Worksheet 1
Self-reference

1) Play "Guess the number" with a classmate. In this game the first player thinks of a natural number between 1 and 15 (including 1 and 15). The second player has to guess it. In each attempt the first player can only answer "higher", "lower" or "correct". The second player gets 1 point for each attempt. The players take turns to be the first and second. The winner is the one with less total points after 2 rounds.

a) What is the best strategy?

You don't have to finish these questions

With the best strategy what is the maximum number of questions needed to guess a number between

b) 1 and 15?

c) 1 and 16?

d) 1 and 2^n-1?

e) 1 and 2^n?

You don't have to finish these questions (many of them are just jokes!)

-Here are some examples of self-referential sentences:

> This sentence contains five words.
> This sentence no verb.
> TLA stands for Three-Letter Acronym.

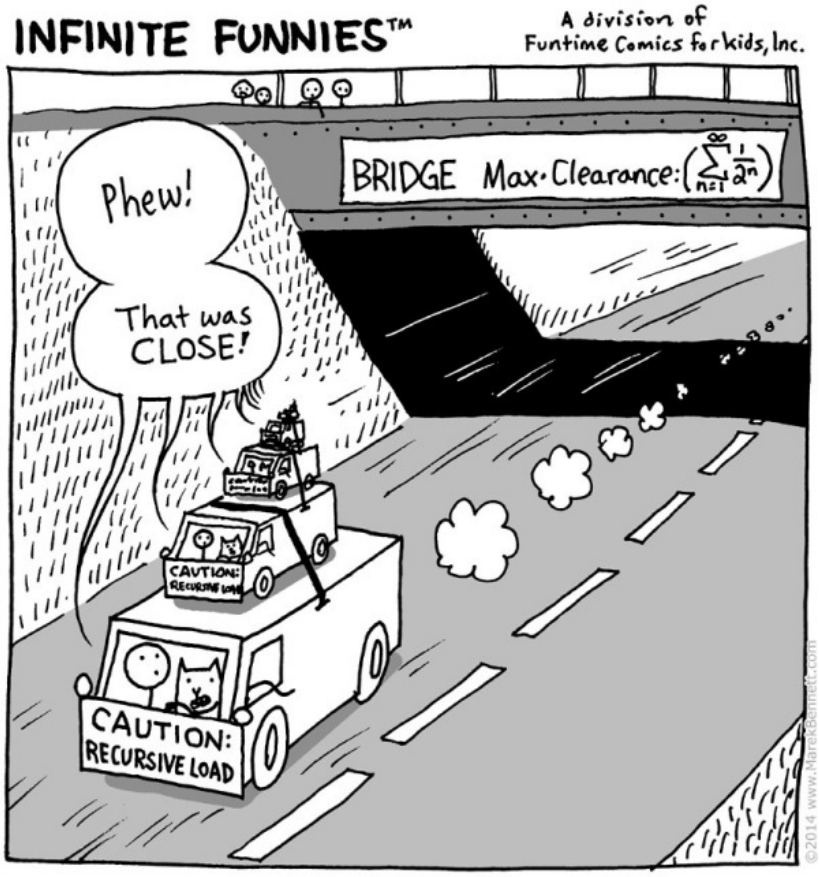
2) Follow these instructions:

Read this sentence and do what it says twice.

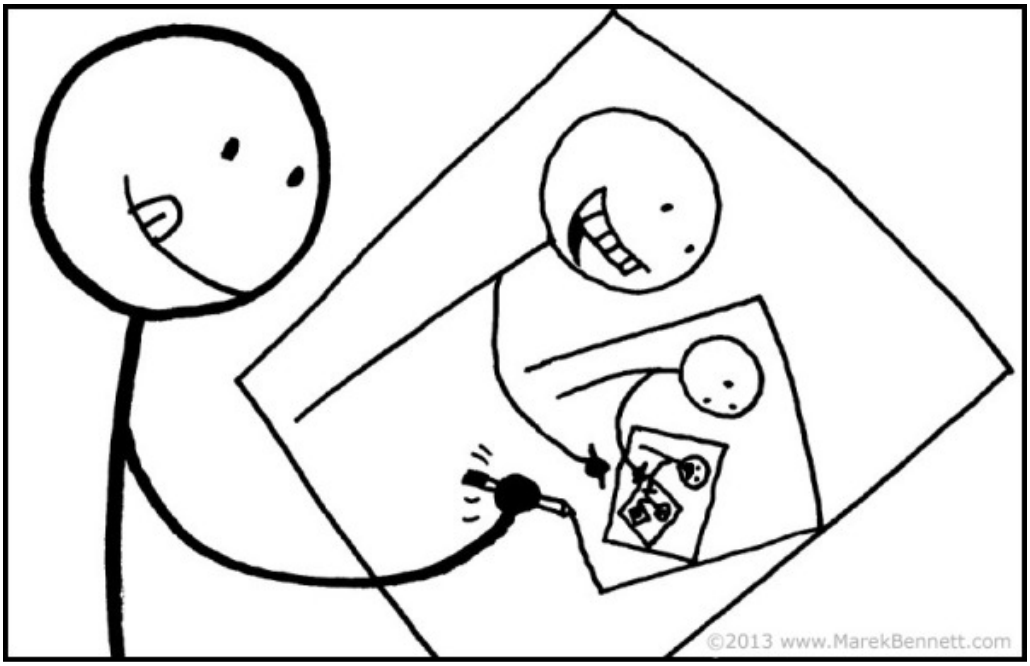
3) If I say to you: "I am lying", is that true or false?

4) Do you see a contradiction between these cartoons?

- http://marekbennett.com/2014/03/06/recursive-load/



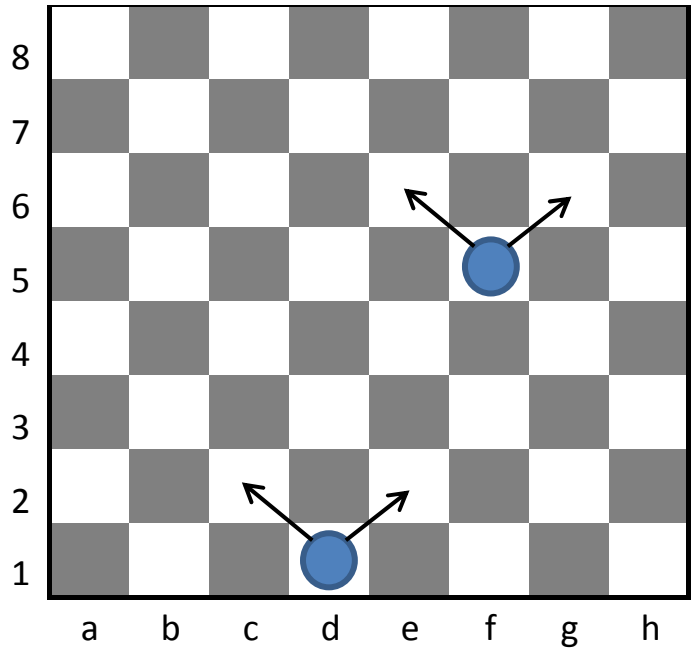
- http://marekbennett.com/2013/12/27/infinite-doodle/



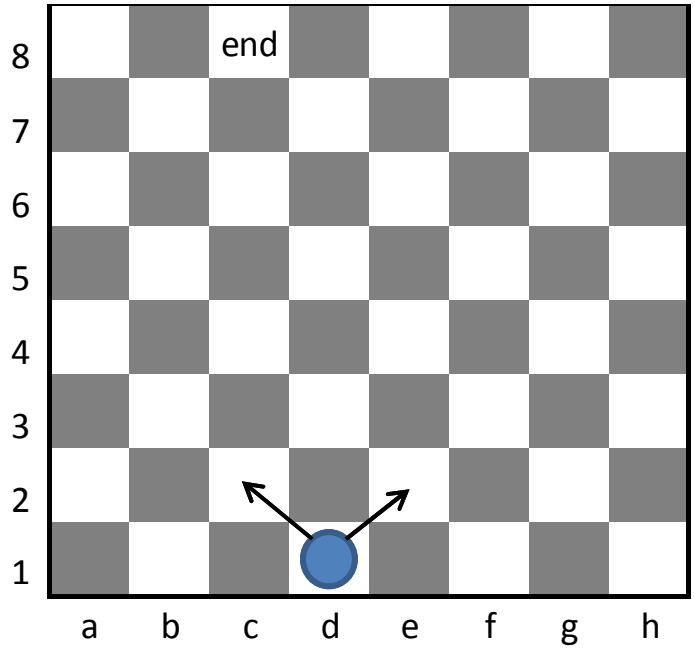
It might not look it, but adding all the details to this drawing took FOREVER.

Worksheet 2
Draughts path counting

In the draughts game an uncrowned piece can only move one step diagonally forward. For example, an uncrowned piece in d1 can only move to c2 or e2 and an uncrowned piece in f5 can only move to e6 or g6:



If you start with an uncrowned piece in d1, how many paths are that end in c8?



Worksheet 3
Recursive play writing

1) Could you write how the blackboard would look after performing the following play? Assume the blackboard is empty before the play.

```
B1(n)
    If n > 0
        Write n in the blackboard
        B1(n-1)
B1(4)
```

2) Could you write how the blackboard would look after performing the following play? Assume the blackboard is empty before the play.

```
B2(n)
    If n > 0
        B2(n-1)
        Write n in the blackboard
B2(4)
```

3) For the following sub-play:

```
B3(n)
    If n > 0
        Write n in the blackboard
        B3(n-1)
        Write n in the blackboard
```

Could you write how the blackboard would look after performing the following plays? Assume the blackboard is empty before the play.

a) B3(1)	b) B3(2)	c) B3(3)	d) B3(4)

Worksheet 4
Algorithms

1) For the following algorithm:

```
P1(n)
    If n = 1
        Write * in a new line in the blackboard
    else
        P1(n-1)
        Write * n times in a new line in the blackboard
        P1(n-1)
```

Could you write how the blackboard would look after running the following statements? Assume the blackboard is empty at the beginning.

a) P1(1)	b) P1(2)	c) P1(3)	d) P1(4)

You don't have to finish these questions.

2) Could you write an algorithm P2 that will give the following outputs:

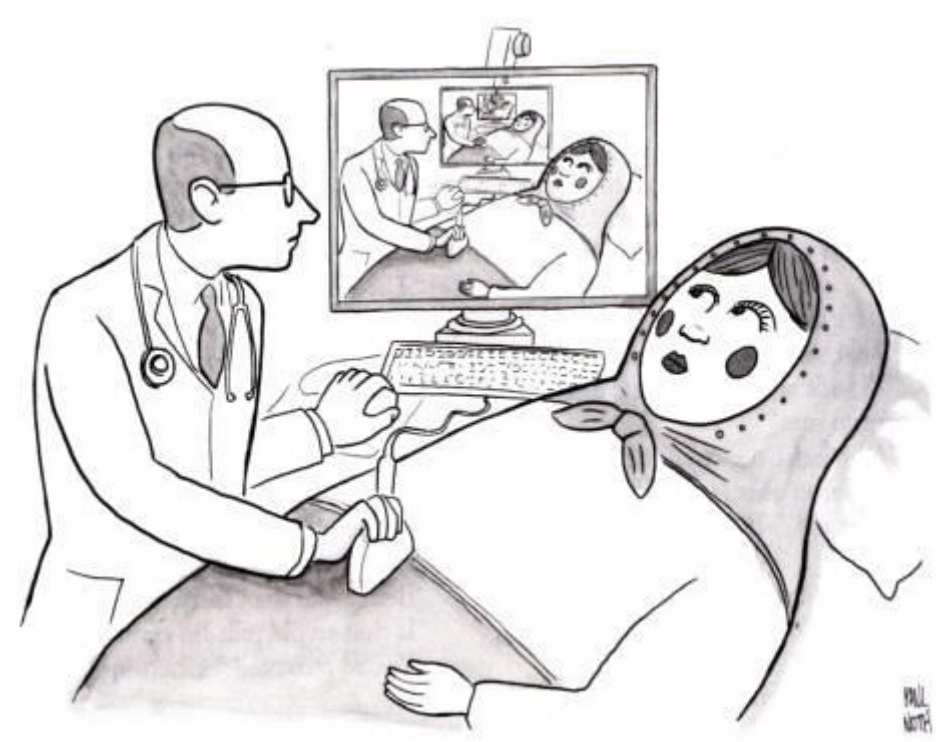
P2(1)	P2(2)	P2(3)	P2(4)
*	** * *	*** ** * * ** * *	**** *** ** * * ** * * *** ** * * ** * *

3) Could you write an algorithm P3 that will give the following outputs:

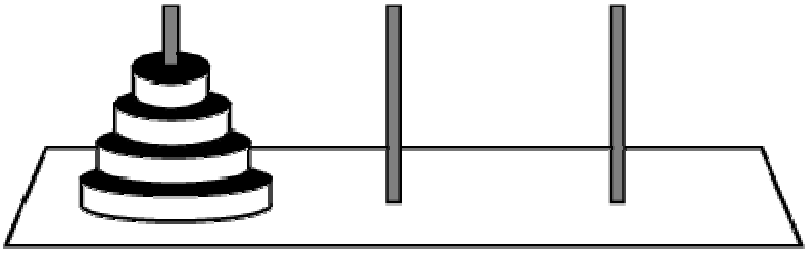
P3(1)	P3(2)	P3(3)	P3(4)
*	* * **	* * ** * * ** ***	* * ** * * ** *** * * ** * * ** *** ****

4) Is there something wrong with this cartoon?

From the New Yorker, 10/Apr/10:



Worksheet 5
Tower of Hanoi



The objective of the puzzle is to move the entire stack from rod 1 to rod 3, obeying the following simple rules:

a. Only one disk can be moved at a time.

b. Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack i.e. a disk can only be moved if it is the uppermost disk on a stack.

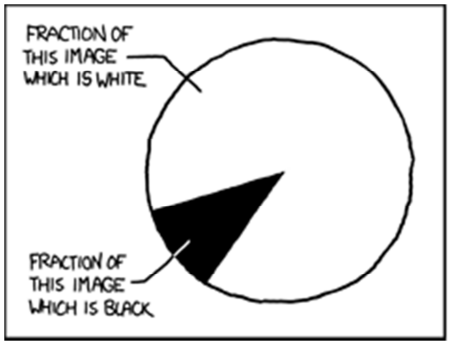
c. No disk may be placed on top of a smaller disk.

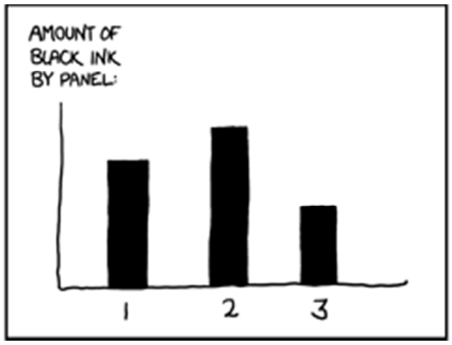
1) Try to solve the puzzle for 4 disks with the paper version that we are providing you. If you have online access you could use this version: http://www.mathsisfun.com/games/towerofhanoi.html.

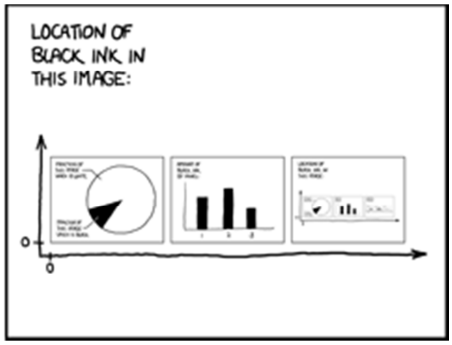
You don't have to finish this question.

2) Write a recursive algorithm to solve a Tower of Hanoi of size n.
3) Conjecture what is the minimum number of moves.

-From xkcd:







Worksheet 6
Turtle programming

Remember, some of the instructions that you can give to the turtle are:

forward(distance)	backward(distance)
left(angle)	right(angle)
penup()	pendown()
if then else	speed(n)
def	

Note: Unfortunately when you use the turtle graphics sometimes there are problems resizing or moving the Python Turtle Graphics window (where the turtle appears). In that case you can click on another window and click back on the Python Turtle Graphics window then resize it and/or move it.

Tip: You can press Alt-p one or more times to bring a previous command, then you can edit it and run it.

1) For the following program:

```
def pol(length, angle, n):
    if n > 0:
        turtle.forward(length)
        turtle.left(angle)
        pol(length, angle, n-1)
```

Could you draw how the screen would look after running the following instructions? Assume the screen is empty at the beginning.

```
pol(100, 120,3)
pol(100, 90,4)
pol(100, 72,5)
pol(100, 60,6)
```

2) What figure would you get if you increase a lot the number of sides?

3) Additionally now we define:

```
def polygon(length, n):
    pol(length, 360/n, n)


def ManyPolygons1(length, n):
    if length>0:
            polygon(length, n)
            ManyPolygons1(length-10, n)

def ManyPolygons2(length, n):
    if n > 2:
            polygon(length, n)
            ManyPolygons2(length, n-1)
```

Could you draw how the screen would look after running the following instructions?

a) ManyPolygons1(100,8)

b) turtle.reset()
   ManyPolygons2(100,8)

Worksheet 7
Turtle programming

Remember, some of the instructions that you can give to the turtle are:

forward(distance)	backward(distance)
left(angle)	right(angle)
penup()	pendown()
if then else	speed(n)
def	

Note: Unfortunately when you use the turtle graphics sometimes there are problems resizing or moving the Python Turtle Graphics window (where the turtle appears). In that case you can click on another window and click back on the Python Turtle Graphics window then resize it and/or move it.

Tip: You can press Alt-p one or more times to bring a previous command, then you can edit it and run it.

1) When we run:
   polygon(100,7)
   the turtle does not go back to its initial position as it does when we run
   polygon(100,6)

   How would you modify the definition of pol:

```
def pol(length, angle, n):
    if n > 0:
            turtle.forward(length)
            turtle.left(angle)
            pol(length, angle, n-1)
```

   to make sure that the turtle goes back to its initial position after drawing the polygon?

You don't have to finish these questions. If you find them hard just try them in your computer.

2) Now to draw spirals we define a function similar to the one to draw polygons but with a new parameter ShrinkingFactor:

```
def spiral(length, angle, n, ShrinkingFactor):
    if n > 0:
            turtle.forward(length)
            turtle.left(angle)
            spiral(length*ShrinkingFactor, angle, n-1, ShrinkingFactor)
```

Could you try to draw how the screen would look after running each of the following instructions?

a) turtle.reset()
   spiral(100,90,10,0.9)

b) turtle.reset()
   spiral(100,60,20,0.95)

c) turtle.reset()
spiral(200,74,150,0.975)

d) turtle.reset()
spiral(50,10,200,0.975)

3) Given the following functions:

```
def a(length, angle, n):
        turtle.color('red')
        if n > 0:
                pol(length/2.0, 360.0/n, n)
                turtle.forward(length)
                turtle.left(angle)
                a(length, angle, n-1)

def b1(length, angle, NumSides, NumCircles):
        if NumCircles > 0:
                polygon(length, NumSides)
                turtle.left(angle)
                b1(length, angle, NumSides, NumCircles-1)

def b(length, NumSides, NumCircles):
        turtle.color('red')
        b1(length, 360.0/NumCircles, NumSides, NumCircles)
```

Can you match the following function calls to the figures?

a(20,3,36)

b(20,36,36)

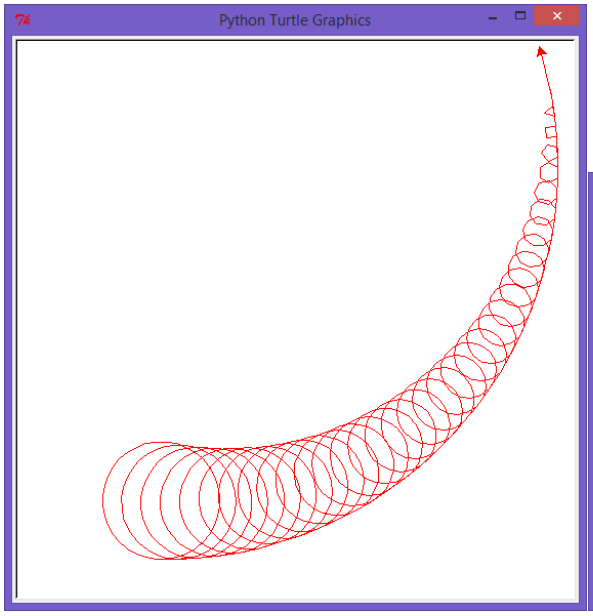


Figure 1

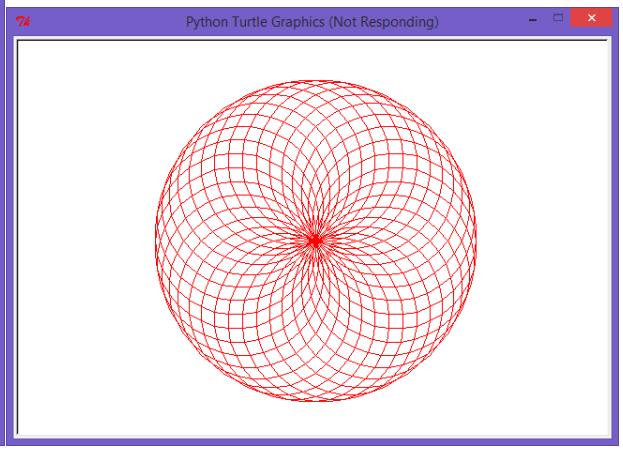
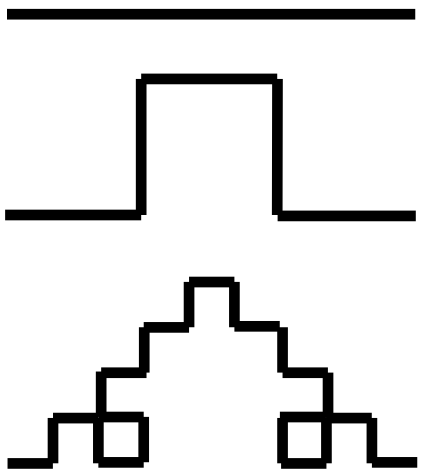


Figure 2

Worksheet 8
Drawing fractals

Koch curve variant reminder

(The first question is on the next page)

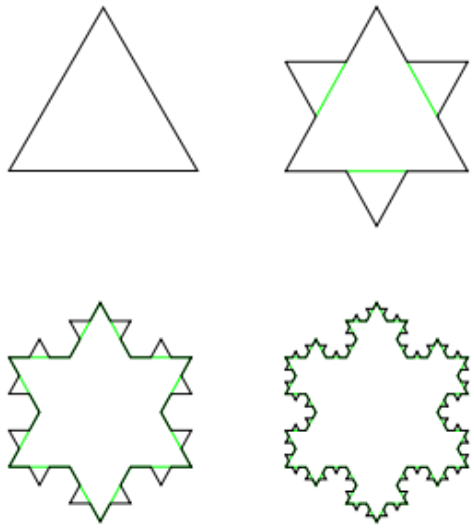


L-system:
start: F
rule: F → F+F−F−F+F
F means "draw forward"
+ means "turn left 90°"
− means "turn right 90°"
Shrinking factor = 1/3

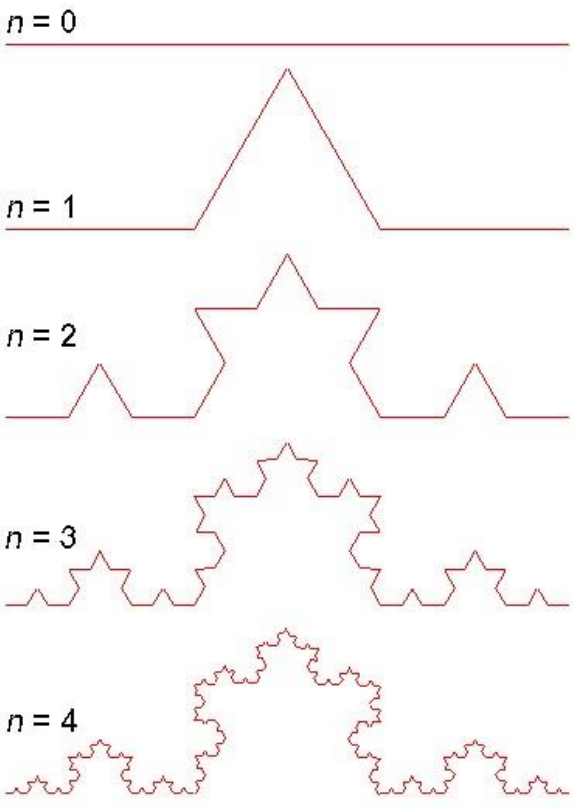
Python program:

```
def koch(length, n):
        if n == 0:
                turtle.forward(length)
        else:
                koch(length/3,n-1)
                turtle.left(90)
                koch(length/3,n-1)
                turtle.right(90)
                koch(length/3,n-1)
                turtle.right(90)
                koch(length/3,n-1)
                turtle.left(90)
                koch(length/3,n-1)
```

1) The first four iterations of the Koch snowflake are:



For just one side of the triangle the first five levels are:



a) Can you define the L-system that generates one side?
b) Can you translate that into a python program?
c) Can you write a python program to draw the full Koch snowflake?

Acknowledgments and references

I would like to thank Professor Edelmira Pasarella from the Polytechnic University of Catalonia for providing me with exercises that she uses in her recursion classes there.

-Recursion jokes: https://recursivelyrecursive.wordpress.com/.

-Mise en abyme: http://en.wikipedia.org/wiki/Mise_en_abyme.

-Liar's paradox: http://en.wikipedia.org/wiki/Liar_paradox.

-Recursion: http://en.wikipedia.org/wiki/Recursion and http://en.wikipedia.org/wiki/Recursion_(computer_science). The latter includes a section on recursion versus iteration.

-*Bachelorette*, included in the DVD *The Work of Director Michel Gondry*, 2003.

-*Rise of the planet of the apes*. Dir. Rupert Wyatt. Twentieth Century Fox, 2011.

-Tower of Hanoi. http://en.wikipedia.org/wiki/Tower_of_Hanoi. There is an online version in http://www.mathsisfun.com/games/towerofhanoi.html.

-History of Logo and Turtle graphics:

http://cyberneticzoo.com/tag/logo-turtle/
http://en.wikipedia.org/wiki/Logo_(programming_language)
http://en.wikipedia.org/wiki/Turtle_(robot)
http://en.wikipedia.org/wiki/Turtle_graphics

-You can download Python and IPython (Interactive Python) from: http://ipython.org/install.html.

Note: Unfortunately when you use the turtle graphics sometimes there are problems resizing or moving the window where the turtle appears. In that case do turtle.done(), resize and/or move the window, go back to the IPython window, press Ctrl-c and continue.

-Spiral photo and several Escher's paintings in Lego: www.andrewlipson.com.

-Fractals: http://en.wikipedia.org/wiki/Self-similarity, http://en.wikipedia.org/wiki/Fractal and http://ecademy.agnesscott.edu/~lriddle/ifs/ifs.htm.

-Mandelbrot set video: http://vimeo.com/6644398.

-Heinz-Otto Peitgen and Dietmar Saupe (Editors), *The Science of Fractal Images*.

-L-systems, including examples of a Koch curve variant, the Sierpinski triangle and the Dragon curve: http://en.wikipedia.org/wiki/L-system.

-Artificial Intelligence: http://en.wikipedia.org/wiki/Artificial_intelligence.

-Programming paradigms: http://en.wikipedia.org/wiki/Programming_paradigm.

-Douglas Hofstadter, *Gödel, Escher, Bach: An Eternal Golden Braid*. Self-reference, recursion, self-reproduction and much more.

-Optical feedback: http://en.wikipedia.org/wiki/Optical_feedback. There are some nice optical feedback videos on youtube.

Answers

Worksheet 1 - Self-reference

1) Play "Guess the number" with a classmate. In this game the first player thinks of a natural number between 1 and 15 (including 1 and 15). The second player has to guess it. In each attempt the first player can only answer "higher", "lower" or "correct". The second player gets 1 point for each attempt. The players take turns to be the first and second. The winner is the one with less total points after 2 rounds.

a) What is the best strategy?

The best strategy is to say 8, then if it is lower say 4 and if it is higher say 12, then, if you haven't guessed say 2, 6, 10 or 14 depending on the case. The idea is to split the set of numbers into two subsets of equal sizes and then do the same again and again until you guess. This strategy is called binary search and is a particular case of the divide and conquer principle. Binary search can easily be written as a recursive program (and as a non-recursive one as well).

With the best strategy what is the maximum number of questions needed to guess a number between

b) 1 and 15?

4. This would be the case if the number is odd.

c) 1 and 16?

5. Following exactly the same strategy we could need 5 if the number is 15 or 16.

d) 1 and 2^n-1?

n. This would be the case if the number is odd. Although if you realize that the other player is doing that the meaning of the "best strategy" could change!

e) 1 and 2^n?

n + 1.

2) Follow these instructions:

Read this sentence and do what it says twice.

If you were to follow the instructions closely you would enter into an infinite loop, that is you would be repeating the same action forever. Sorry.

3) If I say to you: "I am lying", is that true or false?

If it is true then I am lying, therefore it is false.

If it is false then I am lying, therefore it is true.

This is the Liar's paradox, probably the most famous self-referential paradox. Here by paradox we mean a self-contradictory statement.

A version of this paradox is Pinocchio's paradox: Pinocchio says "My nose is growing".

4) Do you see a contradiction between these cartoons?

The first cartoon implies that the height of all the vans is finite, the second one that the time to draw the cartoon was infinite.

Mathematical induction examples

To prove that a statement S(n) is true for all the natural numbers n by mathematical induction you have to prove:

1. The **base case**: S(0) is true or S(1) is true.

2. The **inductive step**: if S(k) is true then S(k+1) is also true.

1) Prove by mathematical induction that:
$2^0 + 2^1 + 2^2 + \ldots + 2^n = 2^{n+1} - 1,\ n \geq 0$

Base case: $2^0 = 1$
$2^{0+1} - 1 = 2 - 1 = 1$
therefore $2^0 = 2^{0+1} - 1$, *that is S(0) is true.*
Inductive step: Let's assume that S(k) is true:
$2^0 + 2^1 + 2^2 + \ldots + 2^k = 2^{k+1} - 1$
If we add 2^{k+1} *to both sides we get:*
$2^0 + 2^1 + 2^2 + \ldots + 2^k + 2^{k+1} = 2^{k+1} - 1 + 2^{k+1}$
$2^0 + 2^1 + 2^2 + \ldots + 2^k + 2^{k+1} = 2^{k+2} - 1$
Then S(k+1) is true. Therefore S(k) $\Rightarrow$ *S(k+1).*

2) Prove by mathematical induction that:
$1 + 2 + \ldots + n = n(n + 1)/2,\ n \geq 1$

Base case: $1 = 1x2/2$
therefore S(1) is true.
Inductive step: Let's assume that S(k) is true:
$1 + 2 + \ldots + k = k(k + 1)/2$
If we add k+1 to both sides we get:
$1 + 2 + \ldots + k + (k+1) = k(k + 1)/2 + (k+1)$
$1 + 2 + \ldots + k + (k+1) = k(k + 1)/2 + 2(k+1)/2$
Then take common factor k+1 on the right side:
$1 + 2 + \ldots + k + (k+1) = (k + 1)(k+2)/2$
Then S(k+1) is true. Therefore S(k) $\Rightarrow$ *S(k+1).*

3) Prove by mathematical induction that:
$1^2 + 2^2 + \ldots + n^2 = n(n + 1)(2n + 1)/6,\ n \geq 1$

Base case: $1^2 = 1x2x3/6$
therefore S(1) is true.

Inductive step: Let's assume that S(k) is true:

$1^2 + 2^2 + \ldots + k^2 = k(k + 1)(2k + 1)/6$

If we add $(k+1)^2$ to both sides we get:

$1^2 + 2^2 + \ldots + k^2 + (k+1)^2 = k(k + 1)(2k + 1)/6 + (k+1)^2$

$1^2 + 2^2 + \ldots + k^2 + (k+1)^2 = (k + 1)(2k^2 + k)/6 + 6(k+1)^2/6$

Then take common factor k+1 on the right side:

$1^2 + 2^2 + \ldots + k^2 + (k+1)^2 = (k + 1)(2k^2 + 7k + 6)/6$

$1^2 + 2^2 + \ldots + k^2 + (k+1)^2 = (k + 1)(k+2)(2k+3)/6$

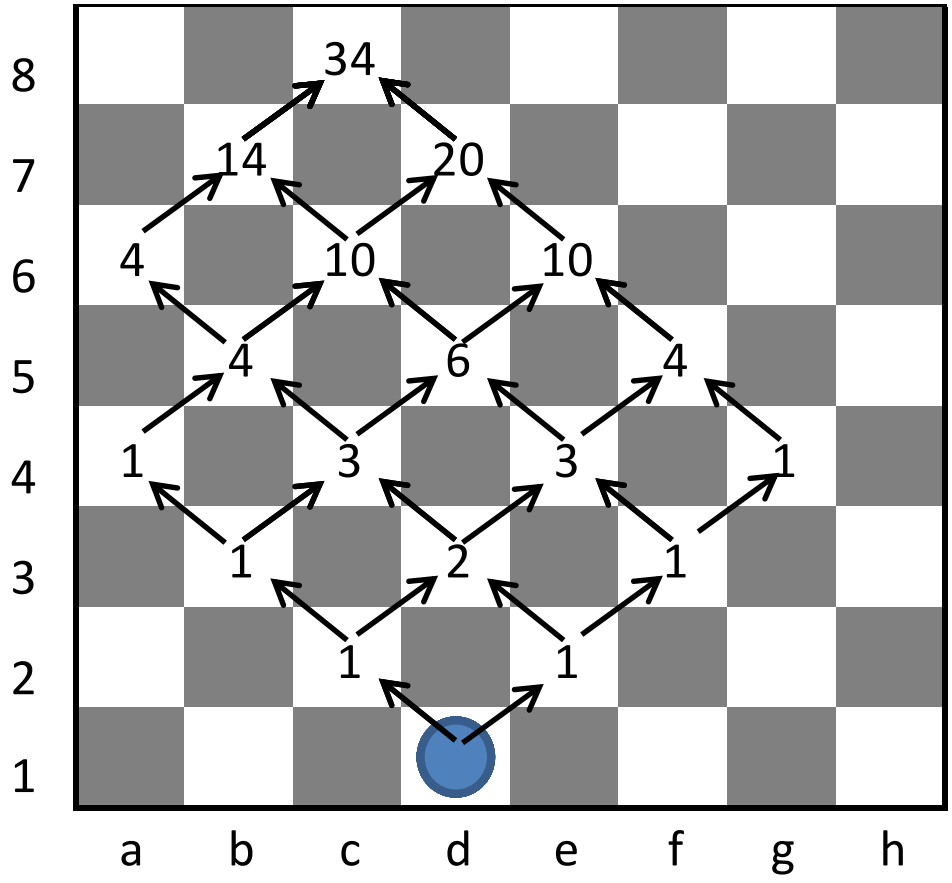
$1^2 + 2^2 + \ldots + k^2 + (k+1)^2 = (k + 1)((k+1)+1)(2(k+1)+1)/6$

Then S(k+1) is true. Therefore S(k) ⇒ S(k+1).

Worksheet 2 - Draughts path counting

If you start with an uncrowned piece in d1, how many paths are that end in c8?

34. As explained in class, applying the divide and conquer principle you get a pattern of numbers very close to Pascal's triangle (it would be the same if the board were infinite).



Worksheet 3 - Recursive play writing

1) Could you write how the blackboard would look after performing the following play? Assume the blackboard is empty before the play.

```
B1(n)
    If n > 0
        Write n in the blackboard
        B1(n-1)
B1(4)
```

Answer: *4*
3
2
1

2) Could you write how the blackboard would look after performing the following play? Assume the blackboard is empty before the play.

```
B2(n)
    If n > 0
        B2(n-1)
        Write n in the blackboard
B2(4)
```

Answer: *1*
2
3
4

3) For the following sub-play:

```
B3(n)
    If n > 0
        Write n in the blackboard
        B3(n-1)
        Write n in the blackboard
```

Could you write how the blackboard would look after performing the following plays? Assume the blackboard is empty before the play.

a) B3(1)	b) B3(2)	c) B3(3)	d) B3(4)
1 1	2 1 1 2	3 2 1 1 2 3	4 3 2 1 1 2 3 4

Worksheet 4 - Algorithms

4) For the following algorithm:

```
P1(n)
    If n = 1
        Write * in a new line in the blackboard
    else
        P1(n-1)
        Write * n times in a new line in the blackboard
        P1(n-1)
```

Could you write how the blackboard would look after running the following statements? Assume the blackboard is empty at the beginning.

a) P1(1)	*b) P1(2)*	*c) P1(3)*	*d) P1(4)*
*	* ** *	* ** * *** * ** *	* ** * *** * ** * **** * ** * *** * ** *

5) Could you write an algorithm P2 that will give the following outputs:

P2(1)	P2(2)	P2(3)	P2(4)
*	** * *	*** ** * * ** * *	**** *** ** * * ** * * *** ** * * ** * *

P2(n)

If n = 1

*Write * in a new line in the blackboard*

else

*Write * n times in a new line in the blackboard*

P2(n-1)

P2(n-1)

6) Could you write an algorithm P3 that will give the following outputs:

P3(1)	P3(2)	P3(3)	P3(4)
*	* * **	* * ** * * ** ***	* * ** * * ** *** * * ** * * ** *** ****

P3(n)

If n = 1

*Write * in a new line in the blackboard*

else

P3(n-1)

P3(n-1)

*Write * n times in a new line in the blackboard*

7) Is there something wrong with this cartoon?

The Russian doll typically has a Russian doll inside but not a doctor.

Worksheet 5 - Tower of Hanoi

1) Try to solve the puzzle for 4 disks with the paper version that we are providing you.

Move disk from stack 1 to stack 2.
Move disk from stack 1 to stack 3.
Move disk from stack 2 to stack 3.

Move disk from stack 1 to stack 2.
Move disk from stack 3 to stack 1.
Move disk from stack 3 to stack 2.

Move disk from stack 1 to stack 2.
Move disk from stack 1 to stack 3.
Move disk from stack 2 to stack 3.

Move disk from stack 2 to stack 1.
Move disk from stack 3 to stack 1.
Move disk from stack 2 to stack 3.

Move disk from stack 1 to stack 2.
Move disk from stack 1 to stack 3.
Move disk from stack 2 to stack 3.

2) Write a recursive algorithm to solve a Tower of Hanoi of size n.

```
Hanoi(n, origin, destination)
    If n = 1
        Move disk from origin to destination
    else
        Hanoi(n-1, origin, 6 – origin – destination)
        Move disk from origin to destination
        Hanoi(n-1, 6 – origin – destination, destination)
```

3) Conjecture what is the minimum number of moves.

2^n-1

Worksheet 6 - Turtle programming

You can download Python and IPython (Interactive Python) from:
http://ipython.org/install.html

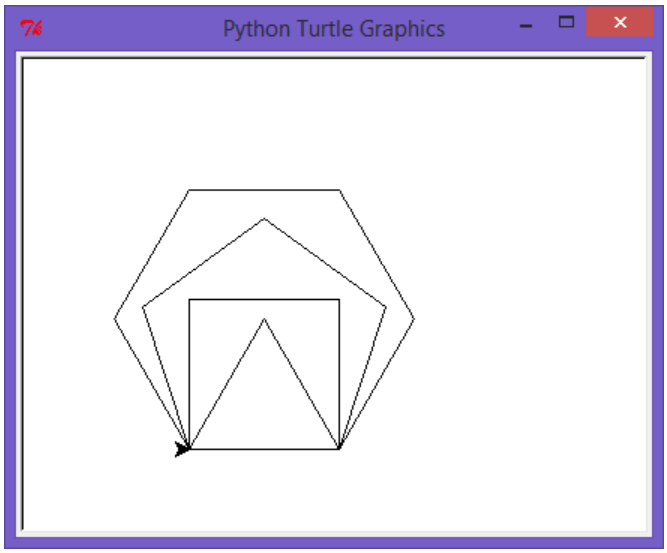
Note: Unfortunately when you use the turtle graphics sometimes there are problems resizing or moving the window where the turtle appears. In that case do turtle.done(), resize and/or move the window, go back to the IPython window, press Ctrl-c and continue.

1) For the following program:

```
def pol(length, angle, n):
    if n > 0:
            turtle.forward(length)
            turtle.left(angle)
            pol(length, angle, n-1)
```

Could you draw how the screen would look after running the following instructions? Assume the screen is empty at the beginning.

```
turtle.reset()
pol(100, 120,3)
pol(100, 90,4)
pol(100, 72,5)
pol(100, 60,6)
```



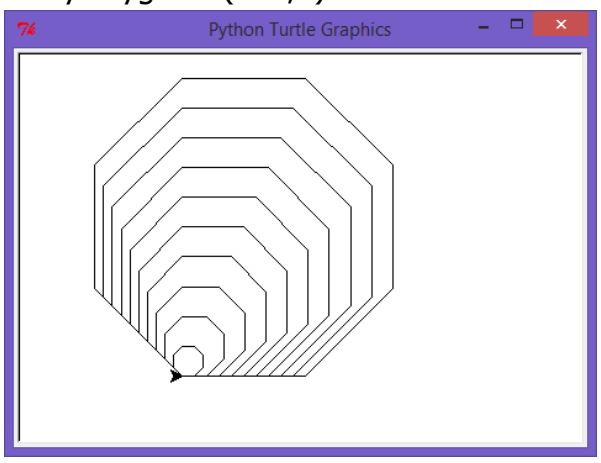
2) What figure would you get if you increase a lot the number of sides?
A circle.

3) Additionally now we define:

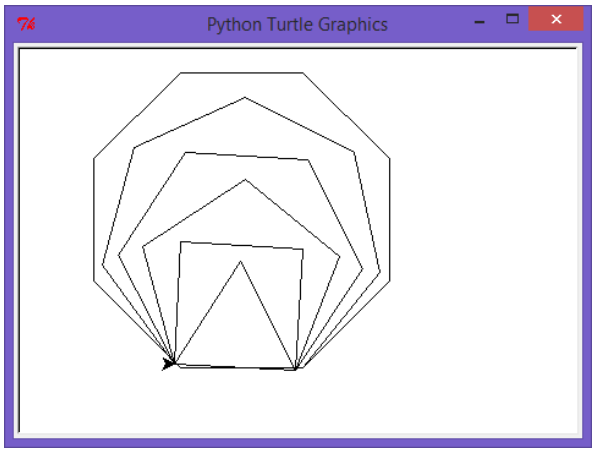
```
def polygon(length, n):
    pol(length, 360/n, n)


def ManyPolygons1(length, n):
    if length>0:
            polygon(length, n)
            ManyPolygons1(length-10, n)

def ManyPolygons2(length, n):
    if n > 2:
            polygon(length, n)
            ManyPolygons2(length, n-1)
```

Could you draw how the screen would look after running the following instructions?

a) ManyPolygons1(100,8)



b) ManyPolygons2(100,8)



4) When we run:
polygon(7)
the turtle does not go back to its initial position as it does when we run
polygon(6)

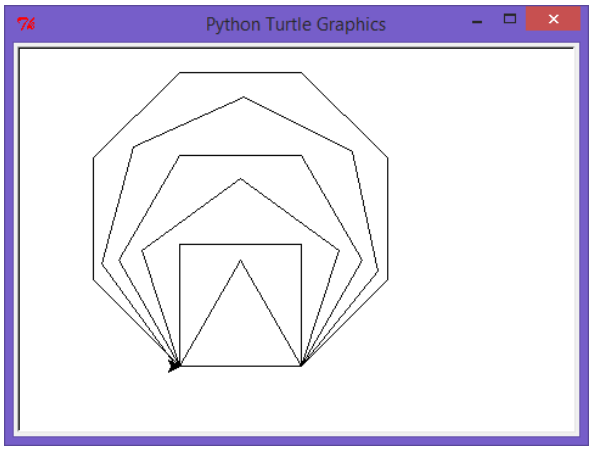
How would you modify the definition of pol:

```
def pol(length, angle, n):
    if n > 0:
            turtle.forward(length)
            turtle.left(angle)
            pol(length, angle, n-1)
```

to make sure that the turtle goes back to its initial position after drawing the polygon?

```
def polBack(length, angle, n):
    if n > 0:
            turtle.forward(length)
            turtle.left(angle)
            polBack(length, angle, n-1)
            turtle.right(angle)
            turtle.backward(length)

def polygonBack(length, n):
    polBack(length, 360/n, n)

def ManyPolygonsBack2(length, n):
    if n > 2:
            polygonBack(length, n)
            ManyPolygonsBack2(length, n-1)
```

Example: ManyPolygonsBack2(100,8)



Note: the problem is due to the fact that python interprets 360/n as an integer division. To do a real division is enough to specify 360.0/n:

```
def polygon(length, n):
    pol(length, 360.0/n, n)
```

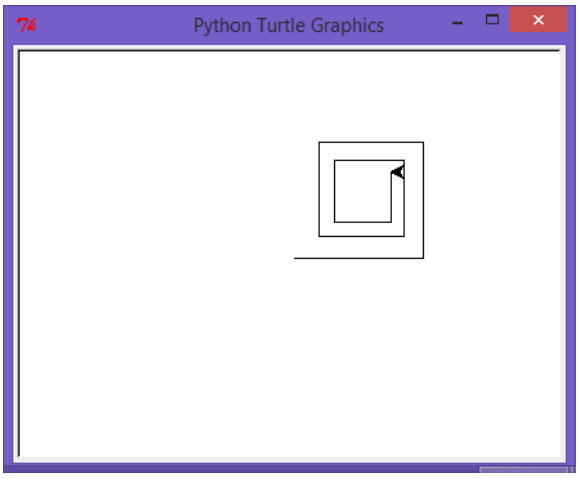
You don't have to finish these questions.

5) Now to draw spirals we define a function similar to the one to draw polygons but with a new parameter ShrinkingFactor:

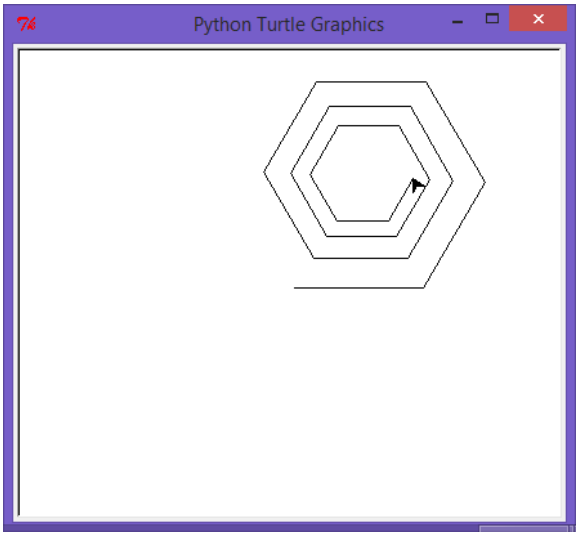
```
def spiral(length, angle, n, ShrinkingFactor):
    if n > 0:
            turtle.forward(length)
            turtle.left(angle)
            spiral(length*ShrinkingFactor, angle, n-1, ShrinkingFactor)
```

Could you try to draw how the screen would look after running each of the following instructions?

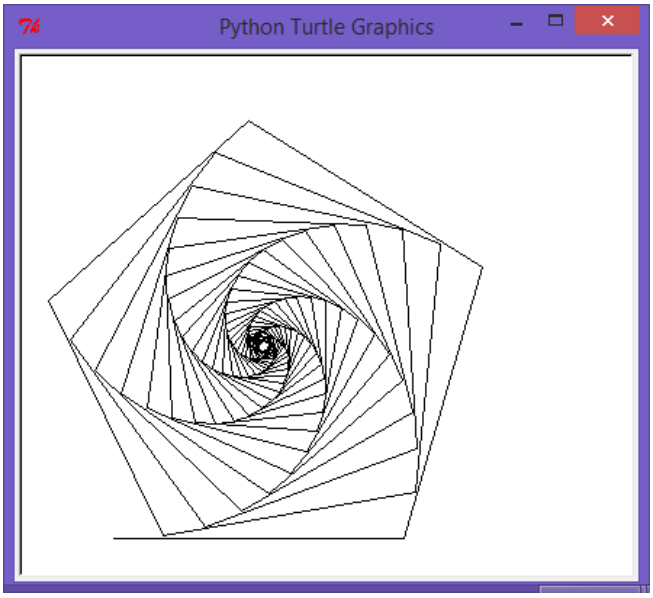
a) turtle.reset()
   spiral(100,90,10,0.9)



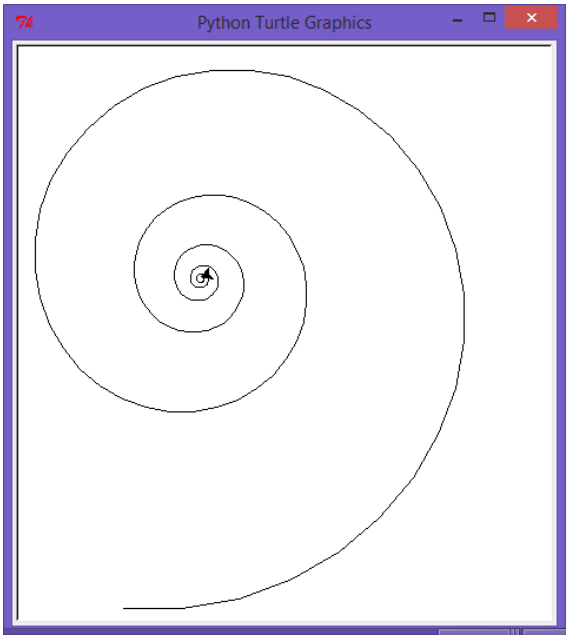
b) turtle.reset()
spiral(100,60,20,0.95)



c) turtle.reset()
spiral(200,74,150,0.975)



d) turtle.reset()
spiral(50,10,200,0.975)



Another function to draw spirals is:

```
def spiral2(length, angle, n, increment):
    if n > 0:
        turtle.forward(length)
        turtle.right(angle)
        spiral2(length + increment, angle, n-1, increment)
```

You could try for example: spiral2(1.25,71.5,360,0.75)

6) Given the following functions:

```
def a(length, angle, n):
        turtle.color('red')
        if n > 0:
                pol(length/2.0, 360.0/n, n)
                turtle.forward(length)
                turtle.left(angle)
                a(length, angle, n-1)

def b1(length, angle, NumSides, NumCircles):
        if NumCircles > 0:
                polygon1(length, NumSides)
                turtle.left(angle)
                b1(length, angle, NumSides, NumCircles-1)

def b(length, NumSides, NumCircles):
        turtle.color('red')
        b1(length, 360.0/NumCircles, NumSides, NumCircles)
```

Can you match the following function calls to the figures?
a(20,3,36)
b(20,36,36)

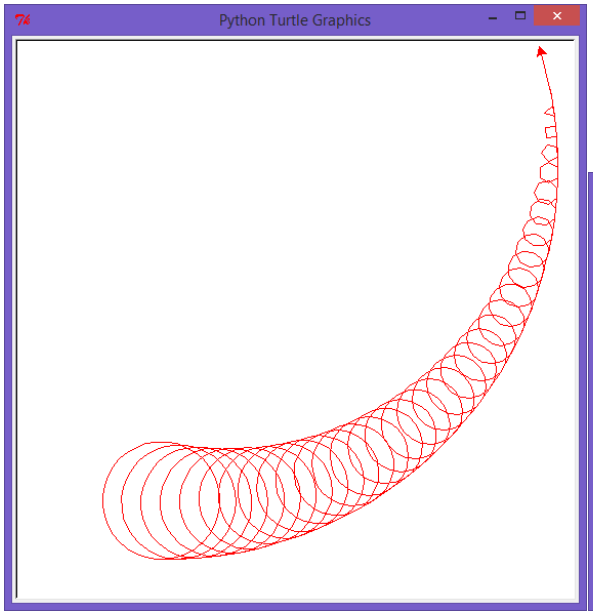


Figure 1

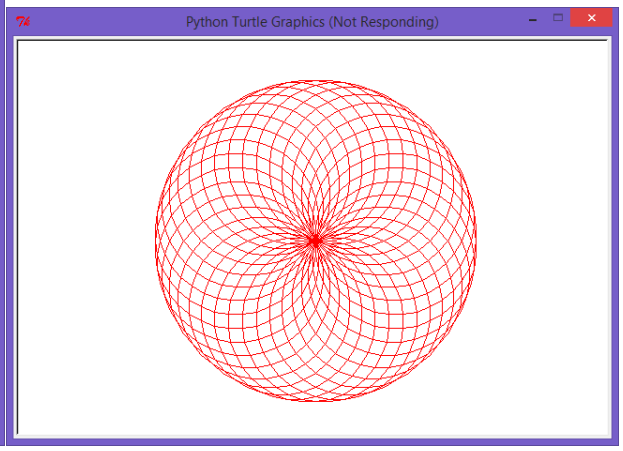
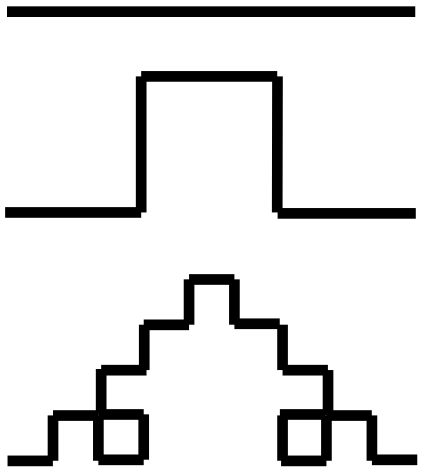


Figure 2

a(20,3,36) draws Figure 1 and b(20,36,36) draws Figure 2.

Worksheet 7 - Drawing fractals

Koch curve variant

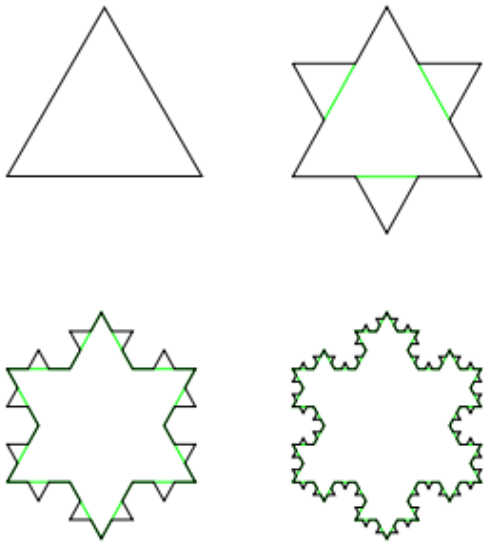


L-system (taken from http://en.wikipedia.org/wiki/L-system):
start: F
rule: F → F+F−F−F+F
F means "draw forward"
+ means "turn left 90°"
− means "turn right 90°"
Shrinking factor = 1/3

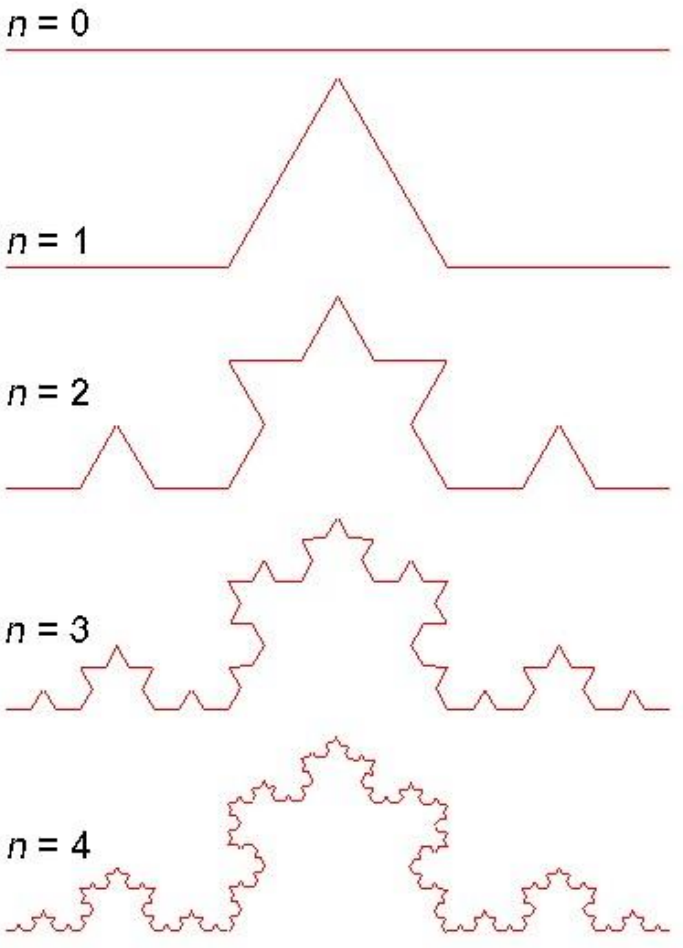
Python program:

```
def koch(length, n):
        if n == 0:
                turtle.forward(length)
        else:
                koch(length/3,n-1)
                turtle.left(90)
                koch(length/3,n-1)
                turtle.right(90)
                koch(length/3,n-1)
                turtle.right(90)
                koch(length/3,n-1)
                turtle.left(90)
                koch(length/3,n-1)
```

1) The first four iterations of the Koch snowflake are:



For just one side of the triangle the first five levels are:



a) Can you define the L-system that generates one side?
   L-system:
   start: F
   rule: F → F+F−F+F
   F means "draw forward"
   + means "turn left 60°"
   − means "turn right 120°"
   Shrinking factor = 1/3

b) Can you translate that into a python program?

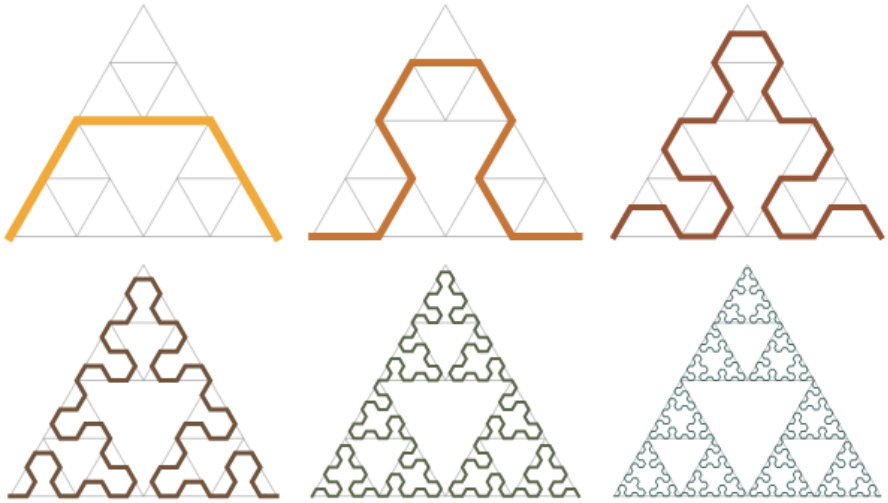
```
def KochSnowflake1(length, n):
        if n == 0:
                turtle.forward(length)
        else:
                KochSnowflake1(length/3,n-1)
                turtle.left(60)
                KochSnowflake1(length/3,n-1)
                turtle.right(120)
                KochSnowflake1(length/3,n-1)
                turtle.left(60)
                KochSnowflake1(length/3,n-1)
```

c) Can you write a python program to draw the full Koch snowflake?

```
def KochSnowflake(length, n):
        KochSnowflake1(length,n)
        turtle.right(120)
        KochSnowflake1(length,n)
        turtle.right(120)
        KochSnowflake1(length,n)
        turtle.right(120)
```

Other fractal and L-system examples

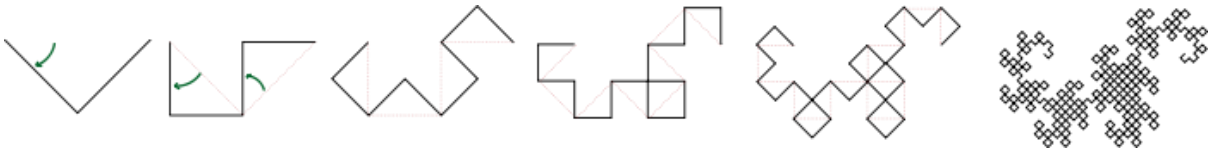
Sierpinski triangle

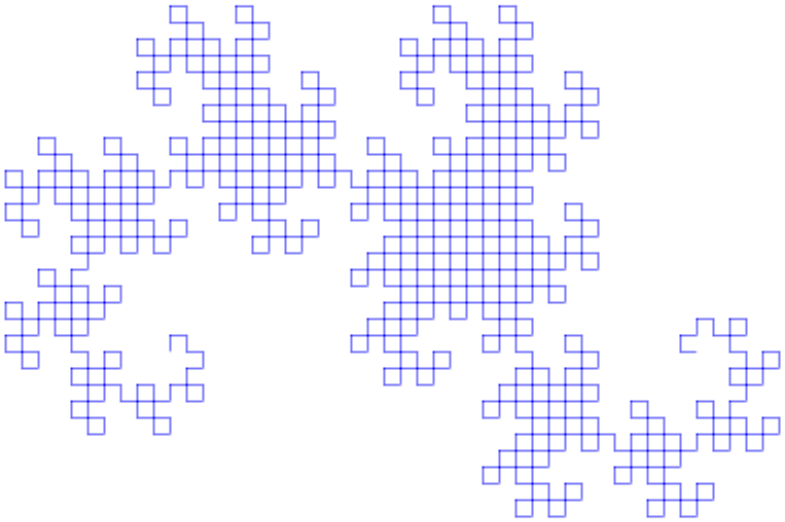


```
# http://en.wikipedia.org/wiki/L-system#Example_5:_Sierpinski_triangle
# The Sierpinski triangle drawn using an L-system.
# variables : A B
# constants : + −
# start  : A
# rules  : (A → B−A−B), (B → A+B+A)
# angle  : 60°
# Here, A and B both mean "draw forward",
# + means "turn left by angle", and − means "turn right by angle"
# Shrinking factor = 1/2

import turtle
def sierpinski(variable, length, n):
        if n == 0:
                turtle.forward(length)
        else:
                if variable == 'A':
                        sierpinski('B',length/2, n-1)
                        turtle.right(60)
                        sierpinski('A',length/2, n-1)
                        turtle.right(60)
                        sierpinski('B',length/2, n-1)
                else:
                        sierpinski('A',length/2, n-1)
                        turtle.left(60)
                        sierpinski('B',length/2, n-1)
                        turtle.left(60)
                        sierpinski('A',length/2, n-1)
```

Dragon curve

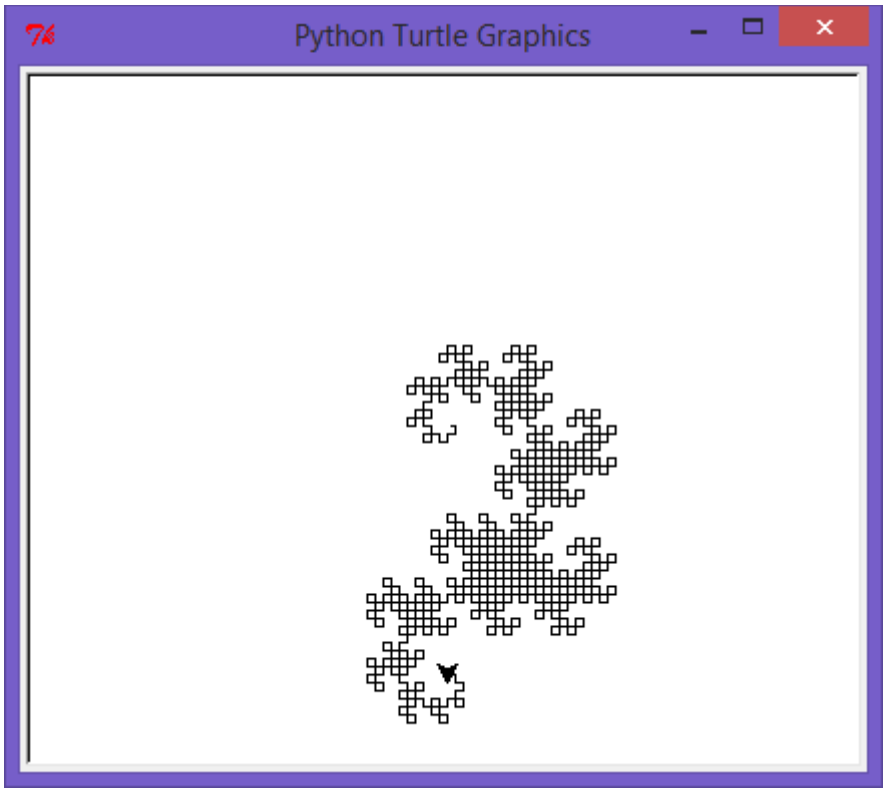
Starting from a base segment, replace each segment by 2 segments with a right angle and with a rotation of 45° alternatively to the right and to the left:





```
# http://en.wikipedia.org/wiki/L-system#Example_6:_Dragon_curve
# start  : X
# rules  : (X → X+YF), (Y → FX-Y)
# F means "draw forward", - means "turn left 90°", + means "turn right 90°".
# X and Y do not correspond to any drawing action.

import turtle
def dragon(variable, length, n):
        if n == 0:
                turtle.forward(length)
        else:
                if variable == 'X':
                        dragon('X',length,n-1)
                        turtle.right(90)
                        dragon('Y',length,n-1)
                        turtle.forward(length)
                else:
                        turtle.forward(length)
                        dragon('X',length,n-1)
                        turtle.left(90)
                        dragon('Y',length,n-1)
```

The output of dragon('X',2,10) is:



The output of dragon('X',1,12) is: