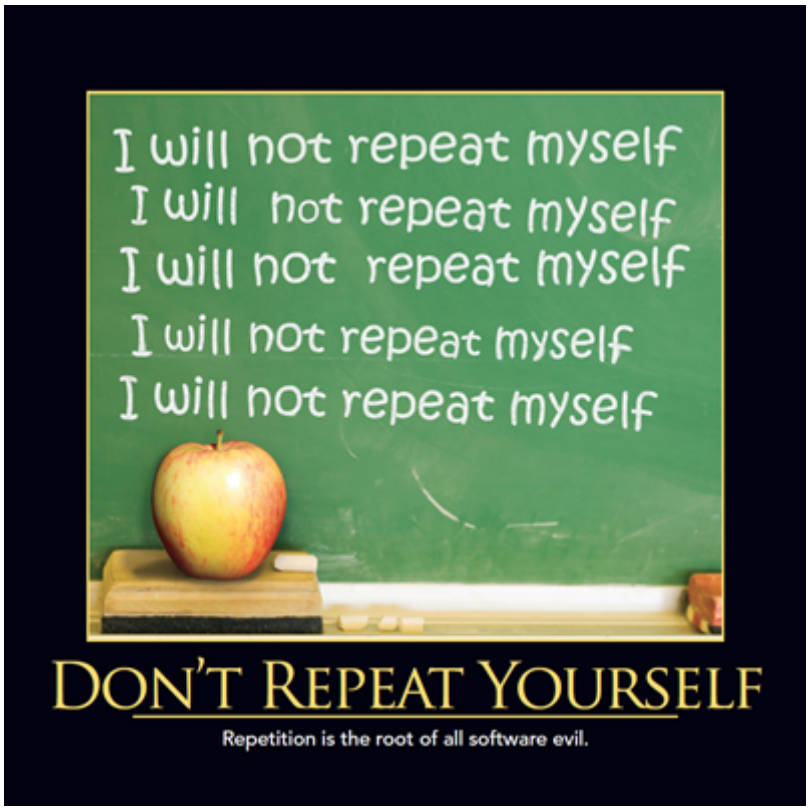
Stage 4 - 중복 코드 제거하기

함수를 배워봅니다. 함수를 사용하여 중복적으로 들어가있는 코드를 하나로 개선하여 보기 좋게 코드를 개선할 수 있습니다.

중복 코드 제거하기

소프트웨어를 개발하는 철학 중에 'DRY'라는 철학이 있습니다. 이는 'Don't Repeat Yourself'를 줄인 말입니다. 소프트웨어를 개발할 때 같은 코드를 두번 사용하는 것과, 복사 붙여넣기를 하지 말라는 이야기입니다. 이전 시간에 배웠던 내장 함수와 외장 함수처럼 한줄로 보기 좋고 이해하기 쉽게끔 만들어서 복사 붙여넣기가 아닌 좋은 코드를 만들어 나가야 합합니다.



Don't repeat yourself

중복 코드 찾아보기

현재 우리의 코드에는 크게 2곳에 똑같은 코드가 들어가 있습니다. 살펴보겠습니다.

숫자 출력

print문을 사용할 때, 사용자, 컴퓨터를 제외하고 모든 부분이 똑같죠?

games/beskin_rabins_game.py

```
for _ in range(size_of_call):
    call += 1
    print("사용자 : '{0}'!!!".format(call))

    if call == 31:
        break
```

games/beskin_rabins_game.py

```
for _ in range(size_of_call):
    call += 1
    print("컴퓨터 : '{0}'!!!".format(call))

    if call == 31:
        break
```

입력 검사

방금 전 Stage 3를 통해 작성한 코드입니다. input문 안에 들어가는 안내문과 조건문에서 사용하는 리스트를 제외하면 완전히 똑같은게 보이시나요?

games/beskin_rabins_game.py

```
while True:
    order = input('''순서를 입력하세요. (선공 1, 후공 0 입력) : ''')
    if order in ['0', '1']:
        order = int(order)
        break
    else:
        print("잘못된 입력입니다. 재입력해주세요.")
```

games/beskin_rabins_game.py

```
while True:
```

```
    size_of_call = input("호출할 개수를 입력하세요 : ")
    if size_of_call in ['1', '2', '3']:
        size_of_call = int(size_of_call)
        break
    else:
        print("잘못된 입력입니다. 재입력해주세요.")
```

함수로 중복 해결하기

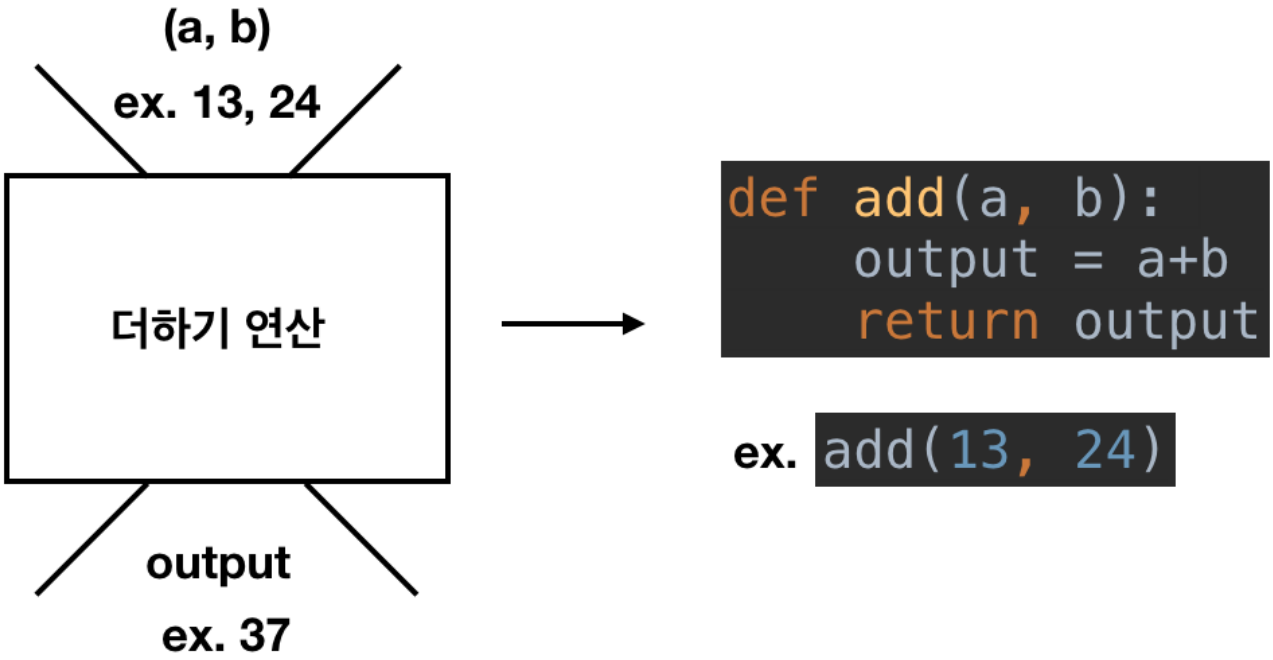
함수를 사용하여 중복을 해결할 수 있습니다. 왜냐하면 함수는 같은 동작에 대해서 미리 정의된 정보를 가지고 똑같이 수행하는 역할을 하기 때문이죠. 지금은 너무 간단한 프로그램이라 함수가 쓸모없어 보일지 모르지만, 나중에 코딩을 하게 되면 거의 모든 것이 함수로 이루어져 있습니다. 그리고 지금까지 우리가 사용한 것들도 내장 함수이고요.

함수 이해하기

함수를 만들어보겠습니다. 함수는 아래와 같은 형태로 생겼습니다.

```
def 함수이름(매개변수1, 매개변수2, ...):
    수행할 문장
    수행할 문장
    수행할 문장
    return 결과
```

중학교 시절 배웠던 함수가 기억나시나요? 바로 그 함수와 똑같습니다. 위의 모양을 중학교때 배우던 함수 그림과 비교해보겠습니다.



수학적 함수와 비교

위와 같이 함수를 정의(def)합니다. 그리고 사용할 때는 우리가 내장 함수와 외장 함수를 사용했던 것 처럼 사용하시면 됩니다. 이제 쫌 함수가 이해되시나요? 위의 예시처럼 우리가 사용했던 내장 함수들도 모두 파이썬을 설치하면 파이썬 내부에 저런 형태로 정의가 되어 있습니다.

숫자 출력 정의하기

매개변수, 아웃풋

함수를 만들기 전에 매개변수는 무엇으로 지정할지, 결과물은 무엇이 나와야할지 결정해야 합니다. 매개변수는 함수 내에서 알 수 없는 값을 외부에서 받아오는 매개체 역할을 합니다. 따라서, 중복되는 코드에서 알 수 없는 변수인 현재 호출되는 숫자와 호출한 수의 크기 대한 정보를 가져와야 합니다. 그리고 가져온 현재 호출 숫자를 입력받은 크만큼 증가시켜 결과물로 넘겨주면 됩니다.

함수 사용

아래와 같이 함수를 정의하고 함수가 제거된 부분에 사용할 수 있습니다.

games/beskin_rabins_game.py

```
1  def call_numbers(size, called):
2      for _ in range(size):
3          called += 1
4          print("'{0}'!!!".format(called))
5
6          if called == 31:
```

```
            break

    return called
```

games/beskin_rabins_game.py

```
call = call_numbers(size_of_call, call)
```

위와 같이 'call_numbers(size_of_call, call)'에 코드가 도착하면 전체 코드에서 ' call_numbers'라는 이름을 가진 함수가 정의되어 있는지 탐색하여, 정의된 함수를 실행시킵니다. 모든 함수의 실행이 종료되면 다시 'call_numbers(size_of_call, call)'이 있는 코드로 돌아와 동작합니다.

입력 검사 정의하기

매개변수, 아웃풋

매개변수는 무엇으로 정의하면 좋을까요? 안내문이 많이 다르죠? 그러면 외부로부터 안내문을 따로 받도록하고요. 유효한 입력 정보를 가진 리스트 (ex. ['0', '1'] , ['1', '2', '3'])을 정보로 받아오면 좋겠네요! 결과물로는 정상적으로 들어온 입력값을 숫자로 바꿔서 전달하고요!

함수 사용

이번에는 함수를 사용하는데 모양이 조금 다릅니다. 매개변수로 들어가는 정보가 이전과 다르기 때문이죠.

games/beskin_rabins_game.py

```
def validate_input(prompt, valid_list):
    while True:
        value = input(prompt)
        if value in valid_list:
            value = int(value)
            break
        else:
            print("잘못된 입력입니다. 재입력해주세요.")
    return value
```

games/beskin_rabins_game.py

```
order = validate_input("순서를 입력하세요. (선공 1, 후공 0 입력) : ", ['0', '1']
```

games/beskin_rabins_game.py

```
size_of_call = validate_input("호출할 개수를 입력하세요 : ", ['1', '2', '3'])
```