

# 파이썬 프로그래밍

## 강의 노트 #06

---

### 함수

# 함수 사용

## □ 함수의 정의

- 수학에서의 함수와 유사함
  - 블랙박스 취급 (내부 구조 몰라도 됨)
  - 입력과 출력만 이해하면 사용할 수 있음
- 정해진 일을 처리하기 위해 작성된 코드의 묶음
- 여태까지는 함수보다는 명령이라는 이름으로 불렸음

## □ 함수 사용

- 전문가에게 작업을 요청하는 것과 유사함
- 함수를 호출한다고 함
- 예

```
>>> n = len("Hello World")
>>> print(n)
11
```

# 함수 사용

- 함수에는 입력과 출력이 있을 수 있음
  - 함수에 전달하는 입력값은 전달 인자(actual argument) 또는 줄여서 인자(argument)라고 부름
  - 함수가 출력으로 되돌려주는 결과값은 반환값(return value)이라고 부름
    - 앞에서 본 len() 함수에서 "Hello World"가 인자, n에 저장되는 문자열의 길이가 반환값
    - print() 함수에 전달되는 변수 n의 값이 print() 함수에 전달되는 입력
- 함수에 전달할 입력이 없다면 빈괄호만 사용

# 명령들을 다시 곱씹어보기

## □ 여태까지 사용했던 명령들

- print()
  - 주어진 내용을 화면에 출력
- type()
  - 주어진 값의 자료형을 알려줌
- input()
  - 사용자가 입력한 내용을 결과로 돌려줌
- int()
  - 주어진 값을 정수로 변환하고 결과로 돌려줌
- float()
  - 주어진 값을 실수로 변환하고 결과로 돌려줌
- str()
  - 숫자를 문자열로 변환 후 결과로 반환

# 명령들을 다시 곱씹어보기

- len()
  - 주어진 문자열을 구성하는 글자의 수를 결과로 반환
- 문자열에 포함되어 있는 명령들
  - startswith(), endswith()
    - 주어진 문자열로 문자열이 시작되는지 혹은 끝나는지 확인하고 True 또는 False 반환
  - index()
    - 주어진 문자열이 포함되어 있으면 그 시작 위치를 반환. 포함되어 있지 않으면 오류 발생
  - strip(), lstrip(), rstrip()
    - 공백 문자 제거, 혹은 주어진 문자 조합의 문자를 문자열 끝에서 제거

# 함수의 사용 이유

---

## □ 함수의 사용 이유

- 코드의 양을 줄일 수 있음
- 반복적으로 사용 가능
- 안전한 코드 작성 가능
- 가독성 증가

# 함수를 구현할 때 고려해야 할 요소들

## ▣ 함수를 구현하기 전에 생각하고 계획할 것

구성 요소	구현할 때 생각해야 하는 내용
이름	함수의 이름을 보고 어떤 일을 할 수 있는 유추할 수 있도록 정함 함수명은 처리 내용을 설명할 수 있도록 하되, 너무 길지 않게 지정
입력(매개변수)	매개 변수에 따라 작업을 다르게 처리하거나 다른 결과를 만들어냄 입력이 필요한지, 그렇다면 어떤 내용을 함수의 입력으로 받을지 판단해서 결정
출력(반환값)	함수의 결과값이 필요한지, 그렇다면 어떤 내용을 반환할지 판단해서 결정

# 함수 정의 방법 및 사용법

## ■ 함수 정의 방법

```
def 함수이름([매개_변수_목록]):  
    코드 블록
```

- def는 함수를 정의하는 키워드
- 함수에 전달받아야 하는 내용이 있으면 괄호 안에 전달 받기 위한 변수 이름을 넣음
  - 주의사항 1: 입력 내용이 없으면 ()만 작성
- 함수가 결과 값을 반환해야 하면 코드 블록에서 return 키워드 사용 가능
  - 주의사항 2: 출력 값이 없으면 return 문 생략 가능

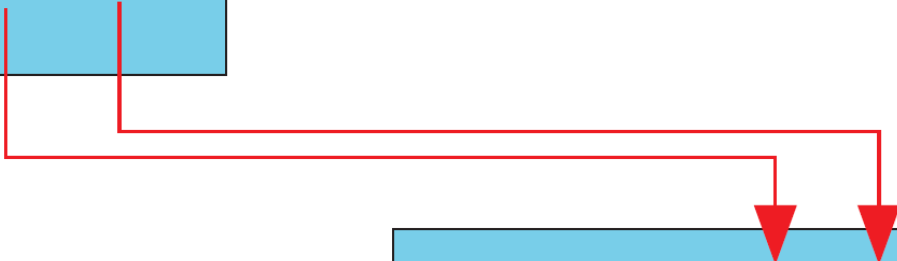


# 함수 정의 방법 및 사용법

- 매개 변수 목록은 변수 이름을 콤마로 분리해서 나열한 것
- 함수를 호출할 때 전달되는 인자값들과 매개 변수는 순서대로 연동됨

```
x = 5  
sum = add(x, 7)
```

```
def add(a, b):  
    sum = a + b  
    return sum
```



The diagram illustrates the argument passing mechanism. Two red lines originate from the arguments 'x' and '7' in the function call 'add(x, 7)'. These lines extend horizontally to the right and then turn vertically downwards as red arrows, pointing to the parameters 'a' and 'b' in the function definition 'def add(a, b):'. This visualizes how the values of 'x' and '7' are passed to 'a' and 'b' respectively when the function is called.

# 함수 사용 방법

# 1. 입력과 출력이 모두 있는 경우  
변수 = 함수이름(입력값)

# 1. 입력과 출력이 모두 있는 경우  
변수 = 함수이름(입력값1, 입력값2, ...)

# 2. 입력은 있고, 출력이 없는 경우  
함수이름(입력값)

# 3. 입력은 없고 출력은 있는 경우  
변수 = 함수이름()

# 4. 입력과 출력이 모두 없는 경우  
함수이름()

# 파이썬 함수

## □ 함수 예

### 1. 입력과 출력이 모두 있는 경우

```
def add(a, b):          # same as
    sum = a + b         # def add(a, b):
    return sum          #     return a + b
x = 5
sum = add(x, 7)
```

```
def add(a, b, c):
    return a + b + c
sum = add(5, 7, 9)
```

- return의 위치는 코드 블록 내에 어디에 위치해도 됨
- return은 결과 값 반환과 함수의 실행 종단을 의미함

# 파이썬 함수

## 2. 입력은 있고 출력이 없는 경우

```
def printWithoutNewLineChar(s):  
    print(s, end = ' ')  
printWithoutNewLineChar("안녕하세요")
```

## 3. 입력은 없고 출력만 있는 경우

```
def returnHelloString():  
    return "안녕하세요"  
s = returnHelloString()  
print(s)
```

## 4. 입력도 없고 출력도 없는 경우

```
def printHello():  
    print("안녕하세요")  
printHello()
```

# 실습문제 1

---

## □ 문제

- 정수 한 개를 함수의 매개 변수를 통해 입력 받고, 윤년인지 확인해서 True 또는 False를 반환하는 함수를 구현하고 이를 검수하는 프로그램 작성
- 윤년의 조건
  - 연도가 4로 나누어지면 윤년
  - 연도가 4로 나누어지면서 100으로 나누어지면 윤년 아님
  - 연도가 400으로 나누어지면 윤년

## □ 요구사항

- 함수에 전달되는 정수는 0보다 큰 양수로 가정

# 실습문제 1

## □ 최종 코드

```
def isLeapYear(year):  
    return (year % 400 == 0) or (year % 4 == 0  
and year % 100 != 0)  
  
print(isLeapYear(1234))  
print(isLeapYear(1900))  
print(isLeapYear(2000))  
print(isLeapYear(2204))
```

## 실습문제 2

### □ 문제

- 두 개 이상의 단어가 있는 문자열을 입력으로 전달받고, 두 번째 단어만 추출해서 반환하는 함수를 구현하고 이를 검수하는 프로그램 작성

### □ 요구사항

- 함수에 전달되는 문자열은 반드시 두 개 이상의 단어로 구성되어 있다고 가정
- 단어는 공백 문자, 웹 문자, 줄바꿈 문자 중 한 개로 분리됨(중복되지 않음)
- 문자열의 양끝에 공백 문자가 있을 수 있음(제거 후 단어를 추출)

## 실습문제 2

### □ 최종 코드

```
def getSecondWord(s):
    s = s.strip()
    pos1 = s.find(' ')
    pos2 = s.find('\t')
    pos3 = s.find('\n')
    # -1보다 큰 값을 반환함
    p1 = max(pos1, pos2, pos3) + 1
    s2 = s[p1:]
    pos1 = s2.find(' ')
    pos2 = s2.find('\t')
    pos3 = s2.find('\n')
    p2 = max(pos1, pos2, pos3)
    if p2 == -1:
        p2 = len(s)
    else:
        p2 += p1
    return s[p1:p2]

print(getSecondWord("what a beautiful day"))
print(getSecondWord("beautiful day"))
```



# 함수에서 return의 또 다른 사용 예

- 출력이 없는 함수에서 return은 함수 실행 도중 종료하는 목적으로 활용 가능

```
# 음수일 때에만 화면에 출력하고 0 이상의 숫자가  
# 인자로 전달되면 무시하는 함수 작성  
def printMinusNumber(num):  
    if num >= 0:  
        return  
    else:  
        print(num)  
  
printMinusNumber(3)  
printMinusNumber(-5)
```

# 입력 인자 전달과 매개변수

- 함수를 호출할 때, "매개변수 이름=인자" 형태로 순서를 바꿔서 지정할 수 있음

```
def add(num1, num2, num3):  
    return num1 + num2 + num3
```

```
s = add(2, 3, 5) # num1=2, num2=3, num3=5  
s = add(num3 = 5, num2 = 3, num1 = 2)
```

# 매개변수에 기본값 지정

- 함수를 구현하면서 매개 변수에 기본값(default value)를 지정 가능
  - 기본값이 지정된 매개변수에 인자가 전달되지 않으면 기본 값 사용
  - 기본값이 지정된 매개변수에 인자가 전달되면, 전달된 값을 사용

```
>>> def printNum(n1 = 1, n2 = 10):  
...     print(f"n1 = {n1}, n2 = {n2}")  
...  
>>> printNum()  
n1 = 1, n2 = 10  
>>> printNum(3, 13)  
n1 = 3, n2 = 13
```

## 실습문제 3

---

- 세 명의 성적을 입력으로 전달받고, 화면에 순서대로 출력하는 함수를 작성하고, 이 함수를 호출하는 코드 구현
- 일반적으로 성적은 내림차순으로 출력하지만, 가끔씩 반대로 출력하는 경우도 있음
  - 매개 변수를 이용해서 결정할 수 있도록 할 것
- 요구사항
  - 매개변수의 기본값을 내림차순으로 지정

## 실습문제 3

### ▣ 최종 코드

```
def CompareScores(score1, score2, score3, order = True)
    if order:
        if score1 >= score2 and score1 >= score3:
            if score2 >= score3:
                print(score1, score2, score3)
            else:
                print(score1, score3, score2)
        elif score2 >= score1 and score2 >= score3:
            if score1 >= score3:
                print(score2, score1, score3)
            else:
                print(score2, score3, score1)
```

```
    else:
        if score1 >= score2:
            print(score3, score1, score2)
        else:
            print(score3, score2, score1)
else: # 오름차순
    if score1 < score2 and score1 < score3:
        if score2 < score3:
            print(score1, score2, score3)
        else:
            print(score1, score3, score2)
    elif score2 < score1 and score2 < score3:
        if score1 < score3:
            print(score2, score1, score3)
        else :
            print(score2, score3, score1)
```

```
else:
    if score1 < score2:
        print(score3, score1, score2)
    else:
        print(score3, score2, score1)
```

```
print("내림차순")
CompareScores(85, 80, 90)
print("오름차순")
CompareScores(85, 80, 90, False)
```

# 변수의 유효 범위

---

- 함수 밖에서 만들어진 변수들은 전역 변수(global variable)라고 부르고, 프로그램 실행이 종료될 때 사라지고, 유효 범위는 프로그램 전체가 됨
  - 유효 범위는 해당 변수가 보이고 사용될 수 있는 영역
- 함수 내부에서 만들어진 변수들은 지역 변수(local variable)라고 부르고, 함수 내부에서만 사용됨



# 변수의 유효 범위

- 함수 내부에서 변수를 사용하면 새로 생성됨

새로운 sum  
변수가 생성됨

```
sum = 0  
def add(a, b, c):  
    sum = a + b + c  
    print(sum)
```

```
add(3, 2, 4)  
print(sum)
```


# 변수의 유효 범위

- 함수 밖에서 만들어진 변수를 함수 내부에서 쓰고 싶다면 `global` 키워드로 다시 선언해야 함

앞에서  
만들어진 `sum`  
변수를 함수  
내부에서  
사용하겠다는  
의미임

```
sum = 0
def add(a, b, c):
    global sum
    sum = a + b + c
    print(sum)

add(3, 2, 4)
print(sum)
```



# 변수의 유효 범위

- 좀 더 바람직한 프로그래밍 방법은 함수 내부에서 외부 변수를 사용하지 않도록 프로그래밍 하는 것
- sum을 바꿀 수 있는 좀 더 좋은 코드의 예

```
sum = 0
def add(a, b, c):
    sum = a + b + c
    return sum

sum = add(3, 2, 4)
print(sum)
```

# 재귀 호출 (Recursion)

- 함수에서 자기 자신을 다시 부르는 것을 재귀호출이라고 함
- 재귀호출을 빠져나갈 수 있도록 검사하고 종료하는 부분이 반드시 존재해야 함 (재귀호출 탈출 조건)

```
def RecursiveFunc():  
    ... # 함수 실행 코드 1  
    RecursiveFunc() # 재귀호출  
    ... # 함수의 실행 코드 2
```

# 재귀 호출

---

- 팩토리얼(factorial)을 구현하는 함수 작성
- $n! = n \times (n - 1) \times (n - 2) \times \dots \times 1$  (where  $n \geq 1$ )  
 $= n \times (n - 1)! \quad (\text{where } n \geq 0, 0! = 1)$
- 알고리즘
  - 만약  $n$ 이 1보다 작거나 같으면 값 1을 반환
  - 만약  $n$ 이 1보다 크다면  $n * (n - 1)!$ 을 반환

# 재귀 호출

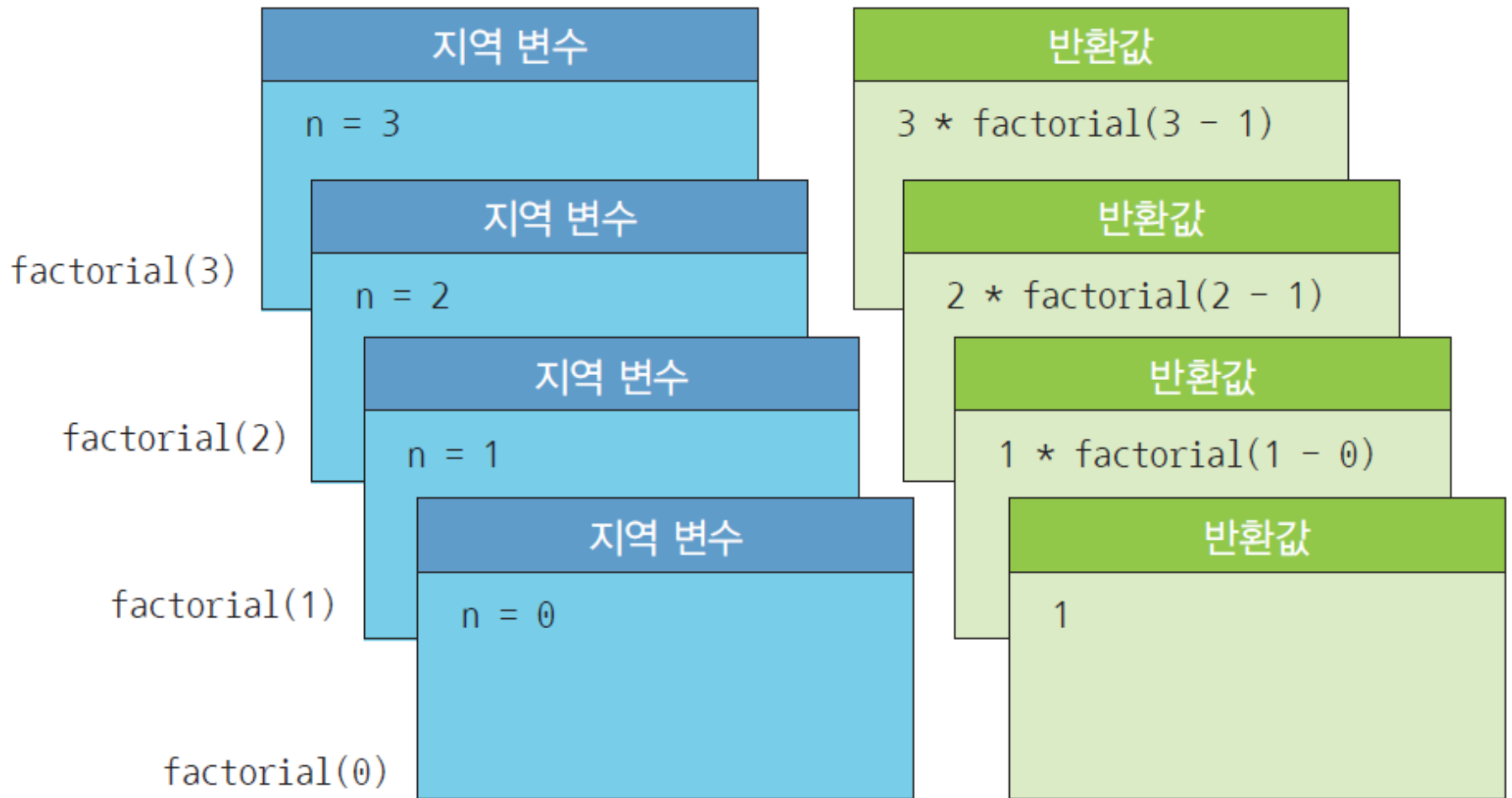
```
def factorial(n):  
    if n == 0:    # 재귀호출 탈출 조건  
        return 1  
    else:  
        return n * factorial(n - 1)  
  
factorial(1)          # 1  
factorial(10)         # 3628800
```

# 재귀 호출

## ▣ 재귀 호출이 일어나는 과정

함수 실행 상태	지역 변수 n	반환값
factorial(3)	$n = 3$	$3 * \text{factorial}(3 - 1)$
factorial(2)	$n = 2$	$2 * \text{factorial}(2 - 1)$
factorial(1)	$n = 1$	$1 * \text{factorial}(1 - 1)$
factorial(0)	$n = 0$	1

# 재귀 호출





# 재귀호출

---

- 최대공약수(Great Common Denominator) 구하기
- 알고리즘
  - $m = n$ 이면  $m$  또는  $n$  반환
  - $m > n$ 이면  $m - n$ 과  $n$ 의 최대공약수 반환
  - $m < n$ 이면  $m$ 과  $n - m$ 의 최대공약수 반환

# 재귀호출

```
def gcd(m, n):  
    if m == n:  
        return m          # 재귀호출 탈출 조건  
    elif m > n:  
        return gcd(m - n, n)  
    else:                  # m < n  
        return gcd(m, n - m)  
  
gcd(12, 4)                # 4  
gcd(12, 18)              # 6
```

# 재귀호출

---

- 피보나치 수열

- 1 1 2 3 5 8 13 21 34 55 89

- 1항과 2항은 1

- 3항 이후부터의 n항은  $(n - 1)$ 항 +  $(n - 2)$ 항

- $f(n) = f(n - 1) + f(n - 2)$

# 재귀호출

```
def fibonacci(n):  
    if n == 1 or n == 2: # 재귀호출 탈출 조건  
        return 1  
    else:  
        return fibonacci(n - 1) + fibonacci(n - 2)
```

fibonacci(1)	# 1
fibonacci(2)	# 1
fibonacci(3)	# 2
fibonacci(6)	# 8

## 실습문제 4

---

### □ 문제

- 인자로 정수 한 개(매개변수 이름은  $n$ )을 입력 받고, 1부터  $n$ 까지의 합을 구해서 반환하는 함수를 재귀 호출을 이용해서 구현

### □ 요구사항

- 인자로 전달되는  $n$ 은 1이상의 정수

## 실습문제 4

### ▣ 최종 코드

```
def Sum(n):  
    if n == 1:  
        return 1  
    return n + Sum(n - 1)  
  
print(Sum(1))  
print(Sum(10))  
print(Sum(100))
```

# 실습문제 5

---

## □ 문제

- 두 개의 정수를 인수로 받아 그 합을 반환하는 함수를 작성하세요.

## □ 요구사항

- `result = add_numbers(3, 5)`
- `print(result)`

# 실습문제 5-1

---

## □ 문제

- 문자열을 인수로 받아 그 문자열을 뒤집어서 반환하는 함수를 작성하세요.

## □ 요구사항

- ("hello") → "olleh"가 출력되어야 합니다.



# 실습문제 6

---

## □ 문제

- 문자열을 인수로 받아 그 문자열의 첫 글자를 대문자로 바꾸어 반환하는 함수를 작성하세요.

## □ 요구사항

- Upper() 활용해보자

# 실습문제 7

---

## □ 문제

- 문자열을 인수로 받아 그 길이가 홀수인지 짝수인지를 반환하는 함수를 작성하세요.

## □ 요구사항

- len() 활용

# 실습문제 8

---

## □ 문제

- 사용자로부터 두 개의 문자열을 입력받아, 두 문자열의 길이를 비교하고 더 긴 문자열을 반환하는 함수를 작성하세요.

## □ 요구사항

- `str1 = input("첫 번째 문자열을 입력하세요: ")`  
`str2 = input("두 번째 문자열을 입력하세요: ")`

## 실습문제 9

---

### □ 문제

- 사용자로부터 문자열과 특정 문자를 입력받아, 해당 문자열 내에 특정 문자가 몇 개 포함되어 있는지 반환하는 함수를 작성하세요.

### □ 요구사항

- `print(f'{char}' 문자의 개수:", result)` 활용

# 실습문제 10

---

## □ 문제

- 사용자로부터 문자열을 입력받아, 해당 문자열의 대문자는 소문자로, 소문자는 대문자로 변환하여 반환하는 함수를 작성하세요.

## □ 요구사항

- `Swapcase()` 활용

# 실습문제 11

---

## □ 문제

- 사용자로부터 숫자를 입력받아, 그 숫자가 양수인지 음수인지 또는 0인지를 판별하는 함수를 작성하세요.

## □ 요구사항

# 실습문제 12

---

## □ 문제

- 사용자로부터 숫자를 입력받아, 그 숫자가 1과 100 사이에 있는지 확인하는 함수를 작성하세요.

## □ 요구사항

# 실습문제 13

---

## □ 문제

- 사용자로부터 두 개의 숫자와 수행할 연산(+, -, \*, /)을 입력받아, 해당 연산을 수행하는 계산기 함수를 작성하세요. 연산 결과를 반환하는 함수를 작성하세요.

## □ 요구사항

- `input("첫 번째 숫자를 입력하세요: ")`
- `input("두 번째 숫자를 입력하세요: ")`
- `input("수행할 연산을 입력하세요 (+, -, *, /): ")`



# 실습문제 14

---

## □ 문제

- 사용자로부터 세 개의 숫자를 입력받아, 세 숫자의 합과 평균을 계산하는 함수를 작성하세요. 평균이 50 이상이면 "평균이 50 이상입니다"를, 그렇지 않으면 "평균이 50 미만입니다"를 출력하세요.

## □ 요구사항

- `print(f"합: {total}, 평균: {avg}, 메시지: {msg}")`

# 실습문제 15

## □ 문제

- 사용자로부터 문자열을 입력받아, 해당 문자열에서 숫자를 모두 추출하고, **추출한 숫자의 합을 계산**하여 반환하는 함수를 작성하세요. 숫자가 없으면 **"숫자가 없습니다"**를 출력하세요.

## □ 요구사항

- `import re`
- `numbers = re.findall(r'\d+', "입력문자열")`

# 실습문제 16

---

## □ 문제

- 사용자로부터 문자열을 입력받아, 해당 문자열에서 단어의 수를 세는 함수를 작성하세요. 단어는 공백으로 구분됩니다.

## □ 요구사항

# 실습문제 17

---

## □ 문제

- 사용자로부터 정수를 입력받아, 해당 정수를 이진수로 재귀적으로 변환하는 함수를 작성하세요.

## □ 요구사항

- $5 \div 2 = 2$ , 나머지 1
- $2 \div 2 = 1$ , 나머지 0
- $1 \div 2 = 0$ , 나머지 1
- 나머지를 역순으로 나열하면 2진수 101

# 실습문제 18

## □ 문제

- 다음은 재귀 함수를 사용, 주식 투자 수익을 계산하는 문제입니다.
- 초기 투자 금액과 연간 수익률을 입력으로 받아,  $n$ 년 후의 투자 수익을 계산하는 함수를 작성하세요

## □ 요구사항

- 재귀함수 활용
- $n$ 년 후의 투자 수익은  $(1 + \text{연간 수익률}) * n-1$ 년 후의 투자 수익

# Geek & geek

