

2024-1

기초 스터디

7. 함수, 재귀

목차

1. 함수
2. 람다 함수
3. 재귀

함수

- 반복 사용되는 **코드 집합**
- 어떤 코드 집합이 수행하는 한가지 기능이 **반복 사용**될 때 그 코드 집합에 이름을 붙여 하나의 함수로 만들 수 있다.

함수 호출

- 함수이름(인자값, 인자값, ...)

```
1 print("hello")
2 a = input()
3 b = len("hello")
4 c = map(int, input().split())
```

함수 선언

```
1 def function_name():  
2     function_code1  
3     function_code2  
4     ...  
5     (return) # return 은 없어도 됨
```

- **def** 키워드를 사용한다.
- 함수 내부 코드는 같은 들여쓰기를 유지한다.
- 함수를 빠져 나올 때, 값을 돌려줄 때는 **return**을 사용한다.

return

- 함수의 실행을 종료하고, 값을 돌려주는 구문
- 함수의 실행을 종료하는 목적만으로도 쓸 수 있다.

```
1 def function(x, y):  
2     if x > y:  
3         return  
4  
5     print(x+y)
```

```
1 def function(x, y):  
2     if x > y:  
3         return x  
4  
5     return y
```

함수 선언

```
1 def add(a, b):  
2     return a + b  
3  
4 a = add(1, 2) # a = 3
```

- 함수 안에서 사용할 값을 **매개변수(parameter)**를 통해 전달받을 수 있다.
- 매개변수를 통해 전달받는 값을 **인자(argument)** 라고 한다.

디폴트 인자 값

```
1 def function(a, b=1, c=1):  
2     return a + b + c  
3  
4 print(function(1))  
5 print(function(1, 2))  
6 print(function(1, 2, 3))
```

- 매개변수에 아무런 값을 넘기지 않았을 때,
기본으로 사용할 값을 지정할 수 있다.

키워드 인자

```
1 def function(a=1, b=1, c=1):  
2     return a + b + c  
3  
4 print(function(b=2))  
5 print(function(c=2, b=3))  
6 print(function(c=2, a=3, b=4))
```

- 함수를 호출할 때 **매개변수 이름을 정해서** 값을 넘기는 것
- 매개변수 순서를 지키지 않아도 된다.
- **print("hello", end=' ')** 에서, **end=' '** 부분이 키워드 인자이다.

람다 함수

- 익명 함수라고도 한다.
- 함수의 이름을 정하지 않고 기능만을 정의한 간단한 형태의 함수이다.

람다 함수

```
1 a = lambda x: x+1
```

- **lambda 매개변수 : 반환값** 형태로 작성한다.
- 위 코드는 아래 함수와 같다.

```
3 def a(x):  
4     return x+1
```

람다 함수

- 함수는 일종의 **데이터**로 보고 변수에 넣을 수 있다.
- 변수에 담은 람다 함수는 **변수명을 이용해 호출**할 수 있다.

```
1 a = lambda x: x+1
2 b = a
3
4 print(a(1))
5 print(b(2))
```

```
>>> %Run ps.py
2
3
```

람다 함수

- 알고리즘 문제를 풀 때는 **복잡한 기준으로 정렬**할 때 자주 사용한다.

```
1 l = [(1, 2), (3, 5), (2, 2)]  
2 l.sort(key=lambda x: (-x[1], -x[0]))  
3 print(l)
```

- (언제 쓰는지 보여주기 위한 예시로 쓴 코드이니 지금은 이해가 안돼도 괜찮습니다.)

재귀 함수

재귀(再歸) 「명사」 원래의 자리로 되돌아가거나 되돌아옴.

- 함수를 정의할 때 자기 자신을 호출하는 것

재귀 함수

- 함수의 정의를 하면서 자기 자신을 호출한다.

```
1 def func(x):  
2     x = x + func(x-1)  
3     return x
```

재귀 함수

- 그런데 실행해보면 에러가 난다.

```
1 def func(x):  
2     x = x + func(x-1)  
3     return x  
4  
5 func(10)
```

```
File <string>, line 2, in func  
File "<string>", line 2, in func  
File "<string>", line 2, in func  
RecursionError: maximum recursion depth exceeded
```


콜 스택

- 함수를 호출(실행)하면 현재 실행하고 있는 함수의 정보(데이터)가 메모리에 저장된다.
- 이 데이터는 **함수가 종료되면 삭제**한다.

```
1 def func(x):  
2     x = x + func(x-1)  
3     return x  
4  
5 func(10)
```

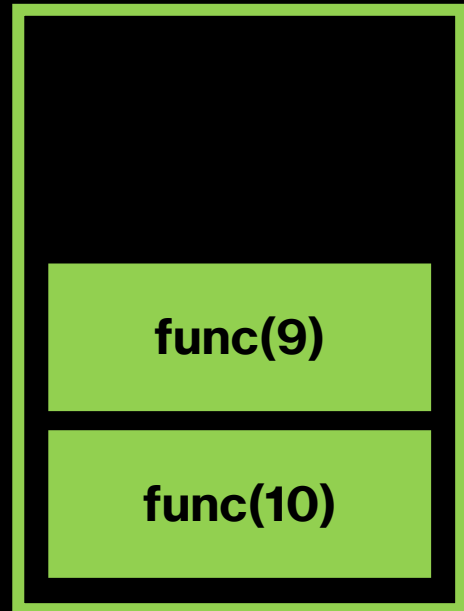


func(10)

콜 스택

- $x = x + \text{func}(x-1)$ 코드를 실행하면서 $\text{func}(9)$ 가 호출된다.
- 아직 $\text{func}(10)$ 은 실행 중이므로, $\text{func}(10)$ 의 데이터 **위에** $\text{func}(9)$ 데이터를 저장한다.

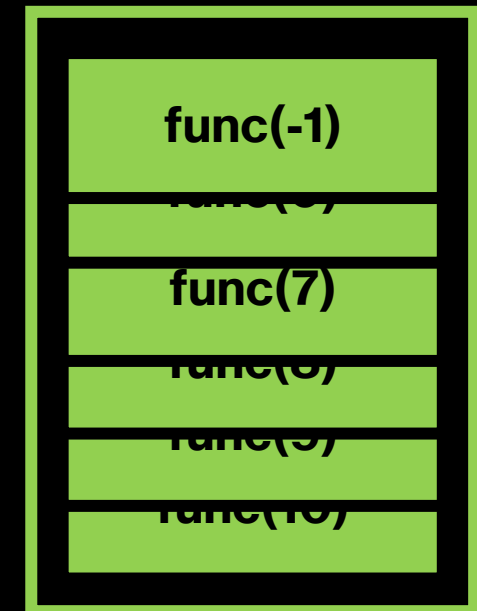
```
1 def func(x):  
2     x = x + func(x-1)  
3     return x  
4  
5 func(10)
```



콜 스택

- 이 과정은 func(8), func(7), ..., func(0), func(-1), .. 순으로 멈추지 않고 반복된다.

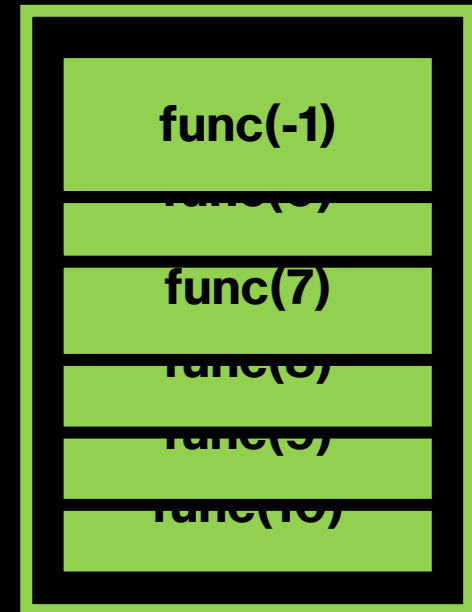
```
1 def func(x):  
2     x = x + func(x-1)  
3     return x  
4  
5 func(10)
```



콜 스택

- 함수의 **호출(콜, call) 정보**를 차곡차곡 **쌓아서 저장**하기 때문에 이 메모리 공간을 '**콜 스택**' 이라고 한다.

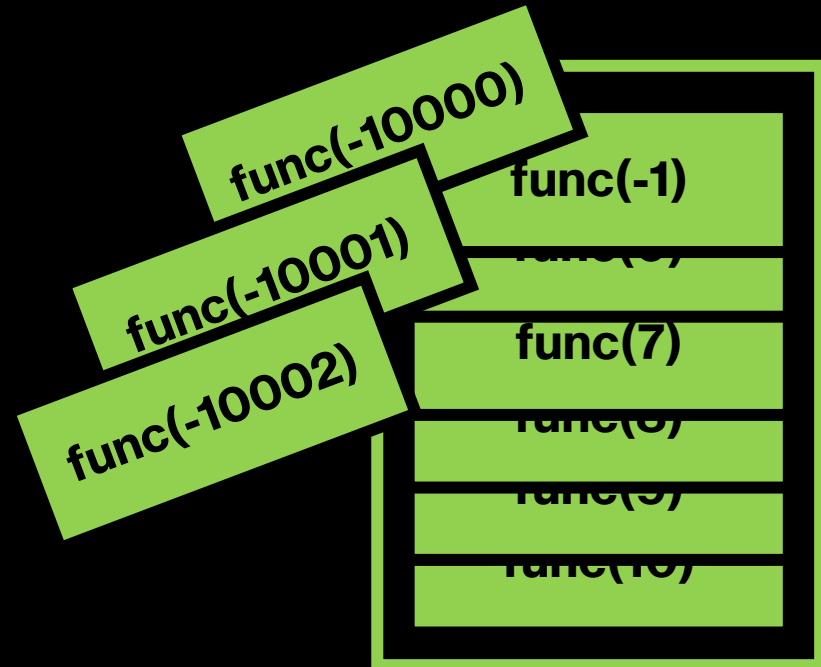
```
1 def func(x):  
2     x = x + func(x-1)  
3     return x  
4  
5 func(10)
```



스택 오버플로우

- 콜 스택의 크기는 한정적이라, 일정 호출 횟수를 넘어가면 함수의 호출 데이터가 밖으로 넘친다. (스택 오버플로우)

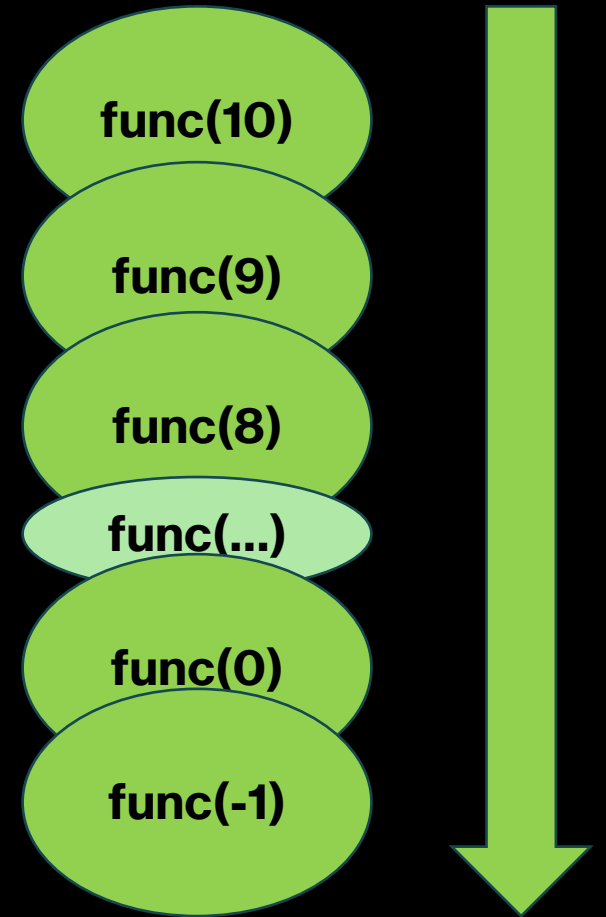
```
1 def func(x):  
2     x = x + func(x-1)  
3     return x  
4  
5 func(10)
```



재귀 깊이

- 콜 스택이 함수의 호출을 아래에서 위로 쌓는 형태로 보여준다면, 재귀 깊이는 함수의 호출을 **위에서 아래로 깊어지는 형태**로 나타낸 것을 말한다.

```
File "<string>", line 2, in func
File "<string>", line 2, in func
File "<string>", line 2, in func
RecursionError: maximum recursion depth exceeded
```



기저 조건

- 스택 오버플로우를 방지하려면 함수를 재귀적으로 호출하지 않고 **빠져 나오는 조건**이 있어야 한다. 이를 **기저 조건** 이라고 한다.

```
1 def func(x):  
2     if x == 0:  
3         return 1  
4  
5     return x * func(x-1)
```

연습 문제 1

- <https://www.acmicpc.net/problem/27433>

팩토리얼 2 성공

0보다 크거나 같은 정수 N 이 주어진다. 이때, $N!$ 을 출력하는 프로그램을 작성하시오.

연습 문제 1

- 한번 풀어봅시다.

연습 문제 1

- 같이 풀어봅시다.

연습 문제 1

- 반복문 정답 코드

```
1 n = int(input())
2 ans = 1
3 for i in range(1, n+1):
4     ans *= i
5 print(ans)
```

- 재귀 정답 코드

```
1 def factorial(n):
2     if n == 0:
3         return 1
4     return n*factorial(n-1)
5
6 n = int(input())
7 print(factorial(n))
```

너무 어려워요..

- 재귀 개념은 처음엔 어려운 게 맞습니다!(저도 그랬어요..)
- 재귀 문제를 풀 땐 각각의 함수가 앞 뒤 호출은 무시하고 독립적으로 **자기 할 일만 한다**고 생각해보세요!

(함수의 호출을 하나하나 따라가면 더 힘들어요!)

쉬어가는 시간


- 컴퓨터 공학과에서 프로젝트를 관리하는 방법
→ **Git & Github**


Git


- 소스코드를 버전별로 관리할 수 있게 도와주는 프로그램


Git


- 프로그램 소스코드는 수시로 바뀌는데 이렇게 복사본을 만들면서 관리할 수 없다.


 [2024-1 기초 스터디] 7. 함수, 재귀함수.pptx

 [2024-1 기초 스터디] 7. 함수, 재귀함수 - 수정본2.pptx

 [2024-1 기초 스터디] 7. 함수, 재귀함수 - 수정본.pptx

 [2024-1 기초 스터디] 7. 함수, 재귀함수 - 수정본 (최종).pptx

 [2024-1 기초 스터디] 7. 함수, 재귀함수 - 수정본 (진짜 최종).pptx

 [2024-1 기초 스터디] 7. 함수, 재귀함수 - 수정본 (진짜 최종22).pptx

Git

- 프로그램 소스코드 파일은 최종 버전 하나만 두고, 코드의 **변경 이력**만 별도로 관리할 수 있다.

```
feat: 2024 SUAPC W 경험 추가
kckc0608 committed 3 months ago

Commits on Feb 15, 2024

feat: 자기소개 웹사이트 프로젝트 설명 업데이트
kckc0608 committed 3 months ago

feat: 2024 경험 데이터 추가
kckc0608 committed 3 months ago

feat: change 2023 experience into json
kckc0608 committed 3 months ago

feat: 2022 경험을 json data 로 이전
kckc0608 committed 3 months ago
```

```
content/experience.json
@@ -16,6 +16,15 @@
16 16         "<a href='https://chinpa.tistory.com/281'>프로젝트 진행 과정 정리</a><br />",
17 17         "<a href='http://pumasi.everdu.com'>프로젝트 API 서버</a><br />"
18 18     ]
19 +     },
20 +     {
21 +         "date": "2024. 2.17",
22 +         "title": "- SUAPC 2024 Winter 18등",
23 +         "content": [
24 +             "신촌 지역 대학생 연합 프로그래밍 대회에서 2명의 팀원과 함께 참가하였습니다. <br />",
25 +             "팀명: 소년가장 <br />",
26 +             "<a href='https://www.acmicpc.net/contest/spotboard/1237'>스코어보드</a>"
27 +         ]
28     }
29 + ]
30 + },
```


Git

- 각각의 변경이력을 ‘커밋(commit)’ 이라고 한다.
원할 때는 이전 변경 시점으로 언제든지 돌아갈 수 있다.

```
feat: 2024 SUAPC W 경험 추가
kckc0608 committed 3 months ago

Commits on Feb 15, 2024

feat: 자기소개 웹사이트 프로젝트 설명 업데이트
kckc0608 committed 3 months ago

feat: 2024 경험 데이터 추가
kckc0608 committed 3 months ago

feat: change 2023 experience into json
kckc0608 committed 3 months ago

feat: 2022 경험을 json data 로 이전
kckc0608 committed 3 months ago
```

```
content/experience.json
@@ -16,6 +16,15 @@
16 16      "<a href='https://chinpa.tistory.com/281'>프로젝트 진행 과정 정리</a><br />",
17 17      "<a href='http://pumasi.everdu.com'>프로젝트 API 서버</a><br />"
18 18      ]
19 19      },
20 20      {
21 21          "date": "2024. 2.17",
22 22          "title": "- SUAPC 2024 Winter 18등",
23 23          "content": [
24 24              "신촌 지역 대학생 연합 프로그래밍 대회에서 2명의 팀원과 함께 참가하였습니다. <br />",
25 25              "팀명: 소년가장 <br />",
26 26              "<a href='https://www.acmicpc.net/contest/spotboard/1237'>스코어보드</a>"
27 27          ]
28 28      }
29 29  ],
30 30  },
```

Github

- Git으로 관리하는 변경 이력은 **내 컴퓨터에 파일로 존재**한다.
- 이 변경 이력을 온라인에 공유/백업하고 싶을 때,
Github를 사용한다.
- 온라인에 똑같은 코드가 공유되므로,
함께 프로젝트를 할 때 **협업하는 도구**로서도 쓸 수 있다.

Github

- 깃허브에는 수많은 개발자들이 자신이 진행한 프로젝트 소스 코드를 공유/저장해둔다.
→ 프로그래머에게는 자신의 **포트폴리오**가 된다.
- 참고로 기초 스터디 강의록은 제 깃허브에도 있습니다!
<https://github.com/kckc0608/hiarc-2024-1-basic-study>

Git & Github 어떻게 공부하나요?

- 무료 강의 (코딩애플, 알파한 코딩사전, 드림 코딩)
- 공식 문서 (<https://git-scm.com/docs>)
- 책

연습 문제 2

- <https://www.acmicpc.net/problem/17478>

재귀함수가 뭔가요? 성공

출력 예시를 보고 재귀 횟수에 따른 챗봇의 응답을 출력한다.

```
어느 한 컴퓨터공학과 학생이 유명한 교수님을 찾아가 물었다.  
"재귀함수가 뭔가요?"  
"잘 들어보게. 옛날옛날 한 산 꼭대기에 이세상 모든 지식을 통달한 선인이 있었어.  
마을 사람들은 모두 그 선인에게 수많은 질문을 했고, 모두 지혜롭게 대답해 주었지.  
그의 답은 대부분 옳았다고 하네. 그런데 어느 날, 그 선인에게 한 선비가 찾아와서 물었어."  
_____"재귀함수가 뭔가요?"  
_____"잘 들어보게. 옛날옛날 한 산 꼭대기에 이세상 모든 지식을 통달한 선인이 있었어.  
____마을 사람들은 모두 그 선인에게 수많은 질문을 했고, 모두 지혜롭게 대답해 주었지.  
____그의 답은 대부분 옳았다고 하네. 그런데 어느 날, 그 선인에게 한 선비가 찾아와서 물었어."  
_____"재귀함수가 뭔가요?"  
_____"재귀함수는 자기 자신을 호출하는 함수라네"  
_____"라고 답변하였지."  
____"라고 답변하였지."  
____"라고 답변하였지."
```

연습 문제 2

- 출력 형식을 보면 아래와 같은 구조가 재귀적으로 반복된다.

[선비 이야기]

1. 재귀가 뭔가요? → 옛날에 어떤 선비가 물어봐서~
2. [선비 이야기]
3. 라고 답변하였지

연습 문제 2

- [선비 이야기]는 현재 호출된 함수가 신경쓸 일이 아니다.
다음 함수가 알아서 잘 출력할 거라고 **믿고 넘겨주자!**

[선비 이야기]

1. 재귀가 뭔가요? 옛날에 어떤 선비가 물어봐서~
2. **[선비 이야기]**
3. 라고 답변하였지

연습 문제 2

어느 한 컴퓨터공학과 학생이 유명한 교수님을 찾아가 물었다.

"재귀함수가 뭔가요?"

"잘 들어보게. 옛날옛날 한 산 꼭대기에 이세상 모든 지식을 통달한 선인이 있었어.
마을 사람들은 모두 그 선인에게 수많은 질문을 했고, 모두 지혜롭게 대답해 주었지.
그의 답은 대부분 옳았다고 하네. 그런데 어느 날, 그 선인에게 한 선비가 찾아와서 물었어."

____ "재귀함수가 뭔가요?"

____ "잘 들어보게. 옛날옛날 한 산 꼭대기에 이세상 모든 지식을 통달한 선인이 있었어.

____ 마을 사람들은 모두 그 선인에게 수많은 질문을 했고, 모두 지혜롭게 대답해 주었지.

____ 그의 답은 대부분 옳았다고 하네. 그런데 어느 날, 그 선인에게 한 선비가 찾아와서 물었어."

_____ "재귀함수가 뭔가요?"

_____ "재귀함수는 자기 자신을 호출하는 함수라네"

_____ 라고 답변하였지.

____ 라고 답변하였지.

라고 답변하였지.

연습 문제 2

이전 코드가 할 일 (신경 x)

"재귀함수가 뭔가요?"

"잘 들어보게. 옛날옛날 한 산 꼭대기에 이세상 모든 지식을 통달한 선인이 있었어.
마을 사람들은 모두 그 선인에게 수많은 질문을 했고, 모두 지혜롭게 대답해 주었지.
그의 답은 대부분 옳았다고 하네. 그런데 어느 날, 그 선인에게 한 선비가 찾아와서 물었어."

선비 이야기

라고 답변하였지.

연습 문제 2

이전 함수가 할 일 (신경 x)

____ "재귀함수가 뭔가요?"

____ "잘 들어보게. 옛날옛날 한 산 꼭대기에 이세상 모든 지식을 통달한 선인이 있었어.

____ 마을 사람들은 모두 그 선인에게 수많은 질문을 했고, 모두 지혜롭게 대답해 주었지.

____ 그의 답은 대부분 옳았다고 하네. 그런데 어느 날, 그 선인에게 한 선비가 찾아와서 물었어."

선비 이야기

____ 라고 답변하였지.

이전 함수가 할 일 (신경 x)

연습 문제 2

이전 함수가 할 일 (신경 x)

----- "재귀함수가 뭔가요?"

----- "재귀함수는 자기 자신을 호출하는 함수라네"

----- 라고 답변하였지.

이전 함수가 할 일 (신경 x)

연습 문제 2

- 같이 풀어봅시다.

연습 문제 2

- 정답 코드

```
n = int(input())
print("어느 한 컴퓨터공학과 학생이 유명한 교수님을 찾아가 물었다.")

2 usages
def f(depth):
    print("____"*depth + "\"재귀함수가 뭔가요?\"")
    if depth == n:
        print("____"*depth + "\"재귀함수는 자기 자신을 호출하는 함수라네\"")
    else:
        print("____"*depth + "\"잘 들어보게. 옛날옛날 한 산 꼭대기에 이세상 모든 지식을 통달한 선인이 있었어.\"")
        print("____"*depth + "마을 사람들은 모두 그 선인에게 수많은 질문을 했고, 모두 지혜롭게 대답해 주었지.")
        print("____"*depth + "그의 답은 대부분 옳았다고 하네. 그런데 어느 날, 그 선인에게 한 선비가 찾아와서 물었어.\")
        f(depth+1)
    print("____"*depth + "라고 답변하였지.")

f(0)
```

연습 문제 2

- Q. 변수 n 은 함수 밖에서 선언했는데, 어떻게 함수 안에서도 사용할 수 있나요?
- A. 파이썬은 변수를 찾을 때 함수 안에 없는 변수는 알아서 함수 밖에서 찾아냅니다!

변수 범위 (scope)

- 파이썬은 **블록 단위(들여 쓰기)**로 변수 범위를 구분한다.

```
1 a = 10
2
3 def f():
4     a = 11
5     print(a)
6
7 f()
8 print(a)
```

```
11
10
```

변수 범위 (scope)

- 자신의 범위에 변수가 없다면 바깥 범위에서 변수를 찾는다.

```
1 a = 10
2
3 def f():
4     print(a)
5
6 f()
7 print(a)
```

```
10
10
```


변수 범위 (scope)

- 변수 값을 읽기만 하거나 (바깥 변수에 접근) 쓰기만 할 때는 (현재 범위에 변수 생성) 문제가 없으나, **변수를 읽기도 하고 쓰기도 할 때는 주의**해야 한다.

```
1 a = 10
2
3 def f():
4     a = a + 11
5     print(a)
6
7 f()
8 print(a)
```

```
>>> %Run -c $EDITOR_CONTENT
Traceback (most recent call last):
  File "<string>", line 7, in <module>
  File "<string>", line 4, in f
UnboundLocalError: local variable 'a' referenced before as
signment
```

변수 범위 (scope)

- **global** 키워드를 사용하면 문제를 해결할 수 있다.

```
1 a = 10
2
3 def f():
4     global a
5     a = a + 11
6     print(a)
7
8 f()
9 print(a)
```

21
21

이해가 안돼요..

- 변수 스코프는 처음엔 헷갈리는 개념이 맞습니다!
- 헷갈리면 **매개변수**를 통해서 바깥 변수의 값을 읽고, **return**을 통해서 바깥 변수에 값을 쓰면 됩니다.

```
1 a = 10
2
3 def f(a):
4     a = a + 11
5     return a
6
7 print(a)
8 a = f(a)
9 print(a)
```

```
10
21
```

Recursion Limit

- 파이썬은 스택 오버플로우를 방지하기 위해 함수의 **재귀 호출 횟수를 제한**한다.
(제한 횟수의 기본 설정값은 버전마다, ide마다 다를 수 있다.)

```
>>> sys.getrecursionlimit()  
1000
```

```
>>> import sys  
>>> sys.getrecursionlimit()  
3000
```

Recursion Limit

- 알고리즘 문제를 풀 때는 입력값이 커지면 1000번은 간단히 넘기기 때문에, 제한을 임의로 늘려주어야 한다.
- sys모듈에 있는 **set recursion limit()** 메서드를 이용한다.

```
1 import sys
2 sys.setrecursionlimit(10**6)
```

Recursion Limit

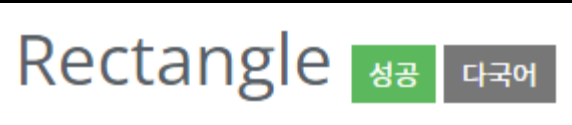
- 제한을 크게 잡으면 그만큼 메모리를 많이 사용하게 된다.
- 보통 **100만 정도로 설정**하면 적절하다.
(혹시 **메모리 초과**가 발생하면 10만 정도로 낮춰보자.)
- 단, 이 설정은 **pypy3 으로 제출할 때는 사용하기 힘들다.**
pypy3은 기본적으로 메모리를 많이 쓰기 때문이다.

Recursion Limit

- 혹시 재귀 함수를 이용해서 문제를 풀다가 런타임 에러가 발생했다면 꼭 재귀 제한을 해제했는지 확인해보자!!
- 직접 테스트할 때는 보통 작은 값으로 테스트를 하기 때문에 재귀 제한 때문에 막힌다는 것을 찾기 힘들다..

연습 문제

- <https://www.acmicpc.net/problem/7113>



- $N \times M$ 직사각형을 가장 큰 정사각형 모양으로 잘라내기
- 자르고 남은 직사각형에 대해 반복
- 얻어낸 정사각형의 총 개수 출력

연습 문제

- 같이 풀어봅시다.

연습 문제

- 정답 코드

```
1 import sys
2 sys.setrecursionlimit(10**6)
3
4 def f(n, m):
5     if n == 1:
6         return m
7     if m == 1:
8         return n
9     if n == m:
10         return 1
11
```

```
12     if n > m:
13         n -= m
14     elif n < m:
15         m -= n
16     return 1 + f(n, m)
17
18 n, m = map(int, input().split())
19 print(f(n, m))
```

이번 주 연습 문제

- 10870 피보나치 수 5
- 2439 별찍기 - 2
- 25501 재귀의 귀재
- 4779 칸토어 집합
- 17910 Joint Attack

이번 주 연습 문제

- 모든 반복문은 재귀로 바꿔서 쓸 수 있습니다.
- 피보나치 수, 별 찍기 문제를 한번 재귀로 풀어보세요!