5

트리 재귀와 반복
Tree Recursion & Iteration

한양대학교 ERICA
기초·융합교육원
프로그래밍개론
2015

컴퓨터공학과 교수
도경구

강의 목차

- 주제 : 트리 재귀와 반복

  1. 피보나찌 수열
  2. 시퀀스
  3. 조합
  4. 프로그래밍 과제 #4 : 슬라이드 퍼즐

5
트리 재귀와 반복
Tree Recursion & Iteration

5-1

피보나찌 수열
Fibonacci Sequence

한양대학교 ERICA
기초·융합교육원
프로그래밍개론
2015

컴퓨터공학과 교수
도경구

피보나찌 수열
Fibonacci Sequence

자연수의 수열로
이전 두 개의 수를 더하여 다음 수를 정하는 수열이다.

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, …

피보나찌 수
Fibonacci Number

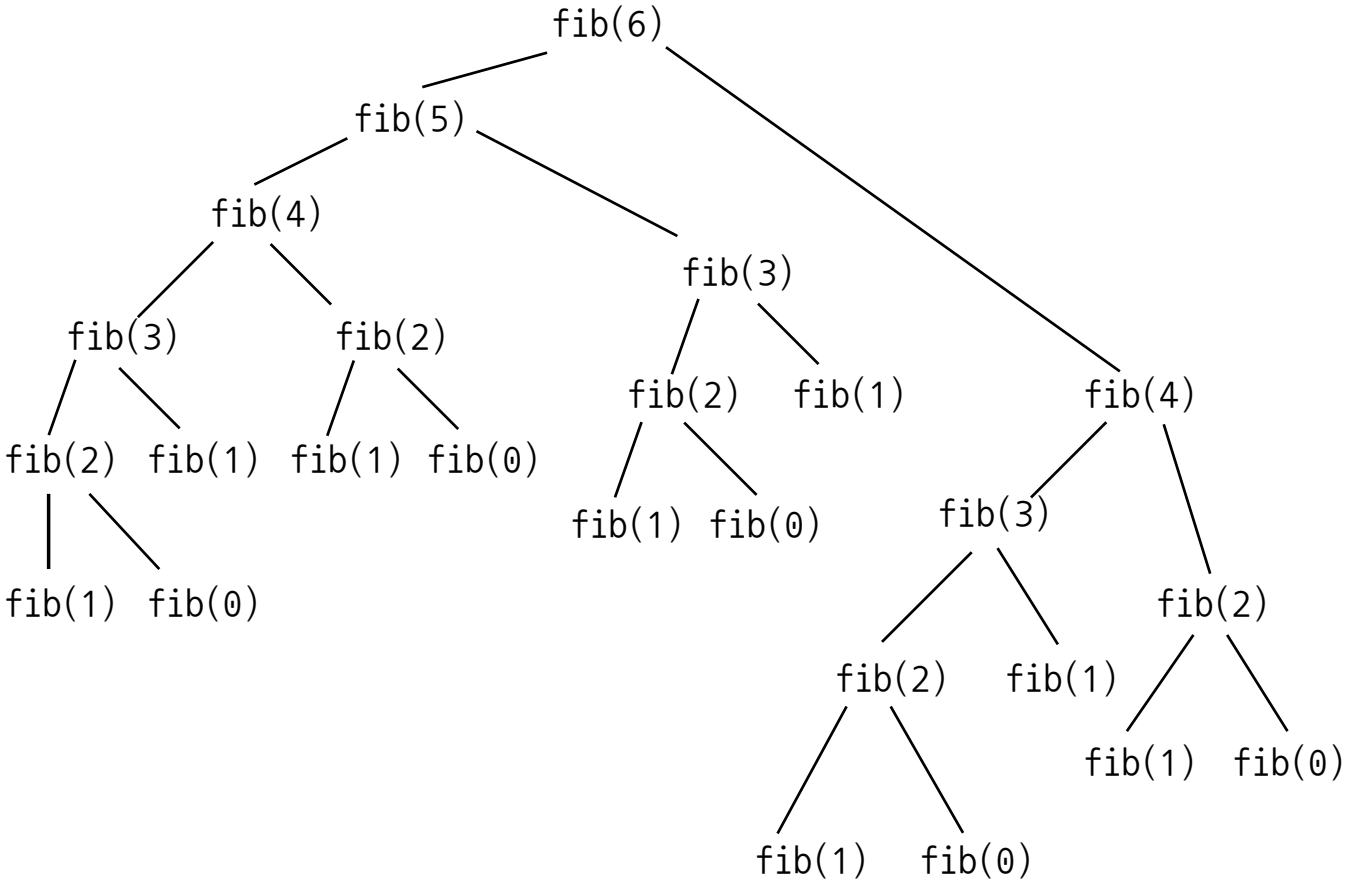
n번째 피보나찌 수는
자연수의 귀납구조를 이용하여
재귀로 정의할 수 있다.

$$Fib(n) = Fib(n-1) + Fib(n-2), \quad n > 1$$
$$Fib(1) = 1$$
$$Fib(0) = 0$$

```python
def fib(n):
    if n > 1:
        return fib(n-1) + fib(n-2)
    else:
        return n
```

실행추적

```
fib(6)
=> fib(5) + fib(4)
=> (fib(4) + fib(3)) + fib(4)
=> ((fib(3) + fib(2)) + fib(3)) + fib(4)
=> (((fib(2) + fib(1)) + fib(2)) + fib(3)) + fib(4)
=> ((((fib(1) + fib(0)) + fib(1)) + fib(2)) + fib(3)) + fib(4)
=> ((2 + fib(2)) + fib(3)) + fib(4)
=> ((2 + (fib(1) + fib(0))) + fib(3)) + fib(4)
=> (3 + fib(3)) + fib(4)
=> (3 + (fib(2) + fib(1))) + fib(4)
=> (3 + ((fib(1) + fib(0)) + fib(1))) + fib(4)
=> 5 + fib(4)
=> 5 + (fib(3) + fib(2))
=> 5 + ((fib(2) + fib(1)) + fib(2))
=> 5 + (((fib(1) + fib(0)) + fib(1)) + fib(2))
=> 5 + (2 + fib(2))
=> 5 + (2 + (fib(1) + fib(0)))
=> 8
```



피보나찌 수
Fibonacci Number

$$Fib(n) = Fib(n-1) + Fib(n-2), \quad n > 1$$
$$Fib(1) = 1$$
$$Fib(0) = 0$$

상향식(Bottom-up)으로 계산하기

n	0	1	2	3	4	5	6	7	8	9	10	…
Fib(n)	0	1	1	2	3	5	8	13	21	34	55	…

피보나찌 수
Fibonacci Number

```
Fib(n) = Fib(n-1) + Fib(n-2),  n > 1
Fib(1) = 1
Fib(0) = 0
```

상향식(Bottom-up)으로 계산하기

n	0	1	2	3	4	5	6	7	8	9	10	···
Fib(n-1)		0	1	1	2	3	5	8	13	21	34	55
Fib(n)	0	1	1	2	3	5	8	13	21	34	55	···

피보나찌 수
Fibonacci Number

```
Fib(n) = Fib(n-1) + Fib(n-2),  n > 1
Fib(1) = 1
Fib(0) = 0
```

상향식(Bottom-up)으로 계산하기

k	0	1	2	3	4	5	6	7	8	9	10	···
old		0	1	1	2	3	5	8	13	21	34	55
new	0	1	1	2	3	5	8	13	21	34	55	···

```
def fib2(n):
    k = 1
    old, new = 0, 1
    while k < n:
        k = k + 1
        old, new = new, old + new
    return new
```

k	0	1	2	3	4	5	6	7	8	9	10	⋯
old		0	1	1	2	3	5	8	13	21	34	55
new	0	1	1	2	3	5	8	13	21	34	55	⋯

5
트리 재귀와 반복
Tree Recursion & Iteration

5-2

시퀀스
Sequence

한양대학교 ERICA
기초·융합교육원
프로그래밍개론
2015

컴퓨터공학과 교수
도경구

시퀀스
Sequence

데이터가 일렬로 나열되어 모여있는 구조

참고

The Python Standard Library
4.6 Sequence Types

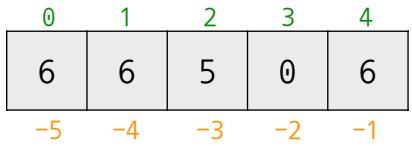
시퀀스의 종류
Sequence

종류	문법	수정가능성
리스트 list	[5, 6, 7, 8, 9]	수정가능 mutable
튜플 tuple	(5,True,"한양대학교")	수정불가능 immutable
정수범위 range	range(i,j,k)	수정불가능 immutable
문자열 string	"한양대학교" '한양대학교'	수정불가능 immutable

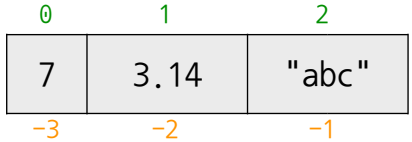
시퀀스
Sequence

내부 원소는 위치번호(index)가 0부터 차례로 지정되어 있음

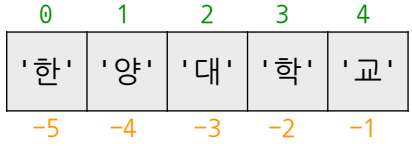
[6, 6, 5, 0, 6]



(7, 3.14, "abc")



'한양대학교'



시퀀스의 종류
Sequence

종류	문법	수정가능성
리스트 list	[5, 6, 7, 8, 9]	수정가능 mutable
튜플 tuple	(5,True,"한양대학교")	수정불가능 immutable
정수범위 range	range(i,j,k)	수정불가능 immutable
문자열 string	"한양대학교" '한양대학교'	수정불가능 immutable

시퀀스 연산

연산	의미
x in s	x가 s에 있으면 True, 없으면 False
x not in s	x가 s에 없으면 True, 있으면 False
s + t	s와 t 나란히 붙이기
s * n	s를 n번 나란히 붙이기

시퀀스 연산

연산	의미
s[i]	0부터 세어서 i째 원소
s[i:j]	위치번호 i부터 j까지 s의 조각 (j째 원소 제외)
len(s)	s의 길이
min(s)	s에서 가장 작은 원소
max(s)	s에서 가장 큰 원소

시퀀스 연산

연산	의미
s.index(x)	s에서 처음 나타나는 x의 위치번호
s.index(x,i)	s의 위치번호 i에서 시작하여 처음 나타나는 x의 위치번호
s.index(x,i,j)	s의 위치번호 i부터 j 앞까지 사이에서 처음 나타나는 x의 위치번호
s.count(x)	s에서 x가 나타나는 총 횟수

정수범위 시퀀스
range Sequence

range(i)	0, 1, 2, ⋯, i-1
range(i,j)	i, i+1, i+2, ⋯, j-1
range(i,j,k)	i, i+k, i+k*2, ⋯, i+k*?(< j)

for 반복문

문법 `for` 〈변수〉 `in` 〈시퀀스〉:
〈몸체〉

의미 〈변수〉를 x이라 하고
〈시퀀스〉를 s라고 하면,
다음을 차례로 실행한다.
s의 첫째 원소를 x로 지정하고 〈몸체〉를 실행하고,
s의 둘째 원소를 x로 지정하고 〈몸체〉를 실행하고,
s의 셋째 원소를 x로 지정하고 〈몸체〉를 실행하고,
…
s의 마지막 원소를 x로 지정하고 〈몸체〉를 실행한다.

```
def fib2(n):
    k = 1
    old, new = 0, 1
    while k < n:
        k = k + 1
        old, new = new, old + new
    return new
```

```
def fib3(n):
    old, new = 0, 1
    for k in range(1,n):
        old, new = new, old + new
    return new
```

```
def fib2(n):
    k = 1
    old, new = 0, 1
    while k < n:
        k = k + 1
        old, new = new, old + new
    return new
```

```
def fib3(n):
    old, new = 0, 1
    for _ in range(1,n):
        old, new = new, old + new
    return new
```

5
트리 재귀와 반복
Tree Recursion & Iteration

5-3

조합
Combination

한양대학교 ERICA
기초·융합교육원
프로그래밍개론
2015

컴퓨터공학과 교수
도경구

조합
Combination

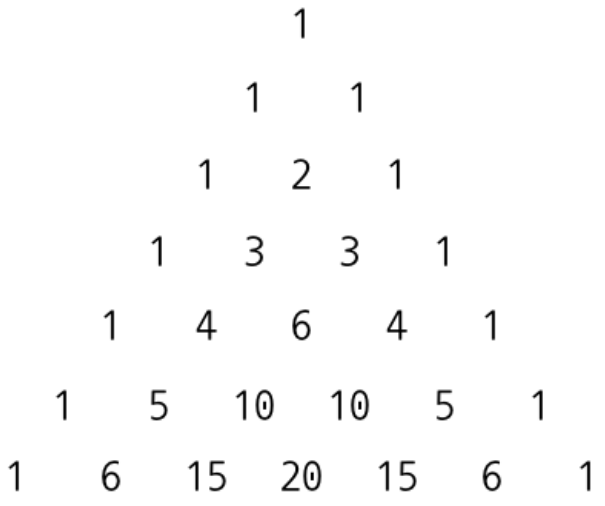
n개에서 순서에 상관없이 r개를 뽑는 가지수

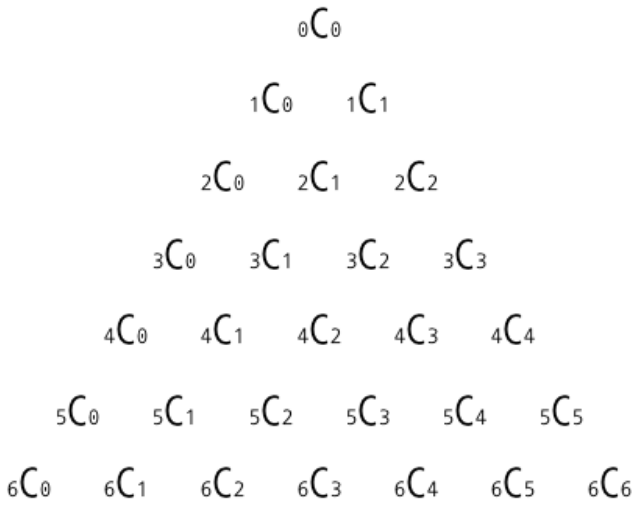
$$ {}_nC_r = {}_{n-1}C_{r-1} + {}_{n-1}C_r \qquad (r \neq 0 \text{ and } r \neq n)$$

$$ {}_nC_0 = 1 \qquad {}_nC_n = 1$$

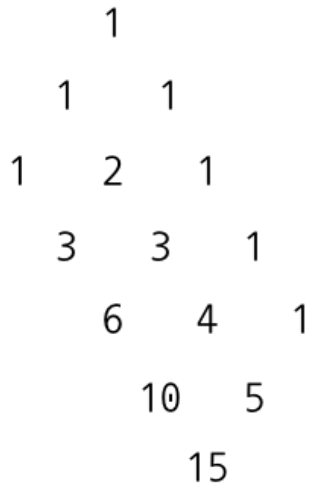
```
def comb(n,r):
    if r != 0 and r != n:
        return comb(n-1,r-1) + comb(n-1,r)
    else:
        return 1
```

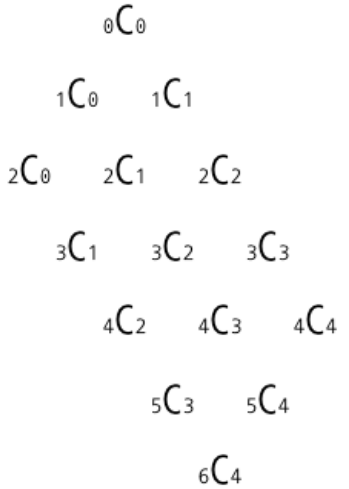
파스칼 삼각형
Pascal's Triangle



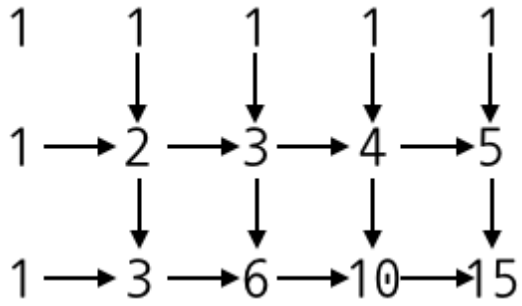


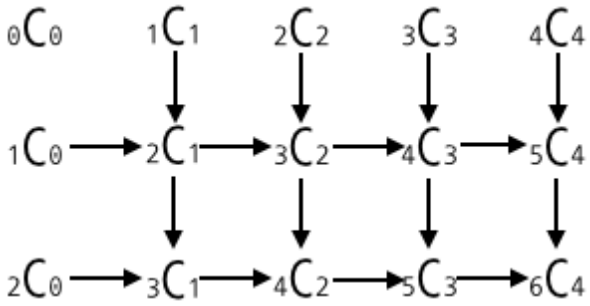
파스칼 삼각형
Pascal's Triangle

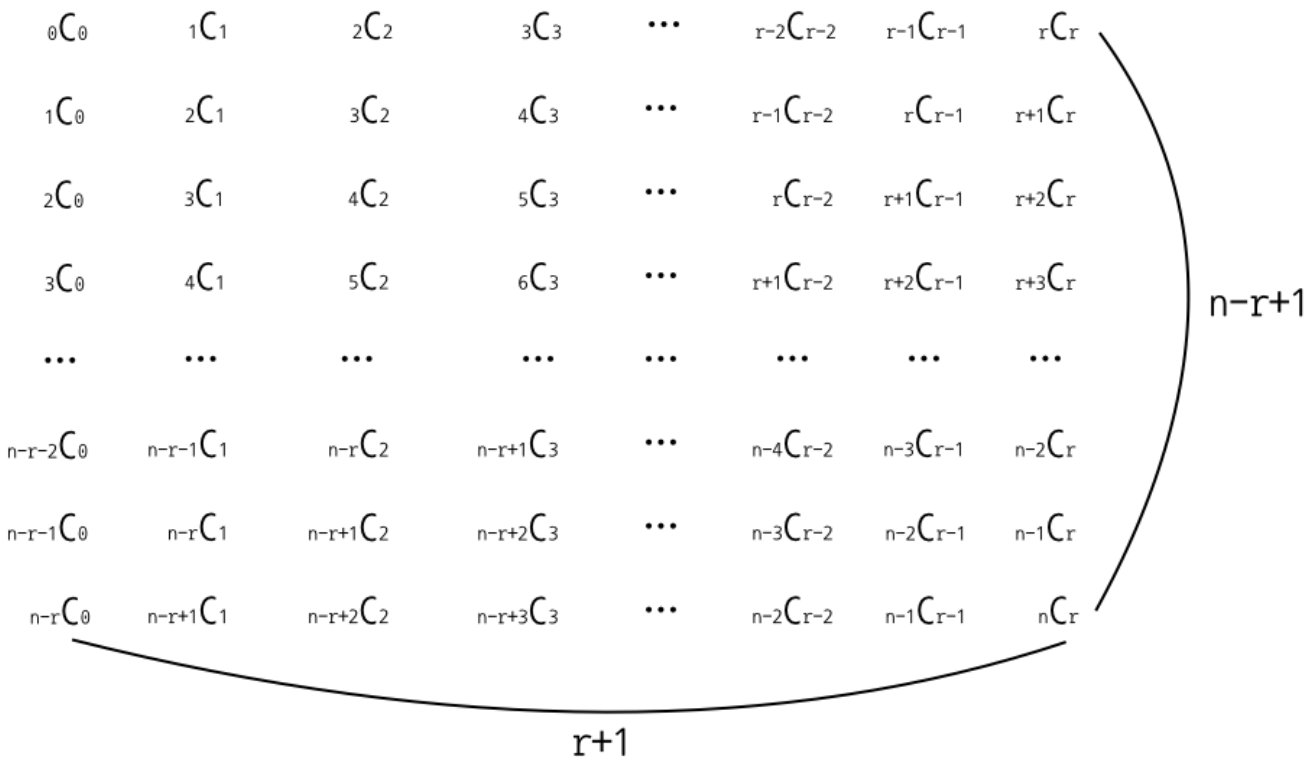




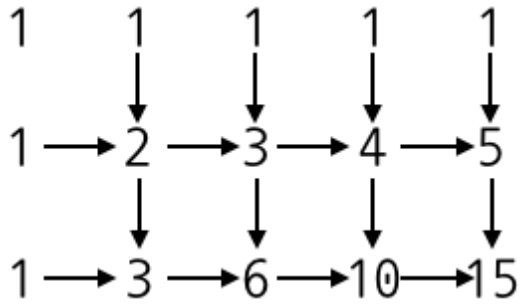
파스칼 삼각형
Pascal's Triangle

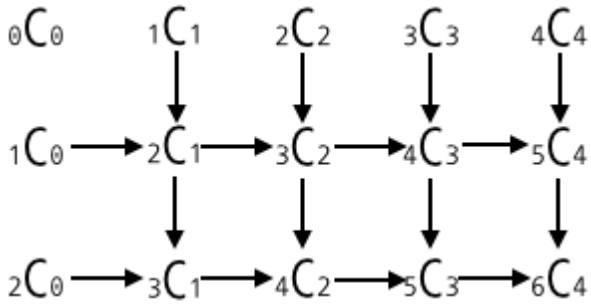






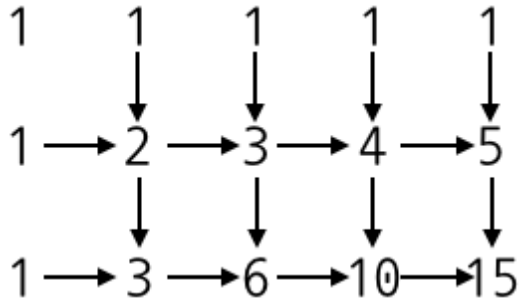
파스칼 삼각형
Pascal's Triangle

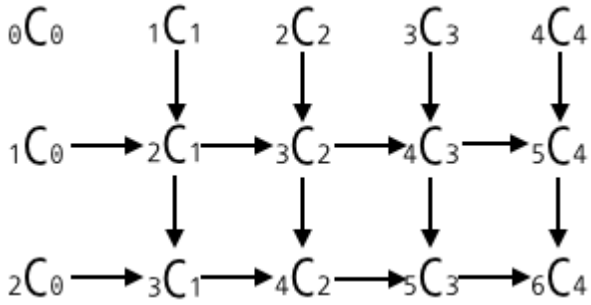





```
[[1, 1, 1, 1, 1],
 [1, 2, 3, 4, 5],
 [1, 3, 6, 10, 15]]
```


파스칼 삼각형
Pascal's Triangle






```
[[1, 1, 1, 1, 1], [1, 2, 3, 4, 5], [1, 3, 6, 10, 15]]
```

```
def pascal(n,r):
    table = [[]]*(n-r+1)
    table[0] = [1]*(r+1)
    for i in range(1,n-r+1):
        table[i] = [1]
    for i in range(1,n-r+1):
        for j in range(1,r+1):
            newvalue = table[i][j-1] + table[i-1][j]
            table[i].append(newvalue)
    return table[n-r][r]
```

```
table = [[],
         [],
         [],
         …,
         []]
```

(n-r+1)개의 빈 리스트

```
def pascal(n,r):
    table = [[]]*(n-r+1)
    table[0] = [1]*(r+1)
    for i in range(1,n-r+1):
        table[i] = [1]
    for i in range(1,n-r+1):
        for j in range(1,r+1):
            newvalue = table[i][j-1] + table[i-1][j]
            table[i].append(newvalue)
    return table[n-r][r]
```

```
table = [[1, 1, …, 1],
         [],
         [],
         …,
         []]
```

맨 위의 행을 r+1개 원소를 모두 1로 초기화

```
def pascal(n,r):
    table = [[]]*(n-r+1)
    table[0] = [1]*(r+1)
    for i in range(1,n-r+1):
        table[i] = [1]
    for i in range(1,n-r+1):
        for j in range(1,r+1):
            newvalue = table[i][j-1] + table[i-1][j]
            table[i].append(newvalue)
    return table[n-r][r]
```

```
table = [[1, 1, ···, 1],
         [1],
         [1],
         ···,
         [1]]
```

맨 왼쪽 열을 모두 1로 초기화

```
def pascal(n,r):
    table = [[]]*(n-r+1)
    table[0] = [1]*(r+1)
    for i in range(1,n-r+1):
        table[i] = [1]
    for i in range(1,n-r+1):
        for j in range(1,r+1):
            newvalue = table[i][j-1] + table[i-1][j]
            table[i].append(newvalue)
    return table[n-r][r]
```

```
table = [[1, 1, ···, 1],
         [1],
         [1],
         ···,
         [1]]
```

왼쪽 값과 윗 값을 더하여
새 값을 결정

```
def pascal(n,r):
    table = [[]]*(n-r+1)
    table[0] = [1]*(r+1)
    for i in range(1,n-r+1):
        table[i] = [1]
    for i in range(1,n-r+1):
        for j in range(1,r+1):
            newvalue = table[i][j-1] + table[i-1][j]
            table[i].append(newvalue)
    return table[n-r][r]
```

새 값을 리스트 뒤에 붙임

```
table = [[1, 1, ⋯, 1],
         [1],
         [1],
         ⋯,
         [1]]
```

```
def pascal(n,r):
    table = [[]]*(n-r+1)
    table[0] = [1]*(r+1)
    for i in range(1,n-r+1):
        table[i] = [1]
    for i in range(1,n-r+1):
        for j in range(1,r+1):
            newvalue = table[i][j-1] + table[i-1][j]
            table[i].append(newvalue)
    return table[n-r][r]
```

행 하나 채우는 내부 반복문

```
table = [[1, 1, ⋯, 1],
         [1],
         [1],
         ⋯,
         [1]]
```

```
def pascal(n,r):
    table = [[]]*(n-r+1)
    table[0] = [1]*(r+1)
    for i in range(1,n-r+1):
        table[i] = [1]
    for i in range(1,n-r+1):
        for j in range(1,r+1):
            newvalue = table[i][j-1] + table[i-1][j]
            table[i].append(newvalue)
    return table[n-r][r]
```

테이블 전체를 채우는
외부 반복문

```
table = [[1, 1, …, 1],
         [1],
         [1],
         …,
         [1]]
```

```
def pascal(n,r):
    table = [[]]*(n-r+1)
    table[0] = [1]*(r+1)
    for i in range(1,n-r+1):
        table[i] = [1]
    for i in range(1,n-r+1):
        for j in range(1,r+1):
            newvalue = table[i][j-1] + table[i-1][j]
            table[i].append(newvalue)
    return table[n-r][r]
```

테이블 맨 아래 행의
오른쪽 끝 값이 내주는 답

```
table = [[1, 1, …, 1],
         [1],
         [1],
         …,
         [1]]
```

5
트리 재귀와 반복
Tree Recursion & Iteration

5-4
프로그래밍 과제 #5

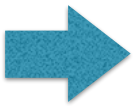
슬라이드 퍼즐

한양대학교 ERICA
기초·융합교육원
프로그래밍개론
2015

컴퓨터공학과 교수
도경구

게임 보드

	0	1	2	3
0		15	14	13
1	12	11	10	9
2	8	7	6	5
3	4	3	2	1



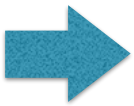
해답 보드

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12
3	13	14	15	

```python
def sliding_puzzle():
    board = create_init_board()
    goal = set_goal_board()
    empty = (0,0)
    while True:
        print_board(board)
        if board == goal:
            print("Congratulations!")
            break
        num = get_number()
        if num == 0:
            break
        pos = find_position(num,board)
        (empty,board) = move(pos,empty,board)
    print("Please come again.")
```

게임 보드

	0	1	2	3
0		15	14	13
1	12	11	10	9
2	8	7	6	5
3	4	3	2	1



해답 보드

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12
3	13	14	15	

```python
def create_init_board():
    return [[0,15,14,13],
            [12,11,10,9],
            [8,7,6,5],
            [4,3,2,1]]
```

```python
def set_goal_board():
    return [[1,2,3,4],
            [5,6,7,8],
            [9,10,11,12],
            [13,14,15,0]]
```

```
def sliding_puzzle():
    board = create_init_board()
    goal = set_goal_board()
    empty = (0,0)
    while True:
        print_board(board)
        if board == goal:
            print("Congratulations!")
            break
        num = get_number()
        if num == 0:
            break
        pos = find_position(num,board)
        (empty,board) = move(pos,empty,board)
    print("Please come again.")
```

게임 보드

	0	1	2	3
0		15	14	13
1	12	11	10	9
2	8	7	6	5
3	4	3	2	1

```
empty = (0,0)
```

```
def sliding_puzzle():
    board = create_init_board()
    goal = set_goal_board()
    empty = (0,0)
    while True:
        print_board(board)
        if board == goal:
            print("Congratulations!")
            break
        num = get_number()
        if num == 0:
            break
        pos = find_position(num,board)
        (empty,board) = move(pos,empty,board)
    print("Please come again.")
```