

10

CHAPTER

재귀 호출

학습목표

- 재귀 호출의 개념과 작동을 이해한다.
- 재귀 호출을 위한 코드 형식을 이해한다.
- 재귀 호출을 다양한 응용 예로 연습한다.

SECTION 00 생활 속 자료구조와 알고리즘

SECTION 01 재귀 호출의 기본

SECTION 02 재귀 호출 작동 방식의 이해

SECTION 03 재귀 호출의 연습

SECTION 04 재귀 호출의 응용

연습문제

응용예제



■ 재귀 알고리즘이란?

- 양쪽에 거울이 있을 때 거울에 비친 자신이 무한 반복해서 비치는 것 또는 마트료시카 인형처럼 동일한 작동을 무한적으로 반복하는 알고리즘을 말함



Section 01 재귀 호출의 기본

■ 재귀 호출의 개념

- 재귀 호출(Recursion)은 자신을 다시 호출하는 것
 - 마지막 종이 상자가 나올 때까지 여러 개 겹쳐진 상자를 꺼내 마지막 상자에 반지 놓는 예

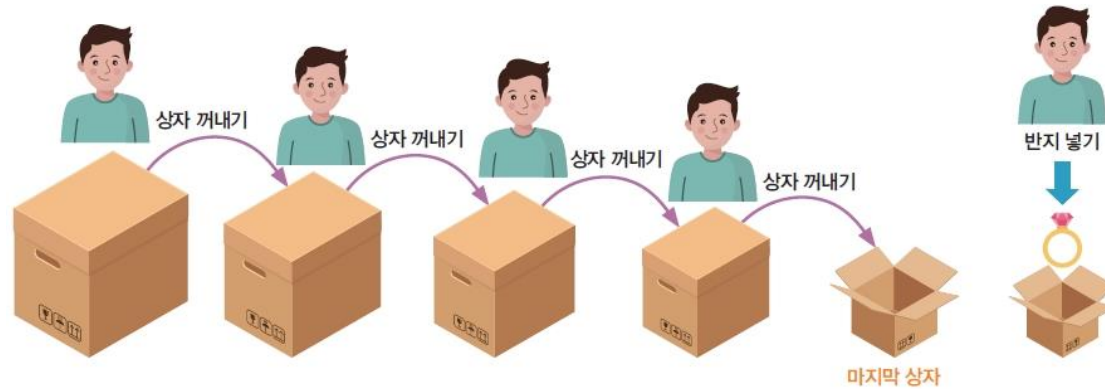


그림 10-1 마지막 상자가 나올 때까지 계속 반복해서 상자를 꺼내 마지막 상자에 반지 놓기

- 꺼낸 순서와 반대로 종이 상자를 다시 넣는 예

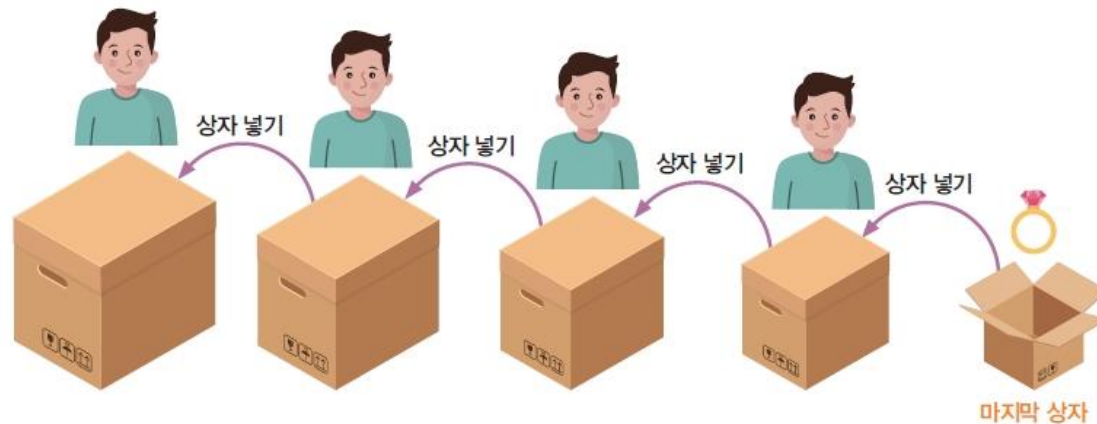


그림 10-2 마지막 상자부터 반복해서 상자 넣기

Section 01 재귀 호출의 기본

■ 재귀 호출의 작동

- 상자를 반복해서 여는 과정을 재귀 호출 형태로 표현



그림 10-3 재귀 함수 형태

Code10-01.py 재귀 호출 함수 기본

```
1 def openBox() :  
2     print("종이 상자를 엽니다. ^^")  
3     openBox()  
4  
5 openBox()    # 처음 함수를 다시 호출
```

Section 01 재귀 호출의 기본

실행 결과

종이 상자를 엽니다. ^^

종이 상자를 엽니다. ^^

...(중략)...

Traceback (most recent call last):

File "C:\CookData\Code10-01.py", line 5, in <module>

openBox() # 처음 함수를 다시 호출

File "C:\CookData\Code10-01.py", line 3, in openBox

openBox()

...(중략)...

[Previous line repeated 1009 more times]

File "C:\CookData\Code10-01.py", line 2, in openBox

print("종이 상자를 엽니다. ^^")

RecursionError: maximum recursion depth exceeded while pickling an object

>>>

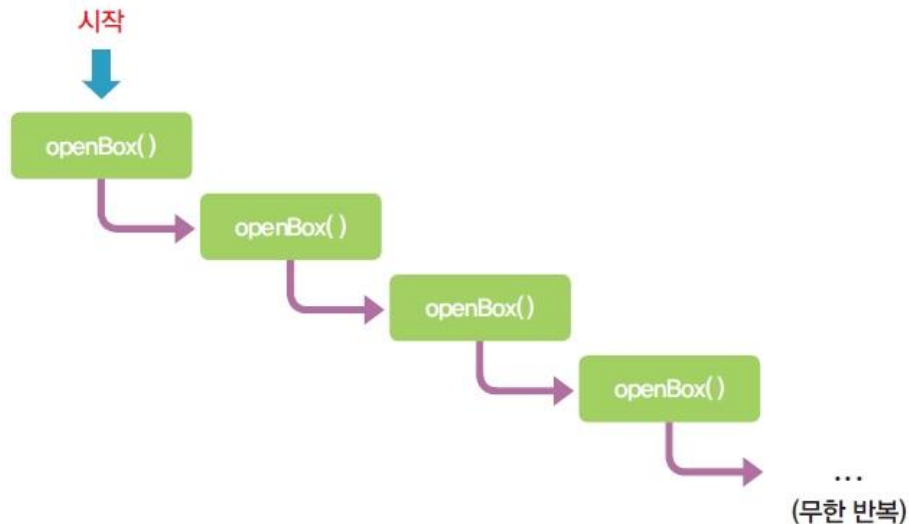


그림 10-4 무한 반복하는 재귀 호출

Section 01 재귀 호출의 기본

- 일반적인 프로그램에서는 무한 반복을 마치고 되돌아가는 조건을 함께 사용함
 - 10회 반복 후 호출한 곳으로 다시 돌아가는 조건을 사용하는 예

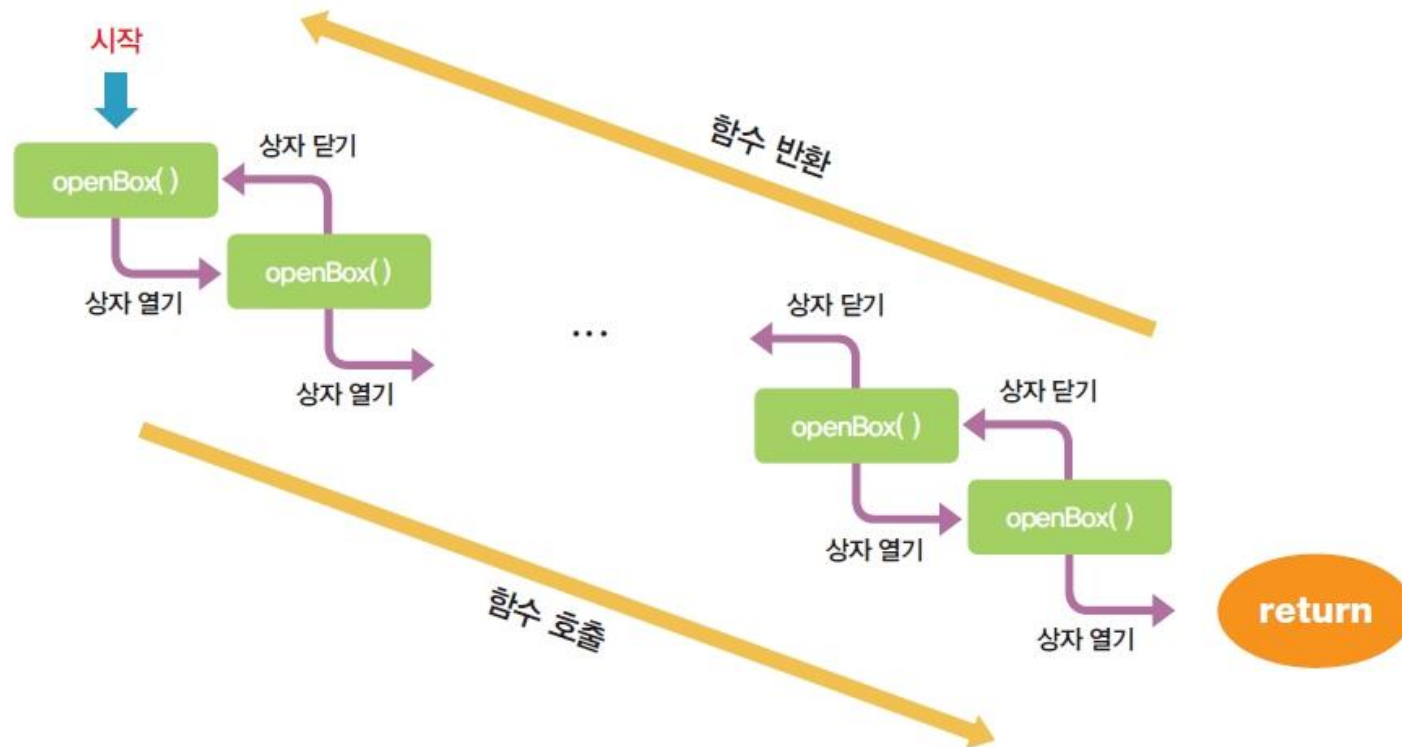


그림 10-5 반환 조건이 추가된 재귀 호출

Section 01 재귀 호출의 기본

Code10-02.py 재귀 호출 함수 기본(반환 조건 추가)

```
1 def openBox() :
2     global count
3     print("종이 상자를 엽니다. ^^")
4     count -= 1
5     if count == 0 :
6         print("** 반지를 넣고 반환합니다. **")
7         return
8     openBox()
9     print("종이 상자를 닫습니다. ^^")
10
11 count = 10
12 openBox()    # 처음 함수를 다시 호출
```

실행 결과

```
종이 상자를 엽니다. ^^
종이 상자를 엽니다. ^^
...(중략)...
종이 상자를 엽니다. ^^
** 반지를 넣고 반환합니다. **
종이 상자를 닫습니다. ^^
...(중략)...
종이 상자를 닫습니다. ^^
종이 상자를 닫습니다. ^^
```

Section 01 재귀 호출의 기본

- 7행의 return 문을 만나면 어디로 돌아가는가(함수의 재귀 호출 방식 이해)

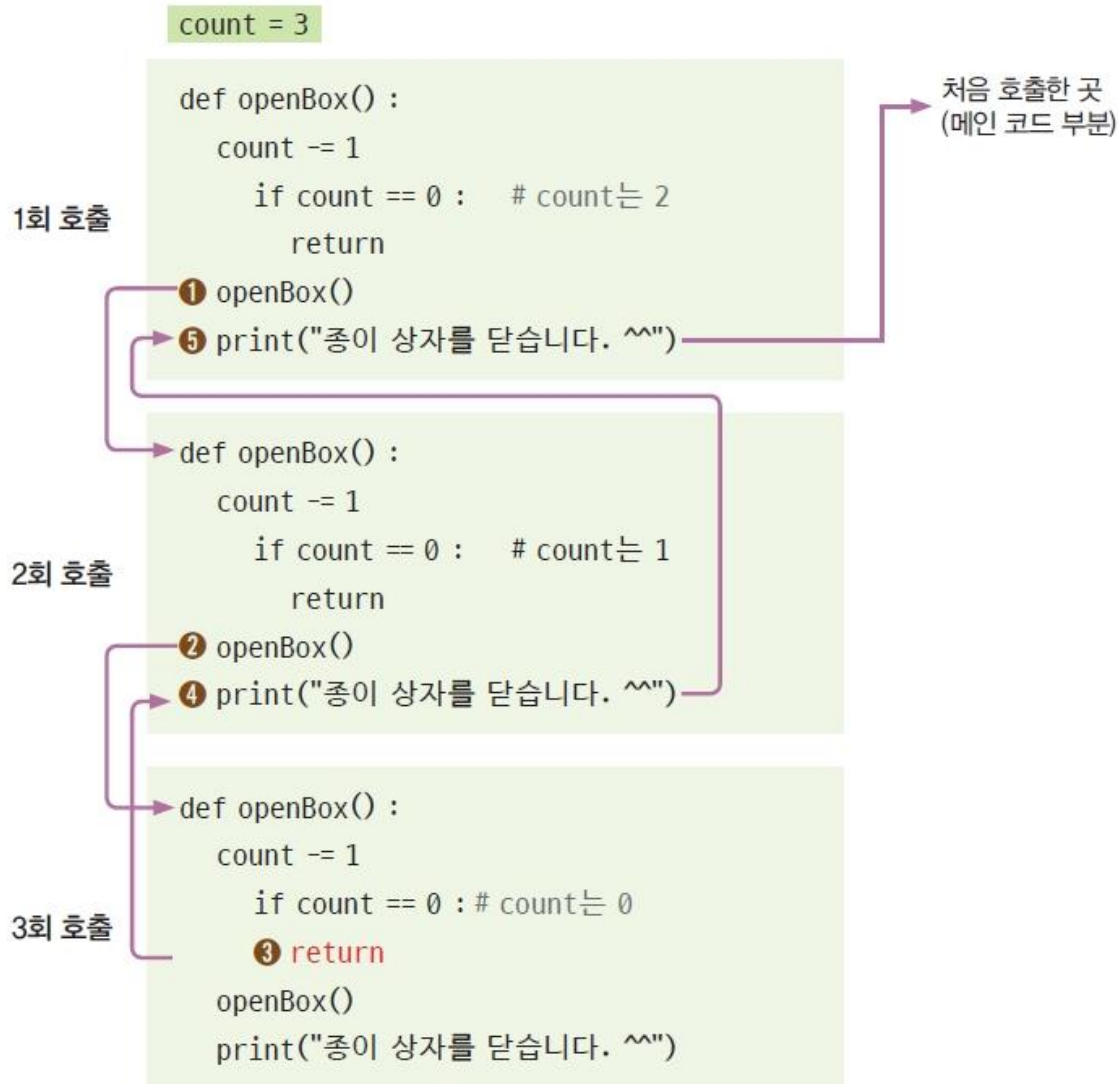


그림 10-6 함수의 재귀 호출 방식 이해

Section 02 재귀 호출 작동 방식의 이해

■ 숫자 합계 내기(1부터 10까지 합계를 구하는 예)

- 반복문을 이용한 구현

```
sumValue = 0
for n in range(10, 0, -1) :
    sumValue += n
print("10+9+...+1 = ", sumValue)
```

실행 결과

10+9+...+1 = 55

- 재귀 함수를 이용한 구현

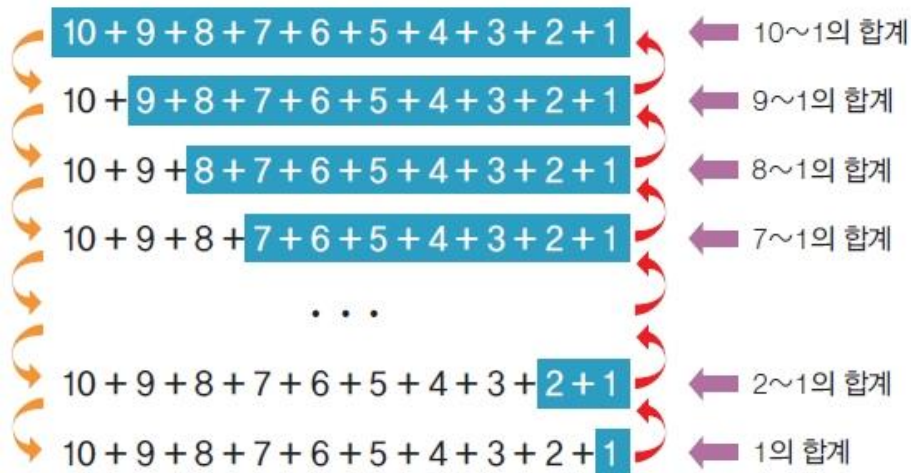


그림 10-7 10부터 1까지 합계

Section 02 재귀 호출 작동 방식의 이해

Code10-03.py 10부터 1까지의 합계를 재귀 호출로 구현

```
1 def addNumber(num) :  
2     if num <= 1 :  
3         return 1  
4     return num + addNumber(num-1)  
5  
6 print(addNumber(10))
```

실행 결과

55

(1단계) 6행의 addNumber(10) 호출
(2단계) 4행의 10 + addNumber(9) 호출
(3단계) 4행의 9 + addNumber(8) 호출
(4단계) 4행의 8 + addNumber(7) 호출
(5단계) 4행의 7 + addNumber(6) 호출
(6단계) 4행의 6 + addNumber(5) 호출
(7단계) 4행의 5 + addNumber(4) 호출
(8단계) 4행의 4 + addNumber(3) 호출
(9단계) 4행의 3 + addNumber(2) 호출
(10단계) 4행의 2 + addNumber(1) 호출
(11단계) 3행의 1 반환
(10단계) 4행의 2 + 1(=3) 반환
(9단계) 4행의 3 + 3(=6) 반환
(8단계) 4행의 4 + 6(=10) 반환
(7단계) 4행의 5 + 10(=15) 반환
(6단계) 4행의 6 + 15(=21) 반환
(5단계) 4행의 7 + 21(=28) 반환
(4단계) 4행의 8 + 28(=36) 반환
(3단계) 4행의 9 + 36(=45) 반환
(2단계) 4행의 10 + 45(=55) 반환
(1단계) 6행의 55 출력

그림 10-8 10부터 1까지 합계 재귀 호출

↓ 5부터 1까지 합계로 단순화

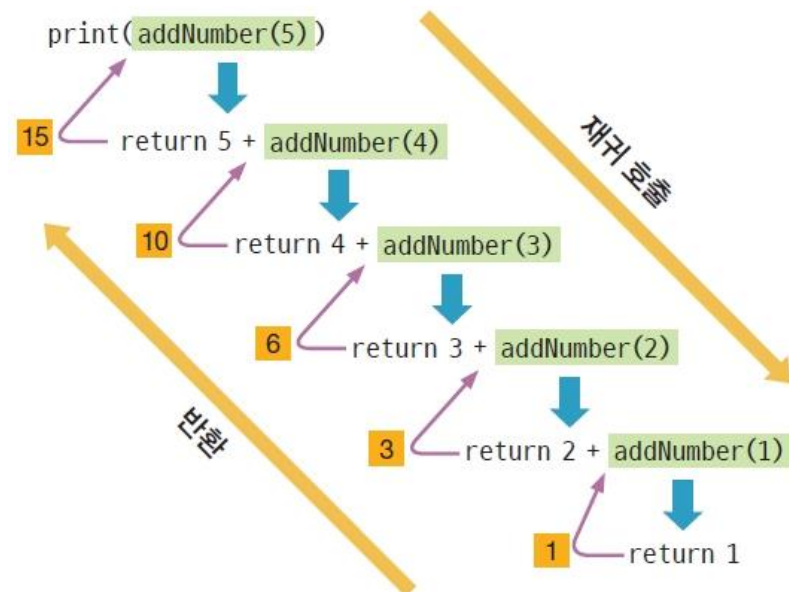


그림 10-9 5부터 1까지 합계 재귀 호출 과정

Section 02 재귀 호출 작동 방식의 이해

SELF STUDY 10-1

Code10-03.py를 수정해서 두 수를 입력받고 두 숫자 사이의 합계를 구하는 코드를 작성하자. 단 입력하는 숫자는 작은 숫자 또는 큰 숫자 중 어느 것을 먼저 입력해도 같다.

실행 결과

```
숫자1-->1  
숫자2-->10  
55
```

```
숫자1-->10  
숫자2-->1  
55
```

Section 02 재귀 호출 작동 방식의 이해

팩토리얼 구하기(10부터 1까지 곱하는 예)

- 반복문을 이용한 구현

```
factValue = 1 # 곱셈이므로 초깃값을 1로 설정
for n in range(10, 0, -1) :
    factValue *= n
print("10*9*...*1 = ", factValue)
```

실행 결과

10*9*...*1 = 3628800

- 재귀 함수를 이용한 구현

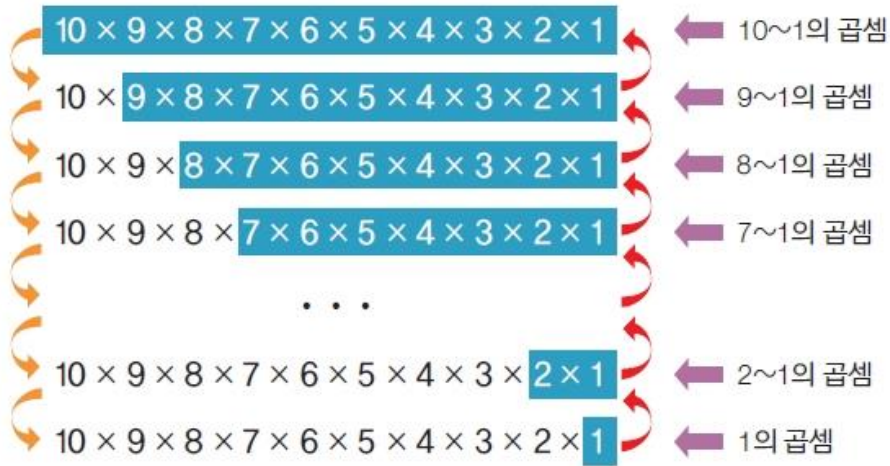


그림 10-10 10! 계산

Section 02 재귀 호출 작동 방식의 이해

- [그림 10-10]의 재귀 호출 과정을 재귀 함수로 작성

```
def factorial(num) :  
    if num <= 1 :  
        return 1  
    return num * factorial(num-1)  
  
print('\n10! = ', factorial(10))
```

Code10-04.py 5!을 재귀 호출로 구현

```
1 def factorial(num) :  
2     if num <= 1 :  
3         print('1 반환')  
4         return 1  
5     print("%d * %d! 호출" % (num, num-1))  
6     retVal = factorial(num-1)  
7  
8     print("%d * %d!(=%d) 반환" % (num, num-1, retVal))  
9     return num * retVal  
10  
11 print('\n5! = ', factorial(5))
```

실행 결과

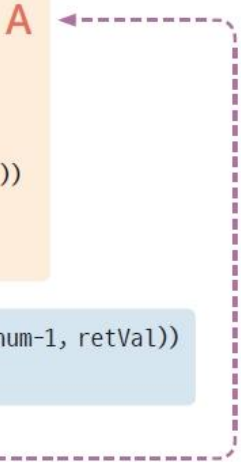
```
5 * 4! 호출  
4 * 3! 호출  
3 * 2! 호출  
2 * 1! 호출  
1 반환  
2 * 1!(=1) 반환  
3 * 2!(=2) 반환  
4 * 3!(=6) 반환  
5 * 4!(=24) 반환  
  
5! = 120
```

Section 02 재귀 호출 작동 방식의 이해

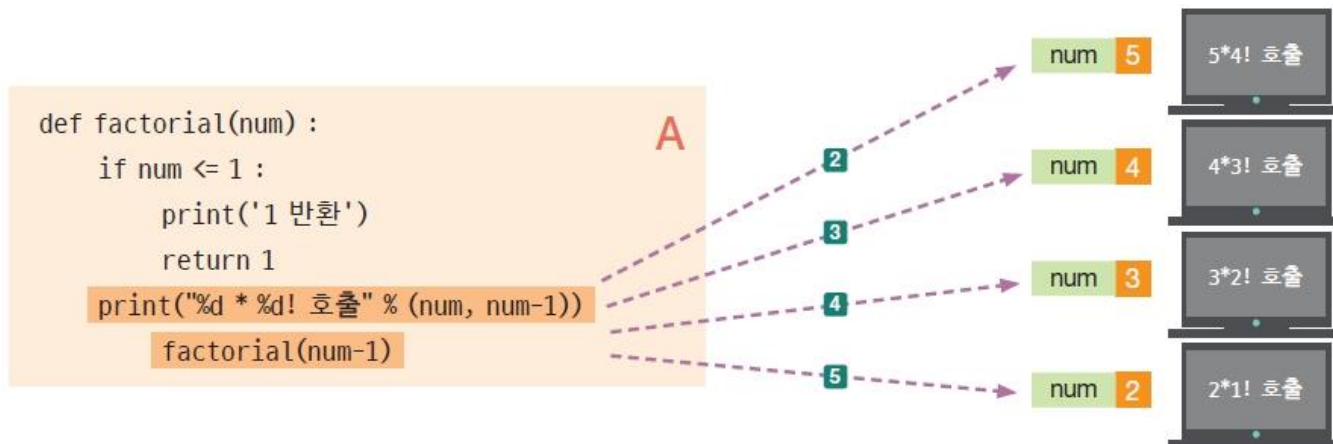
- 5!을 재귀 호출로 구현한 코드의 단계별 작동 과정

1 A부분에서 11행의 factorial(5)를 호출한다.

```
def factorial(num) :  
    if num <= 1 :  
        print('1 반환')  
        return 1  
    print("%d * %d! 호출" % (num, num-1))  
    B retVal = factorial(num-1)  
    print("%d * %d!(=d) 반환" % (num, num-1, retVal))  
    return num * retVal  
  
print('\n5! = ', factorial(5))
```

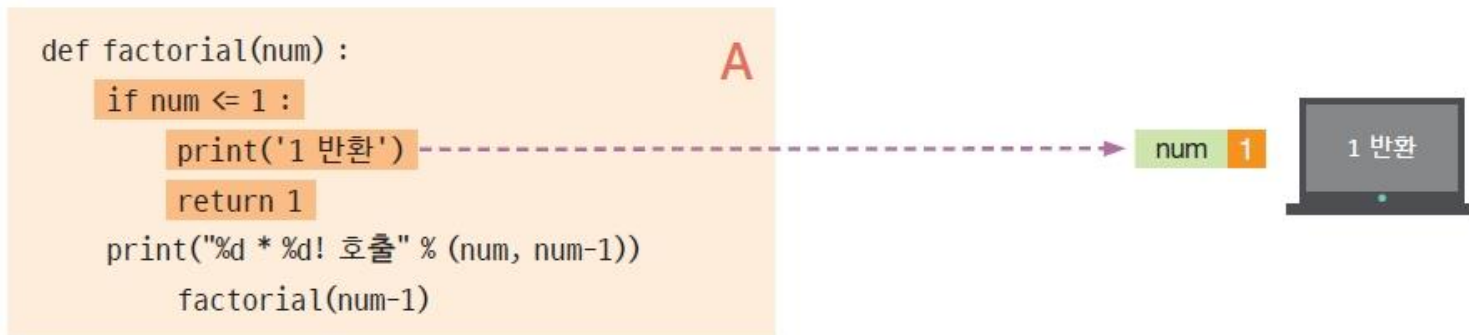


2 ~ 5 num이 5, 4, 3, 2인 상태에서 A 부분의 5~6행 실행(재귀 함수 1~4회 호출)

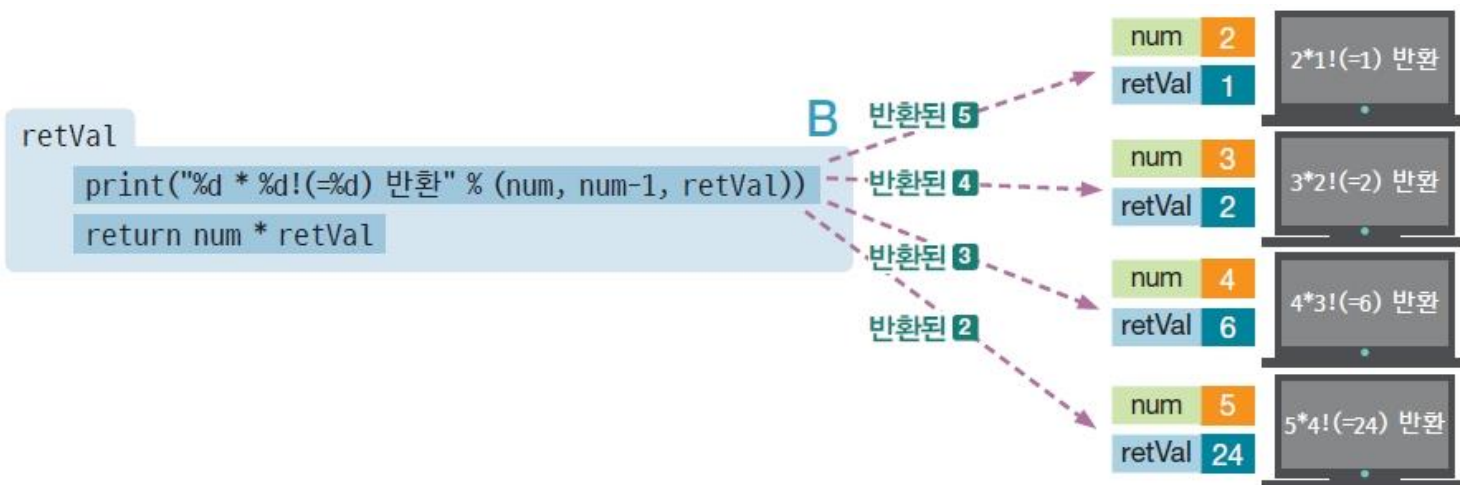


Section 02 재귀 호출 작동 방식의 이해

6 num이 1인 상태에서 A 부분의 2~4행을 실행하여 1 반환(재귀 함수 5회 호출)



반환된 5 ~ 2 num이 2, 3, 4, 5인 상태에서 B 부분의 8~9행 실행



Section 02 재귀 호출 작동 방식의 이해

반환된 **1** 처음 호출한 factorial(5)가 반환되어 최종 5!인 120 출력

```
def factorial(num) :  
    if num <= 1 :  
        print('1 반환')  
        return 1  
    print("%d * %d! 호출" % (num, num-1))
```

A

```
B retVal = factorial(num-1)  
  
print("%d * %d!(=%d) 반환" % (num, num-1, retVal))  
return num * retVal
```

```
print('\n5! = ', factorial(5))
```

120 반환



Section 03 재귀 호출의 연습

■ 우주선 발사 카운트다운

- 우주선 발사를 위해 카운트하는 코드

Code10-05.py 카운트다운을 재귀 호출로 구현

```
1 def countDown(n) :  
2     if n == 0 :  
3         print('발사!!')  
4     else :  
5         print(n)  
6         countDown(n-1)  
7  
8 countDown(5)
```



실행 결과

```
5  
4  
3  
2  
1  
발사!!
```

Section 03 재귀 호출의 연습

별 모양 출력하기

- 입력한 숫자만큼 차례대로 별 모양을 출력하는 코드

Code10-06.py 별 모양 출력을 재귀 호출로 구현

```
1 def printStar(n) :  
2     if n > 0 :  
3         printStar(n-1)  
4         print('★' * n)  
5  
6 printStar(5)
```



실행 결과

```
★  
★★  
★★★  
★★★★  
★★★★★
```

Section 03 재귀 호출의 연습

■ 구구단 출력하기

- 2단부터 9단까지 구구단을 출력하는 코드

Code10-07.py 구구단 출력을 재귀 호출로 구현

```
1 def gugu(dan, num) :  
2     print("%d x %d = %d" % (dan, num, dan*num))  
3     if num < 9 :  
4         gugu(dan, num+1)  
5  
6 for dan in range(2, 10) :  
7     print("## %d단 ##" % dan)  
8     gugu(dan, 1)
```

2단	3단	4단	5단
2 X 1 = 2 2 X 2 = 4 2 X 3 = 6 2 X 4 = 8 2 X 5 = 10 2 X 6 = 12 2 X 7 = 14 2 X 8 = 16 2 X 9 = 18	3 X 1 = 3 3 X 2 = 6 3 X 3 = 9 3 X 4 = 12 3 X 5 = 15 3 X 6 = 18 3 X 7 = 21 3 X 8 = 24 3 X 9 = 27	4 X 1 = 4 4 X 2 = 8 4 X 3 = 12 4 X 4 = 16 4 X 5 = 20 4 X 6 = 24 4 X 7 = 28 4 X 8 = 32 4 X 9 = 36	5 X 1 = 5 5 X 2 = 10 5 X 3 = 15 5 X 4 = 20 5 X 5 = 25 5 X 6 = 30 5 X 7 = 35 5 X 8 = 40 5 X 9 = 45
6단	7단	8단	9단
6 X 1 = 6 6 X 2 = 12 6 X 3 = 18 6 X 4 = 24 6 X 5 = 30 6 X 6 = 36 6 X 7 = 42 6 X 8 = 48 6 X 9 = 54	7 X 1 = 7 7 X 2 = 14 7 X 3 = 21 7 X 4 = 28 7 X 5 = 35 7 X 6 = 42 7 X 7 = 49 7 X 8 = 56 7 X 9 = 63	8 X 1 = 8 8 X 2 = 16 8 X 3 = 24 8 X 4 = 32 8 X 5 = 40 8 X 6 = 48 8 X 7 = 56 8 X 8 = 64 8 X 9 = 72	9 X 1 = 9 9 X 2 = 18 9 X 3 = 27 9 X 4 = 36 9 X 5 = 45 9 X 6 = 54 9 X 7 = 63 9 X 8 = 72 9 X 9 = 81

실행 결과

```
## 2단 ##  
2 x 1 = 2  
2 x 2 = 4  
2 x 3 = 6  
...(중략)...  
9 x 8 = 72  
9 x 9 = 81
```

Section 03 재귀 호출의 연습

SELF STUDY 10-2

Code10-07.py를 수정해서 각 단이 세로로 나오도록 코드를 작성하자.

실행 결과

2x1= 2	3x1= 3	4x1= 4	5x1= 5	6x1= 6	7x1= 7	8x1= 8	9x1= 9
2x2= 4	3x2= 6	4x2= 8	5x2=10	6x2=12	7x2=14	8x2=16	9x2=18
2x3= 6	3x3= 9	4x3=12	5x3=15	6x3=18	7x3=21	8x3=24	9x3=27
2x4= 8	3x4=12	4x4=16	5x4=20	6x4=24	7x4=28	8x4=32	9x4=36
2x5=10	3x5=15	4x5=20	5x5=25	6x5=30	7x5=35	8x5=40	9x5=45
2x6=12	3x6=18	4x6=24	5x6=30	6x6=36	7x6=42	8x6=48	9x6=54
2x7=14	3x7=21	4x7=28	5x7=35	6x7=42	7x7=49	8x7=56	9x7=63
2x8=16	3x8=24	4x8=32	5x8=40	6x8=48	7x8=56	8x8=64	9x8=72
2x9=18	3x9=27	4x9=36	5x9=45	6x9=54	7x9=63	8x9=72	9x9=81

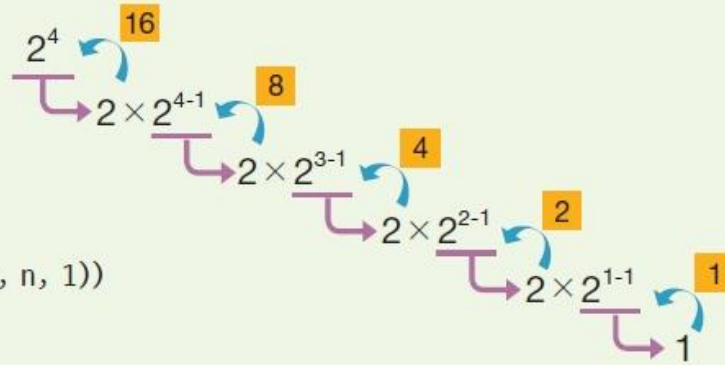
Section 03 재귀 호출의 연습

■ N제곱 계산하기

- N제곱을 계산하는 코드

Code10-08.py N제곱 계산을 재귀 호출로 구현

```
1 tab = ''
2 def pow(x, n) :
3     global tab
4     tab += ' '
5     if n == 0 :
6         return 1
7     print(tab + "%d*%d^(%d-%d)" % (x, x, n, 1))
8     return x * pow(x, n-1)
9
10 print('2^4')
11 print('답 -->', pow(2, 4))
```



실행 결과

```
2^4
  2*2^(4-1)
    2*2^(3-1)
      2*2^(2-1)
        2*2^(1-1)

답 --> 16
```

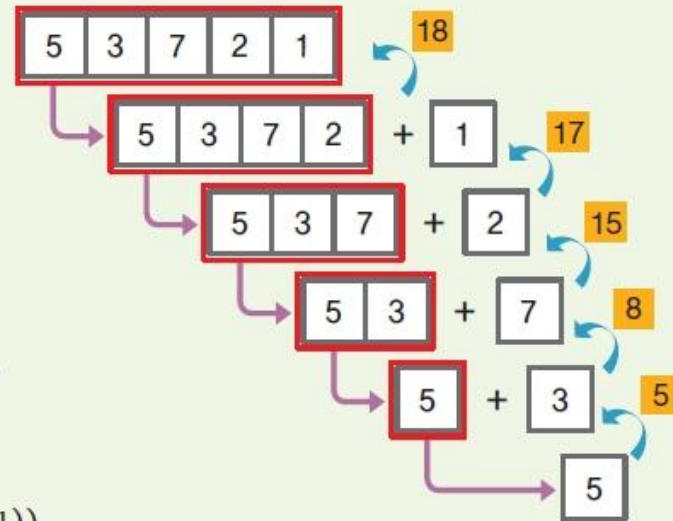
Section 03 재귀 호출의 연습

배열의 합 계산하기

- 랜덤하게 생성한 배열의 합계를 구하는 코드

Code10-09.py 배열의 합계를 재귀 호출로 구현(실행 결과는 실행할 때마다 다름)

```
1 import random
2
3 def arySum(arr, n) :
4     if n <= 0 :
5         return arr[0]
6     return arySum(arr, n-1) + arr[n]
7
8 ary = [random.randint(0, 255) for _ in range
9         (random.randint(10, 20))]
9 print(ary)
10 print('배열 합계 -->', arySum(ary, len(ary)-1))
```



실행 결과

[150, 71, 135, 53, 16, 190, 132, 21, 152, 147, 71, 69, 66, 190, 134, 199, 235, 228]

배열 합계 --> 2259

Section 03 재귀 호출의 연습

■ 피보나치 수

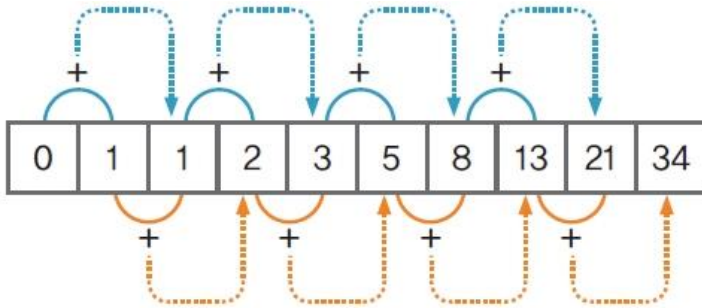


그림 10-11 피보나치 수 구성

Code10-10.py 피보나치 수를 재귀 호출로 구현

```
1 def fibo(n) :  
2     if n == 0 :  
3         return 0  
4     elif n == 1 :  
5         return 1  
6     else :  
7         return fibo(n-1) + fibo(n-2)  
8  
9 print('피보나치 수 --> 0 1 ', end = ' ' )  
10 for i in range(2, 20) :  
11     print(fibo(i), end = ' ' )
```

실행 결과

피보나치 수 --> 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181

Section 04 재귀 호출의 응용

회문 판단하기

- 회문(Palindrome)은 앞에서부터 읽든 뒤에서부터 읽든 동일한 단어나 문장을 의미

회문 예

level
kayak
radar
Borrow or rob
I prefer pi
기러기
일요일
주유소의 소유
다 큰 도라지일지라도 크다
야 너 이번 주 주변이 너야
야 이 달은 밝은 달이야
마지막 날 날 막지 마

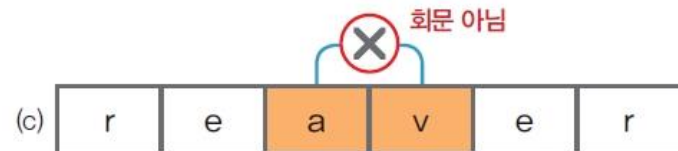
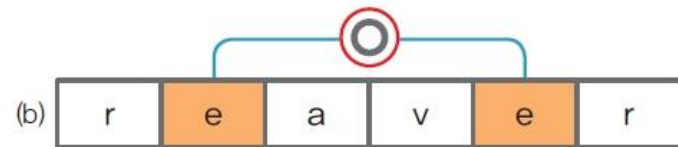
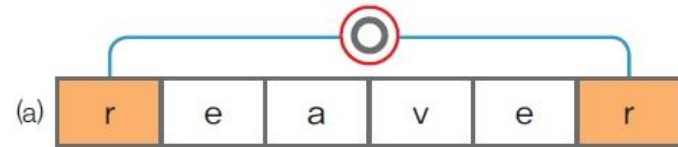


그림 10-12 회문이 아닌 예 : reaver

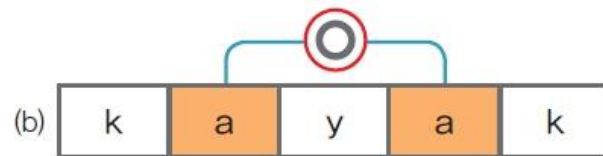
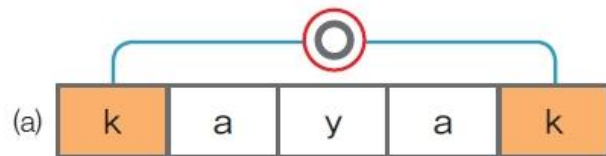


그림 10-13 회문인 예 : kayak

Section 04 재귀 호출의 응용

Code10-11.py 회문 여부를 구별하기(실행 결과는 실행할 때마다 다름)

```
1  ## 클래스와 함수 선언 부분 ##
2  def palindrome(pStr) :
3      if len(pStr) <= 1 :
4          return True
5
6      if pStr[0] != pStr[-1] :
7          return False
8
9      return palindrome(pStr[1:len(pStr)-1])
10
11
12 ## 전역 변수 선언 부분 ##
13 strAry = ["reaver", "kayak", "Borrow or rob", "주유소의 소유주", "야 너 이번 주 주변이 너야", "살금 살금"]
14
15 ## 메인 코드 부분 ##
16 for testStr in strAry :
17     print(testStr, end = '--> ')
18     testStr = testStr.lower().replace(' ', '')
19     if palindrome(testStr) :
20         print('O')
21     else :
22         print('X')
```

실행 결과

```
reaver--> X
kayak--> O
Borrow or rob--> O
주유소의 소유주--> O
야 너 이번 주 주변이 너야--> O
살금 살금--> X
```

■ 프랙탈 그리기

- 프랙탈(Fractal)은 작은 조각이 전체와 비슷한 기하학적인 형태를 의미(자기 유사성)
- 자기 유사성(Self Similarity)은 부분을 확대하면 전체와 동일한 또는 닮은 꼴의 모습을 나타내는 성질이 있음

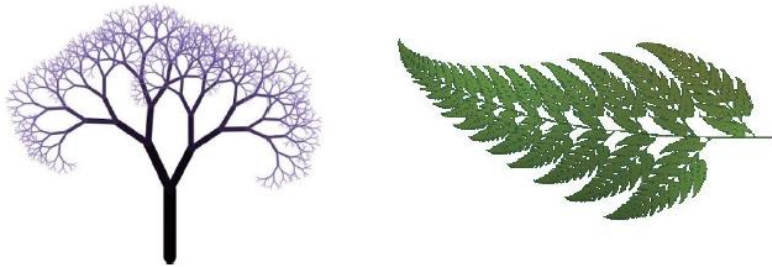


그림 10-14 자연 현상에서 나타나는 프랙탈 형태

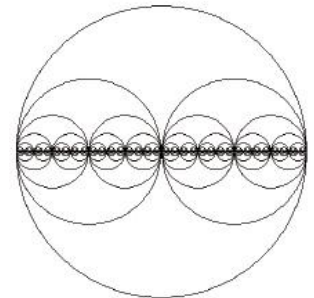
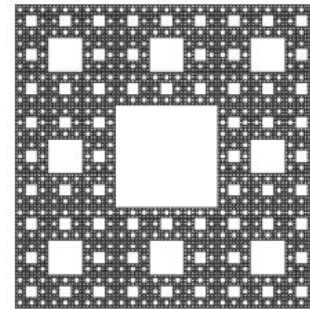


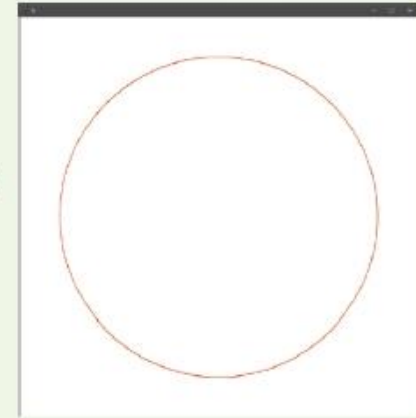
그림 10-15 수학적 도형으로 구성된 프랙탈

Section 04 재귀 호출의 응용

- 1000×1000 크기의 원도 창을 만들고 중앙에 반지름 400 크기의 원을 그리는 코드

Code10-12.py 간단한 원을 그리는 GUI 프로그래밍

```
1 from tkinter import *
2
3 window = Tk()
4 canvas = Canvas(window, height=1000, width=1000, bg='white')
5 canvas.pack()
6
7 cx = 1000//2
8 cy = 1000//2
9 r = 400
10 canvas.create_oval(cx-r, cy-r, cx+r, cy+r, width=2, outline="red")
11
12 window.mainloop()
```



Section 04 재귀 호출의 응용

- 원 도형의 간단한 프랙탈 그리기
 - 하나의 원 안에 작은 원을 2개 좌우로 그리는 것을 재귀 호출로 반복

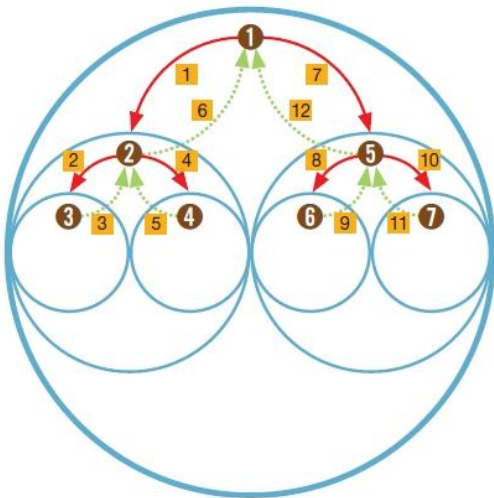
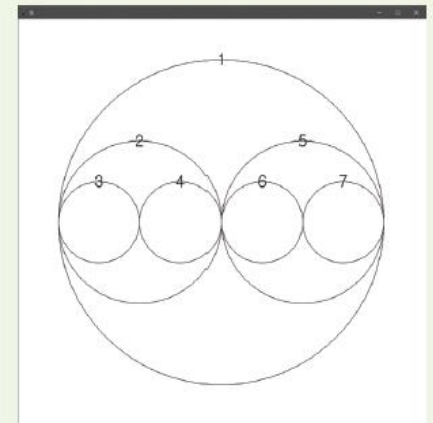


그림 10-16 3단계까지 원을 그리는 재귀 호출 방식

Code10-13.py 3단계의 프랙탈 원 그리기

```
1 from tkinter import *
2
3 ## 클래스와 함수 선언 부분 ##
4 def drawCircle(x, y, r) :
5     global count
6     count += 1
7     canvas.create_oval(x-r, y-r, x+r, y+r)
8     canvas.create_text(x, y-r, text=str(count), font=(' ', 30))
9     if r >= radius/2 :
10         drawCircle(x-r//2, y, r//2)
11         drawCircle(x+r//2, y, r//2)
12
13 ## 전역 변수 선언 부분 ##
14 count = 0
15 wSize = 1000
16 radius = 400
17
18 ## 메인 코드 부분 ##
19 window = Tk()
20 canvas = Canvas(window, height=wSize, width=wSize, bg='white')
21
22 drawCircle(wSize//2, wSize//2, radius)
23
24 canvas.pack()
25 window.mainloop()
```

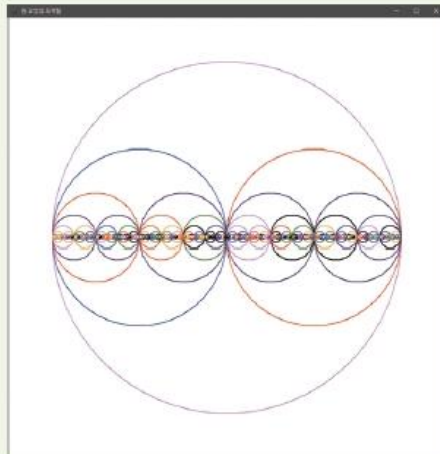


Section 04 재귀 호출의 응용

■ 원 도형의 전체 프랙탈 그리기

Code10-14.py 전체 프랙탈 원 그리기

```
1 from tkinter import *
2 import random
3
4 ## 클래스와 함수 선언 부분 ##
5 def drawCircle(x, y, r) :
6     canvas.create_oval(x-r, y-r, x+r, y+r, width=2, outline=random.choice(colors))
7     if r >= 5 :
8         drawCircle(x+r//2, y, r//2)
9         drawCircle(x-r//2, y, r//2)
10
11 ## 전역 변수 선언 부분 ##
12 colors = ["red", "green", "blue", "black", "orange", "indigo", "violet"]
13 wSize = 1000
14 radius = 400
15
16 ## 메인 코드 부분 ##
17 window = Tk()
18 window.title("원 모양의 프랙탈")
19 canvas = Canvas(window, height=wSize, width=wSize,
20                 bg='white')
21 drawCircle(wSize//2, wSize//2, radius)
22
23 canvas.pack()
24 window.mainloop()
```



응용예제 01 진수 변환하기

난이도 ★ ★ ☆ ☆ ☆

예제 설명

10진수 정수를 입력하면, 2진수/8진수/16진수로 변환되어 출력되는 프로그램을 재귀 함수를 이용하여 작성한다.



실행 결과

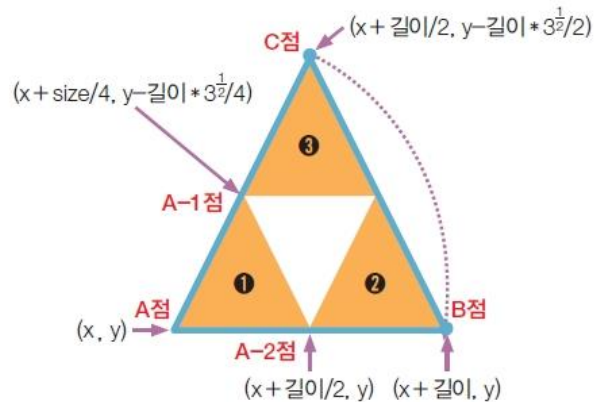
```
Python
File Edit Shell Debug Options Window Help
===== RESTART: C:\CookData\Ex10-01.py =====
10진수 입력 --> 65535
2진수 : 1111111111111111
8진수 : 177777
16진수 : FFFF
>>> |
```

응용예제 02 시에르핀스키 삼각형 그리기

난이도★★★★☆

예제 설명

시에르핀스키(Sierpinski) 삼각형은 다음과 같이 큰 삼각형 안에 다시 정삼각형 3개를 균등하게 생성한다. 변의 길이와 왼쪽 아래 시작점(x, y)이 정해지면 큰 삼각형은 A점, B점, C점을 연결하면 된다. 그리고 다시 안쪽 삼각형 3개는 A점, A-1점, A-2점이 시작점이 되고 변의 길이는 1/2이 된다. 이것을 재귀 호출로 반복하면 프랙탈 모양이 된다. 각 위치를 구하는 수식은 그림에 표현했다.



실행 결과

