Select a group

You can work either with your friend or individually. If you work together with somebody, then please register yourself to a group in Moodle. Make sure that you and your partner are in the same group.

1. Review of previous material

First we review the material covered in this course so far. You can check your answers on the computer.

Write a loop that prints out the numbers 1, 2, ..., 9.

Let \textit{word} be a string variable with some value. Write a statement, after executing of which the first letter of the value of \textit{word} has been removed.

Write a piece of code that prints all elements of a list \textit{t} that are divisible by neither 3 nor 5.

What does the following piece of code output? For each print command write its output after it.

```python
a = [4, 6, 8]
b = a
a[0] = b[1]
print(a)
a[1] = b[2]
print(a)
a[2] = b[0]
print(a)
print(b)
```

How can it be that by changing the elements of the list \textit{a}, the list \textit{b} changes?
2. Chessboard

Modify your second homework program (*home2.py*) in such a way that it draws a colored chessboard of \( m \) by \( n \) cells:

![Chessboard Image](image)

3. Polygons

a) Write a function that instructs the turtle to draw a regular polygon with a given number of vertices and a given side length. For example,

\[
\text{polygon}(3, 10) \quad \text{draws an equilateral triangle with side length 10}
\]
\[
\text{polygon}(4, 100) \quad \text{draws a square with side length 100}
\]
\[
\text{polygon}(5, 200) \quad \text{draws a regular pentagon with side length 200}
\]

b) Write a second function with the same parameters, that draws a non-convex polygon:

![Non-Convex Polygon Image](image)

Since this is feasible only if the number of vertices is odd, the function should check the number of vertices before drawing. If the number of vertices is even, then the function should draw nothing.

*Hint.* At the vertex the turtle should turn \( 180^\circ \times (n-1)/n \) degrees.

c) Using the above functions, draw 30 polygons with random sizes and random numbers of vertices at random locations on the screen.

*Hint.* The turtle can be made move faster with the commands `delay(\theta)` and `speed(\theta)`). Even the command `speed(\theta)` can be used, this makes the turtle to move instantly.
4. Interpreter

To control a device (robot, automaton etc) often a specialized language is used. This language acts as a common medium that is understood both by the device and the human who commands the device, so the communication between them becomes possible. Depending on the nature and the objectives of the tasks to be performed, the language can be designed differently.

In this exercise, the device will be the turtle. The language has only two commands:

- **moveto(x, y)** moves the turtle to the position (x, y) without drawing a line
- **drawto(x, y)** draws the line and moves the turtle from its current position to the position (x, y)

The commands are given in a file, each command on a separate line. For example:

```plaintext
moveto(-100, -100)
drawto(100, -100)
drawto(100, 100)
drawto(-100, 100)
doneto(-100, 100)
```

The turtle starts at the coordinates (0, 0).

Write a program that prompts for a filename, reads commands from the file and moves the turtle according to the commands on the screen.

In essence, our program is the interpreter for the language described above. The interpreter translates the commands into the language understood by the turtle. The same principle is also behind interpreters of other languages; interpreter simply is a program that reads commands and executes them.

*Hint:* see the documentation of the turtle module.

5. Fractals

The following picture shows a fractals at levels 1, 2, 3, and 4:

![Fractals](image)

Write a function that takes side length and level number as arguments and draws the fractal of the given level with the turtle.

A common way to draw fractals is to use recursive functions. A function that calls itself inside its body is called recursive. For example, the following function computes and returns the value of the product $1 \times 2 \times 3 \times \ldots \times n$, where $n$ is a non-negative integer.

```python
def fact(n):
    if n == 0:
        return 1
```
else:
    return fact(n - 1) * n

If n is 0, then it simply returns 1. Otherwise it computes the product $1 \times 2 \times 3 \times \ldots \times (n - 1)$, using the same function, and then multiplies the result with $n$. You can try to run the above function in Thonny step by step with the debugger to see how it works.

To draw the fractal above, we can start at simpler fractals, which are shown on the following picture at levels 1, 2, 3, and 4:

The algorithm can be as follows:
1. draw a fractal of level $n - 1$
2. turn a little less than 90 degrees
3. draw a fractal of level $n - 1$
4. turn back sharply
5. draw a fractal of level $n - 1$
6. turn
7. draw a fractal of level $n - 1$

The starting point, the fractal of level 0, is simply a line segment of given length.

Implement this algorithm in Python as a function. Then write the second function that draws the fractal square of given size and level.