CHAPTER.20

재귀 호출의 연습 및 응용

학습 내용

[1] 재귀 호출의 연습

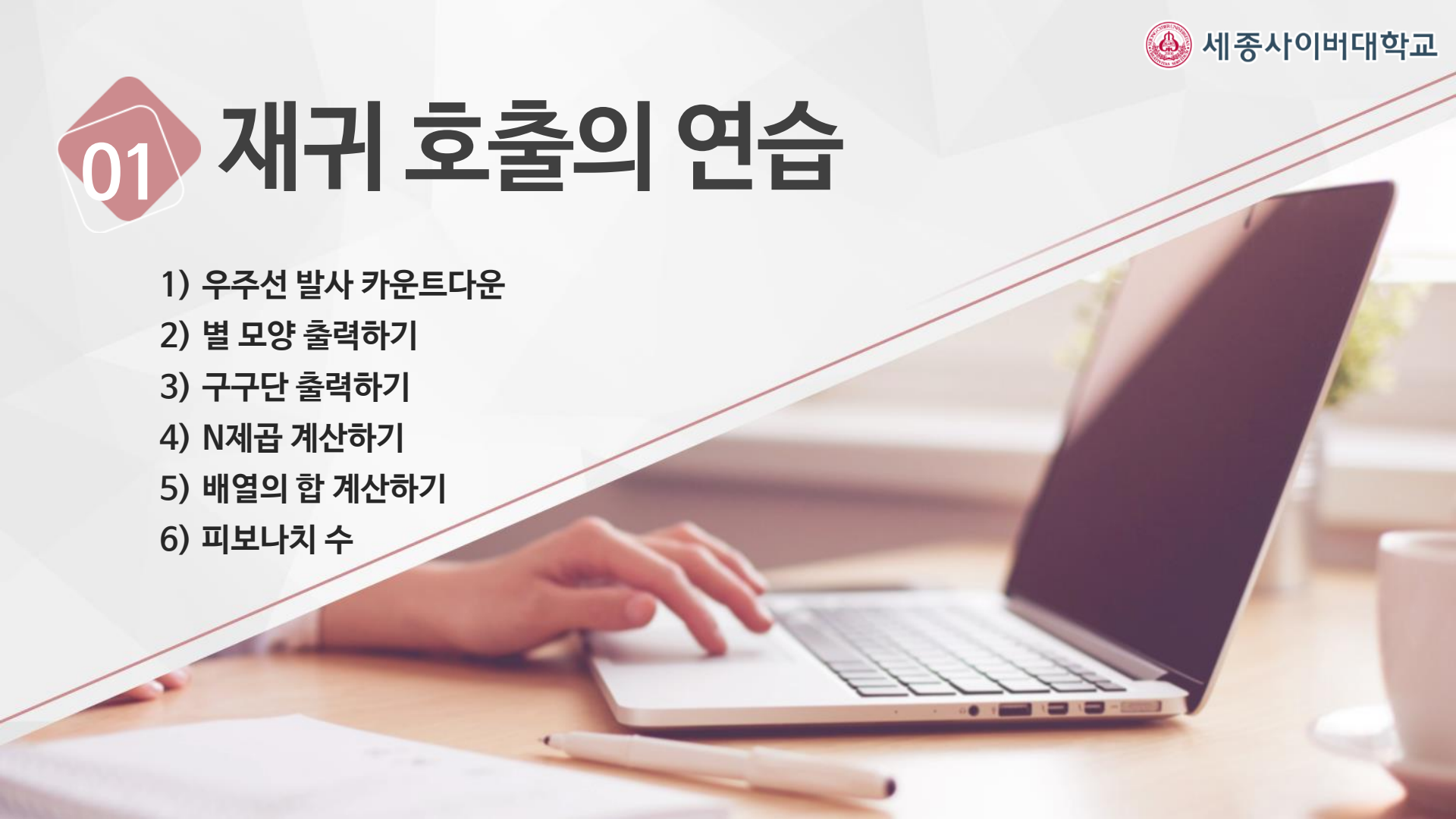
[2] 재귀 호출의 응용

학습 목표

- 재귀 호출을 다양한 응용 예로 연습할 수 있다.

01 재귀 호출의 연습

1) 우주선 발사 카운트다운
2) 별 모양 출력하기
3) 구구단 출력하기
4) N제곱 계산하기
5) 배열의 합 계산하기
6) 피보나치 수



1] 우주선 발사 카운트다운

우주선 발사를 위해 카운트하는 코드

- 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)
```

1] 우주선 발사 카운트다운

우주선 발사를 위해 카운트하는 코드

- 카운트다운을 재귀 호출로 구현

실행 결과

```
5
4
3
2
1
발사!!
```

2] 별 모양 출력하기

입력한 숫자만큼 차례대로 별 모양을 출력하는 코드

- 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)
```

입력한 숫자만큼 차례대로 별 모양을 출력하는 코드

- 별 모양 출력을 재귀 호출로 구현

실행 결과

```
★
★★
★★★
★★★★
★★★★★
```

3] 구구단 출력하기

2단부터 9단까지 구구단을 출력하는 코드

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

3] 구구단 출력하기

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)
```

구구단 출력을 재귀 호출로 구현

실행 결과

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

구구단 출력하기 실습

앞쪽의 소스를 수정해서 각 단이 세로로 나오도록 코드를 작성하자.

실행 결과

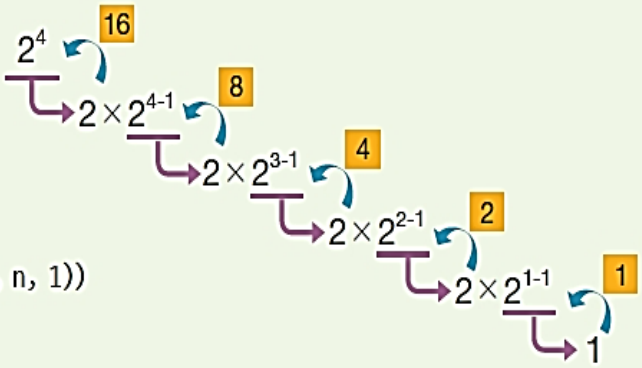
```
2x1= 2  3x1= 3  4x1= 4  5x1= 5  6x1= 6  7x1= 7  8x1= 8  9x1= 9
2x2= 4  3x2= 6  4x2= 8  5x2=10  6x2=12  7x2=14  8x2=16  9x2=18
2x3= 6  3x3= 9  4x3=12  5x3=15  6x3=18  7x3=21  8x3=24  9x3=27
2x4= 8  3x4=12  4x4=16  5x4=20  6x4=24  7x4=28  8x4=32  9x4=36
2x5=10  3x5=15  4x5=20  5x5=25  6x5=30  7x5=35  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
```

구구단 출력하기 실습

N제곱을 계산하는 코드

- 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))
```



N제곱을 계산하는 코드

- N제곱 계산을 재귀 호출로 구현

실행 결과

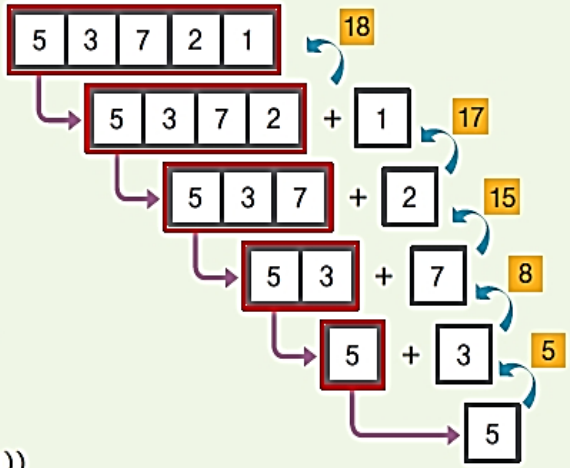
```
2^4
    2*2^(4-1)
        2*2^(3-1)
            2*2^(2-1)
                2*2^(1-1)
답 --> 16
```

5] 배열의 합 계산하기

랜덤하게 생성한 배열의 합계를 구하는 코드

- 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
       (random.randint(10, 20))]
9  print(ary)
10 print('배열 합계 -->', arySum(ary, len(ary)-1))
```



5] 배열의 합 계산하기

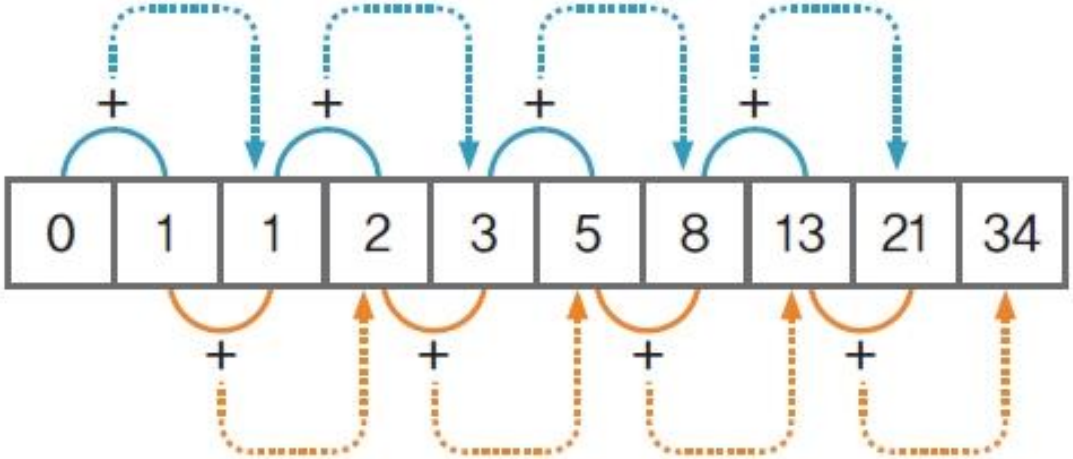
랜덤하게 생성한 배열의 합계를 구하는 코드

- 배열의 합계를 재귀 호출로 구현(실행 결과는 실행할 때마다 다름)

실행 결과

```
[150, 71, 135, 53, 16, 190, 132, 21, 152, 147, 71, 69, 66, 190, 134, 199, 235, 228]
배열 합계 --> 2259
```

피보나치 수 구성



6] 피보나치 수

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
```

02 재귀 호출의 응용

1) 회문 판단하기
2) 프랙탈 그리기
3) 재귀 호출 응용 연습하기

1] 회문 판단하기

회문(Palindrome)

앞에서부터 읽든 뒤에서부터 읽든 동일한 단어나 문장을 의미

- 회문의 예

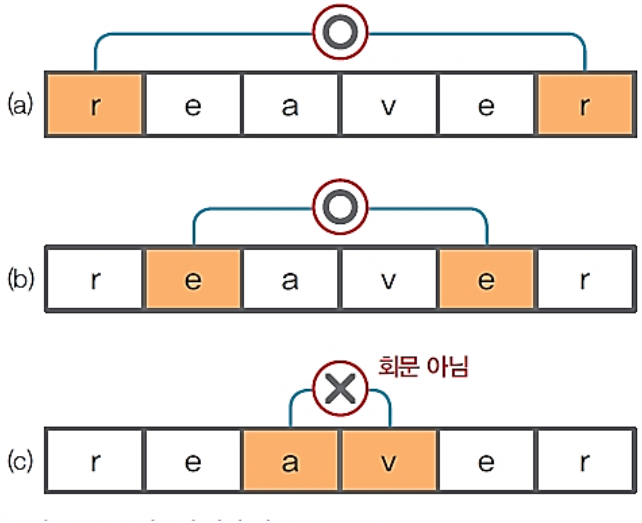
```
level
kayak
radar
Borrow or rob
I prefer pi
기러기
일요일
주유소의 소유
다 큰 도라지일지라도 큰다
야 너 이번 주 주번이 너야
야 이 달은 밝은 달이야
마지막 날 날 막지 마
```

1] 회문 판단하기

회문(Palindrome)

앞에서부터 읽든 뒤에서부터 읽든 동일한 단어나 문장을 의미

- 회문이 아닌 예 : reaver

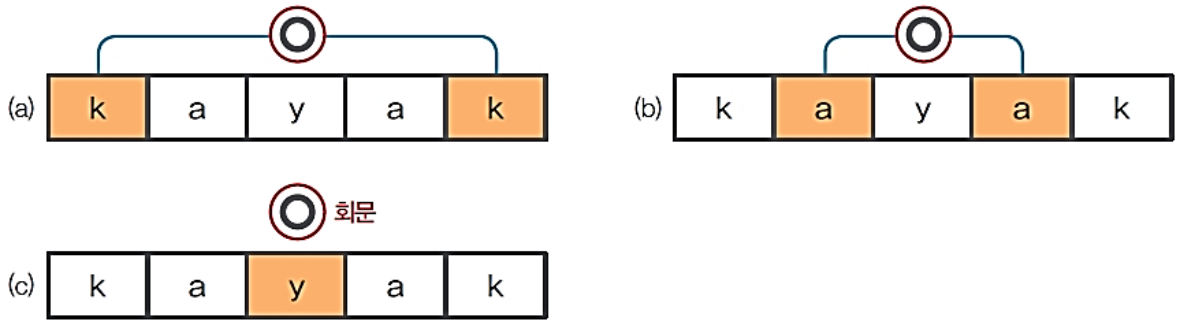


1] 회문 판단하기

회문(Palindrome)

앞에서부터 읽든 뒤에서부터 읽든 동일한 단어나 문장을 의미

- 회문인 예 : kayak



1] 회문 판단하기

회문 여부를 구별하기 (실행 결과는 실행할 때마다 다름)

```
## 클래스와 함수 선언 부분 ##
def palindrome(pStr) :
    if len(pStr) <= 1 :
        return True

    if pStr[0] != pStr[-1] :
        return False

    return palindrome(pStr[1:len(pStr)-1])


## 전역 변수 선언 부분 ##
strAry = ["reaver", "kayak", "Borrow or rob", "주유소의 소유주", "야 너 이번 주 주번이 너야", "살금 살금"]

```

회문 여부를 구별하기 (실행 결과는 실행할 때마다 다름)

```
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 주번이 너야--> O
살금 살금--> X
```

2] 프랙탈 그리기

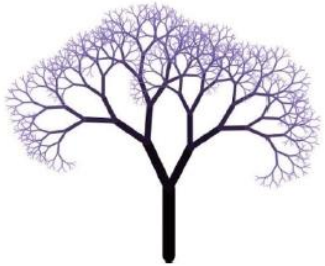
프랙탈 (Fractal)

작은 조각이 전체와 비슷한 기하학적인 형태를 의미
(자기 유사성)

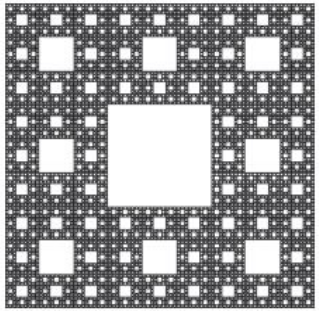
자기 유사성 (Self Similarity)

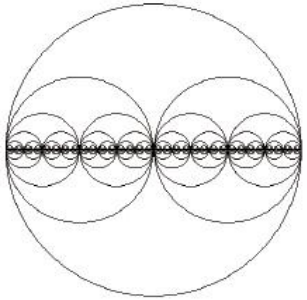
부분을 확대하면 전체와 동일한 또는 닮은 꼴의 모습을
나타내는 성질이 있음

자연 현상에서 나타나는 프랙탈 형태



수학적 도형으로 구성한 프랙탈





2] 프랙탈 그리기

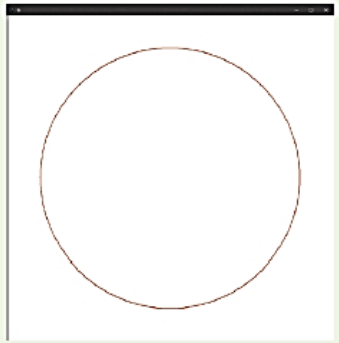
1000×1000 크기의 윈도 창을 만들고
중앙에 반지름 400 크기의 원을 그리는 코드

- 구글 colab이 아니라 파이썬 IDE에서 수행해야 함
- 그래픽 display는 로컬 머신에 있고 Colab의 Python 코드는 GCP의 가상 머신에서 실행되기 때문

1000×1000 크기의 윈도 창을 만들고 중앙에 반지름 400 크기의 원을 그리는 코드

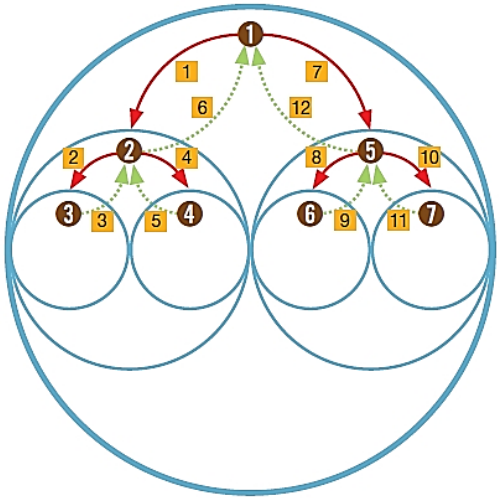
- 간단한 원을 그래는 GUI 프로그래밍

```
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()
```



원 도형의 간단한 프랙탈 그리기

- 하나의 원 안에 작은 원을 2개 좌우로 그리는 것을 재귀 호출로 반복
- 3단계까지 원을 그리는 재귀 호출 방식



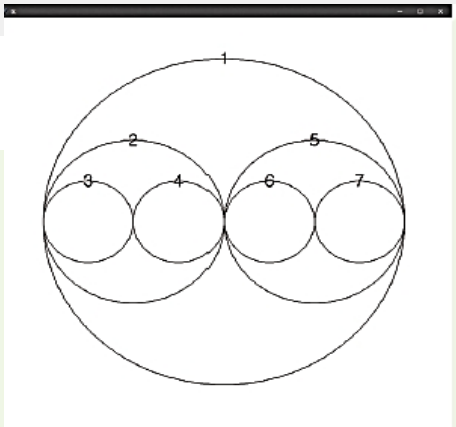
원 도형의 간단한 프랙탈 그리기

- 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
```

원 도형의 간단한 프랙탈 그리기

- 3단계의 프랙탈 원 그리기



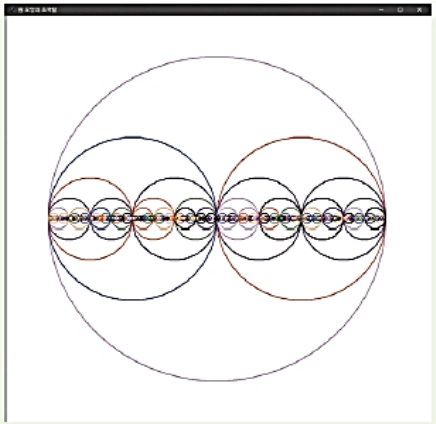

```
13 ## 전역 변수 선언 부분 ##
14 count = 0
15 wSize = 1000
16 radius = 400
17
18 ## 메인 코드 부분 ##
19 window = Tk()
20 canvas = Canvas(window, height=wSize, width=wSize, bg='white')
21
22 drawCircle(wSize//2, wSize//2, radius)
23
24 canvas.pack()
25 window.mainloop()
```

원 도형의 전체 프랙탈 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
```

원 도형의 전체 프랙탈 그리기

```
15
16 ## 메인 코드 부분 ##
17 window = Tk()
18 window.title("원 모양의 프랙탈")
19 canvas = Canvas(window, height=wSize, width=wSize,
   bg='white')
20
21 drawCircle(wSize//2, wSize//2, radius)
22
23 canvas.pack()
24 window.mainloop()
```



3] 재귀 호출 응용 연습하기

재귀 호출 응용 실습

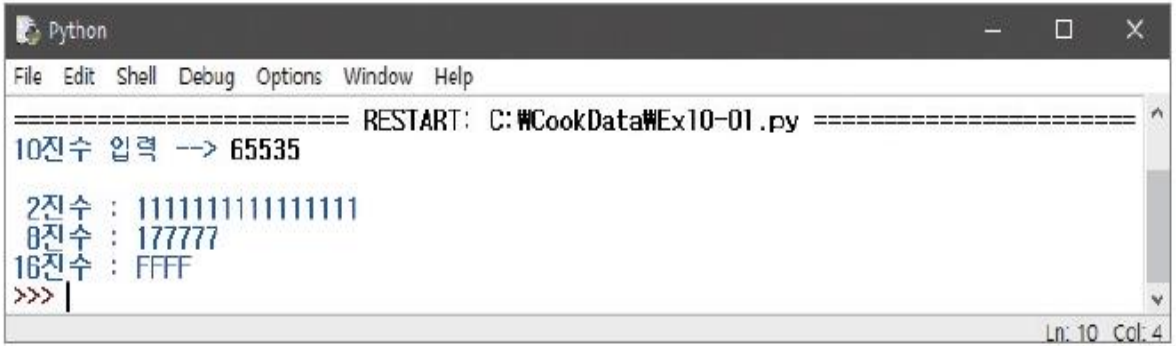
10진수 정수를 입력하면, 2진수/8진수/16진수로 변환되어 출력되는 프로그램을 재귀 함수를 이용하여 작성한다.

3] 재귀 호출 응용 연습하기

재귀 호출 응용 실습

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

실행 결과



3] 재귀 호출 응용 연습하기

재귀 호출 응용 실습

학습평가

Q1 Q2

Q1

입력한 숫자의 크기만큼 별 모양을 출력하는 함수의 (1)~(3)을 완성하시오.

```
def printStar(n):
  if (   1   ):
    print('*' * (  2  ))
    printStar(   3   )

printStar(5)
```

(1) _____

(2) _____

(3) _____

학습평가

Q1 Q2

Q1

입력한 숫자의 크기만큼 별 모양을 출력하는 함수의 (1)~(3)을 완성하시오.

```
def printStar(n):
  if (   1   ):
    print('*' * (  2  ))
    printStar(   3   )

printStar(5)
```

(1) n>=1

(2) n

(3) n-1

정 답 n>=1 / n / n-1

해 설 재귀 호출은 자기 자신을 호출해야 합니다.

학습평가

Q2

1부터 1000까지 임의의 숫자 배열의 합계를 재귀 호출로 구현한 코드의 (1)을 완성하시오.

```
import random                    (1)

def arySum(arr, n):
    if n <= 0:
        return arr[0]
    return (          1          )

ary = [random.randint(1,1000) for _ in
range(random.randint(10,20))]
print(ary)
print('배열 합계-->', arySum(ary, len(ary)-1))
```

학습평가

Q1 Q2

Q2

1부터 1000까지 임의의 숫자 배열의 합계를 재귀 호출로 구현한 코드의 (1)을 완성하시오.

```
import random

def arySum(arr, n):
    if n <= 0:
        return arr[0]
    return (          1          )

ary = [random.randint(1,1000) for _ in
range(random.randint(10,20))]
print(ary)
print('배열 합계-->', arySum(ary, len(ary)-1))
```

(1)

```
arySum(arr, n-1) +
arr[n]
```

학습평가

정 답 arySum(arr, n-1) + arr[n]

해 설 재귀 호출은 자기 자신을 호출해야 합니다.

재귀 호출의 연습

우주선 발사 카운트다운

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

재귀 호출의 연습

별 모양 출력하기

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