CSCI 2202 Computer Modelling for Scientists Lab 2
Turtle Geometry Part 2: Polygons, Spirals and Pursuit Curves

Lab Goals:

- Using loops `for; while` in programs.
- Understand how to use the Python API.
- Learn how to use the features of Turtle using the documentation
- Learn alternate methods for making shapes.
- Understanding pursuit curves.

Part I: Drawing Polygons

In the previous lab, you learned how to use the basic features of the Turtle module, which included functions such as `forward(), left(), right()` *etc.* In this lab, you will going to look at the rest of the commands available for `turtle`. To start, visit the following URL:

https://docs.python.org/3/library/turtle.html

The link above site is a part of what is known as the *Python Application Program Interface (API).* An API is a specific kind of documentation which tells a user what a particular Python module or language feature consists of. In this case, we are looking at the API for the Turtle module.

Each API page is formatted in a similar fashion, typically starting with an Introduction which explains what the module does and how to use it, followed by a list of all the *methods* available to the module. A *method* is an action a given object can perform. In the case of `Turtle`, `forward(x)` is a method that moves the turtle forward a distance x on the screen.

Look through the Turtle API and try out some of the methods for yourself. In addition to the methods you learnt in the previous lab, some useful methods to understand would be: `setheading(), heading(), distance(), and position()`.

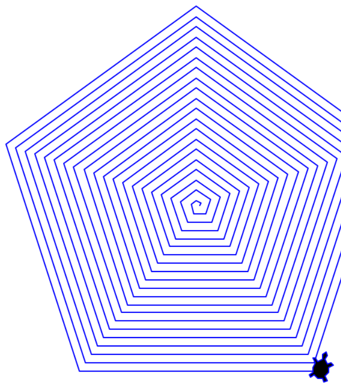
Ex. 1 (a) Start by modifying `polygon.py` from the last lab. The modified program should ask the user to input a value for the variable `numSides`.
(the Python command `numSides = int(input(''Enter the number of sides ''))` *does the job.* The `input` *always* reads user input as a *String.* The `int` to the left of the `input` *casts* the input string into an integer. If on the other hand, the user was supposed to enter *real* values, the `int` should be replaced with `float`).

Ex. 1 (b) Use the program created in **Ex 1(a)** to draw *(i)* a pentagon, *(ii)* a hexagon *(iii)* an octagon it (iv) a decagon (10-sided fig.) and *(v)* a dodecagon (12-sided figure). *Make sure that all the polygons fit on the turtle screen.* The figures will need to be scaled. (*Hint: Notice that the square (of side 100) has a perimeter of 400. Pick the perimeter of the polygon and then find the appropriate side-length.).* **Submit images for each figure**.

Ex. 1 (c) Use your program to draw a (approximate) circle. **Submit code and image**.

Ex 2(a) Create a program `spiral.py` that draws (outward growing) spiral polygons.

- *Notice* that you <u>do not know the exact number of times</u> the turtle will repeat the moves and turns.
- Therefore you cannot use a `for` loop. *However, you know about a loop that can be controlled by a condition.*
- Use the condition that *the drawing should stop when the* `side-length` *of the spiral polygon reaches (say) 200.* An example of a pentagonal spiral is shown below.



Part II: Drawing Pursuit Curves

Imagine a dog chasing a car. A *pursuit curve* is the path traced out by the dog as it chases the car. The dog *always* runs directly towards the car along the *line of sight*.

Ex 3 Create a program `turtleChase.py`. In this program, you will trace the pursuit curve of a dog chasing a car. The car travels, with uniform speed, from the bottom left of the screen $(-X/2, 0)$ to the bottom right of the screen $(X/2, 0)$. See figures on the next page.

- Place the *car i.e. the "Red" turtle* at $(-X/2, 0)$ (The centre of the screen is at $(0, 0)$ and
- The "Red" turtle moves to the right, a total distance X across the screen, at a constant speed, leaving a Red line along its path.
- The *dog i.e. the Blue turtle* is initially placed at $(0, X/2)$.
- The Blue turtle runs *along the line of sight* towards the Red turtle, for a short time/distance.
- In this time the Red turtle has moved, so the Blue turtle should point towards the Red turtle (recall `setheading()`) and runs towards the Red turtle for a short time/distance.
- As long as your program runs, the Blue turtle points towards the Red turtle and run to it, continuing the chase for the *same short time/distance as above.*
- The Blue turtle should leave a blue line tracing its path.

•The turtles positions initially and finally with the paths traced out: