

# Parameter & Argument

소프트웨어 끝대 강의

청주대학교 노기섭 교수

([kafa46@cju.ac.kr](mailto:kafa46@cju.ac.kr))

# Contents

- Parameter와 Argument 개념 이해하기
- Terminal 환경에서 Argument 작동 원리
- Python STL Lib 이용한 Argument 관리
- Python argparse 모듈을 이용한 Argument 관리

# Parameter와 Argument 개념 이해하기

# Parameter 정의 및 특징

## ■ 정의

- 수학과 통계학에서 어떠한 시스템이나 함수의 특정한 성질을 나타내는 변수
- 일반적으로는  $\theta$ 라고 표현되며, 다른 표시는 각각 독특한 뜻을 지닌다.

([from online wiki](#))

- 매개변수(媒介變數), 파라미터(parameter), 모수(母數), 조변수(祖變數) 등 용어를 혼용해서 사용 (전부 맞다는 의미)

## ■ 특징

- Parameter 에 따라서 함수의 관계가 결정됨
- 우리나라 교과서에서는 상수로 표현하기도 함
- 기원은 종속변수이지만 외형은 독립변수
- 동형 구조를 결정하는 역할
- 구체적인 예시 → 다음 슬라이드 참고

뭔 말여??

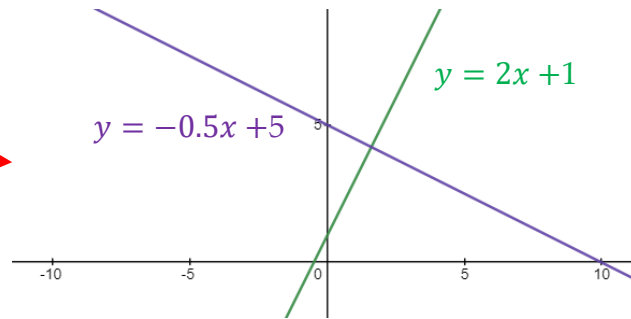


# Parameter 예시

## ■ 일차 함수

$$y = f(x) = ax + b,$$

where  $\theta = \{a, b\}$



## ■ 이차 함수

$$y = f(x) = ax^2 + bx + c,$$

where  $\theta = \{a, b, c\}$

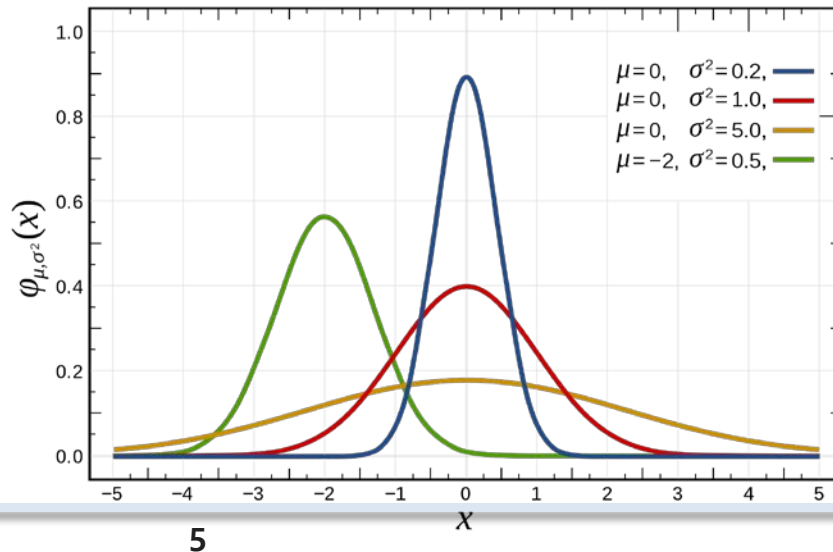


## ■ 확률밀도 함수 (정규 분포)

$$y = f(x) = N(X = x | \mu, \sigma^2)$$

$$= \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right],$$

where  $\theta = \{\mu, \sigma^2\}$



# Parameter 숨은 의미

## ■ 함수 특징을 결정짓는 매개(중매자)

- 매개(媒介): 둘 사이에서 양편의 관계를 맺어 줌

## ■ 함수에서 Parameter

- $y = f_{\theta}(x)$ 
  - 의미: Parameter  $\theta$ 를 갖는 함수  $f(x)$
  - $f(x)$ 는  $\theta$ 를 매개로  $x$ 와  $y$ 의 관계가 결정된다.

(예시)

$$\begin{aligned} y &= f_{\theta}(x) = ax + b, \text{ where } \theta = \{2, 3\} \\ &= f(x) = 2x + 3 \end{aligned}$$

# 지금까지 생각했던 Parameter

$$(예시) \ y = f(x) = ax^2 + bx + c$$

## ■ 지금까지 했던 방식

- Parameter  $a, b, c$  를 상수로 제시  $\rightarrow$  입력  $x$  가 있는 경우 출력  $y$  를 찾아라.

## ■ 앞으로 할 방식

- 입력  $x$  를 고정하고  $\rightarrow$  Parameter  $a, b, c$  를 변화시키면서 출력  $y$  를 관찰하라.

## ■ 인공지능 (머신러닝) 접근 법


- 입력  $x$ , 출력  $y$  가 포함된 과거 데이터를 주고  $\rightarrow$  파라미터  $\theta$  를 찾아라.

# 프로그래밍에서 Parameter 의미

## ■ Function in Programming

```
def add(x, y):  
    '''더하기 함수'''  
    result = x + y  
    return result  
  
def test():  
    value = add(2, 3)
```

- 수학에서 말하는 엄밀한 의미의 Parameter는 아닙니다.
- Parameter 'x' 와 'y'를 통해 add 함수의 입력/출력 관계를 매개한다.
- 그러므로 'x' 와 'y'는 add 함수의 Parameter (매개변수)이다.

- 
- Parameter 로 지정된 값을 argument라고 부릅니다.
  - 함수 test 에서는 parameter 값을 2와 3으로 지정하여 add 함수의 결과를 리턴 받습니다.
  - 여기서 2와 3은 add 함수의 arguments 입니다.



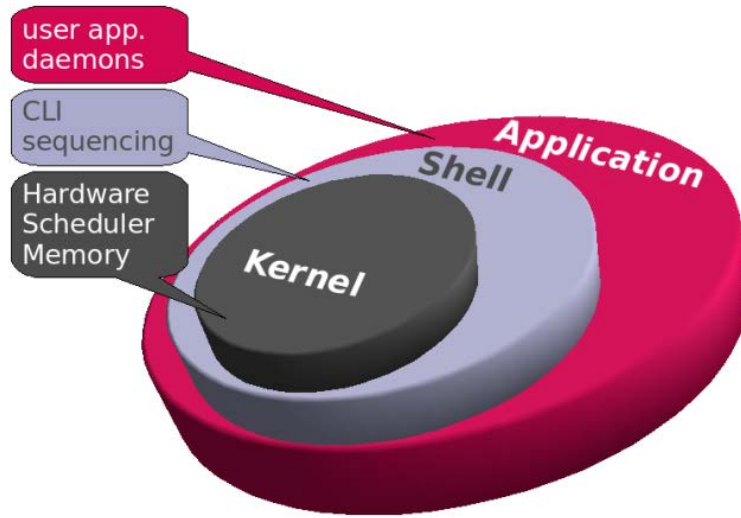
# Programming 관점에서 정리하면...

| Term      | Korean               | Meaning                                      |
|-----------|----------------------|--|
| Parameter | 매개변수                 | 함수와 메서드 입력 변수<br>(Variable) 이름(name)         |
| Argument  | 전달인자,<br>인자 값,<br>인자 | 함수와 메서드의 입력 값(Value)<br>해당 Parameter에 대입되는 값 |

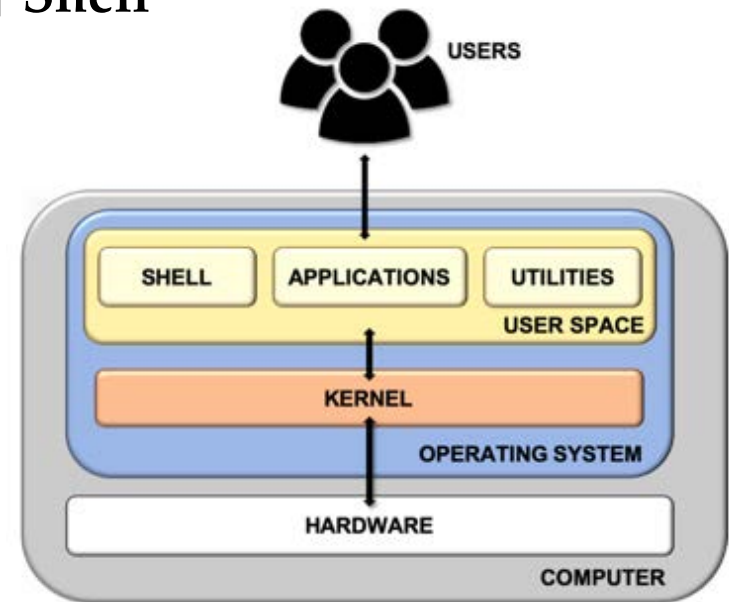
# Terminal 환경에서 Argument 작동 원리

# OS, Shell, Terminal

## ■ 컴퓨터 구조: Operating System (OS) 와 Shell



이미지 출처: <https://jaguhiremath62.medium.com/difference-between-kernel-and-shell-718b3de15be6>



이미지 출처: <https://www.futurelearn.com/info/courses/linux-for-bioinformatics/0/steps/202947>

- Shell: 운영체제 외부에서 사용자의 명령을 받아 해석 → OS 전달 → 결과 수령
- Shell을 이용하는 방법
  - CLI: Command Line Interface (PowerShell, Bash, Git Bash, 윈도우 CMD 창 등)
  - GUI: Graphic User Interface (파일 탐색기, 작업관리자, 네트워크 관리자 등)

# CLI 환경에서 Software를 작동하는 원리

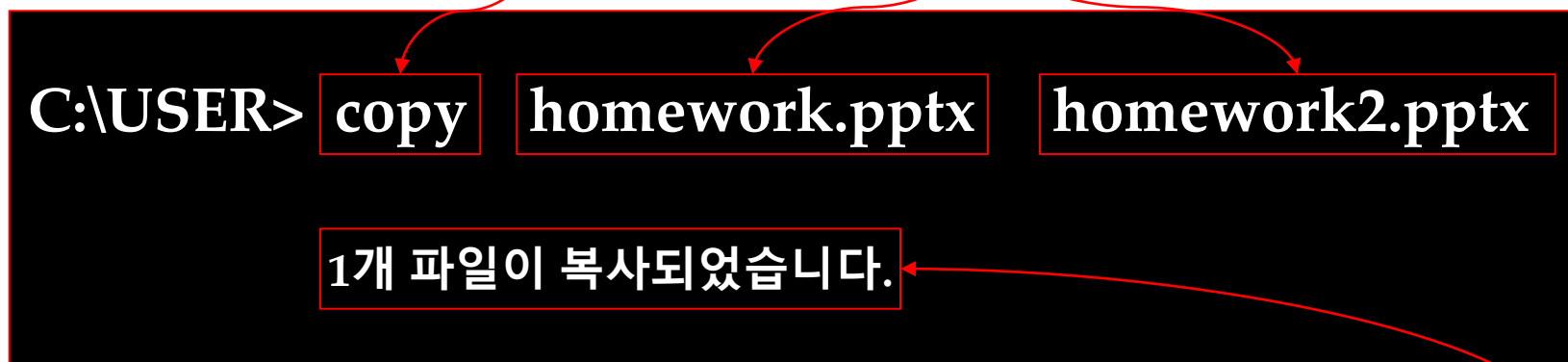
## ■ 터미널 환경에서 Shell을 통해 Software 작동시키는 방법

(예) 윈도우 OS에서 파일을 복사하는 'copy' 라는 소프트웨어를 작동시키려면?

(윈도우 CMD Shell 이용하는 경우)

OS에 전달할 명령어

명령어 실행에 필요한 Arguments



CMD Shell은 윈도우 운영체제에게 copy 소프트웨어 실행을 요청  
copy 소프트웨어 실행에 필요한 Arguments를 같이 전달

윈도우 운영체제가 수행한 결과를 받아 사용자에게 전달

# Python 명령어 실행하기

## ■ Python 설치했다는 의미?

- Python 언어를 작동하기 위한 다양한 라이브러리(library)를 설치하고
- 운영체제에 'python' 이라는 명령어를 등록했다는 의미

```
C:\USER> python
Python 3.11.5 (tags/v3.11.5:f3256153, Oct 9 2023,
16:14:13) [MSC v.1968 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license"
for more information.
>>>
```

OS에 'python' 이라는 명령어만 전달하고  
Argument는 전달하지 않은 상황  
→ Python Shell 실행

OS는 명령어에 해당하는 SW를 찾고  
SW 실행에 필요한 Parameters를  
전달받은 Arguments에서 찾아서 대입

```
C:\USER> python --version
Python 3.11.5
```

OS에 'python' 이라는 명령어와 '--version'  
이라는 Argument를 전달한 상황  
→ Python 버전을 알려줌

```
C:\USER> python test.py
Hello world!
```

OS에 'python' 이라는 명령어와 'test.py'라는  
Argument를 전달한 상황  
→ 'test.py' 스크립트를 찾아서 실행

# 내가 만든 Python 스크립트 실행하기

내가 만든 python script → add.py

```
def add(a, b):  
    result = a + b  
    print(f'Result: {result}')  
    return a + b  
  
if __name__ == '__main__':  
    add(2, 3)
```

문제점???



사용자가 원하는 Arguments를  
직접 전달하지 못한다!!

Solution 1.

사용자 대화형 코드 추가

```
if __name__ == '__main__':  
    data = input('두 수를 입력하세요: ')  
    a, b = data.split(' ')  
    print(f'a: {a}, b: {b}')  
    add(a, b)
```

터미널에서 add.py 실행

```
C:\USER> python add.py
```

Result: 5

OS에 'python' 이라는 명령어와 'add.py'라는  
Argument를 전달한 상황

→ 'add.py' 스크립트를 찾아서 실행

Solution 2.

스크립트 실행에 필요한 값을  
Arguments로 넘긴다.

```
C:\USER> python add.py 2 3  
Result: 6
```

# 교수님~ 뭐가 좋은 건가요^^. ??

## ■ User Interactive Code가 좋은 경우

- 프로그램이 간단할 경우
- 사용자 입력 요청이 몇 개 안될 경우
- 단일 모듈로 설계된 경우

## ■ Argument 전달이 좋은 경우

- 프로그램이 복잡하고 클 경우
  - 실행에 필요한 정보를 한꺼번에 제공 → 틀린 입력이 있다면 바로 알 수 있음
  - 기본(default) 설정 가능
- 프로그램 실행 중 사용자의 선택에 따라 실행해야 하는 경우
  - 사용자에게 언제 물어볼지 모르기 때문에 계속 기다림?
  - 한참 돌아가다가 사용자가 잘못 입력하면?
    - 처음부터 다시 실행하여 물어볼 때까지 기다림?

# Python STL Lib 이용한 Argument 관리



# 터미널에서 Argument 전달받는 방법

## ■ Python에서 제공하는 표준 라이브러리 사용

- sys.argv

## ■ 사용법

- 터미널에서 argument를 입력하면 리스트로 데이터를 python 스크립트 내부로 전달
- Python 스크립트는 전달 받은 argument를 프로그램 내부 작동에 활용

## ■ 실습: 구구단 프로그램

- 사용자 입력에 따라 원하는 단을 출력
- 2~9 숫자 중 원하는 구구단을 입력 (다수 입력 가능)
- 아무것도 입력하지 않을 경우 2단부터 9단까지 모두 출력
- 컬럼(세로) 수를 지정할 수 있어야 함

# Python argparse 모듈을 이용한 Argument 관리

# Python에서 제공하는 표준 라이브러리

## ■ Python에서 제공하는 표준 라이브러리는 매우 불편

- 대부분의 parsing 처리를 개발자가 해야 함
  - 필수 입력, 선택 입력, 기본 값 설정 등
- 에러처리도 개발자가 해야 함
- 도움말 처리도 개발자가 해야 함

## ■ 결론

- sys.argv 는 간단한 argument 전달에만 사용하자!
- 보다 편리한 모듈이 있다면 가져다 쓰자!
- solution → argparse 모듈 (다음 슬라이드 참고)

# argparse

## ■ argparse 모듈

- Python 프로그램을 CLI 환경에서 쉽게 사용하도록 도와준다.
- 프로그램에 필요한 인자(argument)를 정의하면 sys.argv를 어떻게 파싱할지를 알려줌

내가 만든 파이썬 프로그램(예시)

```
import sys

file_path = sys.argv[1]

if len(sys.argv) != 2:
    print("Insufficient arguments")
    sys.exit()

print("File path : " + file_path)
```

파이썬 프로그램 실행 (CLI 환경)

```
>>> python test.py /home/limsee/test.json

"File path : /home/limsee/test.json"
```

sys.argv[0] → 내 프로그램의 이름(스크립트명)이 전달  
sys.argv[1] → 첫번째 전달 인자(argument)

python 파일명 옵션들...

```
python train.py --epochs 50 --batch-size 64 --save-dir weights
```

머신러닝/딥러닝에서 흔히 사용하는 명령어....

위와 같은 parameter와 argument 값을 편리하게 이용할 수 있을까?

# argparse 사용

## ■ argparse는 python에서 기본적으로 제공

### 내가 만든 Python 프로그램

```
import argparse

parser = argparse.ArgumentParser(description='Argparse Tutorial')
# argument는 원하는 만큼 추가한다.
parser.add_argument('--print-number', type=int,
                    help='an integer for printing repeatably')

args = parser.parse_args()

for i in range(args.print_number):
    print('print number {}'.format(i+1))
```

- 인자의 이름에는 -와 \_을 쓸 수 있다.
- 단, python 기본 문법은 변수명에 -를 허용하지 않기 때문에,
- 인자의 이름에 -가 들어갔다면
- **args**인자로 접근하려면 -를 \_로 바꿔 주어야 한다.

### CLI 환경에서 실행한 결과

```
> python argparseTest.py -h
usage: argparseTest.py [-h] [--print-number PRINT_NUMBER]

Argparse Tutorial

optional arguments:
  -h, --help            show this help message and exit
  --print-number PRINT_NUMBER
                        an integer for printing repeatably

> python argparseTest.py --print-number 5
print number 1
print number 2
print number 3
print number 4
print number 5
```

# argparse 인자값 설명

## 클래스

### ArgumentParser()

해당 객체에는 아래와 같이 입력받고 있습니다.

- **prog**: 프로그램의 이름 (기본값: `sys.argv[0]`)
  - 기본값으로 실행한 스크립트파일명을 노출. 작성 시 스크립트 파일 대신 입력한 값이 노출
- **usage**: 프로그램 사용법을 설명하는 문자열 (기본값: 파서에 추가된 인자로부터 만들어지는 값)
  - 사용방법을 노출. 기본값으로 실행한 파일 + 입력한 인자값들을 노출
- **description**: 인자 도움말 전에 표시할 텍스트 (기본값: none)
  - 스크립트에 `-h` 옵션을 주어 실행 시, usage 아래에 노출
- **epilog**: 인자 도움말 후에 표시할 텍스트 (기본값: none)
- **parents**: `ArgumentParser` 객체들의 리스트이고, 이 들의 인자들도 포함
- **formatter\_class**: 도움말 출력을 사용자 정의하기 위한 클래스
- **prefix\_chars**: 선택 인자 앞에 붙는 문자 집합 (기본값: '-').
- **fromfile\_prefix\_chars**: 추가 인자를 읽어야 하는 파일 앞에 붙는 문자 집합 (기본값: `None`).
- **argument\_default**: 인자의 전역 기본값 (기본값: `None`)
- **conflict\_handler**: 충돌하는 선택 사항을 해결하기 위한 전략 (일반적으로 불필요함)
- **add\_help**: 파서에 `-h/--help` 옵션을 추가 (기본값: `True`)
- **allow\_abbrev**: 약어가 모호하지 않으면 긴 옵션을 축약할 수 있도록 함. (기본값: `True`)

## ArgumentParser 클래스의 메서드

### add\_argument()

해당 메서드는 아래와 같이 입력받고 있습니다.

- **name or flags**: 옵션 문자열의 이름이나 리스트, 예를 들어 `foo` 또는 `-f, --foo`.
- **action**: 명령행에서 이 인자가 발견될 때 수행 할 액션의 기본형.
- **nargs**: 소비되어야 하는 명령행 인자의 수.
- **const**: 일부 action 및 nargs 를 선택할 때 필요한 상숫값.
- **default**: 인자가 명령행에 없는 경우 생성되는 값.
- **type**: 명령행 인자가 변환되어야 할 형.
- **choices**: 인자로 허용되는 값의 컨테이너.
- **required**: 명령행 옵션을 생략 할 수 있는지 아닌지 (선택적일 때만).
- **help**: 인자가 하는 일에 대한 간단한 설명.
- **metavar**: 사용 메시지에 사용되는 인자의 이름.
- **dest**: `parse_args()` 가 반환하는 객체에 추가될 어트리뷰트의 이름.

# argparse 실습



수고하셨습니다 ..^^..