CHAPTER.19

재귀 호출의 기본 및 작동 방식의 이해

학습 내용

[1] 재귀 호출의 기본

[2] 재귀 호출 작동 방식의 이해

학습 목표

- 재귀 호출의 개념과 작동을 설명할 수 있다.
- 재귀 호출을 위한 코드 형식을 설명할 수 있다.

01 재귀 호출의 기본

1) 재귀 알고리즘이란?
2) 재귀 호출의 개념
3) 재귀 호출의 작동

1] 재귀 알고리즘이란?

> **양쪽에 거울이 있을 때 거울에 비친 자신이 무한 반복해서 비치는 것**

1] 재귀 알고리즘이란?

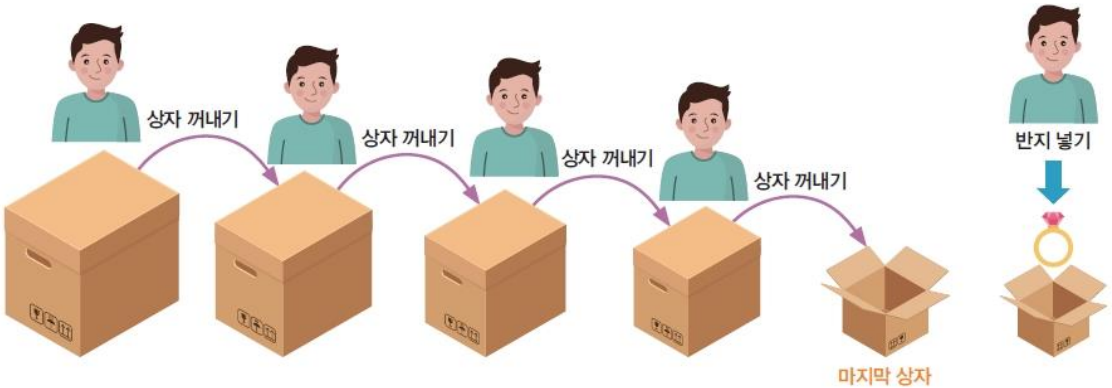
마트료시카 인형처럼 동일한 작동을 무한적으로 반복하는 알고리즘

2] 재귀 호출의 개념

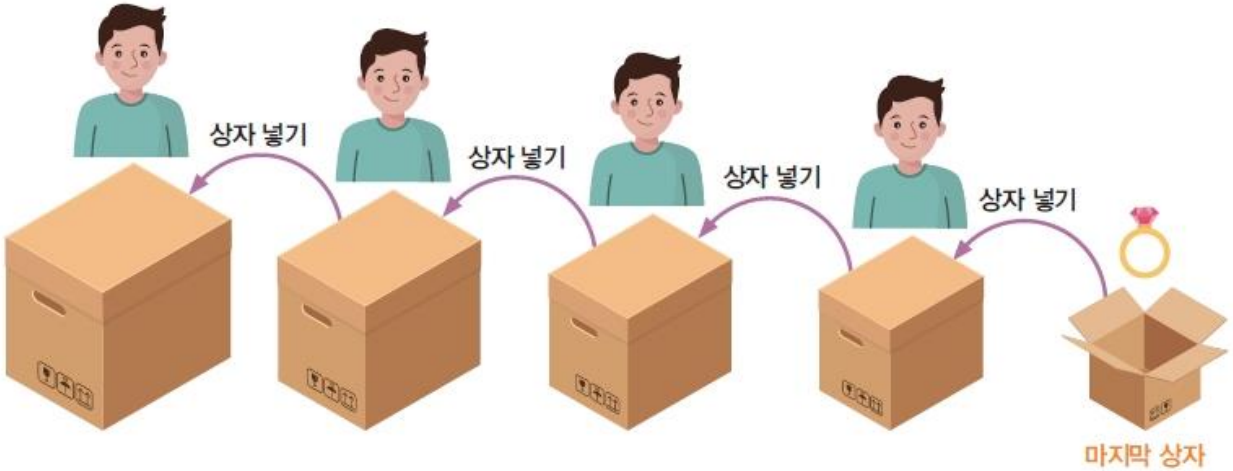
재귀 호출(Recursion)

자신을 다시 호출하는 것

마지막 종이 상자가 나올 때까지 여러 개 겹쳐진 상자를 꺼내 마지막 상자에 반지를 넣는 예



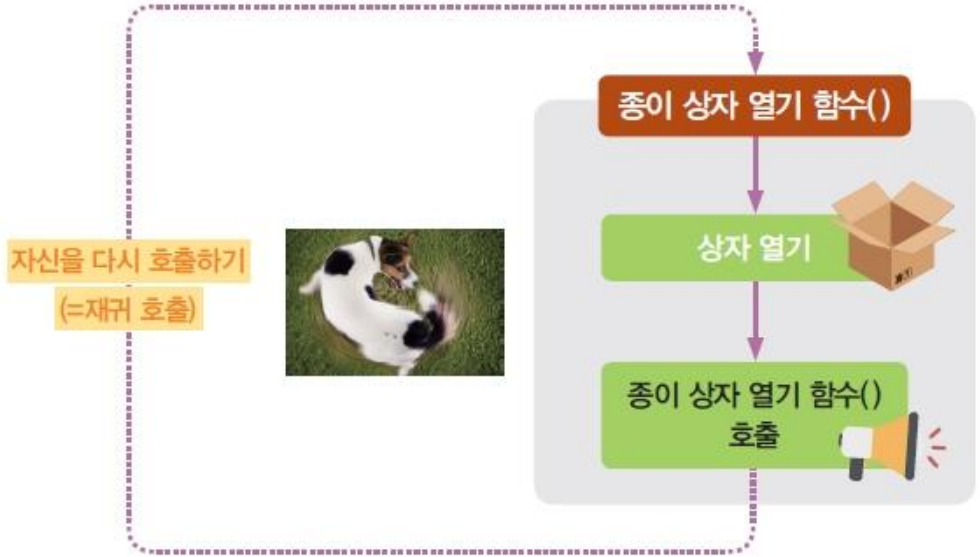
꺼낸 순서와 반대로 종이 상자를 다시 넣는 예



3] 재귀 호출의 작동

상자를 반복해서 여는 과정을 재귀 호출 형태로 표현

- 재귀 함수 형태



3] 재귀 호출의 작동

재귀 호출 함수 기본

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

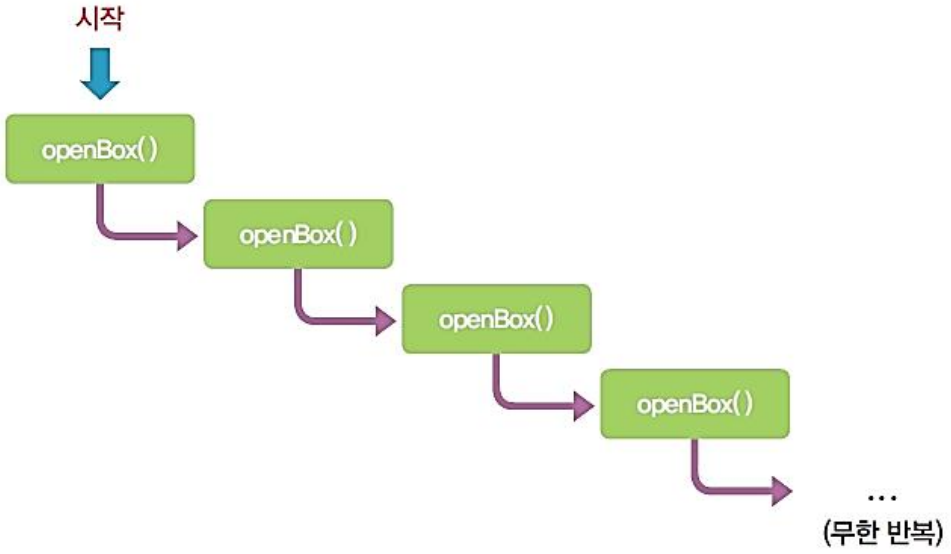
재귀 호출 함수 기본

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
>>>
```

재귀 호출 함수 기본

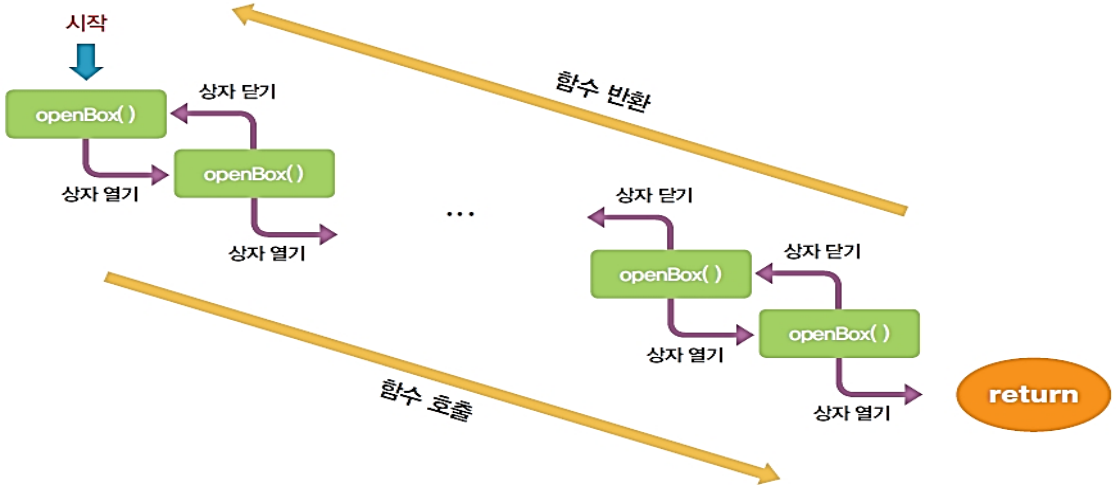
- 무한 반복하는 재귀 호출



3] 재귀 호출의 작동

일반적인 프로그램에서는 무한 반복을 마치고 되돌아가는 조건을 함께 사용함

- 10회 반복 후 호출한 곳으로 다시 돌아가는 조건을 사용하는 예



3] 재귀 호출의 작동

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
```

재귀 호출 함수 기본(반환 조건 추가)

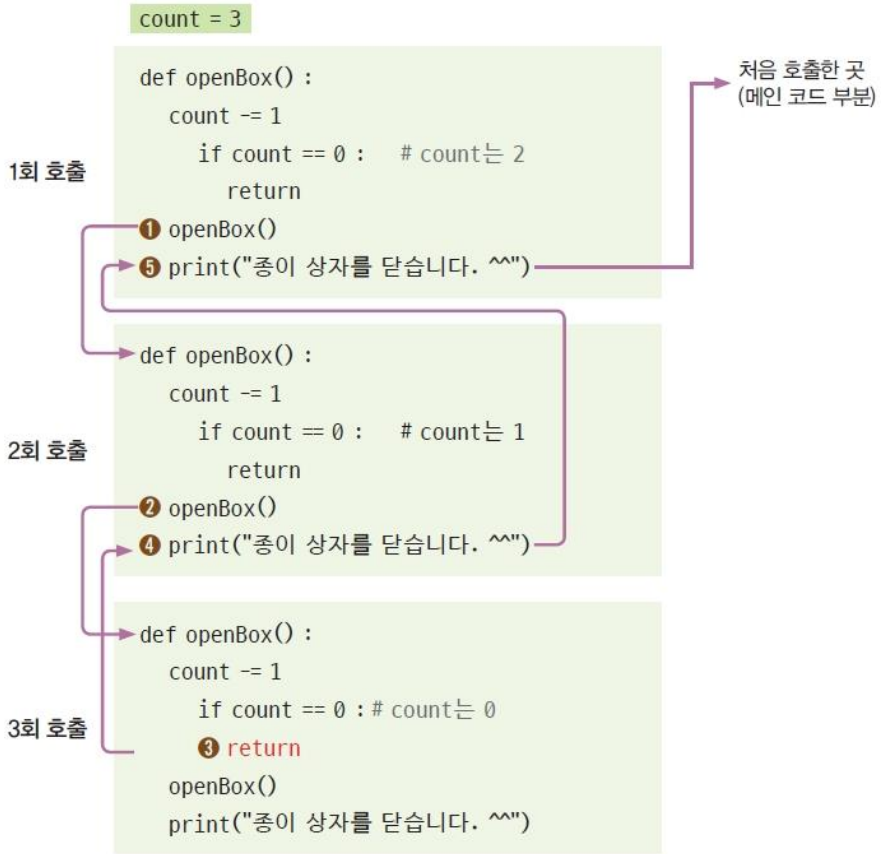
실행 결과

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

3] 재귀 호출의 작동

재귀 호출 함수 기본(반환 조건 추가)

- 7행의 return 문을 만나면 어디로 돌아가는가? (함수의재귀호출방식이해)



02

재귀 호출 작동 방식의 이해

1) 숫자 합계 내기(1부터 10까지 합계를 구하는 예)

2) 팩토리얼 구하기(10부터 1까지 곱하는 예)

1] 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)
```

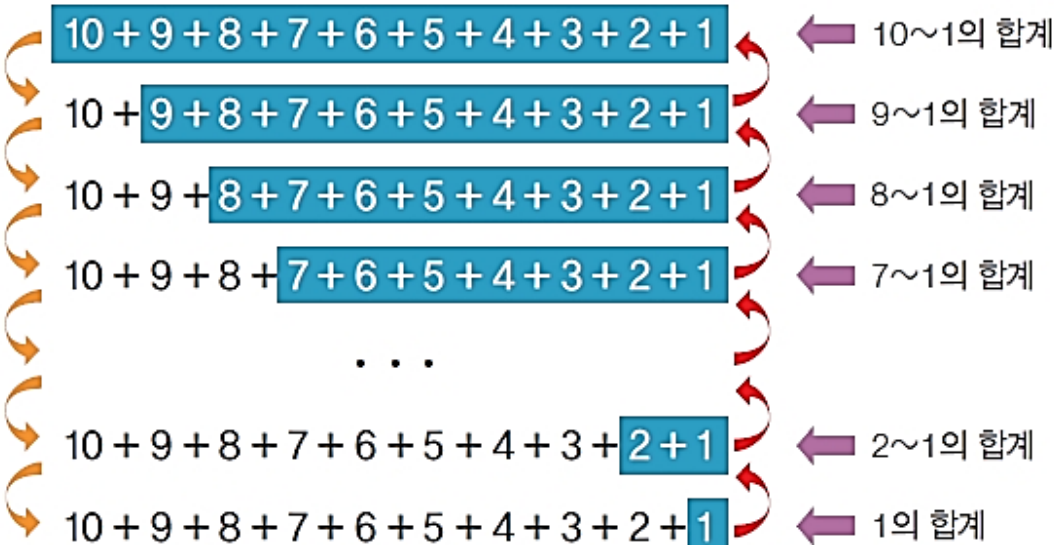
실행 결과

```
10+9+…+1 = 55
```

1] 숫자 합계 내기(1부터 10까지 합계를 구하는 예)

재귀 함수를 이용한 구현

- 10부터 1까지의 합계



1] 숫자 합계 내기(1부터 10까지 합계를 구하는 예)

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

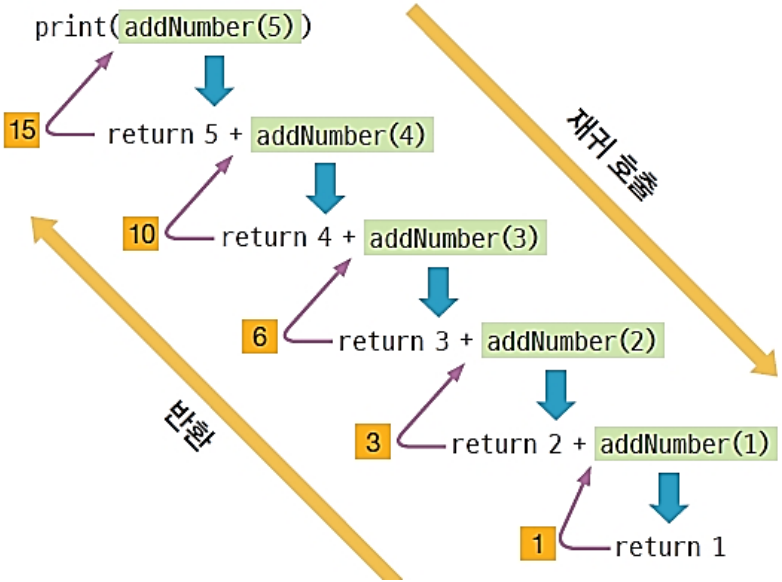
```
55
```

(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

1] 숫자 합계 내기(1부터 10까지 합계를 구하는 예)

5부터 1까지의 합계로 단순화

- 5부터 1까지의 합계 재귀 호출 과정



1] 숫자 합계 내기(1부터 10까지 합계를 구하는 예)

숫자 합계 내기 실습

앞쪽의 소스를 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
```

1] 숫자 합계 내기(1부터 10까지 합계를 구하는 예)

숫자 합계 내기 실습

2] 팩토리얼 구하기(10부터 1까지 곱하는 예)

반복문을 이용한 구현

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

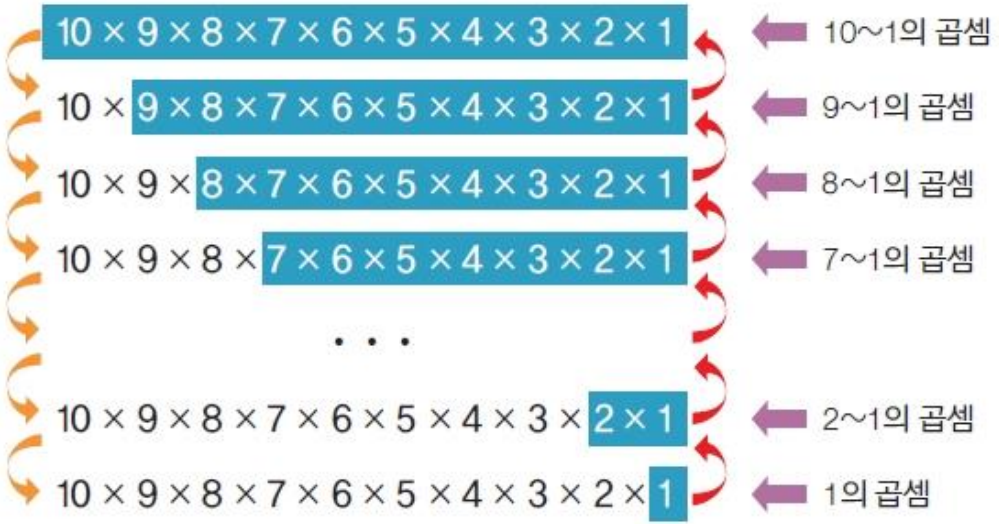
실행 결과

```
10*9*…*1 = 3628800
```

2] 팩토리얼 구하기(10부터 1까지 곱하는 예)

재귀 함수를 이용한 구현

- 10! 계산



2] 팩토리얼 구하기(10부터 1까지 곱하는 예)

재귀 호출 과정을 재귀 함수로 작성

```
def factorial(num) :
    if num <= 1 :
        return 1
    return num * factorial(num-1)

print('\n10! = ', factorial(10))
```

2] 팩토리얼 구하기(10부터 1까지 곱하는 예)

재귀 호출 과정을 재귀 함수로 작성

- 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
```

2] 팩토리얼 구하기(10부터 1까지 곱하는 예)

세종사이버대학교

5!을 재귀 호출로 구현한 코드의 단계별 작동 과정

1 A부분에서 11행의 factorial(5)를 호출

```
def factorial(num) :                                    A
    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] 팩토리얼 구하기(10부터 1까지 곱하는 예)

세종사이버대학교

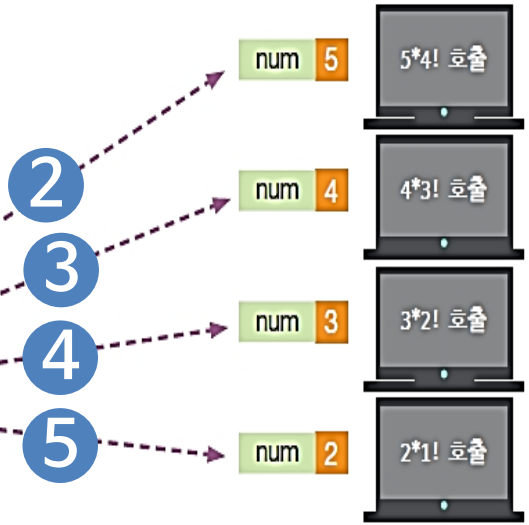
5!을 재귀 호출로 구현한 코드의 단계별 작동 과정

2 3 4 5

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

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

A



2] 팩토리얼 구하기(10부터 1까지 곱하는 예)

5!을 재귀 호출로 구현한 코드의 단계별 작동 과정

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

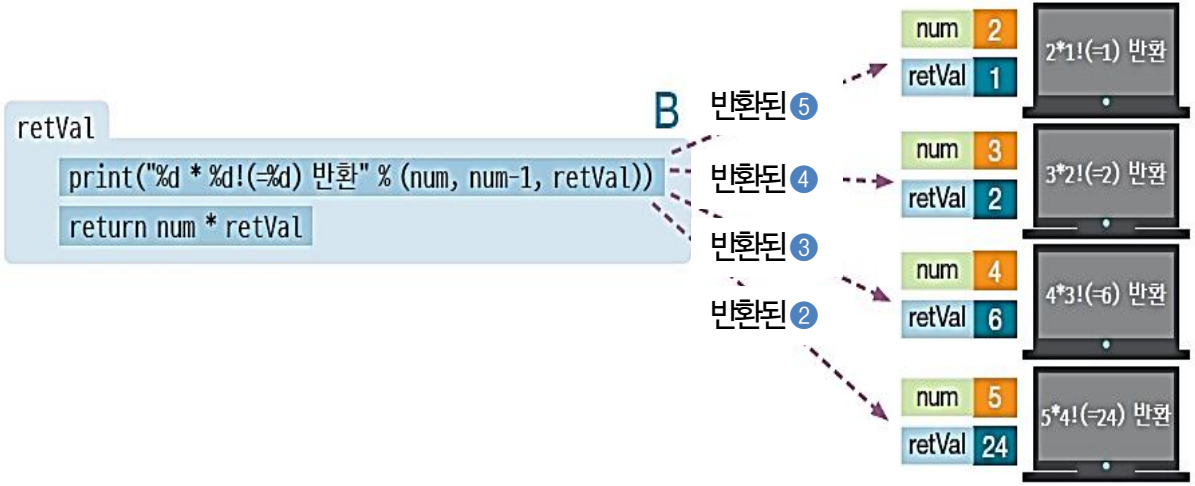
```
def factorial(num) :                                    A
    if num <= 1 :
        print('1 반환')
        return 1
    print("%d * %d! 호출" % (num, num-1))
    factorial(num-1)
```

2] 팩토리얼 구하기(10부터 1까지 곱하는 예)

세종사이버대학교

5!을 재귀 호출로 구현한 코드의 단계별 작동 과정

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



2] 팩토리얼 구하기(10부터 1까지 곱하는 예)

세종사이버대학교

5!을 재귀 호출로 구현한 코드의 단계별 작동 과정

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

```
def factorial(num) :                                    A
    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))
```

120 반환

학습평가

Q1 Q2 Q3

Q1 재귀 호출 관련 설명으로 가장 거리가 먼 것은?

1. 자기 자신을 다시 호출하는 개념이다.
2. 재귀 호출의 무한 반복 코드를 작성하면,
   파이썬은 어느 정도 반복하다가 자동 종료시킨다.
3. 재귀 호출의 무한 반복을 마치려면 return 문을 사용한다.
4. 재귀 호출을 위한 예약어는 for 또는 while이다.

학습평가

Q1 Q2 Q3

Q1 재귀 호출 관련 설명으로 가장 거리가 먼 것은?

1. 자기 자신을 다시 호출하는 개념이다.
2. 재귀 호출의 무한 반복 코드를 작성하면, 파이썬은 어느 정도 반복하다가 자동 종료시킨다.
3. 재귀 호출의 무한 반복을 마치려면 return 문을 사용한다.
4. 재귀 호출을 위한 예약어는 for 또는 while이다.

정 답 2 재귀 호출의 무한 반복 코드를 작성하면, 파이썬은 어느 정도 반복하다가 자동 종료시킨다.

해 설 무한 반복을 마치고 되돌아가는 조건을 함께 사용합니다.

학습평가

Q2

10부터 100까지 합계를 구하는 코드를 (1)과 (2)에 채우시오.

```
def addNumber(num):
  if num <= 10:
    (      1      )
  return num + (       2       )

print (addNumber(100))
```

(1)

(2)

학습평가

Q1 Q2 Q3

Q2

10부터 100까지 합계를 구하는 코드를 (1)과 (2)에 채우시오.

```
def addNumber(num):
  if num <= 10:
    (      1      )
  return num + (       2       )

print (addNumber(100))
```

(1) return 10

(2) addNumber (num-1)

정 답 return 10 / addNumber(num-1)

해 설 return 문장에서 자기자신을 호출해야 합니다.

학습평가

Q3

팩토리얼을 구하는 코드를 (1)~(3)에 채우시오.

```
def factorial(num):
  if (     1     ):
    return 1
  retVal = (     2     )
  return (     3     )
```

(1)

(2)

(3)

학습평가

Q3

팩토리얼을 구하는 코드를 (1)~(3)에 채우시오.

```
def factorial(num):
  if (     1     ):
    return 1
  retVal = (     2     )
  return (     3     )
```

(1) num <= 1

(2) Factorial (num-1)

(3) num * retVal

정 답 num <= 1 / factorial(num-1) / num * retVal

해 설 팩토리얼을 구할 때 자기 자신을 호출해야 합니다.

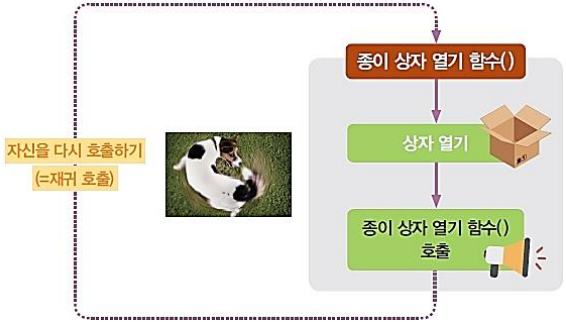
재귀 호출의 기본

재귀 호출(Recursion)

- 자신을 다시 호출하는 것

재귀 호출의 작동

- 상자를 반복해서 여는 과정을 재귀 호출 형태로 표현



재귀 호출의 기본

재귀 호출의 작동

- 일반적인 프로그램에서는 무한 반복을 마치고 되돌아가는 조건을 함께 사용함
- 반환 조건이 추가된 재귀 호출