

## 06-2 다양한 함수의 사례

정수가 홀수인지 확인하는 함수



# 정수가 홀수인지 확인하는 함수

## 함수를 만들 때 필요한 4가지 요소

1. 홀수를 판단하는 기준 → 어떤 수를 **2로 나눈 나머지가 1이면 홀수**, 아니면 짝수
2. 함수 이름 → **is\_odd\_number**(홀수 입니까?)
3. 함수 입력값(인수) → **숫자 데이터 1개**(정수)
4. 함수 출력값(리턴값) → **불 데이터 1개**(홀수인 경우 True/ 짝수인 경우 False)

## 함수 실행 예시

```
is_odd_number(3) → True  
is_odd_number(2) → False
```



[손코딩 실습] 정수가 홀수인지 확인하는 함수(p.281)

```
01 def is_odd_number(arg):
```

```
02     if arg % 2 == 1:
```

```
03         return True
```

```
04     return False
```

```
05
```

```
06 print(is_odd_number(3))
```

```
07 print(is_odd_number(2))
```



영어 알파벳을 감싸는 테두리를 출력하는 함수



# 영어 알파벳을 감싸는 테두리를 출력하는 함수

## 함수 실행 예시

Hello, World!

\*\*\*\*\*

\* Hello, World! \*

\*\*\*\*\*

\*\*\*\*\*

323

\* 323 \*

\*\*\*\*\*

## 함수를 만들 때 필요한 4가지 요소

1. 테두리 길이 변경 → 주어진 **문자열의 길이**에 따라 테두리 길이 변경
2. 함수 이름 → **get\_bordered\_str**(테두리 그려진 문자열을 얻는다)
3. 함수 입력값(인수) → 테두리를 그릴 **문자열 1개**
4. 함수 출력값(리턴값) → 테두리가 그려진 **문자열 1개**



[손코딩 실습] 영어 알파벳을 감싸는 테두리를 출력하는 함수(p.283)

```
01 def get_bordered_str(arg):
02     result = ""
03     sep = "*"
04     length = len(arg)
05     result = result + (sep * (length + 4) + "\n")
06     result = result + (sep + " " + arg + " " + sep + "\n")
07     result = result + (sep * (length + 4))
08     return result
09
10 print(get_bordered_str("Hello, World!"))
11 print(get_bordered_str("323"))
```



리스트 요소들의 합계와 평균을 구하는 함수



# 리스트 요소들의 합계와 평균을 구하는 함수

## 함수 실행 예시

[3, 2] → {"합계": 5, "평균": 2.5}  
[-1, 0, 1, 2, 3] → {"합계": 5, "평균": 1.0}  
[] → "[오류] 요소의 개수가 0 입니다."

## 함수를 만들 때 필요한 4가지 요소

1. 리스트 요소가 없는 경우 → 리스트 **요소의 개수를 계산**하고, 0인 경우 오류 리턴
2. 함수 이름 → **get\_sum\_and\_average** (합계, 평균을 얻는다)
3. 함수 입력값(인수) → 숫자 데이터가 저장된 **리스트 1개**
4. 함수 출력값(리턴값) → 합계와 평균이 저장된 **딕셔너리 1개**



[손코딩 실습] 리스트 요소들의 **합계**와 **평균**을 구하는 함수 (p.285)

```
01 def get_sum_and_average(arg):
02     length = len(arg)
03     if length == 0:
04         return "[오류] 요소의 개수가 0 입니다"
05
06     total = 0
07     for x in arg:
08         total = total + x
09     return {"합계": total, "평균": total / length}
10
11 print(get_sum_and_average([]))
12 print(get_sum_and_average([3, 2]))
13 print(get_sum_and_average([-1, 0, 1, 2, 3]))
```



초를 시간, 분으로 변환하는 함수



# 초를 시간, 분으로 변환하는 함수

## 함수 실행 예시

3 → 3 초  
323 → 5 분 23 초  
 $60 * 60 + 323 * 2$  → 1 시간 10 분 46 초

## 함수를 만들 때 필요한 4가지 요소

1. 초를 시간, 분으로 변환 → 정수 나누기 연산자(`//`), 나머지 연산자(`%`)를 활용
2. 함수 이름 → **convert\_seconds** (초를 변환한다)
3. 함수 입력값(인수) → 변환할 초를 나타내는 **정수 1개**
4. 함수 출력값(리턴값) → 초가 시간, 분으로 변환된 **문자열 1개**



## [손코딩 실습] 초를 시간, 분으로 변환하는 함수 (p.287)

```
01 def convert_seconds(arg):
02     if arg < 60:
03         # 60초 미만이라면, 초만 출력
04         return str(arg) + " 초"
05
06     seconds = arg % 60
07     minutes = arg // 60
08     if minutes < 60:
09         # 60분 미만이라면, 분과 초를 출력
10         return str(minutes) + " 분 " + str(seconds) + " 초"
11
12     # 그 외의 경우, 시간, 분, 초를 출력
13     hours = minutes // 60
14     minutes = minutes % 60
15     return str(hours) + " 시간 " + str(minutes) + " 분 " + str(seconds) + " 초 "
16
17 print(convert_seconds(3))
18 print(convert_seconds(60))
19 print(convert_seconds(323))
20 print(convert_seconds(60 * 60 + 323 * 2))
```