[함수의 고급]

- 터틀 그래픽스
- 재귀호출 함수
- map 함수
- eval 함수

재귀호출(되부름) 함수

- 재귀호출함수(recursion, recursive funtion)는 되부름 함수 또는 순환호출함수 라고도 함
- 함수가 자기 자신을 부르게 작성된 함수
- 재귀호출함수를 작성할 때는 함수가 종료되는 조건을 반드시 포함하도록 해야 함
- 그렇지 않으면 계속 자기 자신을 불러 시스템 자원을 소진시켜 비정상적으로 끝나게 됨
- 재귀호출을 사용한다고 해서 기억장소(storage)가 절약되거나 속도가 빨라지지 않음
- 하지만 프로그램이 간결해지고 이해하기가 쉬운 장점이 있음

하노이 탑 프로그램(전형적인 예)

```
# hanoi.py: Hanoi tower
# a에 있는 n개의 원판을 b를 이용해 c로 옮김
def move(n, a, b, c):
    if n > 0:
        move(n-1, a, c, b) # a에 있는 n-1개의 원판을 c를 이용해 b로 옮김
        print("Move a disk from %s to %s" %(a,c))
        move(n-1, b, a, c) # b에 있는 n-1개의 원판을 a를 이용해 c로 옮김

move(3, 'a', 'b', 'c') # a에 있는 3개의 원판을 b를 이용해 c로 옮김
```

factorial을 재귀호출로 작성해 보자 :

- n! = n * (n-1)!인 특성을 이용해 n!을 계산하는 재귀호출 함수 factorial(n)을 작성
- 단 0! = 1임

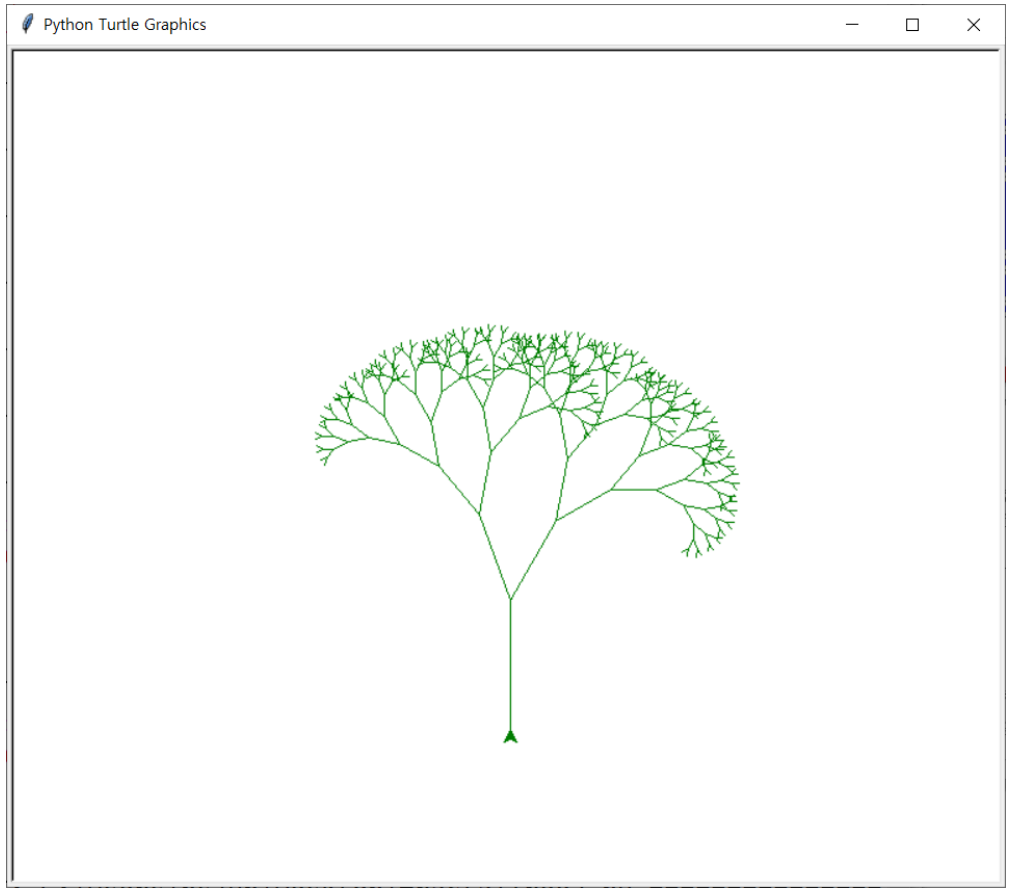
```
def factorial(n):
    if n == 0:
        return 1 # 재귀 함수 종료 조건. n=0이면 종료
    else:
        return n * factorial(n-1) # n! = n x (n-1)!

print(factorial(5))
```

x ** n 계산하는 재귀함수 작성

- x ** n = x * (x ** (n-1))인 특성을 이용해 x ** n을 계산하는 재귀호출 함수 pow(x, n)을 작성
- 단 0 ** 0은 계산하지 않음

프랙털 트리

가지의 끝에서 가지가 둘로 갈라져 자라는 나무를 모델링 해보자.



알고리즘

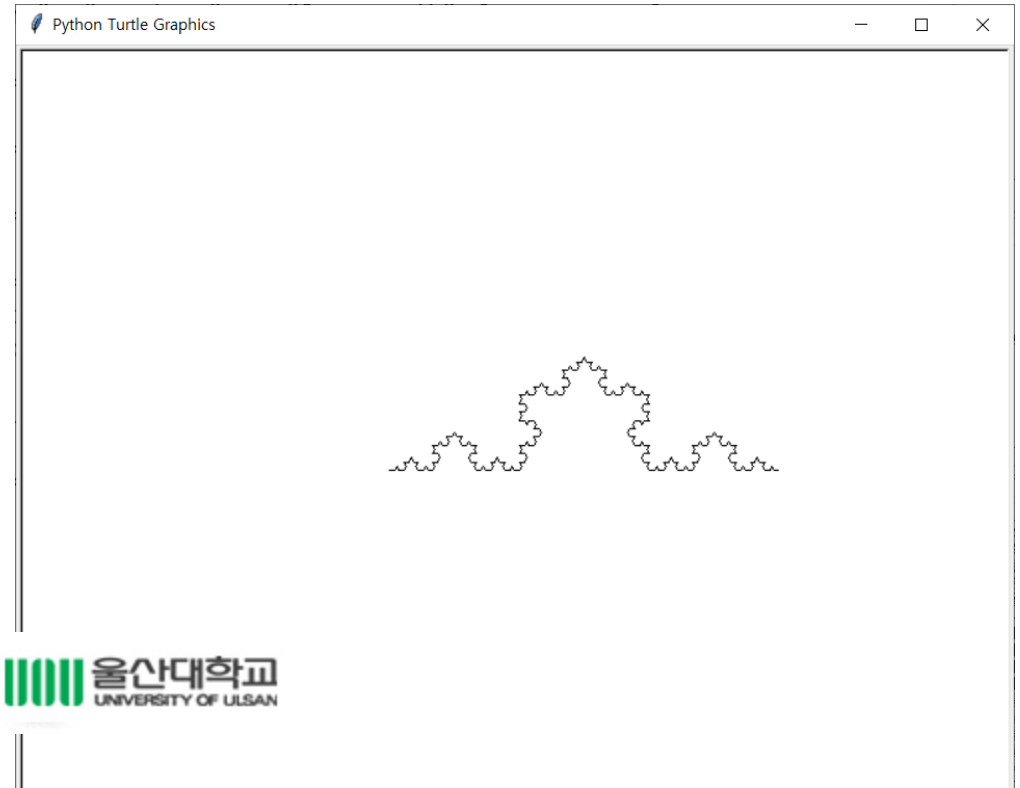
- 임의의 수직선 하나를 그린다. 이것이 줄기 또는 "0 세대" 가지
- 가지의 끝에서 임의의 비율로 축소된 두 가지(그 다음 세대 가지)를 그린다. 가지가 벌어지는 각도도 임의로 정한다.
- 정해진 조건까지 이 과정을 반복한다(재귀호출 횟수 결정됨)
    i. 가지의 길이가 5 이상이면 그린다
    ii. 재귀호출 횟수를 실행시 정해준다.

소스코드

```
import turtle

# A recursive function to draw a fractal tree
def fractal_tree(branch_length, t):
    if branch_length > 5:
        # Draw the main branch
        t.forward(branch_length)
        t.right(20)

        # Recursively call for the right branch
        fractal_tree(branch_length - 15, t)

        # Come back and draw the left branch
        t.left(40)
        fractal_tree(branch_length - 15, t)

        # Return to the original position
        t.right(20)
        t.backward(branch_length)

# Set up the turtle canvas
screen = turtle.Screen()
screen.title("Fractal Tree")
screen.setup(width=800, height=600)
t = turtle.Turtle()
t.speed('fastest')
t.penup()
t.color('green')
t.goto(0, -200)  # Start at the bottom of the screen
t.left(90)
t.pendown()

# Draw the tree
fractal_tree(100, t)

w open until closed
)
```

Koch fractal curve 그리기 _

아래 표시한 것을 Koch curve라고 한다. Koch curve를 그리려면 한 변의 길이가 일정한(예:300)인 선분에서 출발한다. 선분을 3등분하여 가운데의 선분을 한 변으로 하는 정삼각형을 그리지만 밑변은 그리지 않는다. 이 과정을 계속 반복하면 아래 그림을 얻을 수 있다. 이런 커브를 자기 상사성(닮음)이 있는 프랙털(fractal)이라 한다.



source

```
import turtle

def koch(leng, n):
    if n == 0:
        t.forward(leng)
    else:
        koch(leng/3,n-1)
        t.left(60)
        koch(leng/3,n-1)
        t.right(120)
        koch(leng/3,n-1)
        t.left(60)
        koch(leng/3,n-1)

# n = int(input("recursion depth:"))
# leng = int(input("side length:"))
n = turtle.numinput("recursion depth", "Your input:", 3, minval=0, maxval=5)
leng = turtle.numinput("side length", "Your input:", 300, minval=100, maxval=400)

t=turtle.Turtle()
t.speed(0)
t.ht()

t.up()
t.goto(-100, 0)
t.pd()
)
oop()
```

map 함수

- map(func,seq)은 seq의 각 요소에 대해 func를 호출하여 실행하고 결과를 iterable로 반환

```
>>> a = map(str.upper, 'abc')
>>> list(a)
['A', 'B', 'C']
```

eval 함수

- eval(str)은 실행이 가능한 문자열 str을 입력으로 받아 그 문자열을 실행한 결과값을 반환

```
>>> eval('1+2*3')    # 수식의 계산
7
>>> x, n = eval(input('x의 n 제곱은?:')); print(x**n)
x의 n 제곱은?:2, 5    # x, n을 튜플로 입력받아
32                    # x**n 출력
```

- 편해 보이지만 문자열 표현을 그대로 실행하기 때문에 매우 조심해 사용해야 함
- 사용자가 악의를 가지고 사용할 경우 시스템에 해로울 수 있음
- 다음을 셀에서 실행해 보라.

```
eval("__import__('os').system('dir /p')")
```

울산대학교 UNIVERSITY OF ULSAN 신 시스템의 파일을 모두 지우는 명령어를 사용하면 파일이 모두 지워지게 된다.