UNISTUDY

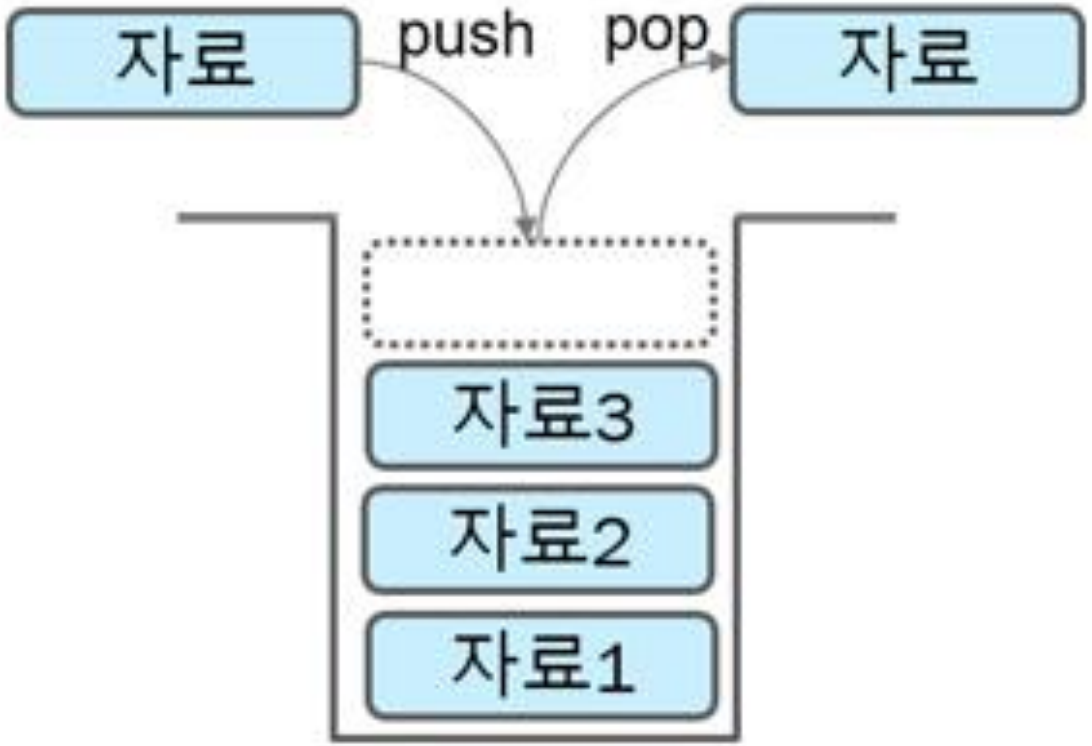
파이썬

19강. 재귀함수

1. 재귀함수

- 재귀함수(Recursive function)는 함수 내부에서 자신의 함수를 반복적으로 호출하는 함수를 의미한 다. 재귀함수는 반복적으로 호출하기 때문에 반드시 함수 내에는 반복을 탈출(exit)은 조건이 필수이며, 반복적으로 변수를 조금씩 변경하여 연산을 수행하는 알고리즘에서 이용된다.
- 예를 들면 1에서 n 까지 1씩 증가하는 수열의 합, 팩토리얼(Factorial) 계산 등의 사례를 들 수 있다.
- 한편 재귀함수에서 재귀호출이 발생할 때 생성된 자료는 스택(Stack)이라는 메모리 영역에 저장된다.
- 스택 영역에 자료가 입력(push)되고 출력(pop)되는 과정을 나타낸다.

1. 재귀함수



2. 카운트

- 1에서 n까지 정수를 카운트(count)하는 과정을 살펴보면 변수의 값을 반복적으로 1씩 증가하고(조금 씩 변경) 이를 출력한다. 이러한 연산과정은 재귀함수를 적용하여 문제를 해결할 수 있다.

chapter05.lecture.step04_inner_func.py

```
# (1) 재귀함수 정의 : 1~n 카운트
def Counter(n):
    if n == 0 :
        return 0 # 종료 조건
    else :
        Counter(n-1) # 재귀호출

# (2) 함수 호출1
print('n=0 : ', Counter(0))

# (3) 함수 호출2
Counter(5)
```

Python Console

```
n=0 :  0

1 2 3 4 5
```

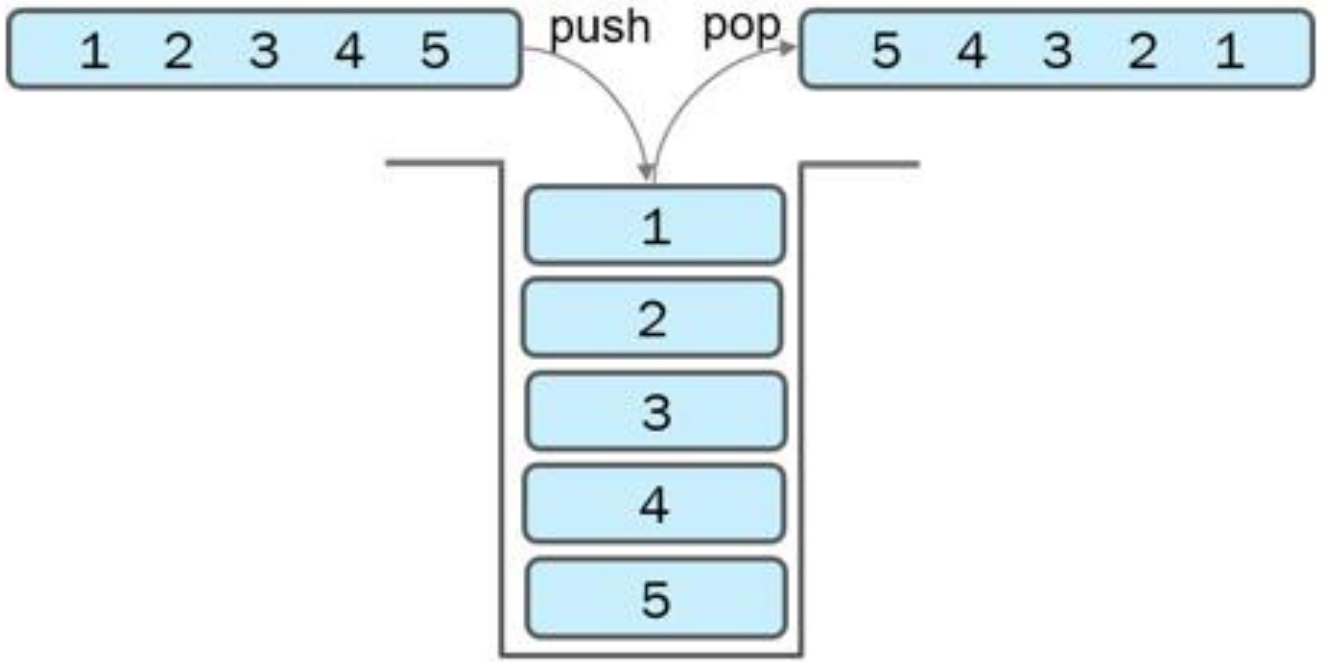
2. 카운트

- 해설 숫자 카운트 예
- # (1) 재귀함수 정의
- 매개변수 n을 갖는 Counter() 함수는 종료조건 'n==0'을 포함하고 있다. 실인수 0으로 함수를 호출하면 재귀호출 없이 곧바로 종료 조건에 의해서 함수는 종료된다. 하지만 실인수가 1이상이면 재귀호출이 발생한다.
- # (2) 함수 호출1
- 실인수 0으로 재귀함수를 호출하면 종료 조건에 의해서 0이 반환되고 출력된다.
- # (3) 함수 호출2
- 실인수 5로 재귀함수를 호출하는 과정을 단계로 알아본다.
- 단계1: 실인수 5로 재귀함수를 호출 하면 종료 조건이 거짓이므로 else 영역의 재귀호출이 발생 한다. 이때 최초로 넘어온 실인수 n=5는 스택(stack) 영역에 저장된다.
- 단계2: 재귀호출은 (n-1) 수식에 의해서 Counter(4) 형식으로 자신의 함수를 호출한다. 이때 n=4 가 스택에 저장된다.

2. 카운트

- 단계3: 두 번째 재귀호출은 (n-1) 수식에 의해서 Counter(3) 형식으로 자신의 함수를 호출한다. 이때 n=3이 스택에 저장된다.
- 단계4: 세 번째 재귀호출은 (n-1) 수식에 의해서 Counter(2) 형식으로 자신의 함수를 호출한다. 이때 n=2가 스택에 저장된다.
- 단계5: 네 번째 재귀호출은 (n-1) 수식에 의해서 Counter(1) 형식으로 자신의 함수를 호출한다. 이때 n=1이 스택에 저장된다.
- 단계6: 다섯 번째 재귀호출은 (n-1) 수식에 의해서 n=0이 되기 때문에 종료 조건(n==0)이 참(True) 이 된다. 재귀호출 과정에서 종료 조건이 참이면 지금까지 스택에 누적된 값을 역순으로 꺼내서(pop) 함수 호출 부분으로 반환한다.

2. 카운트



3. 누적합

- 1에서 n까지 정수를 카운트 하는 과정을 앞에서 살펴보았다. 이렇게 카운트한 값을 누적하여 반환하는 경우에도 재귀함수를 적용하여 문제를 해결할 수 있다.

chapter05.lecture.step04_inner_func.py

```
# (1) 재귀함수 정의 : 1~n 누적합(1+2+3+4+5=15)
def Adder(n):
    if n == 1 : # 종료 조건
        return 1
    else :
        result = n + Adder(n-1) # 재귀호출

        print(n, end = ' ')  # (4) 스택 영역  2 3 4 5
        return result

# (2) 함수 호출1
print('n=1 :', Adder(1))

# (3) 함수 호출2
print('\nn=5 :', Adder(5))
```

Python Console

```
n=1 : 1

n=5 : 15
```

3. 누적합

- 해설 1~n 정수 누적합 예
- # (1) 재귀함수 정의
- 매개변수 n을 갖는 Adder() 함수는 종료조건 'n==1'을 포함하고 있다. 실인수 1로 함수를 호출하면 재귀호출 없이 곧바로 종료 조건에 의해서 함수는 종료된다. 하지만 실인수가 2이상의 정수이면 재귀호 출이 발생 한다.
- # (2) 함수 호출1
- 실인수 1로 재귀함수를 호출하면 종료 조건에 의해서 1이 반환되고 출력된다.
- # (3) 함수 호출2
- 실인수 5로 재귀함수를 호출하는 과정을 단계로 알아본다.
- 단계1: 실인수 5로 재귀함수를 호출 하면 종료 조건이 거짓이므로 else 영역의 재귀호출이 발생 한다. 이때 최초로 넘어온 실인수 n=5는 스택(stack) 영역에 저장된다.

3. 누적합

- 단계2: 재귀호출은 (n-1) 수식에 의해서 Counter(4) 형식으로 자신의 함수를 호출한다. 이때 n=4
- 가 스택에 저장된다.
- 단계3: 두 번째 재귀호출은 (n-1) 수식에 의해서 Counter(3) 형식으로 자신의 함수를 호출한다. 이때 n=3이 스택에 저장된다.
- 단계4: 세 번째 재귀호출은 (n-1) 수식에 의해서 Counter(2) 형식으로 자신의 함수를 호출한다. 이때 n=2가 스택에 저장된다.
- 단계5: 네 번째 재귀호출은 (n-1) 수식에 의해서 n=1이 되기 때문에 종료 조건(n==1)이 참(True)이 된다. 재귀호출 과정에서 종료 조건이 참이면 return 1의 값이 재귀호출 부분으로 반환되고, 스택 영역의 값을 역순으로 꺼내서(pop) ‘result = n + Adder(n-1)’ 문장으로 누적합이 계산된다.

3. 누적합

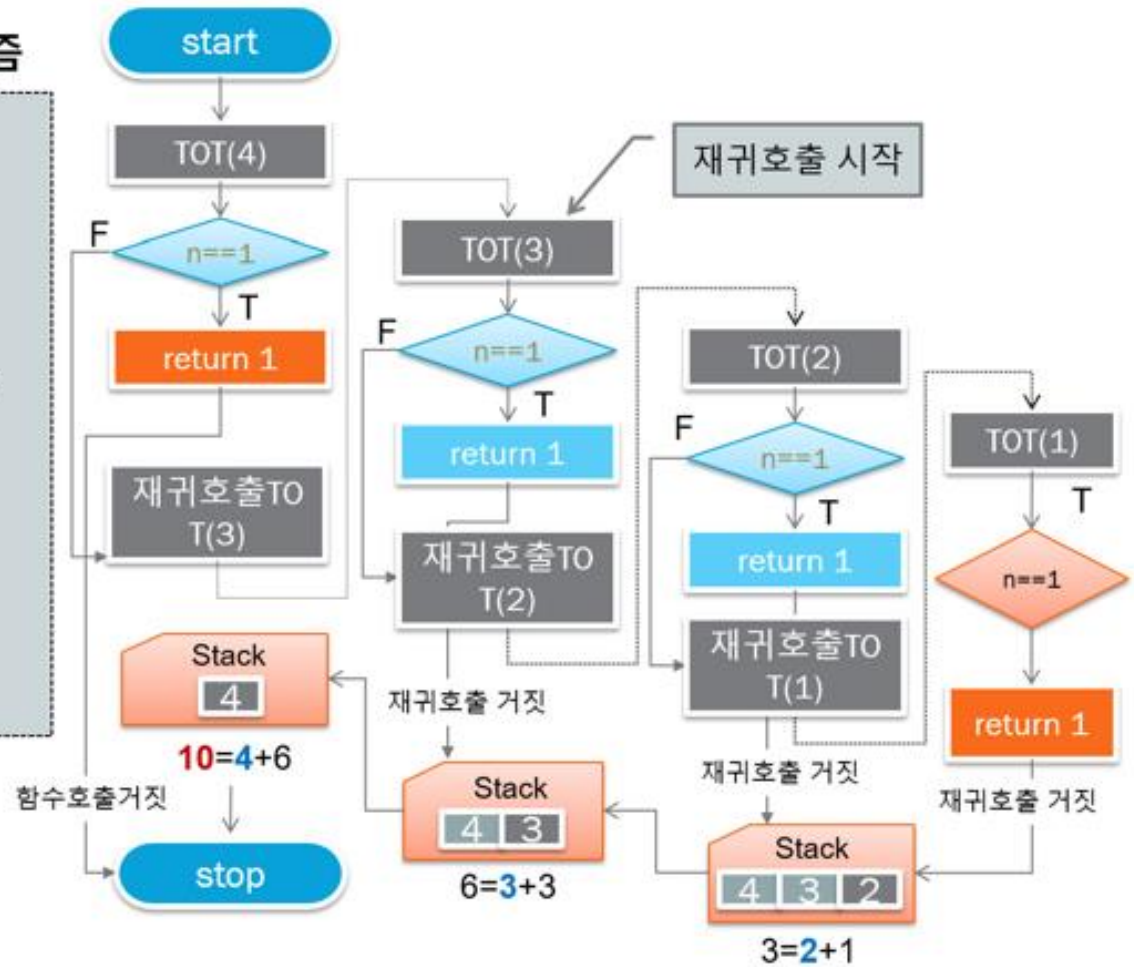
- 위 5단계가 수행되는 과정에서 누적합을 계산하는 'result = n + Adder(n-1)' 명령문의 수행과 정을 보면 재귀호출이 먼저 수행된 후 종료 조건이 참이면 그 때 스택의 값을 역순으로 꺼내서 나중에 result에 누적 연산이 수행되는 것을 볼 수 있다.
- 재귀호출 알고리즘의 수행과정을 나 타내고 있다.

3. 누적합

● 재귀호출 알고리즘

```
def TOT(n):
    if n == 1 :
        return 1
    else :
        tot = TOT(n-1) + n
        return tot

tot = TOT(4) # start
print('덧셈 결과 =', tot)
# 덧셈 결과 = 10
```



THANK YOU