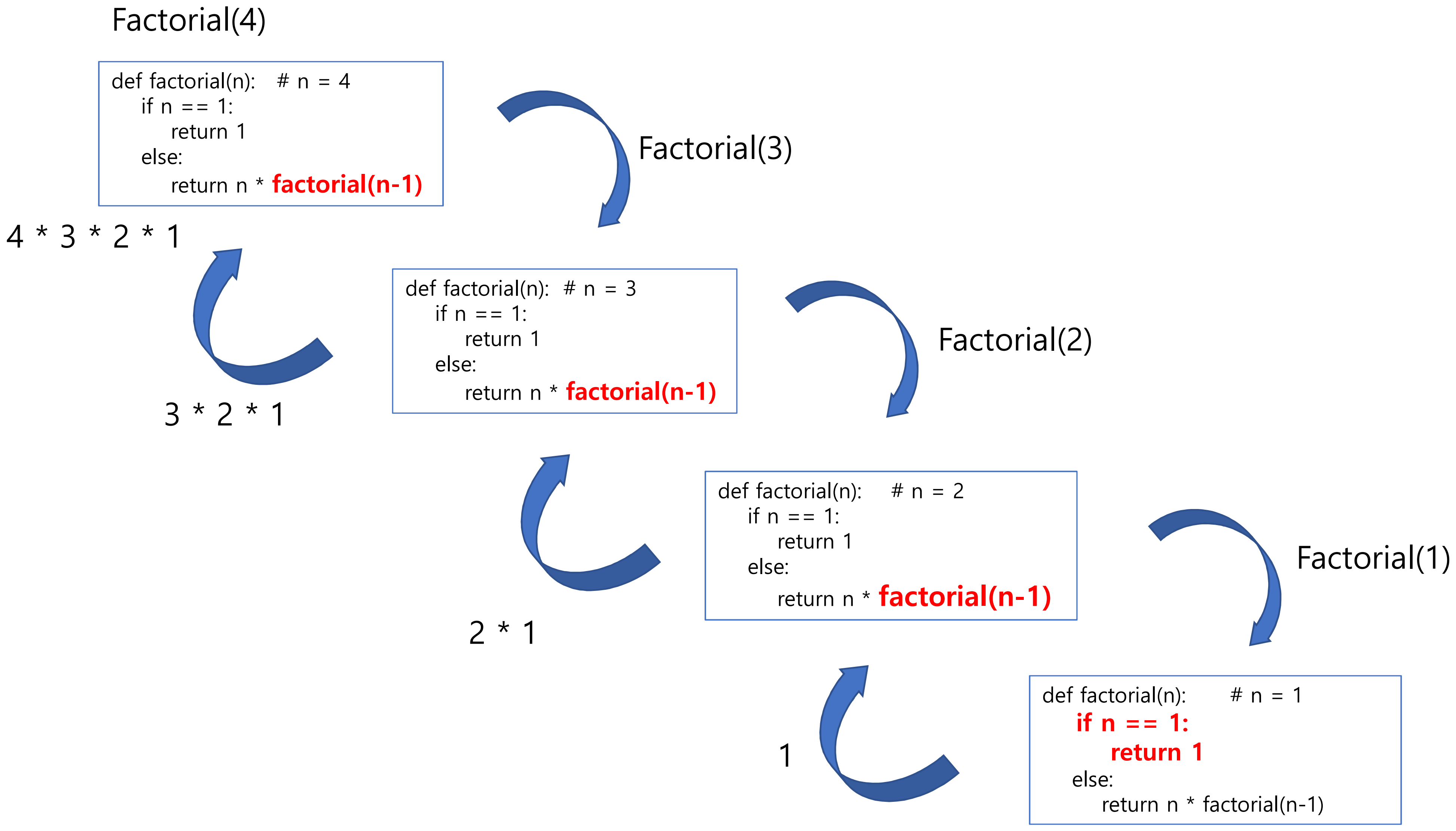


```
def factorial(n):   # n = 4
    if n == 1:
        return 1
    else:
        return n * factorial(n-1)
```

```
def factorial(n):  # n = 3
    if n == 1:
        return 1
    else:
        return n * factorial(n-1)
```

```
def factorial(n):     # n = 2
    if n == 1:
        return 1
    else:
        return n * factorial(n-1)
```

```
def factorial(n):        # n = 1
    if n == 1:
        return 1
    else:
        return n * factorial(n-1)
```

Stack 구조

```
def factorial(n):       # n = 1
    if n == 1:
        return 1
    else:
        return n * factorial(n-1)
```

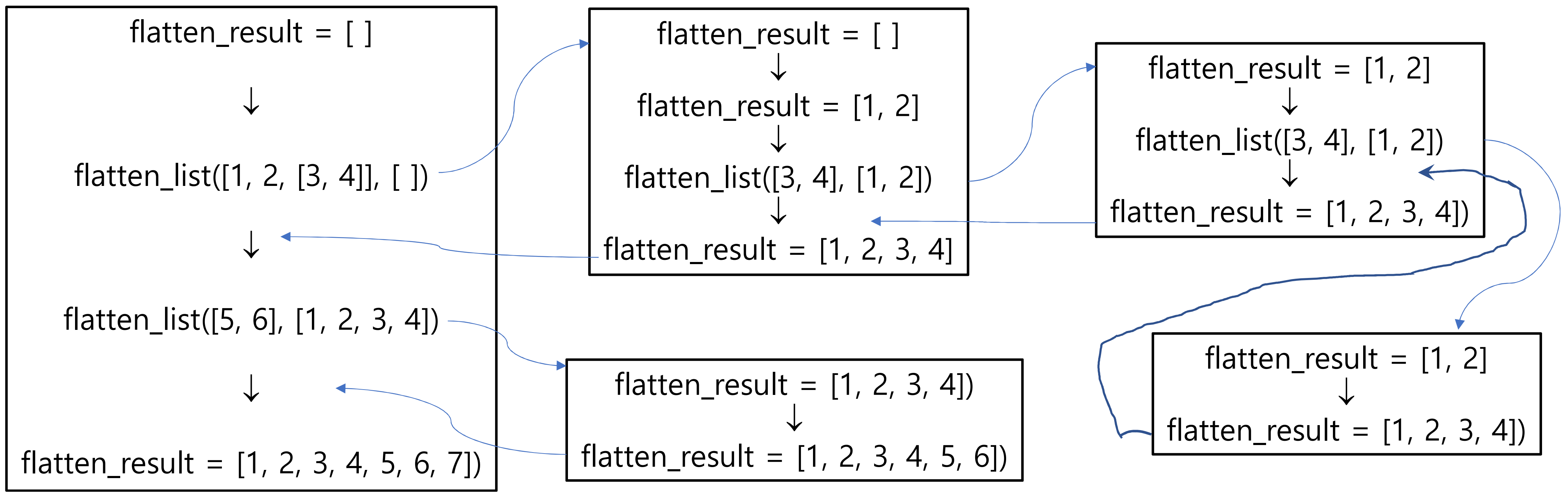
```
def factorial(n):     # n = 2
    if n == 1:
        return 1
    else:
        return n * factorial(n-1)
```

```
def factorial(n):   # n = 3
    if n == 1:
        return 1
    else:
        return n * factorial(n-1)
```

```
def factorial(n):    # n = 4
    if n == 1:
        return 1
    else:
        return n * factorial(n-1)
```

Flattening List

- Flattening([[1, 2, [3, 4]], [5, 6], 7])
- flattening_result = [Flattening([1, 2, [3, 4]]), Flattening([5, 6]), 7]
- flattening_result = [1, 2, Flattening([3, 4])], Flattening([5, 6]), 7]
- flattening_result = [1, 2, 3, 4, Flattening([5, 6]), 7]
- flattening_result = [1, 2, 3, 4, 5, 6, 7]


flatten_result = [ ]
↓
flatten_list([1, 2, [3, 4]], [ ])
↓
flatten_list([5, 6], [1, 2, 3, 4])
↓
flatten_result = [1, 2, 3, 4, 5, 6, 7])
flatten_result = [ ]
↓
flatten_result = [1, 2]
↓
flatten_list([3, 4], [1, 2])
↓
flatten_result = [1, 2, 3, 4]
flatten_result = [1, 2]
↓
flatten_list([3, 4], [1, 2])
↓
flatten_result = [1, 2, 3, 4])
flatten_result = [1, 2]
↓
flatten_result = [1, 2, 3, 4])
flatten_result = [1, 2, 3, 4])
↓
flatten_result = [1, 2, 3, 4, 5, 6])

Flattening Dictionary

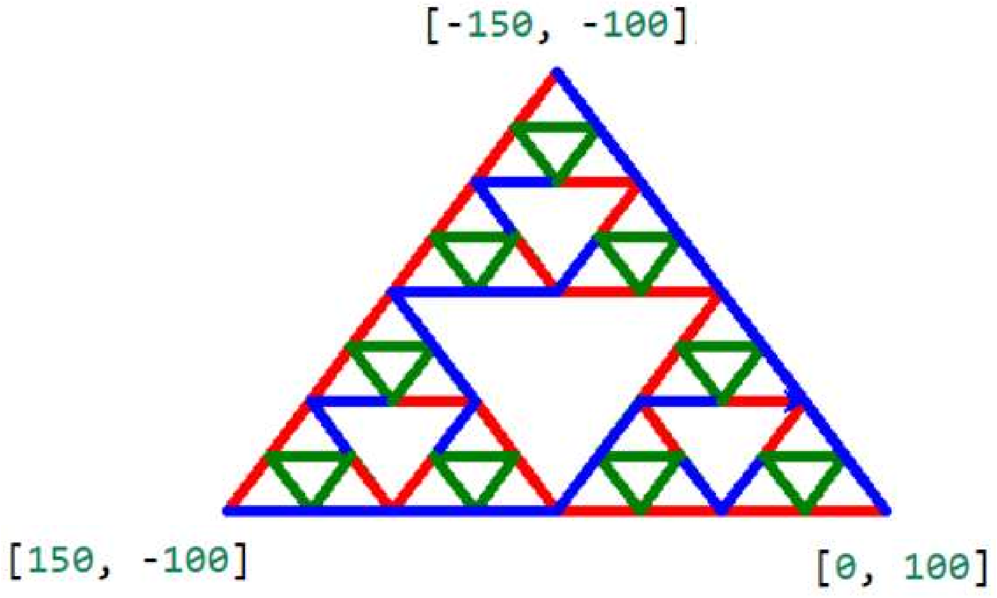
- {'a': 1, 'b': {'x': 2, 'y': 3}, 'c': 4} ==> {'a': 1, 'b.x': 2, 'b.y': 3, 'c': 4}
- Flatten({'a': 1, 'b': {'x': 2, 'y': 3}, 'c': 4}**, None**)
- Flatten({'a': 1, Flatten({'x': 2, 'y': 3}**, ‘b’**), 'c': 4})
- Flatten({'a': 1, ‘b.x': 2, ‘b.y’: 3, 'c': 4})
- {'a': 1, ‘b.x': 2, ‘b.y’: 3, 'c': 4}

turtle

- Turtle 객체의 최초 위치 (0, 0)
- 최초 방향 : right
- 최초 pen 상태 : down

- 최초 pen methods
  - penup()
  - pendown()
  - pensize()

- 주요 methods
  - forward(distance), right(angle), left(angle), goto(x, y), circle(radius)
  - speed(s)
  - done() - 일시정지



```
def drawTurtle(points):
    t.penup()
    t.setpos(points[0][0], points[0][1])
    t.pendown()
    t.color('red')
    t.goto(points[1][0], points[1][1])
    t.color('green')
    t.goto(points[2][0], points[2][1])
    t.color('blue')
    t.goto(points[0][0], points[0][1])
```

```
def getMid(p1, p2):
    return ((p1[0]+p2[0])/2, (p1[1]+p2[1])/2)

def Sierpinski(points, n):
    drawTurtle(points)
    if n > 0:
        Sierpinski([points[0],
                        getMid(points[0], points[1]),
                        getMid(points[0], points[2])], n-1)
        Sierpinski([points[1],
                        getMid(points[1], points[0]),
                        getMid(points[1], points[2])], n-1)
        Sierpinski([points[2],
                        getMid(points[2], points[1]),
                        getMid(points[2], points[0])], n-1)

Sierpinski([[0, 100], [-150, -100], [150, -100]], 3)
```