Lab 7
Turtle Graphics, Functions, Lists

Overview

In this lab, we do some simple Turtle Graphics to get a nice effect without too much work.
You will define several "utility" functions to focus on the main task: a program that loops indefinitely

Lab 7-1.py

Loop

Ask for various characteristics of the drawing such as the angle to turn at.
Draw as requested
Ask whether to do another drawing

Lab 7-2.py

Set up an empty favorites list
Loop

Ask for various characteristics of the drawing such as the angle to turn at.
Draw as requested
Ask whether to save this drawing as a favorite
If yes,
append angle to favorites list
Ask whether to do another drawing

After the loop

Display the favorites; "for" syntax works nicely here.

```
for my_angle in favorites:
      print("Angle:", my_angle, "degrees")
# end for
```

Extra credit

Ask if the person using the program would like the item re-displayed.
If so, do it.

List Basics

Create an empty list

```
favorites = []
```

Add on to the end of a list

```
favorites.append(angle)
```

Loop through a list

```
for saved_angle in favorites:
      # do something with item:
      print("Angle:", saved_angle)
      # do you want to see it redisplayed?"
      print("Redisplay ?")
      redisplay = get_ok()
      if redisplay == 'y':
          draw_picture(angle, 500)  # if you have a function defined to do this
      # end if
# end for
```

Some useful functions - general

```
import turtle

def get_ok():
    ''' ( ) -> str
        Display a prompt message to person running the program
        return 'y' or 'n'
    '''
    ok = ''
    while ok != 'y' and ok != 'n':
        ok = input("Do again ? Press y or n ")
        ok = ok.lower() + "x" # make lowercase; at least 1 char long
        ok = ok[0]  # get 1st char. only
    # end while
    return ok
# end def

def get_int(prompt_message):
    ''' (str) -> int
        Display prompt message (with trailing space)
        to person using program
        Return user typed value converted to int
    '''
    msg = prompt_message + " "
    temp = input(msg)
    return int(temp)
# end def

def get_float(prompt_message):
    ''' (str) -> float
        Display prompt message (with trailing space)
        to person using program
        Return user typed value converted to float
    '''
    temp = input(prompt_message + " ")
    return float(temp)
# end def


def ask_y_or_n(question):
    ''' (str) -> str
        Display a prompt question + " Press y or n "
        to person running the program; return 'y' or 'n'
    '''
    answer = ''
    while answer != 'y' and answer != 'n':
        answer = input(question + " Press y or n ")
        answer = answer.lower() + "x" # make lowercase; at least 1 char long
        answer = answer[0]  # get 1st char. only
    # end while
    return answer
# end def
```

Nice looking graphics from a simple loop:
Repeat a large number of times
Move the turtle forward (a little further each time through the loop)
Turn the turtle

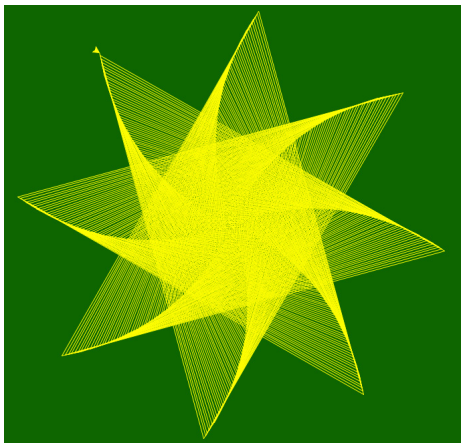
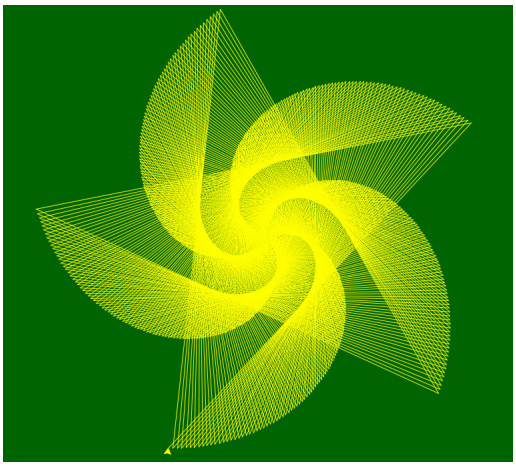


Image resulted from an angle of 135.05



and this from an angle of 144.02

Lab7-1.py main program (after various function definitions)

```
more = 'y'
while more == 'y':
    turtle.reset()   # Erase screen for fresh start
    turtle.bgcolor('dark green')  # background color
    angle = get_float("Turn angle (91 - 160, works OK, 98, 135.1, 144.2 good)")
    number_of_lines = get_int("Draw how many lines (300 - 1000 work well)")

    # sample code adapted from a StackOverflow demo with 98 degree angle

    for i in range(number_of_lines):
        turtle.color('yellow')
        turtle.forward(i)
        turtle.right(angle)
    # end for
    more = get_ok()
# end while

print("Finished")
```

Play around with your lab 7-1; see which angles produces pleasant results.

You could write down results, but you are using a computer, and you can make it remember the nice angles.

You are now ready for **Lab7-2.py**
General strategy
Before the loop

#Create an empty favorites list
`favorites = []`

Inside the loop

After the "for" loop that draws the picture
Ask whether to add this picture to favorites
You will use a variation on get_ok(), **`ask_y_or_n(question)`** to get a yes or no response.
If yes, append the angle you used to the favorites list.
`favorites.append(angle)` `# Remember this angle by adding it to the list`

Here is some code that follows the image drawing; put it just before the "more = get_ok()"

```
save_favorite = ask_y_or_n("Save this picture as a favorite?")
if save_favorite == 'y':
    favorites.append(angle)
# end if
```

Finally, after the big while loop ends, you need to display your list of favorites, somewhat like this:

```
count = 0
number_of_lines = 600
for angle in favorites:
    print(count, "Picture from angle", angle)
    count += 1
# end for
```

Lab7.2 Extra credit
Instead of just printing a favorite angle, do so, then **ask if you should display the image.**
You could just duplicate the "for" loop that creates the image. A better bet: make a display_picture(angel) function. Then call that function (and call it also from the big "while" loop).

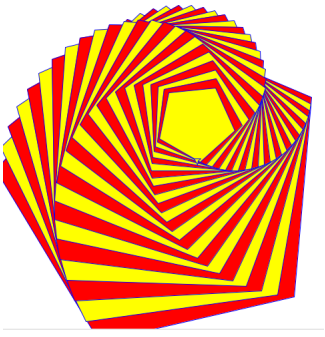
Grading rubric
Lab 7-1.py

10 points	Collection of useful functions like get_float(prompt)
5 points	Interact with user for angle to turn, number of lines to draw.
10 points	Display the image caused by turtle forward, turtle turn repeatedly.

Lab 7-2.py	Add a favorites feature:
5 points	Create a favorites list
5 points	Append angle data to favorites list.
5 points	Display list of favorite angles

Lab 7 Extra Credit

10 points	Re-create the image of each saved favorite user wants to see again.
10 points	Draw a spiral image using pentagons or hexagons (5 or 6 sided figures with 2 alternating colors



```
def draw_polygon(length, n_sides):
    ''' (number, int) -> None
        Draw a polygon with n_side
        Each side is length pixels long
    '''
    angle = 360 / n_sides
    for i in range(n_sides):
        turtle.forward(length)
        turtle.left(angle)
    # end for
# end def

def invisible_move(distance):
    '''(number) -> None
        Move distance without leaving a trace
        when done, pen is down, ready to draw
    '''
    turtle.penup()
    turtle.forward(distance)
    turtle.pendown()
# end def
```

5 points Modify your spiral design to treat the entire spiral as one fill object, similar to this: