Lab 7: Pattern Drawing using Loops

Recap – Some things to remember when using a while loop

The syntax of a while loop is as follows:

```
while <condition>:          # must always end with a colon (:)
    <statement>             # can have one or many statements
    …
```

Example 1:

To print numbers 1 to 3 on screen using a while loop, we can try the following:

```
counter = 1
while counter <=3:
    print(counter)  # will print the counter value
    counter = counter + 1
```

Example 2:

To print numbers 1 to 3 on screen using a while loop, we can **also** try using a variable to store the number of times to loop. Type in the following:

```
numToLoop = 3
counter = 1

while counter <= numToLoop:
    print(counter)  # will print the counter value
    counter = counter + 1
```

Example 3:

To ask the user to enter the number of numbers they wish to print, type in the following:

```
numToLoop = int(input('Please enter the number of times to loop:'))
counter = 1

while counter <= numToLoop:
    print(counter)  # will print the counter value
    counter = counter + 1
```

Example 4:

To ask the user to enter the number of numbers they wish to print, and including an error check, type in the following:

```
numToLoop = int(input('Please enter the number of times to loop (2-5):'))
while numToLoop < 2 or numToLoop>5:
    numToLoop = int(input('Invalid!. Enter the number between(2-5):'))

counter = 1
while counter <= numToLoop:
    print(counter)  # will print the counter value
    counter = counter + 1
```

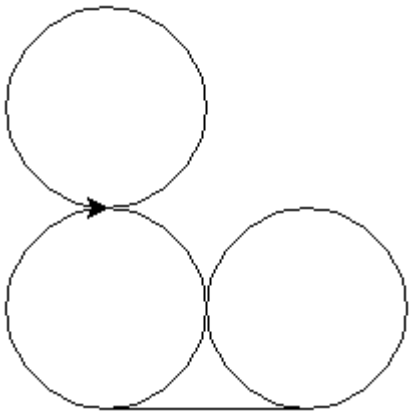
How to draw a circle?

The **circle(radius)** function draws a circle on the current position in the direction that the arrow head is facing.

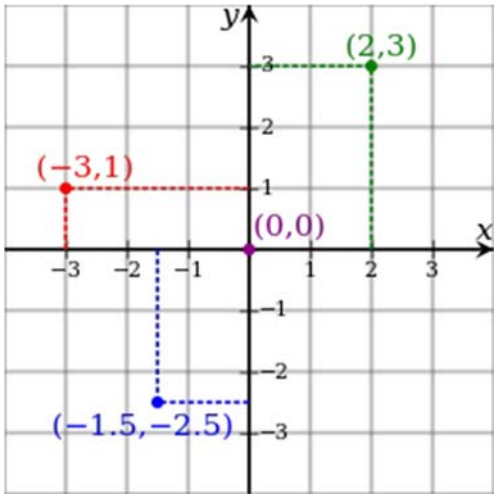
What is the radius? The distance between the center point and the edge.
What is the diameter? Twice of the radius.

```
>>> from turtle import *
>>> pos()        # will display the current position of turtle
(0.00,0.00)
>>> circle(50)  # will draw a circle with radius 50
>>> goto(-100,0)
>>> pos()
(-100.00,0.00)
>>> circle(50)
>>> penup()     # can also use pu()
>>> goto(-100,100)
>>> pendown()  # can also use pd()
>>> circle(50)
```

Sample output:



Cartesian coordinate system



Note:

- x value increases to the right y value increases upwards
- two points on the same horizontal line have the same y value two points on the same vertical line have the same x value

Task1:

Create a new sript called Lab7Task1.py to include the code to draw three circles, on the same horizontal line. The start point of the first circle is (0, -100) and the radius is 50.

```
from turtle import *

#create the variables
radius = 50
x, y = 0, -100 # notice that we have assigned multiple values

# move to the starting point without drawing lines
penup()
goto(x,y)
pendown()
circle(radius) #draw the first circle

# move to the starting point of second circle
x = x + (radius *2)
penup()
goto(x,y)
pendown()
circle(radius) #draw the second circle

# move to the starting point of third circle
x = x + (radius *2)
penup()
goto(x,y)
pendown()
circle(radius) #draw the third circle
```

There are so much repeated code in above example. The repeated code can be avoided by using a loop, as follows:

Model answer:

```
from turtle import *

#create the variables
radius = 50
x, y = 0, -100      # notice that we have assigned multiple values


count=0
while count <3:      # will loop 3 times
    penup()
    goto(x,y)
    pendown()

    circle(radius)  #draw the circle

    x = x + (radius *2) # starting point of next circle
    count = count +1
```

Task 2: Modify **Lab7Task 1.py** and add another loop that draws four vertical circles next to the last circle. (**Hint**: Since the circles go vertically up this time, the value of y should be incremented)

Note: at the end of last loop, x has been set to a new location. You can start from here.

Model answer:

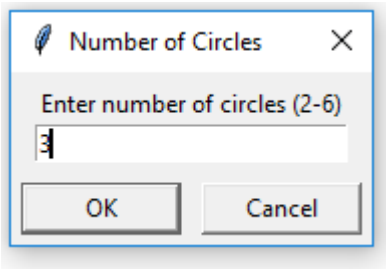
```
.
.
.
count=0
while count <4:      # will loop 4 times
    penup()
    goto(x,y)
    pendown()

    circle(radius)  #draw the circle

    y = y + (radius *2) # starting point of next circle
    count = count +1
```

Task 3: Create a new script called **Lab7Task 3.py** that will ask the user to enter the number of circles. Then it should draw the specified number of circles in a vertical line connecting the circles. The radius of each circle is 10.

Answer: To ask for user input use a textinput

```
from turtle import *

#create the variables
radius = 10
numCircles = int(textinput('Number of Circles','Enter number of circles (2-6)'))

while numCircles <2 or numCircles >6:
    numCircles = int(textinput('Number of Circles','Invalid!! Enter between(2-6)'))


x, y = 0, 0     # notice that we have assigned multiple values



count=0
while count <numCircles:     # will loop for specified number of circles
    penup()
    goto(x,y)
    pendown()

    circle(radius)  #draw the circle

    y = y + (radius *2) # starting point of next circle
    count = count +1
```

Sample output:



turtle output:

Bonus Task: Modify **Lab7Task3.py** so that the program will ask the user for the number of circles to draw and validate that it must be in the range 2 - 6.

```
while numCircles <2 or numCircles >6:
    numCircles = int(textinput('Number of Circles','Invalid!! Enter between(2-6)'))
```

Question: Where will you place the above code?

Let the radius control how many circles to draw

Task 4: Create a new script called **Lab7Task 4.py**. Instead of drawing a specific number of circles, modify the program such that the radius increases for each circle by 10 in a horizontal line. The program should stop when the radius is larger than 100. (Note: This should give you an idea of how to approach one of the questions of Assignment 1)

```
from turtle import *

#create the variables
radius = 10
radiusIncrement = 10
maxRadius = 80
x, y = -400, 0      # notice that we have assigned multiple values

while radius <maxRadius: # will loop till radius exceeds maxRadius
    penup()
    goto(x,y)
    pendown()
    circle(radius)  #draw the circle

    x = x + (radius*2) + radiusIncrement # starting point of next circle
    radius = radius + radiusIncrement
```