x+25, y-50, x+50, y)
```

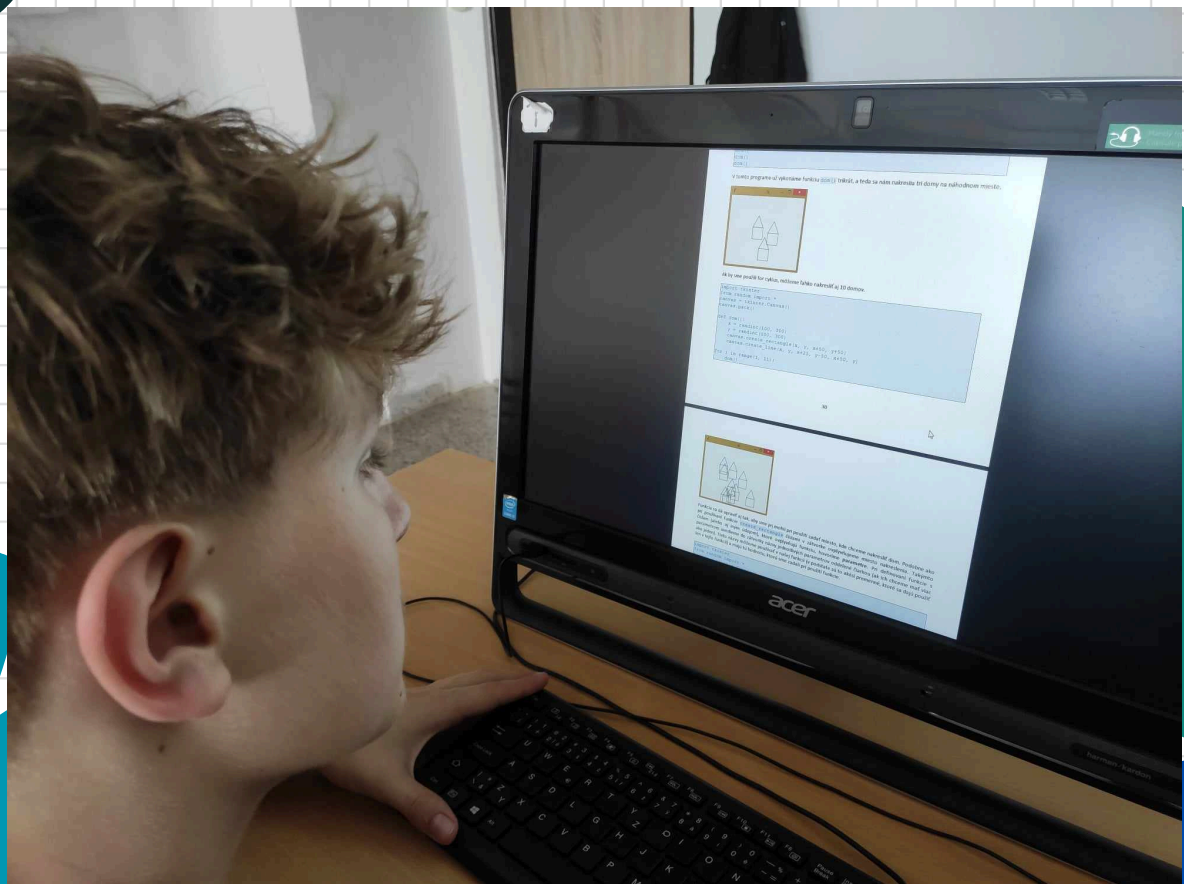
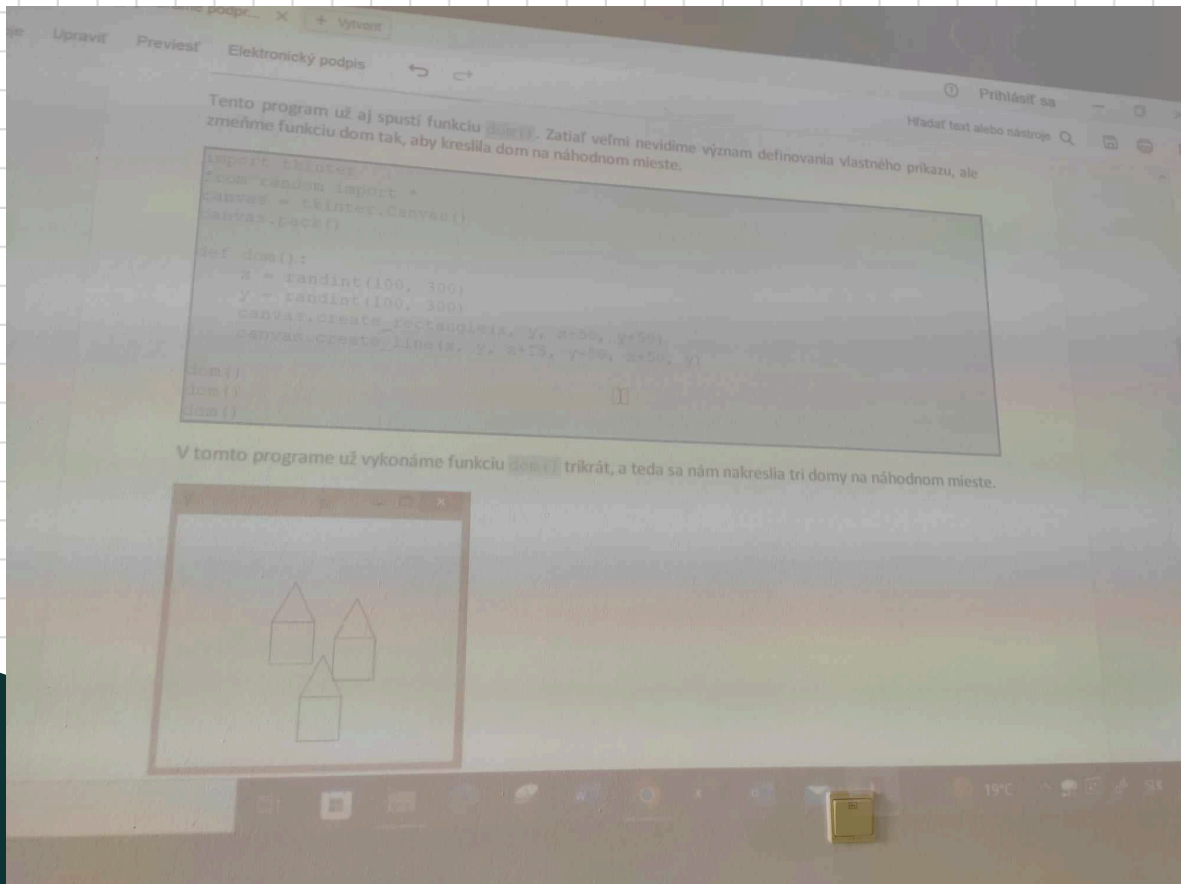




ANNEX



PHOTOS FROM THE LESSON IMPLEMENTATIONS

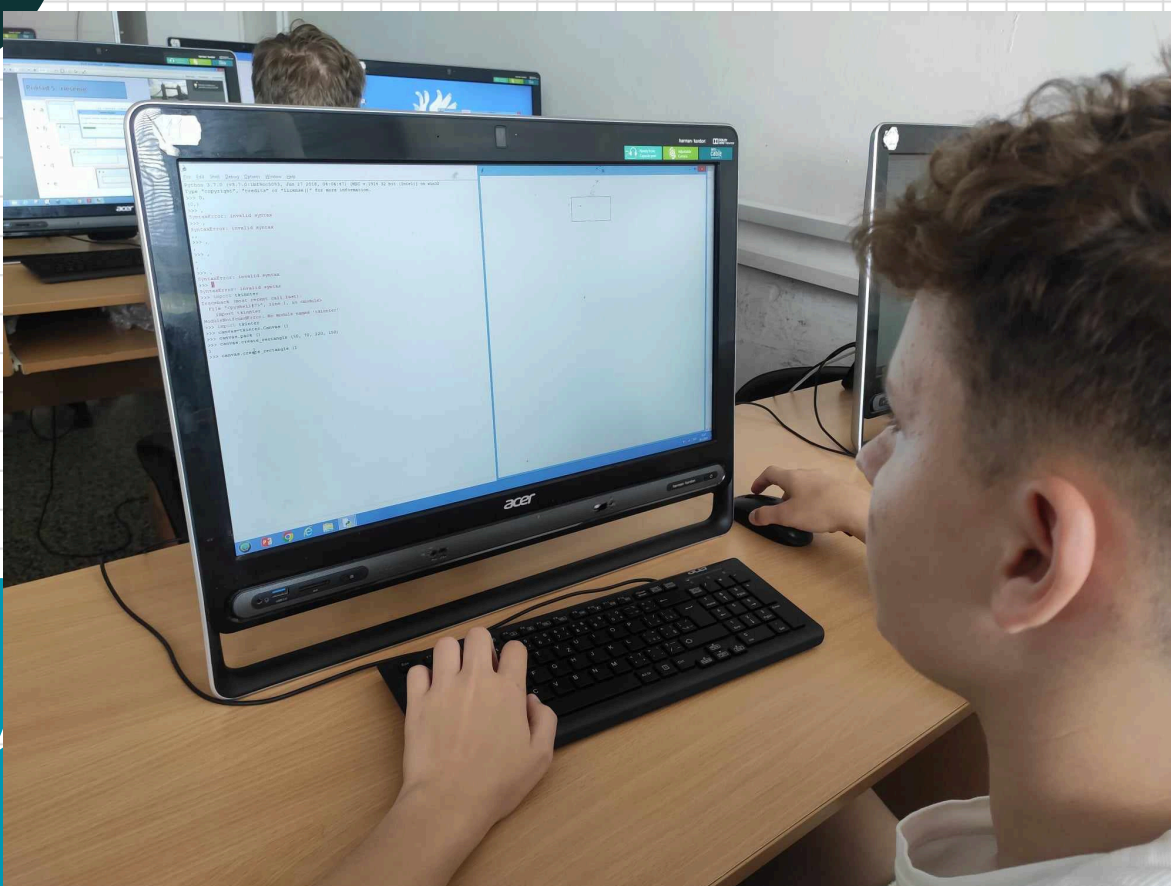
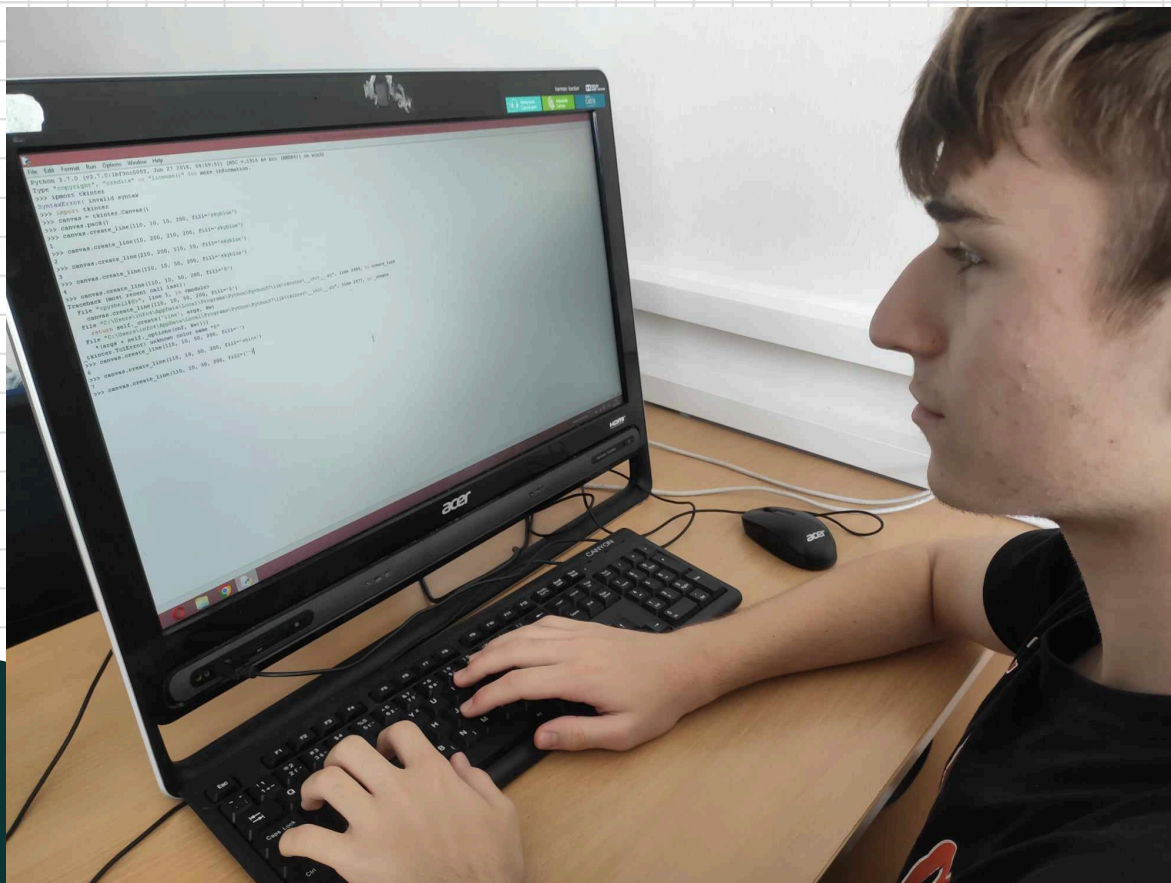




ANNEX



PHOTOS FROM THE LESSON IMPLEMENTATIONS





ANNEX



STUDENTS OUTPUT

THE CALL MECHANISM IS ILLUSTRATED BY THE CALL `ZOPIS_HVIEZDIČEK(30)`:
THE RETURN ADDRESS OF THE CALL IS REMEMBERED
A NEW VARIABLE NUMBER (FORMAL PARAMETER) IS CREATED AND THE
VALUE OF THE ACTUAL PARAMETER 30 IS ASSIGNED TO IT
ALL STATEMENTS IN THE FUNCTION DEFINITION (FUNCTION BODY) ARE
EXECUTED

ALL VARIABLES THAT WERE CREATED DURING THE EXECUTION OF THE
FUNCTION WILL BE CANCELED

CONTROL IS RETURNED TO WHERE THE FUNCTION WAS CALLED

LET'S WRITE A NEW FUNCTION `RED_CIRCLE()`, WHICH WILL HAVE TWO
PARAMETERS: THE COORDINATES OF THE CENTER OF THE CIRCLE. THE
FUNCTION DRAWS A CIRCLE WITH A RADIUS OF 10 AND WITH THE GIVEN
CENTER:

```
DEF CERVENY_KRUH(X, Y):
```

```
    CANVAS.CREATE_OVAL(X - 10, Y - 10, X + 10, Y + 10, FILL='RED') AND WE  
CAN CALL IT, FOR EXAMPLE, LIKE THIS:
```

```
IMPORT TKINTER DEF CERVENY_KRUH(X, Y):
```

```
    CANVAS.CREATE_OVAL(X - 10, Y - 10, X + 10, Y + 10, FILL='RED')
```

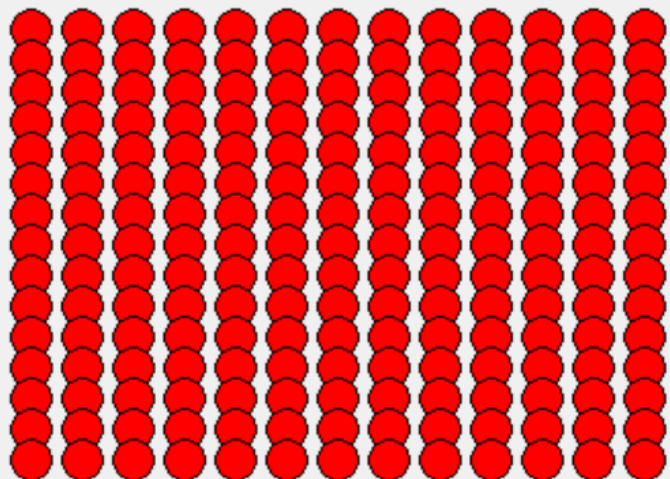
```
CANVAS = TKINTER.CANVAS() CANVAS
```

```
FOR X IN RANGE(30, 350, 25):
```

```
    FOR Y IN RANGE(20, 240, 15):
```

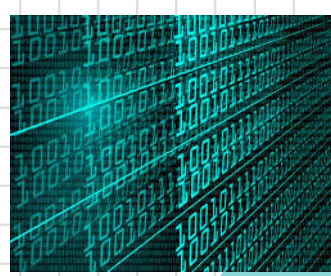
```
        RED_CIRCLE(X, Y)
```

```
TKINTER.MAINLOOP()
```





LESSON PLAN



SUBJECT: PROGRAMMING

TOPIC: PROGRAMMING IN PYTHON - WE CREATE SUBPROGRAMS.

LEARNING OBJECTIVES:.

- **1. UNDERSTAND THE BASIC PRINCIPLES OF CREATING SUBROUTINES IN PYTHON USING THE TKINTER LIBRARY**
- 2. TO LEARN HOW TO USE FUNCTIONS WITH PARAMETERS IN PYTHON.**

TOOLS:

**A COMPUTER WITH PYTHON AND THE TKINTER LIBRARY INSTALLED.
PYTHON SAMPLE TEXTBOOK WORKSHEET
WITH EXERCISES**

**INTRO: USING DIFFERENT COLORS IN
DRAWING:**

**A FUNCTION IS A NAMED BLOCK OF
COMMANDS (SUBROUTINE). WE DESCRIBE
(DEFINE) IT WITH A SPECIAL
CONSTRUCTION:**

- **WE CAN CALL THE FUNCTION ONLY WHEN
PYTHON ALREADY KNOWS ITS DEFINITION**

LESSON STRUCTURE: CONNECTION (4 MINUTES)

ACQUIRED SYNTACTIC RULES

- **METHOD OF WRITING THE SUBPROGRAM DEFINITION: HEADER AND BODY OFFSET OF ALL COMMANDS IN THE BODY OF THE SUBPROGRAM FROM THE LEFT MARGIN METHOD OF WRITING THE SUBPROGRAM CALL (EMPTY PARENTHESES AFTER THE NAME OF THE SUBPROGRAM)**

CONVERSATION WITH STUDENTS ABOUT THE ADVANTAGES OF USING SUBROUTINES IN PYTHON: - SUBROUTINES CAN USE THE CANVAS GLOBAL VARIABLE

WHEN FORMATTING THE SUBPROGRAM ENTRY, WE USE THE <TAB> AND <ENTER> KEYS

- **RESEARCH (12 MINUTES) - RESEARCHING NEW COMMANDS - WITH THE HELP OF SUBPROGRAMS, DRAWING IS DONE IN THE GRAPHICS AREA9. NOW YOU WILL CREATE SUBROUTINES THAT DRAW IN THE GRAPHICS WINDOW**
- PRESENTATION (15 MINUTES) - SUMMARY AND EXPLANATION OF KNOWLEDGE FROM THE SECTION INVESTIGATION OF THE ISSUE**

CREATE A NEW KRESBA_PODPROGRAM.PY PROGRAM AND TRY:

- **IMPORT TKINTER**
 - **CANVAS = TKINTER.CANVAS()**
 - **CANVAS.PACK()**
 - **DEF CHAIRS():**
 - **CANVAS.CREATE_RECTANGLE(10, 20, 30, 40, FILL='RED')**
 - **CHAIR()**
 - **IN THIS TASK, THE BODY OF THE SUBROUTINE CONTAINS A SINGLE COMMAND - TO DRAW A RECTANGLE.**
 - **ELABORATION (9 MINUTES) - SUPPORT FOR STUDENTS IN FURTHER INDEPENDENT CREATION OF PYTHON SUBPROGRAMS**
 - **ASSESSMENT (5 MINUTES) - SOLVING THE SELF-ASSESSMENT TEST, DISCUSSING THE ANSWERS.**
- EXPOSURE PHASE - EXPOSURE OF NEW COMMANDS**
- I WILL BRIEFLY NOTE THE BASIC TERMS ON THE BOARD**



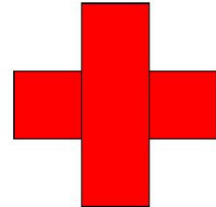
LESSON PLAN



5. STUDENT WORKSHEET

Task: Create a new kriz.py program and define a kriz subprogram in it, which will draw a red cross when called:

```
import tkinter
canvas = tkinter.Canvas()
canvas.pack()
def kriz():
    canvas.create_rectangle(150 - 90, 100 - 30, 150 + 90, 100 + 30, fill='red')
    canvas.create_rectangle(150 - 30, 100 - 90, 150 + 30, 100 + 90, fill='red')
kriz()
```

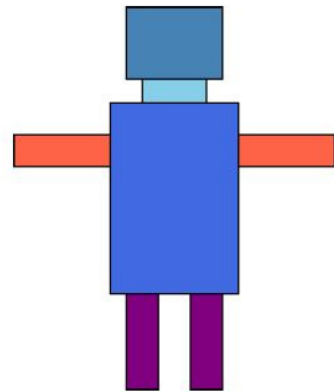


In this task, the subroutine body contains a sequence of two statements. Each time this subroutine is called, first, the first rectangle is drawn, and then the second rectangle is drawn through it.

2. Task: Create a new robot.py program that will be able to draw a robot. There will be four subroutines in the program - head, arms, legs, body - and each of them will be able to draw a part of the robot. When you call them in the following order:

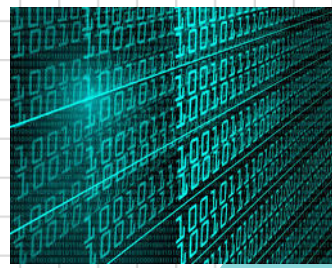
```
head()
arms()
legs()
body()
the whole robot is drawn.
```

```
import tkinter
canvas = tkinter.Canvas()
canvas.pack()
def hlava():
    canvas.create_rectangle(160, 40, 200, 100, fill='sky blue')
    canvas.create_rectangle(150, 10, 210, 55,
    fill='steel blue')
def telo():
    canvas.create_rectangle(140, 70, 220, 190, fill='royal blue')
def ruce():
    canvas.create_rectangle(80, 90, 280, 110, fill='tomato')
def nohy():
    canvas.create_rectangle(150, 160, 170, 250, fill='purple')
    canvas.create_rectangle(190, 160, 210, 250, fill='purple')
head()
arm()
legs()
body()
```





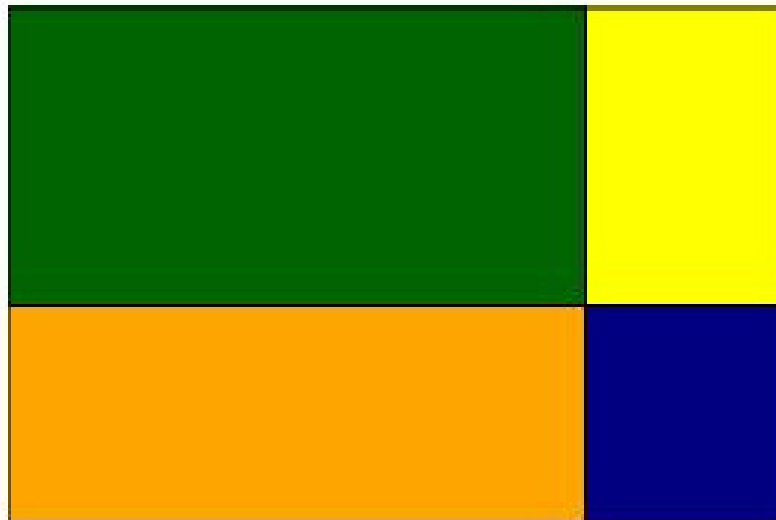
LESSON PLAN



5. STUDENT WORKSHEET

3. Task: Create a new program `sportová_vlajka.py`, which will draw the sports flag of your class.

The flag will consist of four colored rectangles that touch each other at a single point as in the image below:
The program randomly chooses the coordinates x, y , which represent the point of contact of all four rectangles.
The outer dimensions of the flag should be 300×200 ; determine the colors on the flag at your discretion.

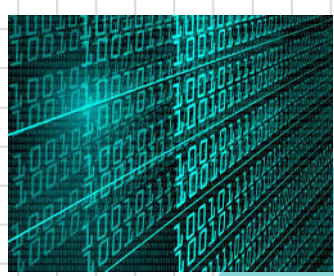


```
import tkinter
import random
canvas = tkinter.Canvas()
canvas.pack()
x = random.randint(50, 250)
y = random.randint(50, 150)
canvas.create_rectangle(10, 10, x, y, fill='darkgreen')
canvas.create_rectangle(x, 10, 310, y, fill='yellow')
canvas.create_rectangle(10, y, x, 210, fill='orange')
canvas.create_rectangle(x, y, 310, 210, fill='navy')
```

The outer vertices of the rectangle can be chosen arbitrarily. In our solution, the upper left vertex has coordinates $[10, 10]$ and since the dimensions of the rectangle should be 300×200 , the opposite vertex (lower right) has coordinates $[310, 210]$. We generated the inner randomly chosen point so that it is at least 50 from the edges of the rectangle, so each of the four areas will have dimensions of at least 50×50 .



ANNEX



PHOTOS FROM THE LESSON IMPLEMENTATIONS

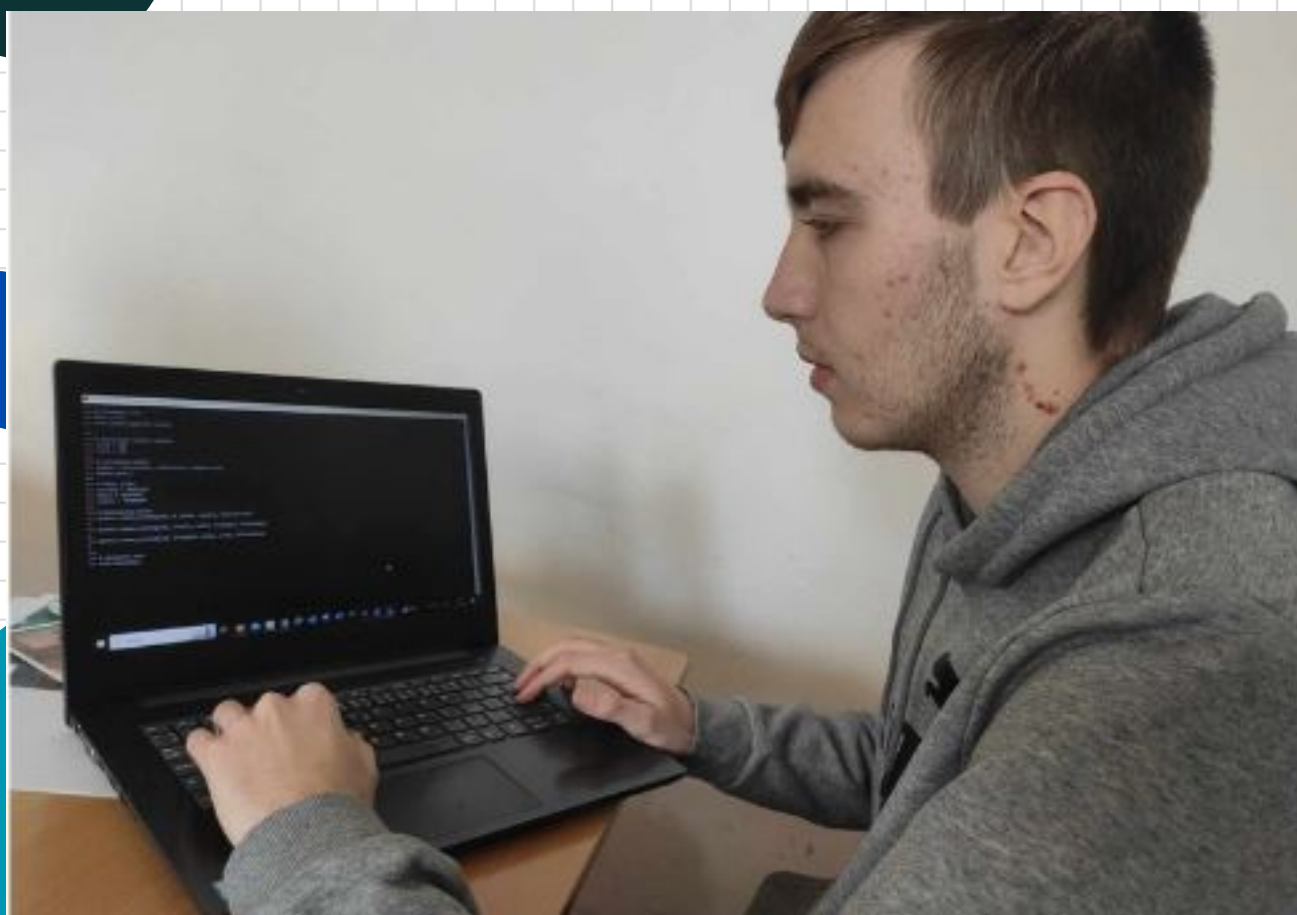




ANNEX



PHOTOS FROM THE LESSON IMPLEMENTATIONS





ANNEX

STUDENTS OUTPUT

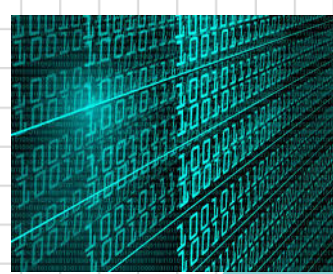
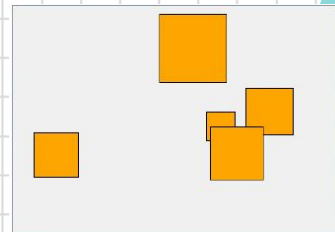
MODIFY THE RANDOM_SQUARES.PY PROGRAM SO THAT SQUARES ARE DRAWN NOT ONLY AT RANDOM POSITIONS, BUT ALSO SO THAT EACH SQUARE HAS A RANDOM SIZE FROM THE INTERVAL 10 TO 100:

```
DEF RANDOM_SQUARE():
```

```
X = RANDOM.RANDINT(10, 300) Y = RANDOM.RANDINT(10, 200) A = RANDOM.RANDINT(10, 100)
CANVAS.CREATE_RECTANGLE(X, Y, X + A, Y + A,
FILL='ORANGE ')
```

```
IMPORT TKINTER IMPORT RANDOM CANVAS =
TKINTER.CANVAS() CANVAS.PACK() DEF
RANDOM_THURSDAY():
```

```
X = RANDOM.RANDINT(10, 300) Y = RANDOM.RANDINT(10, 200) A = RANDOM.RANDINT(10, 100)
CANVAS.CREATE_RECTANGLE(X, Y, X + A, Y + A,
FILL='ORANGE ') DEF RANDOM_OBDĽŽNIK():
X = RANDOM.RANDINT(10, 300) Y = RANDOM.RANDINT(10, 200) A = RANDOM.RANDINT(10, 100) B =
RANDOM.RANDINT(10, 100) CANVAS.CREATE_RECTANGLE(X,
Y, X + A, Y + B, FILL='LIME GREEN') RANDOM_OBDĽŽNIK()
RANDOM_OBDĽŽNIK() RANDOM_OBDĽŽNIK()
RANDOM_OBDĽŽNIK() RANDOM_OBDĽŽNIK() RANDOM_OBDĽŽNIK()
RANDOM_OBDĽŽNIK() RANDOM_OBDĽŽNIK() RANDOM_OBDĽŽNIK()
```



Lesson plan

Introductory Lessons Series on Programming in C

Topic: Getting Started with C

Lesson #1

Lesson Focus and Goals:

This lesson introduces students to the C programming language, its history, and its basic syntax. By the end of this lesson, students should be able to write and run a simple "Hello, World!" program

Materials Needed:

- Computers with C compiler installed (e.g., GCC)
- Projector and screen
- Whiteboard and markers
- Handouts of basic C syntax and structure

Learning Objectives:

- Introduce C
- Introduce the Notepad & GCC
- Write "Hello, World!" program.

Structure / Activity:

1. Introduction to C (15 minutes)
 - Discussion (5 minutes): Begin with a brief history of the C programming language. Discuss its importance and how it laid the foundation for many modern programming languages. Highlight key applications, such as operating systems (e.g., UNIX, Linux) and embedded systems.
 - Presentation (10 minutes): Use a projector to display where C is used today and discuss why learning C is essential. Provide examples of popular software and systems written in C.
2. Basic Syntax and Structure (20 minutes)
 - Explanation (10 minutes): Introduce the structure of a basic C program. Explain the role of `#include`, the `main()` function, and basic syntax rules. Provide examples of syntax errors and how to avoid them.
 - Example (10 minutes): Write a simple "Hello, World!" program on the whiteboard or projector screen. Break down each line of code, explaining its purpose. Encourage students to ask questions about each part of the code.
3. Hands-On Activity (20 minutes)
 - Activity (15 minutes): Have students write their own "Hello, World!" program on their computers. Guide them through setting up their development environment (installing a C compiler like GCC if necessary). Ensure each student can compile and run their program successfully.
 - Peer Review (5 minutes): Pair students up to review each other's code. Encourage them to discuss any issues or questions they encountered.
4. Q&A and Recap (5 minutes)
 - Q&A (3 minutes): Open the floor for any questions regarding the lesson. Address common issues or confusion points that students might have.
 - Recap and Homework (2 minutes): Summarize the key points of the lesson. For homework, ask students to research a simple C program online, modify it slightly (e.g., change the output message), and run it.

Lesson plan - handout

Introductory Lessons Series on Programming in C

Topic: Getting Started with C

Handout #1

1. Introduction to C

- Key Points:
 - C is a foundational programming language used in many applications, including operating systems and embedded systems.
 - Understanding C helps in learning other programming languages like C++, Java, and Python.
- Example Programs:
 - "Hello, World!": The simplest C program.

```
#include <stdio.h>

int main() {
    printf("Hello,
World!\n");
    return 0;
}
```

2. Basic Syntax and Structure

- Key Points:
 - Every C program must include `#include <stdio.h>` and a `main()` function.
 - The `main()` function is the starting point of any C program.
- Activity:
 - Write and run the "Hello, World!" program.
 - Identify the purpose of each line in the program.

3. Homework:

- Research a simple C program online, modify it slightly (e.g., change the output message), and run it.

Lesson plan

Introductory Lessons Series on Programming in C

Topic: Variables and Data Types

Lesson #2

Lesson Focus and Goals:

Teach students about variables, data types, and basic input/output operations in C.

Materials Needed:

- Computers with C compiler installed (e.g., GCC)
- Projector and screen
- Whiteboard and markers
- Handouts with examples of variable declarations and common input/output operations

Learning Objectives:

Learn about variables/data types
Write program with printf

Structure / Activity:

1. Review of Previous Lesson (5 minutes)
 - Discussion (5 minutes): Recap the basic structure of a C program and the "Hello, World!" program from the previous lesson. Emphasize the importance of understanding syntax before moving on to more complex concepts.
2. Introduction to Variables and Data Types (15 minutes)
 - Explanation (10 minutes): Introduce the concept of variables as placeholders for data. Explain the difference between different data types (e.g., int, float, char, double) and their typical uses.
 - Example (5 minutes): Write a simple program that declares different types of variables and uses printf to output their values. Demonstrate how to assign values to variables and perform basic arithmetic operations.
3. Hands-On Activity #1: Variable Declaration (15 minutes)
 - Activity (10 minutes): Have students declare variables of different data types in a new program. Ask them to perform basic operations, such as addition and multiplication, and print the results.
 - Peer Review (5 minutes): Students exchange their programs and check if the variable declarations and operations are correct. Encourage them to discuss any differences or mistakes.
4. Introduction to Basic Input/Output (10 minutes)
 - Explanation (5 minutes): Introduce the scanf function for taking user input. Explain how it works in conjunction with printf to create interactive programs.
 - Example (5 minutes): Write a program that asks the user for their age and then prints it back to them using scanf and printf.
5. Hands-On Activity #2: Interactive Program (15 minutes)
 - Activity (10 minutes): Students will modify their previous program to include user input. For example, ask the user for two numbers, perform an operation on them (like addition), and print the result.
 - Support (5 minutes): Circulate the room to help students with any syntax or logic errors they encounter.
6. Q&A and Recap (5 minutes)
 - Q&A (3 minutes): Allow students to ask questions about variables, data types, or input/output operations.
 - Recap and Homework (2 minutes): Summarize the lesson. Assign homework where students create a program that takes user input for three different data types and outputs them in a formatted string.

Lesson plan - handout

Introductory Lessons Series on Programming in C

Topic: Variables and Data Types

Handout #2

1. Introduction to Variables

- Key Points:
 - Variables are placeholders for data. In C, each variable has a specific data type.
 - Common data types: int (integer), float (floating-point number), char (character), double (double-precision floating-point).

Example code:

```
int age = 25;
float height = 5.9;
char initial = 'A';
double distance = 12345.67;
```

2. Input and Output

- Key Points:
 - Use printf to output data and scanf to take input.

Example code:

```
int age;
printf("Enter your age: ");
scanf("%d", &age);
printf("You entered: %d\n", age);
```

3. Homework:

- Create a program that takes user input for three different data types and outputs them in a formatted string.

Lesson plan

Introductory Lessons Series on Programming in C

Topic: Control Structures

Lesson #3

Lesson Focus and Goals:

This lesson introduces students to control structures in C, including if-else statements and loops. By the end of the lesson, students should be able to use these structures to control the flow of their programs.

Materials Needed:

- Computers with C compiler installed (e.g., GCC)
- Projector and screen
- Whiteboard and markers
- Handouts with examples of if-else statements and loops

Learning Objectives:

Learn about basic Input/Output
Learn about the if/ if-else statements

Structure / Activity:

1. Review of Previous Lesson (5 minutes)
 - Discussion (5 minutes): Recap the concepts of variables, data types, and basic input/output. Highlight how these elements will be used in conjunction with control structures.
2. Introduction to If-Else Statements (15 minutes)
 - Explanation (10 minutes): Explain the syntax and usage of if-else statements. Discuss the concept of conditional logic and how it allows programs to make decisions.
 - Example (5 minutes): Write a program that asks the user for a number and prints whether it is even or odd. Explain how the if-else statement is used to check conditions.
3. Hands-On Activity #1: Conditional Logic (15 minutes)
 - Activity (10 minutes): Students write a program that takes user input and uses if-else statements to make decisions (e.g., check if a number is positive, negative, or zero).
 - Peer Review (5 minutes): Have students exchange their programs and review the logic used. Encourage them to test each other's programs with different inputs.
4. Introduction to Loops (15 minutes)
 - Explanation (10 minutes): Introduce for and while loops. Explain how loops are used to repeat actions in a program and discuss scenarios where each type of loop is appropriate.
 - Example (5 minutes): Write a program that prints numbers from 1 to 10 using a for loop. Then, rewrite it using a while loop.
5. Hands-On Activity #2: Looping Structures (15 minutes)
 - Activity (10 minutes): Students write a program that uses a loop to calculate the factorial of a number or generate a multiplication table.
 - Support (5 minutes): Assist students with debugging their loops and ensuring the logic is correct.
6. Q&A and Recap (5 minutes)
 - Q&A (3 minutes): Address any questions about control structures. Discuss common mistakes, such as infinite loops or improper use of if-else.
 - Recap and Homework (2 minutes): Summarize the lesson. Assign homework where students write a program using both if-else statements and loops to create a simple calculator.

Lesson plan - handout

Introductory Lessons Series on Programming in C

Topic: Control Structures

Handout #3

1. If-Else Statements

- Key Points:
 - if-else statements allow the program to make decisions based on conditions.
- Example Code:

```
int number;  
printf("Enter a number: ");  
scanf("%d", &number);  
  
if (number % 2 == 0) {  
    printf("The number is even.\n");  
} else {  
    printf("The number is odd.\n");  
}
```

2. Loops

- Key Points:
 - Loops allow the program to repeat actions.
Common loops are for and while.
 -
- Example Code:

```
for (int i = 1; i <= 10; i++) {  
    printf("%d\n", i);  
}
```

3. Homework:

- Write a program using both if-else statements and loops to create a simple calculator.

Your paragraph text

Lesson plan

Introductory Lessons Series on Programming in C

Topic: Functions

Lesson #4

Lesson Focus and Goals:

This lesson teaches students about functions in C, their importance, and how to create and use them. By the end of the lesson, students should be able to define and call basic functions in their programs.

Materials Needed:

- Computers with C compiler installed (e.g., GCC)
- Projector and screen
- Whiteboard and markers
- Handouts with examples of function definitions and function calls

Learning Objectives:

Learn to define basic functions in C
Hands-on practice

Structure / Activity:

1. Review of Previous Lesson (5 minutes)
 - Discussion (5 minutes): Quickly review control structures, emphasizing how functions will help organize and reuse code in more complex programs.
2. Introduction to Functions (15 minutes)
 - Explanation (10 minutes): Define functions, explaining the purpose of dividing code into smaller, reusable blocks. Discuss the syntax of a function, including return types, function names, and parameters.
 - Example (5 minutes): Write a simple function that calculates the square of a number. Explain how the function is defined, called, and how it returns a value.
3. Hands-On Activity #1: Writing Functions (15 minutes)
 - Activity (10 minutes): Have students write their own function that performs a specific task, such as converting Celsius to Fahrenheit. Ensure they understand how to pass parameters and return values.
 - Peer Review (5 minutes): Students exchange their programs and test each other's functions.
4. Expanding on Functions (15 minutes)
 - Explanation (10 minutes): Discuss how to use multiple functions in a single program. Explain how functions can call other functions and how to manage function dependencies.
 - Example (5 minutes): Write a program that includes two functions: one for input validation and another for performing a calculation. Demonstrate how these functions interact.
5. Hands-On Activity #2: Function Interaction (15 minutes)
 - Activity (10 minutes): Students will create a program that uses at least two functions working together, such as a simple calculator that validates input before performing operations.
 - Support (5 minutes): Provide guidance on debugging and understanding function scopes.
6. Q&A and Recap (5 minutes)
 - Q&A (3 minutes): Address any questions regarding functions. Discuss common issues, such as function overloading and naming conflicts.
 - Recap and Homework (2 minutes): Summarize the lesson. For homework, ask students to modify a program to include functions for input, processing, and output.

Lesson plan - handout

Introductory Lessons Series on Programming in C

Topic: Functions

Handout #4

1. Introduction to Functions

- Key Points:
 - Functions are reusable blocks of code that perform specific tasks.
 - Functions have a return type, name, and parameters.
- Example Code:

```
int square(int num) {  
    return num * num;  
}  
  
int main() {  
    int result = square(5);  
    printf("Square of 5 is %d\n", result);  
    return 0;  
}
```

2. Writing and Calling Functions

- Activity:
 - Write a function that converts Celsius to Fahrenheit and calls it in `main()`.

3. Homework:

- Modify a program to include functions for input, processing, and output.

Lesson plan

Introductory Lessons Series on Programming in C

Topic: Arrays and Strings

Lesson #5

Lesson Focus and Goals:

This lesson introduces students to arrays and strings in C, explaining how to declare, initialize, and use them in programs. By the end of the lesson, students should be able to manipulate arrays and strings in basic C programs.

Materials Needed:

- Computers with C compiler installed (e.g., GCC)
- Projector and screen
- Whiteboard and markers
- Handouts with examples of array and string operations

Learning Objectives:

- Introduce arrays and strings
- Hands-on practice

Structure / Activity:

1. Review of Previous Lesson (5 minutes)
 - Discussion (5 minutes): Recap the concept of functions and their usage in C. Highlight how functions can simplify working with arrays and strings.
2. Introduction to Arrays (15 minutes)
 - Explanation (10 minutes): Explain what arrays are, including their syntax and how to declare and initialize them. Discuss different types of arrays, such as single-dimensional and multi-dimensional arrays.
 - Example (5 minutes): Write a program that uses an array to store and sum a list of numbers. Explain each step of the program, from declaration to iteration and summation.
3. Hands-On Activity #1: Array Manipulation (15 minutes)
 - Activity (10 minutes): Students write a program that uses an array to store user input (e.g., a list of grades) and then calculates the average.
 - Support (5 minutes): Help students debug issues such as array out-of-bounds errors and incorrect indexing.
4. Introduction to Strings (15 minutes)
 - Explanation (10 minutes): Discuss how strings are represented in C using character arrays. Explain how to declare, initialize, and manipulate strings.
 - Example (5 minutes): Write a program that takes a string input from the user and prints it in reverse. Walk students through the logic of string manipulation.
5. Hands-On Activity #2: String Operations (15 minutes)
 - Activity (10 minutes): Students write a program that asks for the user's name and then performs various operations, such as counting the number of characters or finding the first vowel.
 - Support (5 minutes): Provide assistance with string termination issues and common mistakes in string manipulation.
6. Q&A and Recap (5 minutes)
 - Q&A (3 minutes): Allow students to ask questions about arrays and strings. Discuss the importance of understanding these concepts for more advanced programming.
 - Recap and Homework (2 minutes): Summarize the lesson. Assign homework where students write a program that uses both arrays and strings to manage a simple contact list.

Lesson plan - handout

Introductory Lessons Series on Programming in C

Topic: Arrays and Strings

Handout #5

1. Introduction to Arrays

- Key Points:
 - Arrays are collections of variables of the same type.
 - Arrays are zero-indexed; the first element is at index 0.
- Example Code:

```
int numbers[5] = {1, 2, 3, 4, 5};  
printf("First number: %d\n", numbers[0]);
```

2. Introduction to Strings

- Key Points:
 - Strings in C are arrays of characters ending with a null character `\0`.
- Example Code:

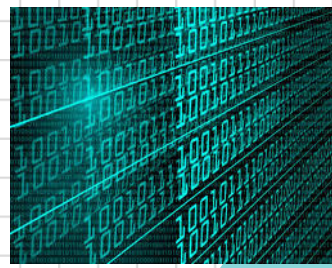
```
char name[20];  
printf("Enter your name: ");  
scanf("%s", name);  
printf("Hello, %s!\n", name);
```

3. Homework:

- Write a program that uses both arrays and strings to manage a simple contact list.



LESSON PLAN



SUBJECT: SOFTWARE DEVELOPMENT AND CODING

TOPIC: INTRODUCTORY LESSONS ON THE PYTHON PROGRAMMING LANGUAGE - LESSON 1

LESSON FOCUS AND OBJECTIVES:

THIS LESSON FAMILIARIZES STUDENTS WITH THE FUNDAMENTALS OF THE PYTHON PROGRAMMING LANGUAGE, ENCOMPASSING ITS HISTORY, KEY FEATURES, AND THE PROCESS OF WRITING AN INITIAL PYTHON PROGRAM.

BY THE CONCLUSION OF THIS LESSON, STUDENTS SHOULD BE CAPABLE OF COMPREHENDING THE FUNDAMENTAL SYNTAX OF THE PYTHON LANGUAGE AND CONSTRUCTING AND EXECUTING A BASIC "HELLO, WORLD!" PROGRAM.

ESSENTIAL MATERIALS:

- COMPUTERS EQUIPPED WITH PYTHON (E.G., VISUAL STUDIO CODE, PYCHARM, OR OTHER DEVELOPMENT ENVIRONMENTS)
- PROJECTOR AND SCREEN.
- CHALKBOARD AND MARKERS
- FUNDAMENTAL PYTHON SYNTAX AND STRUCTURAL ASSIGNMENTS

1. OVERVIEW OF PYTHON (15 MINUTES)

TALK (5 MINUTES): BEGIN WITH A CONCISE HISTORY OF THE PYTHON PROGRAMMING LANGUAGE, CREATED BY GUIDO VAN ROSSUM IN THE LATE 1980S. EXAMINE THE EVOLUTION OF THE LANGUAGE AND ITS RISE TO PROMINENCE, ATTRIBUTED TO ITS USER-FRIENDLY NATURE AND EXTENSIVE APPLICABILITY. EMPHASIZE HOW PYTHON HAS ESTABLISHED A FOUNDATION FOR CONTEMPORARY PROGRAMMING LANGUAGES AND ITS PARTICULAR POPULARITY IN SCIENTIFIC COMPUTING, WEB DEVELOPMENT, AND DATA PROCESSING.

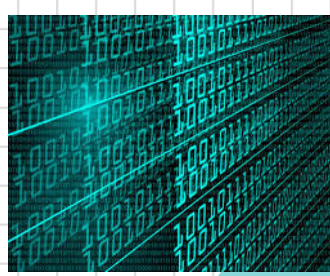
PRESENTATION (10 MINUTES): PROVIDE AN OVERVIEW OF THE CURRENT APPLICATIONS OF PYTHON ACROSS DIVERSE INDUSTRIES. EXPLAIN THE ADVANTAGES OF LEARNING PYTHON AND PRESENT EXAMPLES OF WIDELY-USED PYTHON-BASED APPLICATIONS, INCLUDING DATA ANALYTICS FOR YOUTUBE, INSTAGRAM, AND NASA.

2. FUNDAMENTAL SYNTAX AND STRUCTURE (20 MINUTES)

EXPLANATION (10 MINUTES): DEMONSTRATE THE STRUCTURE OF A BASIC PYTHON PROGRAM. CLARIFY THE FUNCTION OF THE IMPORT STATEMENT, THE SIGNIFICANCE OF THE MAIN() FUNCTION, AND THE SYNTACTICAL FEATURES OF PYTHON, PARTICULARLY THE IMPORTANCE OF INDENTATION. DISCUSS HOW PYTHON PROCESSES EXPRESSIONS AND THE COMMON ERRORS THAT MAY ARISE. HIGHLIGHT STRAIGHTFORWARD MISTAKES, SUCH AS IMPROPER INDENTATION, AND PROVIDE GUIDANCE ON HOW TO PREVENT THEM. WRITE A "HELLO, WORLD!" PROGRAM ON THE BOARD OR PROJECTOR SCREEN. PROVIDE A DETAILED EXPLANATION OF THE FUNCTION OF EACH LINE OF CODE, INCLUDING THE OPERATION OF THE PRINT() FUNCTION, THE HANDLING OF STRINGS, AND THE INTERPRETATION OF THE OUTPUT RESULT. ENCOURAGE STUDENTS TO INQUIRE ABOUT EACH COMPONENT OF THE CODE.



LESSON PLAN



3. HANDS-ON ACTIVITY (20 MINUTES)

ACTIVITY (15 MINUTES): INSTRUCT STUDENTS TO CREATE THEIR OWN "HELLO, WORLD!" PROGRAM IN PYTHON. ASSIST THEM IN CONFIGURING THEIR DEVELOPMENT ENVIRONMENT, SUCH AS IDLE OR PYCHARM, AND ENSURE THAT EACH STUDENT CAN EXECUTE THEIR PROGRAM SUCCESSFULLY. CONFIRM THAT ALL STUDENTS COMPREHEND THE FUNCTIONALITY OF THE CODE AND ARE PROFICIENT IN NAVIGATING THE PYTHON ENVIRONMENT.

PEER REVIEW (5 MINUTES): PAIR STUDENTS TO EVALUATE ONE ANOTHER'S CODE. ENCOURAGE THEM TO ENGAGE IN DISCUSSIONS REGARDING ANY QUESTIONS OR ISSUES THAT MAY ARISE AND ASSIST EACH OTHER IN ENHANCING THE CODE.

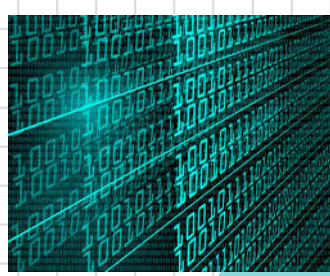
4. QUESTIONS, ANSWERS, AND SUMMARY (5 MINUTES)

QUESTIONS AND ANSWERS (3 MINUTES): INVITE STUDENTS TO POSE QUESTIONS. ADDRESS PREVALENT ISSUES OR AREAS OF CONFUSION THAT MAY HAVE EMERGED DURING THE LESSON. FACILITATE OPPORTUNITIES FOR STUDENTS TO SHARE THEIR FEEDBACK REGARDING THEIR COMPREHENSION OF THE MATERIAL.

SUMMARY AND HOMEWORK (2 MINUTES): SUMMARIZE THE KEY POINTS OF THE LESSON, EMPHASIZING FUNDAMENTAL PYTHON SYNTAX AND THE SIGNIFICANCE OF THE "HELLO, WORLD!" PROGRAM. FOR HOMEWORK, INSTRUCT STUDENTS TO LOCATE A SIMPLE PYTHON PROGRAM ONLINE, ALTER ITS OUTPUT MESSAGE, AND EXECUTE IT. THEY SHOULD BRING THE MODIFIED CODE TO THE NEXT CLASS AND BE READY TO PRESENT IT.



LESSON PLAN – HANDOUT



SUBJECT: SOFTWARE DEVELOPMENT AND CODING

TOPIC: INTRODUCTORY LESSONS ON THE PYTHON PROGRAMMING LANGUAGE – LESSON 1

1. AN OVERVIEW OF PYTHON

MAIN POINTS:

PYTHON IS AN ACCESSIBLE, HIGH-LEVEL PROGRAMMING LANGUAGE UTILIZED ACROSS VARIOUS DOMAINS, INCLUDING WEB DEVELOPMENT, DATA PROCESSING, SCIENTIFIC COMPUTING, AND ARTIFICIAL INTELLIGENCE.

THE INCEPTION OF PYTHON CAN BE TRACED TO THE LATE 1980S, WHEN GUIDO VAN ROSSUM DESIGNED THE LANGUAGE TO PRIORITIZE EASE OF LEARNING AND READABILITY.

COMMON APPLICATIONS OF PYTHON ENCOMPASS WIDELY USED PLATFORMS SUCH AS YOUTUBE, INSTAGRAM, AND NASA DATA ANALYTICS.

EXAMPLE PROGRAM:

"HELLO, WORLD!" – A QUINTESSENTIAL EXAMPLE OF A BASIC PYTHON PROGRAM:

```
PRINT("GREETINGS, WORLD!")
```

2. FUNDAMENTAL SYNTAX AND STRUCTURE

MAIN POINTS:

IN PYTHON, THE ORGANIZATION OF THE CODE IS SIGNIFICANTLY INFLUENCED BY INDENTATION. INDENTATION DELINEATES BLOCKS OF CODE, MAKING ITS CORRECT USAGE IMPERATIVE. THE PRINT() FUNCTION IS AMONG THE FUNDAMENTAL TOOLS IN PYTHON, ENABLING THE DISPLAY OF TEXT AND VARIOUS DATA ON THE SCREEN.

PYTHON DOES NOT NECESSITATE SEMICOLONS AT THE CONCLUSION OF LINES, IN CONTRAST TO NUMEROUS OTHER PROGRAMMING LANGUAGES.

ACTIVITY:

CREATE AND EXECUTE A "HELLO, WORLD!" PROGRAM WITHIN ITS DEDICATED DEVELOPMENT ENVIRONMENT.

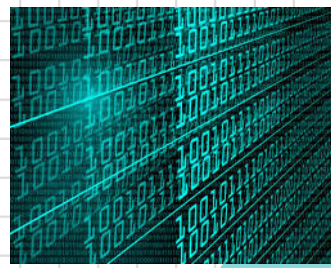
ASCERTAIN THE FUNCTION OF EACH LINE OF THE PROGRAM AND ENSURE COMPREHENSION OF ITS OPERATION.

3. HOMEWORK

DEVELOP A PYTHON PROGRAM THAT OUTPUTS "HELLO, WORLD!" AND PRESENTS A PERSONALIZED MESSAGE, SUCH AS YOUR NAME OR A FAVORITE QUOTE, IN PLACE OF THE STANDARD MESSAGE. EXPERIMENT WITH THE PRINT() FUNCTION BY UTILIZING VARIOUS QUOTATION MARKS OR INCORPORATING A NEW LINE TO DISPLAY SUPPLEMENTARY TEXT. SAVE THE REVISED PROGRAM AND BRING IT TO THE NEXT CLASS FOR PRESENTATION.



LESSON PLAN



SUBJECT: SOFTWARE DEVELOPMENT AND CODING

TOPIC: VALUES AND DATA TYPES IN THE PYTHON PROGRAMMING LANGUAGE – LESSON 2

LESSON FOCUS AND OBJECTIVES:

AN OVERVIEW OF THE FUNDAMENTAL DATA TYPES UTILIZED IN PYTHON, ENCOMPASSING INTEGERS, FLOATS, STRINGS, AND BOOLEANS.

BY THE CONCLUSION OF THE LESSON, STUDENTS SHOULD BE ABLE TO IDENTIFY AND ACCURATELY UTILIZE FUNDAMENTAL DATA TYPES, AS WELL AS COMPREHEND HOW OPERATIONS ALTER THE TYPE OF THESE VALUES.

ESSENTIAL MATERIALS:

- **COMPUTERS EQUIPPED WITH PYTHON (E.G., VISUAL STUDIO CODE, PYCHARM, OR OTHER DEVELOPMENT ENVIRONMENTS)**
- **PROJECTOR OR DISPLAY**
- **CHALKBOARD AND MARKERS**
- **ANNEXES DISPLAYING THE DATA TYPES**

1. OVERVIEW OF VALUES AND TYPES (15 MINUTES)

DISCUSSION (5 MINUTES): INITIATE A BRIEF DISCUSSION ON THE SIGNIFICANCE OF VALUES IN PROGRAMMING AND THE IMPORTANCE OF DATA TYPES. EXPLORE HOW VALUES REPRESENT DATA WITHIN PROGRAMS AND HOW DATA TYPES DELINEATE THE METHODS BY WHICH THAT DATA CAN BE PROCESSED.

PRESENTATION (10 MINUTES): ILLUSTRATE VARIOUS DATA TYPES, INCLUDING INTEGERS, FLOATING-POINT NUMBERS, STRINGS, AND BOOLEANS. EXAMINE THE PRIMARY CHARACTERISTICS AND APPLICATIONS OF EACH TYPE. PROVIDE EXAMPLES OF THEIR PRACTICAL USE, SUCH AS IN MATHEMATICAL OPERATIONS, WORD PROCESSING, AND CONDITIONAL LOGIC.

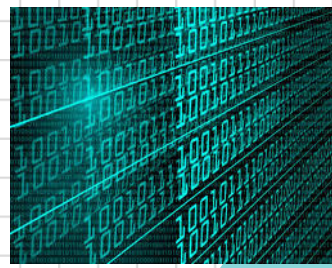
2. FUNDAMENTAL SYNTAX AND STRUCTURE (20 MINUTES)

EXPLANATION (10 MINUTES): DEMONSTRATE THE DEFINITION AND UTILIZATION OF BASIC DATA TYPES IN PYTHON. CLARIFY HOW THE OUTCOME OF AN EXPRESSION VARIES BASED ON THE DATA TYPE EMPLOYED. FOR INSTANCE, ILLUSTRATE HOW THE ADDITION OF AN INTEGER AND A FLOATING-POINT NUMBER YIELDS A RESULT, AND WHAT OCCURS WHEN ATTEMPTING TO CONCATENATE AN INTEGER WITH A STRING.

DEMONSTRATE STRAIGHTFORWARD EXAMPLES OF UTILIZING VARIOUS TYPES OF VALUES ON A BOARD OR PROJECTOR SCREEN. FOR INSTANCE, ILLUSTRATE THE ADDITION OF TWO NUMBERS, THE CONCATENATION OF TWO TEXTS, AND THE APPLICATION OF LOGICAL OPERATORS TO DEFINE CONDITIONS. EMPHASIZE THAT THE ACCURATE APPLICATION OF VALUES AND TYPES IS CRUCIAL FOR THE PROPER FUNCTIONING OF PROGRAMS.



LESSON PLAN



3. PRACTICAL EXERCISE (20 MINUTES)

ACTIVITY (15 MINUTES): INSTRUCT STUDENTS TO COMPOSE BASIC PYTHON CODE THAT INCORPORATES VARIOUS VALUE TYPES, INCLUDING INTEGERS, FLOATING-POINT NUMBERS, STRINGS, AND BOOLEANS. ASSIST THEM IN CONFIGURING THEIR DEVELOPMENT ENVIRONMENT AND ENSURE THAT EACH STUDENT CAN EXECUTE THEIR PROGRAM SUCCESSFULLY. VERIFY THAT STUDENTS GRASP THE OUTCOMES AND THE DISTINCTIONS AMONG EACH VALUE TYPE.

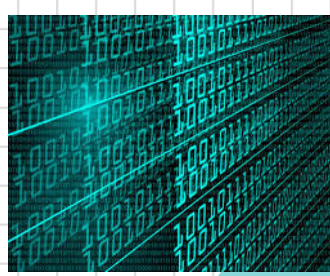
4. QUESTIONS, ANSWERS, AND SUMMARY (5 MINUTES)

QUESTIONS AND ANSWERS (3 MINUTES): INVITE STUDENTS TO POSE QUESTIONS. ADDRESS PREVALENT ISSUES OR AREAS OF CONFUSION THAT MAY HAVE EMERGED DURING THE LESSON. FACILITATE OPPORTUNITIES FOR STUDENTS TO SHARE THEIR FEEDBACK REGARDING THEIR COMPREHENSION OF THE MATERIAL.

SUMMARY AND HOMEWORK (2 MINUTES): RECAP THE KEY CONCEPTS OF THE LESSON, EMPHASIZING THE SIGNIFICANCE OF VARIOUS DATA TYPES AND THEIR APPROPRIATE APPLICATION. FOR HOMEWORK, INSTRUCT STUDENTS TO CREATE A BASIC PYTHON PROGRAM THAT INCORPORATES DIFFERENT TYPES OF VALUES. FOR INSTANCE, THEY COULD DEVELOP A PROGRAM THAT COMPUTES THE SUM OF TWO NUMBERS AND SUBSEQUENTLY PRESENTS A MESSAGE WITH THE OUTCOME.



LESSON PLAN – HANDOUT



SUBJECT: SOFTWARE DEVELOPMENT AND CODING

TOPIC: VALUES AND DATA TYPES IN THE PYTHON PROGRAMMING LANGUAGE – LESSON 2

KEY HIGHLIGHTS:

VALUES ARE ESSENTIAL COMPONENTS OF PROGRAMMING THAT SIGNIFY DATA.

PYTHON ACCOMMODATES A VARIETY OF DATA TYPES, INCLUDING INTEGERS, FLOATING-POINT NUMBERS, STRINGS, AND BOOLEANS.

DATA TYPES DELINEATE THE MANNER IN WHICH DATA CAN BE PROCESSED AND PRESENTED WITHIN PROGRAMS.

EXAMPLE PROGRAM:

```
lesson_plan_values_and_types.py > ...  
1  # Examples of different types of values  
2  x = 5                # integer  
3  y = 3.14             # floating-point number  
4  name = "Python"     # string  
5  is_fun = True        # boolean  
6  
7  print(x + y)         # Adding two numbers  
8  print(name + " is cool!") # Concatenating text  
9  print(is_fun and (x > 2)) # Logical expression
```

2. APPLIED ASSIGNMENT

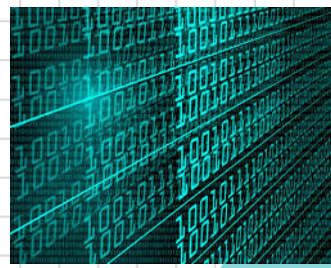
TASK: DEVELOP A PYTHON PROGRAM THAT INCORPORATES VALUES OF VARIOUS TYPES. FOR INSTANCE, SUM TWO NUMBERS, PRESENT A MESSAGE ON THE SCREEN, AND EMPLOY A LOGICAL OPERATOR TO EVALUATE A CONDITION.

3. HOMEWORK

DEVELOP A PROGRAM THAT COMPUTES THE AVERAGE OF THREE NUMBERS AND SUBSEQUENTLY PRESENTS THE RESULT IN A MESSAGE. EXPERIMENT WITH VARIOUS TYPES OF VALUES AND OPERATORS. SAVE THE PROGRAM AND BRING IT TO THE NEXT CLASS FOR PRESENTATION.



LESSON PLAN



SUBJECT: SOFTWARE DEVELOPMENT AND CODING

TOPIC: VARIABLES IN THE PYTHON PROGRAMMING LANGUAGE – LESSON 3

LESSON FOCUS AND OBJECTIVES:

1. AN OVERVIEW OF THE CONCEPT OF VARIABLES, INCLUDING THEIR FUNCTION AND APPLICATION IN PYTHON PROGRAMMING.

BY THE CONCLUSION OF THE LESSON, STUDENTS SHOULD BE CAPABLE OF CREATING VARIABLES, ASSIGNING VALUES, AND COMPREHENDING HOW VARIABLES CAN BE UTILIZED IN VARIOUS SECTIONS OF A PROGRAM.

ESSENTIAL MATERIALS:

- **COMPUTERS EQUIPPED WITH PYTHON (E.G., VISUAL STUDIO CODE, PYCHARM, OR OTHER DEVELOPMENT ENVIRONMENTS)**
- **PROJECTOR OR DISPLAY**
- **CHALKBOARD AND MARKERS**
- **ANNEXES DISPLAYING THE DATA TYPES**

1. OVERVIEW OF CHANGELINGS (15 MINUTES)

DISCUSSION (5 MINUTES): BEGIN BY DEFINING A VARIABLE AND ITS SIGNIFICANCE IN PROGRAMMING. EXPLORE HOW VARIABLES FACILITATE THE STORAGE AND MANIPULATION OF DATA WITHIN PROGRAMS.

PRESENTATION (10 MINUTES): DESCRIBE THE VARIOUS TYPES OF VARIABLES AND ELUCIDATE THEIR FUNCTIONALITY. FOR INSTANCE, DEMONSTRATE THE PROCESS OF STORING NUMERICAL, TEXTUAL, OR LOGICAL VALUES IN A VARIABLE. HIGHLIGHT THE CONVENTIONS FOR NAMING VARIABLES AND THE BEST PRACTICES ASSOCIATED WITH THEM.

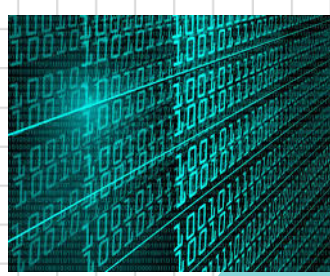
2. UTILIZING VARIABLES IN PYTHON (20 MINUTES)

EXPLANATION (10 MINUTES): ILLUSTRATE THE PROCESS OF CREATING AND UTILIZING VARIABLES IN PYTHON. DISCUSS VARIABLE VALUE ASSIGNMENT, THE REASSIGNMENT OF VARIABLE VALUES, AND THE CONVERSION AMONG VARIOUS VARIABLE TYPES.

PRESENT STRAIGHTFORWARD EXAMPLE CODE ON THE BOARD OR PROJECTOR SCREEN THAT ILLUSTRATES THE CREATION, ASSIGNMENT, AND UTILIZATION OF VARIABLES WITHIN A PROGRAM. CLARIFY HOW A VARIABLE CAN BE EMPLOYED IN VARIOUS OPERATIONS AND HOW VARIABLES ENHANCE THE EFFICIENCY OF THE CODE.



LESSON PLAN



3. PRACTICAL EXERCISE (20 MINUTES)

ACTIVITY (15 MINUTES): INSTRUCT STUDENTS TO COMPOSE A BASIC PYTHON PROGRAM THAT INCORPORATES MULTIPLE VARIABLES. FOR INSTANCE, THEY SHOULD CREATE SEVERAL VARIABLES WITH DIVERSE VALUE TYPES AND SUBSEQUENTLY EXECUTE OPERATIONS ON THOSE VARIABLES. ASSIST THEM IN CONFIGURING THEIR DEVELOPMENT ENVIRONMENT AND ENSURE THAT EACH STUDENT CAN SUCCESSFULLY RUN THEIR PROGRAM.

PAIR WORK (5 MINUTES): ORGANIZE STUDENTS INTO PAIRS TO EVALUATE ONE ANOTHER'S CODE. ENCOURAGE THEM TO ENGAGE IN DISCUSSIONS REGARDING ANY QUESTIONS OR ISSUES THAT MAY ARISE AND ASSIST EACH OTHER IN ENHANCING THEIR CODE.

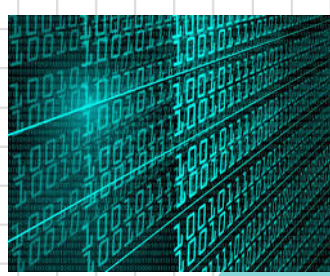
4. QUESTIONS, ANSWERS, AND SUMMARY (5 MINUTES)

QUESTIONS AND ANSWERS (3 MINUTES): INVITE STUDENTS TO POSE QUESTIONS. ADDRESS PREVALENT ISSUES OR AREAS OF CONFUSION THAT MAY HAVE EMERGED DURING THE LESSON. FACILITATE OPPORTUNITIES FOR STUDENTS TO SHARE THEIR FEEDBACK REGARDING THEIR COMPREHENSION OF THE MATERIAL.

SUMMARY AND HOMEWORK (2 MINUTES): RECAP THE KEY CONCEPTS OF THE LESSON, EMPHASIZING THE FUNDAMENTALS OF VARIABLE USAGE AND THEIR SIGNIFICANCE. FOR HOMEWORK, INSTRUCT STUDENTS TO DEVELOP A PYTHON PROGRAM THAT INCORPORATES VARIOUS TYPES OF VARIABLES AND EXECUTES DIFFERENT OPERATIONS ON THEM.



LESSON PLAN – HANDOUT



SUBJECT: SOFTWARE DEVELOPMENT AND CODING

TOPIC: VARIABLES IN THE PYTHON PROGRAMMING LANGUAGE – LESSON 3

1. VARIABLES

KEY HIGHLIGHTS:

VARIABLES SERVE AS CONTAINERS FOR DATA DURING THE EXECUTION OF A PROGRAM. WHEN CREATING VARIABLES, WE ASSIGN THEM A NAME AND DESIGNATE THE DESIRED VALUE. VARIABLE NAMES ADHERE TO PYTHON'S SYNTAX RULES, FACILITATING THE IDENTIFICATION AND UTILIZATION OF DATA WITHIN YOUR PROGRAM.

EXAMPLE PROGRAM:

```
variables_example.py > ...  
1  # Example of using variables  
2  x = 10                      # integer variable  
3  y = 5.5                    # floating-point variable  
4  name = "Alice"             # string variable  
5  is_happy = True            # boolean variable  
6  
7  # Performing operations with variables  
8  sum_result = x + y  
9  greeting = "Hello, " + name  
10 status = is_happy and (x > y)  
11  
12 print(sum_result)          # Output the result of the addition  
13 print(greeting)            # Output the greeting message  
14 print(status)              # Output the logical operation result
```

2. APPLIED ASSIGNMENT

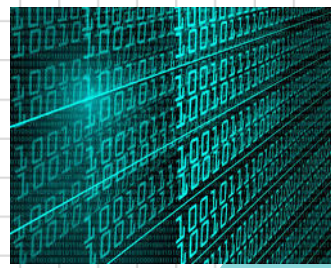
TASK: DEVELOP A PYTHON PROGRAM THAT INCORPORATES VALUES OF VARIOUS TYPES. FOR INSTANCE, SUM TWO NUMBERS, PRESENT A MESSAGE ON THE SCREEN, AND EMPLOY A LOGICAL OPERATOR TO EVALUATE A CONDITION.

3. HOMEWORK

DEVELOP A PROGRAM TO COMPUTE THE AREA AND PERIMETER OF A CIRCLE BASED ON A SPECIFIED RADIUS, UTILIZING VARIABLES TO STORE THE VALUES. SAVE THE PROGRAM AND BRING IT TO THE NEXT CLASS FOR PRESENTATION.



LESSON PLAN



SUBJECT: SOFTWARE DEVELOPMENT AND CODING

TOPIC: OPERATOR SIGNALS AND OPERANDS IN THE PYTHON PROGRAMMING LANGUAGE – LESSON 4

LESSON FOCUS AND OBJECTIVES:

INTRODUCTION TO THE VARIOUS OPERATIONAL SIGNALS (ARITHMETIC, COMPARATIVE, LOGICAL, ETC.) AND THEIR APPLICATION IN PYTHON.

BY THE CONCLUSION OF THE LESSON, STUDENTS SHOULD BE CAPABLE OF CONSTRUCTING AND ASSESSING SIMPLE EXPRESSIONS WHILE COMPREHENDING THE RELATIONSHIP BETWEEN VARIOUS OPERATIONS AND OPERANDS.

ESSENTIAL MATERIALS:

- **COMPUTERS EQUIPPED WITH PYTHON (E.G., VISUAL STUDIO CODE, PYCHARM, OR OTHER DEVELOPMENT ENVIRONMENTS)**
- **PROJECTOR OR DISPLAY**
- **CHALKBOARD AND MARKERS**
- **APPENDIX DISPLAYING OPERATIONAL SIGNALS AND OPERANDS**

1. OVERVIEW OF OPERATION SIGNALS AND OPERANDS (15 MINUTES)

DISCUSSION (5 MINUTES): START BY DEFINING OPERATION SIGNS AND OPERANDS, AND THEIR APPLICATION IN PROGRAMMING. ILLUSTRATE WITH EXAMPLES FROM DAILY LIFE, SUCH AS EXECUTING MATHEMATICAL OPERATIONS.

PRESENTATION (10 MINUTES): DEMONSTRATE THE VARIOUS OPERATIONAL SIGNALS (E.G., ADDITION, SUBTRACTION, MULTIPLICATION, DIVISION, ETC.) AND THEIR SYNTAX IN PYTHON. EXAMINE HOW THESE OPERATORS FUNCTION WITH THEIR OPERANDS AND HIGHLIGHT THE DISTINCTIONS BETWEEN ARITHMETIC AND LOGICAL OPERATORS.

2. UTILIZING OPERANDS AND OPERATORS IN PYTHON (20 MINUTES)

EXPLANATION (10 MINUTES): ILLUSTRATE THE CREATION AND UTILIZATION OF BASIC EXPRESSIONS IN PYTHON THROUGH THE USE OF OPERATORS AND OPERANDS. CLARIFY THE PROCEDURE FOR EVALUATING EXPRESSIONS AND HOW THESE OPERATIONS YIELD RESULTS.

PRESENT STRAIGHTFORWARD EXAMPLE CODE ON THE WHITEBOARD OR PROJECTOR SCREEN THAT ILLUSTRATES THE USE OF FUNDAMENTAL ARITHMETIC AND LOGICAL OPERATORS IN PYTHON. CLARIFY THE FUNCTIONALITY OF THESE OPERATIONS WITH VARIOUS TYPES OF OPERANDS, INCLUDING NUMBERS AND BOOLEANS.



LESSON PLAN



3. HANDS-ON ACTIVITY (20 MINUTES)

ACTIVITY (15 MINUTES): INSTRUCT STUDENTS TO COMPOSE A BASIC PYTHON PROGRAM THAT INCORPORATES VARIOUS OPERATION SYMBOLS AND OPERANDS. FOR INSTANCE, THEY SHOULD PERFORM SIMPLE ARITHMETIC CALCULATIONS AND FORMULATE EXPRESSIONS UTILIZING LOGICAL OPERATORS. ASSIST THEM IN CONFIGURING THEIR DEVELOPMENT ENVIRONMENT AND ENSURE THAT EACH STUDENT CAN EXECUTE THEIR PROGRAM SUCCESSFULLY.

PAIR WORK (5 MINUTES): ORGANIZE STUDENTS INTO PAIRS TO REVIEW ONE ANOTHER'S CODE. ENCOURAGE THEM TO ENGAGE IN DISCUSSIONS REGARDING ANY QUESTIONS OR ISSUES THAT MAY ARISE AND ASSIST EACH OTHER IN ENHANCING THEIR CODE.

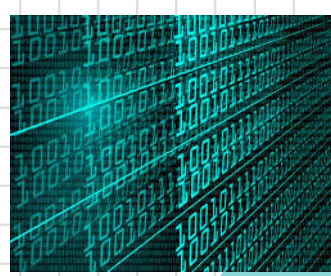
4. QUESTIONS, ANSWERS, AND SUMMARY (5 MINUTES)

QUESTIONS AND ANSWERS (3 MINUTES): INVITE STUDENTS TO POSE QUESTIONS. ADDRESS PREVALENT ISSUES OR AREAS OF CONFUSION THAT MAY HAVE EMERGED DURING THE LESSON. FACILITATE OPPORTUNITIES FOR STUDENTS TO SHARE THEIR FEEDBACK REGARDING THEIR COMPREHENSION OF THE MATERIAL.

SUMMARY AND HOMEWORK (2 MINUTES): RECAP THE KEY CONCEPTS OF THE LESSON, EMPHASIZING THE APPLICATION OF OPERATORS AND OPERANDS IN PYTHON. FOR HOMEWORK, INSTRUCT STUDENTS TO DEVELOP A PYTHON PROGRAM THAT INCORPORATES A RANGE OF ARITHMETIC AND LOGICAL EXPRESSIONS.



LESSON PLAN – HANDOUT



SUBJECT: SOFTWARE DEVELOPMENT AND CODING

TOPIC: OPERATOR SIGNALS AND OPERANDS IN THE PYTHON PROGRAMMING LANGUAGE – LESSON 4

1. OPERATIONAL INDICATORS AND OPERANDS

KEY HIGHLIGHTS:

THROUGH THE UTILIZATION OF OPERATION SIGNALS, WE CAN EXECUTE A RANGE OF OPERATIONS INVOLVING OPERANDS, WHICH MAY INCLUDE NUMBERS, TEXT, OR LOGICAL VALUES. ARITHMETIC OPERATIONS, SUCH AS ADDITION AND SUBTRACTION, ALONG WITH LOGICAL OPERATORS LIKE AND, OR, AND NOT, SERVE AS ESSENTIAL INSTRUMENTS IN PROGRAMMING. IT IS ESSENTIAL TO COMPREHEND HOW THESE OPERATIONS ALTER THE VALUE AND TYPE OF THE OPERANDS.

EXAMPLE PROGRAM:

```
variables_example.py > ...
1  # Example of using variables
2  x = 10                      # integer variable
3  y = 5.5                    # floating-point variable
4  name = "Alice"             # string variable
5  is_happy = True            # boolean variable
6
7  # Performing operations with variables
8  sum_result = x + y
9  greeting = "Hello, " + name
10 status = is_happy and (x > y)
11
12 print(sum_result)           # Output the result of the addition
13 print(greeting)             # Output the greeting message
14 print(status)               # Output the logical operation result
```

2. APPLIED ASSIGNMENT

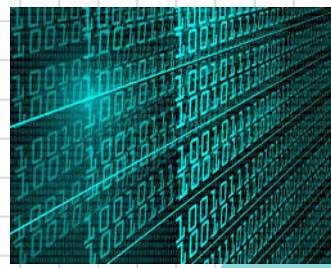
TASK: DEVELOP A PYTHON PROGRAM THAT EMPLOYS VARIOUS OPERATION SYMBOLS AND OPERANDS. FOR INSTANCE, EXECUTE ADDITION, DIVISION, AND LOGICAL OPERATIONS, FOLLOWED BY THE EVALUATION OF EXPRESSIONS.

3. HOMEWORK

DEVELOP A PROGRAM THAT COMPUTES THE AVERAGE OF TWO NUMBERS AND DISPLAYS THE RESULT. SUBSEQUENTLY, CONDUCT A LOGICAL ASSESSMENT TO DETERMINE IF THE AVERAGE EXCEEDS A SPECIFIED VALUE. SAVE THE PROGRAM AND BRING IT TO THE NEXT CLASS FOR PRESENTATION.



LESSON PLAN



SUBJECT: SOFTWARE DEVELOPMENT AND CODING

TOPIC: CONDITIONAL STATEMENTS (IF-ELSE) IN THE PYTHON PROGRAMMING LANGUAGE – LESSON 5

LESSON FOCUS AND OBJECTIVES:

THIS LESSON FAMILIARIZES STUDENTS WITH THE APPLICATION OF CONDITIONAL STATEMENTS IN THE PYTHON PROGRAMMING LANGUAGE, ENCOMPASSING THE KEYWORDS IF, ELIF, AND ELSE.

BY THE CONCLUSION OF THE LESSON, STUDENTS SHOULD BE CAPABLE OF IMPLEMENTING BASIC CONDITIONAL LOGIC IN THEIR PROGRAMS AND COMPREHENDING THE DECISION-MAKING PROCESS WITHIN CODE.

ESSENTIAL MATERIALS:

- COMPUTERS EQUIPPED WITH PYTHON (E.G., VISUAL STUDIO CODE, PYCHARM, OR OTHER DEVELOPMENT ENVIRONMENTS)
- PROJECTOR OR DISPLAY
- CHALKBOARD AND MARKERS
- ATTACHMENT DISPLAYING CONDITIONAL DIRECTIVES

1. OVERVIEW OF CONDITIONAL INSTRUCTIONS (15 MINUTES)

DISCUSSION (5 MINUTES): START BY ELUCIDATING THE NECESSITY OF CONDITIONAL STATEMENTS IN PROGRAMMING. EXPLORE THE SIGNIFICANCE OF DECISION-MAKING AND HOW PROGRAMS EMPLOY LOGICAL EXPRESSIONS TO DETERMINE THE APPROPRIATE ACTIONS TO EXECUTE.

PRESENTATION (10 MINUTES): PROVIDE AN OVERVIEW OF THE STRUCTURE OF CONDITIONAL STATEMENTS IN PYTHON. CLARIFY THE FUNCTIONS OF THE IF, ELIF, AND ELSE KEYWORDS, AND ILLUSTRATE WITH REAL-LIFE SCENARIOS THAT NECESSITATE DECISION-MAKING. SUBSEQUENTLY, PRESENT EXAMPLE CODE IN PYTHON THAT CORRESPONDS TO THESE SCENARIOS.

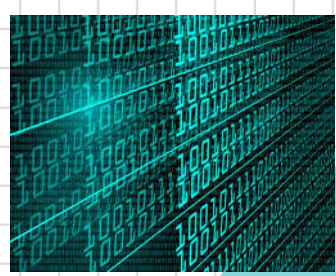
2. IMPLEMENTING CONDITIONAL STATEMENTS IN PYTHON (20 MINUTES)

EXPLANATION (10 MINUTES): DEMONSTRATE THE APPLICATION OF CONDITIONAL STATEMENTS IN PYTHON. CLARIFY THE FUNCTIONALITY OF THE IF-ELSE STRUCTURE AND ILLUSTRATE HOW LOGICAL EXPRESSIONS CAN BE EMPLOYED TO EVALUATE CONDITIONS. PROVIDE EXAMPLES FROM VARIOUS SCENARIOS, SUCH AS ASSESSING THE TIME OF DAY BASED ON THE HOUR OR DETERMINING WHETHER AN INDIVIDUAL HAS PASSED AN EXAM BASED ON THEIR SCORE.

EXAMPLE (10 MINUTES): DISPLAY STRAIGHTFORWARD EXAMPLE CODE ON THE BOARD OR PROJECTOR SCREEN THAT ILLUSTRATES THE APPLICATION OF THE IF-ELSE STRUCTURE. FOR INSTANCE, A PROGRAM THAT DETERMINES WHETHER AN INDIVIDUAL QUALIFIES FOR A DISCOUNT OR ASSESSES WHETHER A NUMBER IS POSITIVE, NEGATIVE, OR ZERO.



LESSON PLAN



3. PRACTICAL EXERCISE (20 MINUTES)

ACTIVITY (15 MINUTES): INSTRUCT STUDENTS TO COMPOSE A STRAIGHTFORWARD PYTHON PROGRAM THAT INCORPORATES AN IF-ELSE STRUCTURE. FOR INSTANCE, THEY COULD CREATE A PROGRAM THAT EVALUATES THE USER'S AGE TO DETERMINE ELIGIBILITY FOR ENTRY INTO A SPECIFIC EVENT. ASSIST THEM IN CONFIGURING THEIR DEVELOPMENT ENVIRONMENT AND ENSURE THAT EACH STUDENT CAN EXECUTE THEIR PROGRAM SUCCESSFULLY.

PAIR WORK (5 MINUTES): ORGANIZE STUDENTS INTO PAIRS TO REVIEW ONE ANOTHER'S CODE. ENCOURAGE THEM TO ENGAGE IN DISCUSSIONS REGARDING ANY QUESTIONS OR ISSUES THAT MAY ARISE AND ASSIST EACH OTHER IN ENHANCING THEIR CODE.

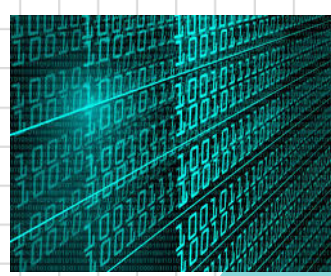
4. QUESTIONS, ANSWERS, AND SUMMARY (5 MINUTES)

QUESTIONS AND ANSWERS (3 MINUTES): INVITE STUDENTS TO POSE QUESTIONS. ADDRESS PREVALENT ISSUES OR AREAS OF CONFUSION THAT MAY HAVE EMERGED DURING THE LESSON. FACILITATE OPPORTUNITIES FOR STUDENTS TO SHARE THEIR FEEDBACK REGARDING THEIR COMPREHENSION OF THE MATERIAL.

SUMMARY AND ASSIGNMENT (2 MINUTES): RECAP THE KEY CONCEPTS OF THE LESSON, EMPHASIZING THE SIGNIFICANCE OF CONDITIONAL STATEMENTS. FOR HOMEWORK, INSTRUCT STUDENTS TO DEVELOP A PYTHON PROGRAM THAT SIMULATES A BASIC DECISION-MAKING PROCESS USING AN IF-ELSE STRUCTURE.



LESSON PLAN – HANDOUT



SUBJECT: SOFTWARE DEVELOPMENT AND CODING

TOPIC: CONDITIONAL STATEMENTS (IF-ELSE) IN THE PYTHON PROGRAMMING LANGUAGE – LESSON 5

1. CONTINGENT GUIDELINES

MAIN POINTS:

CONDITIONAL STATEMENTS ENABLE THE PROGRAM TO MAKE DECISIONS BASED ON THE FULFILLMENT OF SPECIFIC CONDITIONS.

THE KEYWORDS IF, ELIF, AND ELSE ARE EMPLOYED TO MANAGE VARIOUS SCENARIOS WITHIN YOUR CODE.

BOOLEAN EXPRESSIONS DICTATE THE EXECUTION OF SPECIFIC SEGMENTS OF CODE.

EXAMPLE PROGRAM:

```
if_else_examples.py > ...  
1  # Example of using if-else statements to determine the time of day  
2  hour = int(input("Enter the current hour (0-23): "))  
3  
4  if 5 <= hour < 12:  
5      print("Good morning!")  
6  elif 12 <= hour < 18:  
7      print("Good afternoon!")  
8  elif 18 <= hour < 22:  
9      print("Good evening!")  
10 else:  
11     print("Good night!")
```

2. PRACTICAL ASSIGNMENT

TASK: DEVELOP A PYTHON PROGRAM THAT EMPLOYS AN IF-ELSE STRUCTURE TO EVALUATE MULTIPLE CONDITIONS. FOR INSTANCE, DETERMINE WHETHER A NUMBER IS EVEN OR ODD, OR ASSESS A PERSON'S ELIGIBILITY FOR A DISCOUNT BASED ON THEIR AGE.

3. HOMEWORK

DEVELOP A PROGRAM THAT EVALUATES A TEMPERATURE AND DETERMINES WHETHER THE WEATHER IS CLASSIFIED AS COLD, WARM, OR HOT. EMPLOY AN IF-ELSE STRUCTURE FOR THE DECISION-MAKING PROCESS AND SAVE THE PROGRAM FOR PRESENTATION IN THE UPCOMING CLASS.



01. 01. 2024 –31. 12. 2025



**Funded by
the European Union**



**Spojená škola
Kollárova 17, Sečovce**

**Móricz Zsigmond
Református Kollégium,
Gimnázium, Technikum,
Általános Iskola és Óvoda**



**SOLSKI CENTER
SLOVENJ GRADEC**

2023-2-SK01-KA210-VET-000184496