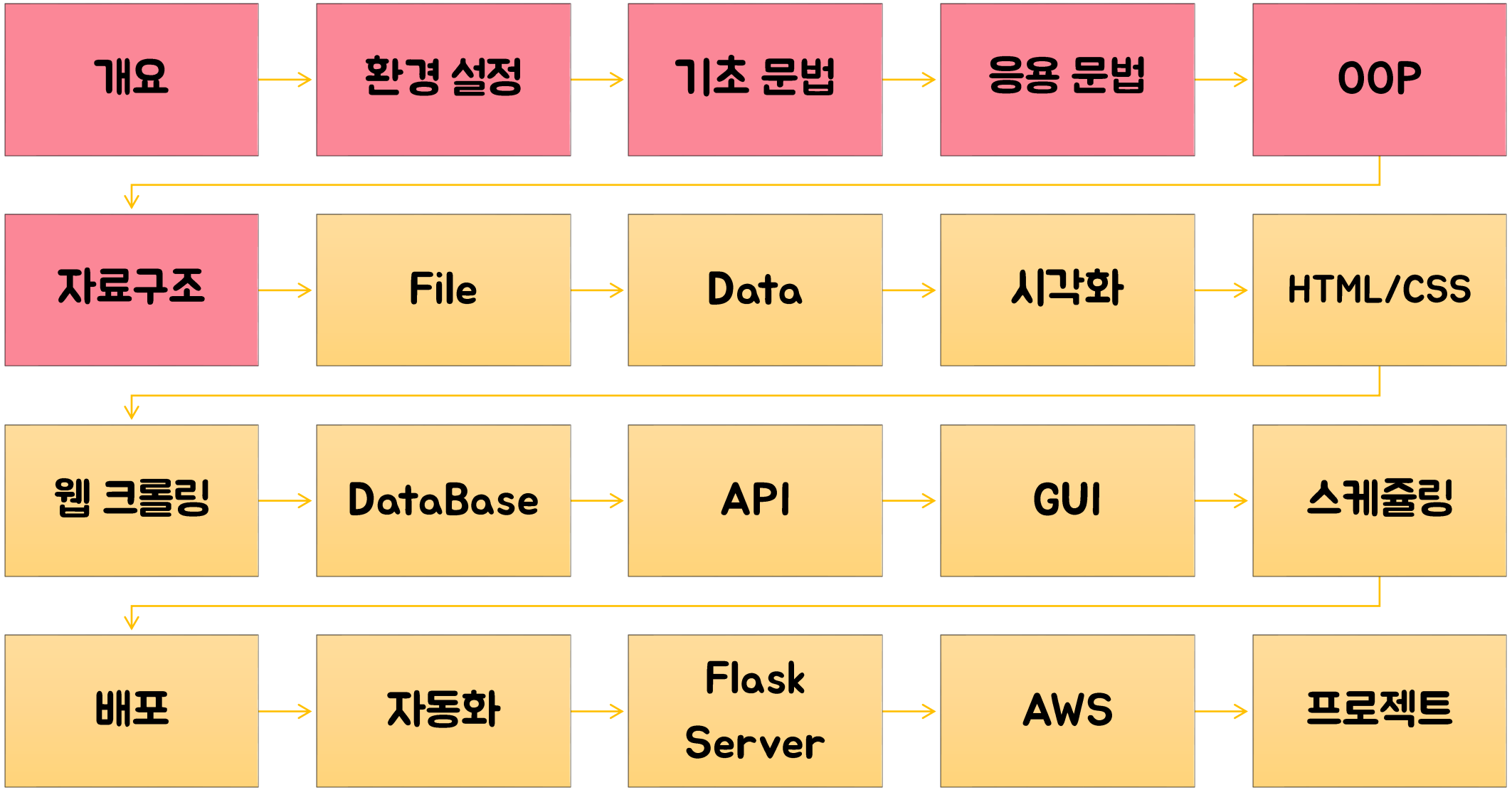
python

대한상공회의소
서울기술교육센터

나 예 호 교수

python 수업 진행

대한상공회의소
서울기술교육센터



```python
file = open("sample.txt", "r")   # 읽기 모드로 파일 열기
content = file.read()            # 파일 내용 읽기
print(content)
file.close()                     # 파일 닫기 (꼭 필요!)
```

"r" : 읽기 모드 (기본값)
"w": 쓰기 모드 (기존 내용 삭제 후 새로 작성)
"a": 추가 모드 (기존 내용 유지, 추가로 작성)
"b": 바이너리 모드

```python
# 모든 내용을 한 번에 읽기
file = open("sample.txt", "r")
print(file.read())
file.close()

# 한 줄씩 읽기
file = open("sample.txt", "r")
print(file.readline())
file.close()

# 모든 줄을 리스트로 읽기
file = open("sample.txt", "r")
lines = file.readlines()
print(lines)
file.close()
```

```
Python is fun!
Learning file I/O.
Keep practicing!
```

```
Python is fun!
```

```
['Python is fun!\n', 'Learning file I/O.\n', 'Keep practicing!']
```

```python
# 파일에 문자열 쓰기 (기존 내용 삭제됨)
file = open("output.txt", "w")
file.write("Hello, Python!\n")
file.write("파일 입출력 예제입니다.")
file.close()

# 여러 줄 쓰기
lines = ["첫 번째 줄\n", "두 번째 줄\n", "세 번째 줄\n"]
file = open("output.txt", "w")
file.writelines(lines)
file.close()
```

with open구문

file close 실수를 미연에 방지할 수 있음

```
with open("output.txt", "r") as file:
    content = file.read()
    print(content)  # 파일 자동 닫힘
```

have_it_all.txt 파일을 만들고 다음 내용을 저장한 후,
다시 읽어 출력하세요.

All you can imagine
All, no matter what your path is
If you believe it then anything can happen

number.txt 파일을 만들고 1~100까지의 숫자를 한 줄에 하나씩 파일에 저장 후, 다시 읽어서 출력하시오

```python
# 1~100까지 숫자 저장하기
with open("numbers.txt", "w") as file:
    for i in range(1, 101):
        file.write(f"{i}\n")

# 저장된 숫자 읽어서 출력하기
with open("numbers.txt", "r") as file:
    for line in file:
        print(line.strip())  # 줄 끝의 개행 문자 제거 후 출력
```

exam.txt 파일을 열고
개행 문자 제거 후 첫 번째 줄만 출력하시오

```
# sample.txt 파일 읽기
with open("sample.txt", "r") as file:
    first_line = file.readline()  # 첫 번째 줄만 읽기
    print(first_line.strip())     # 개행 문자 제거 후 출력
```

exam.txt 파일을 열고
각 줄의 문자 수(공백 포함) 출력하기

```
줄 1의 문자 수: 14
줄 2의 문자 수: 18
줄 3의 문자 수: 16
```

```python
# sample.txt 파일에서 각 줄의 문자 수 출력하기
with open("sample.txt", "r") as file:
    for line_number, line in enumerate(file, start=1):
        char_count = len(line.strip())  # 개행 문자 제거 후 문자 수 계산
        print(f"줄 {line_number}의 문자 수: {char_count}")
```

enumerate()

- 순서가 있는 자료형(list, set, tuple, dictionary, string)을 입력으로 받았을 때, 인덱스와 값을 포함하여 리턴
- 인덱스와 값을 동시에 접근

output.txt 파일에
사용자가 입력한 문장 3개를 저장하는 프로그램을 작성하세요.

```
1번째 문장을 입력하세요: 오늘의
2번째 문장을 입력하세요: 점심 메뉴는
3번째 문장을 입력하세요: 뭘까요?
문장이 output.txt 파일에 저장되었습니다.
```

```python
# 사용자로부터 입력받은 문장 3개를 저장하기
with open("output.txt", "w") as file:
    for i in range(3):
        sentence = input(f"{i+1}번째 문장을 입력하세요:")
        file.write(sentence + "\n")

print("문장이 output.txt 파일에 저장되었습니다.")
```

```
text = "    Hello, Python!    "
print(text.strip())    # 앞뒤 공백 제거
print(text.lstrip())   # 왼쪽 공백 제거
print(text.rstrip())   # 오른쪽 공백 제거
```

```
Hello, Python!
Hello, Python!
    Hello, Python!
```

sample2.txt 파일에서
Python이라는 단어가 포함된 줄만 찾아 출력하기

```
I love Python programming.
Python is versatile and powerful.
```

```python
# sample2.txt 파일 읽기
with open("sample2.txt", "r") as file:
    for line in file:
        if "Python" in line:  # 'Python'이 포함된 줄 찾기
            print(line.strip())
```

split(): 문자열을 리스트로 분리
join(): 리스트를 다시 문자열로 결합

```python
sentence = "Python is fun"
words = sentence.split()              # 공백 기준으로 분리
print(words)                          # ['Python', 'is', 'fun']

new_sentence = " ".join(words)        # 리스트를 공백으로 연결
print(new_sentence)                   # Python is fun
```

사용자에게 입력받은 문자열을 공백 기준으로 나눠 리스트로 출력하기

```
문자열을 입력하세요:  I Love you
['I', 'Love', 'you']
```

```python
# 사용자 입력 받기
user_input = input("문자열을 입력하세요: ")

# 공백 기준으로 나눠 리스트로 변환
words = user_input.split()

# 결과 출력
print(words)
```

fruits = ["apple", "banana", "cherry"]
리스트의 요소를 연결해
"apple,banana,cherry"과 같이 출력하시오

```python
# 리스트 정의
fruits = ["apple", "banana", "cherry"]

# 쉼표로 구분된 문자열로 변환
result = ",".join(fruits)

# 결과 출력
print(result)
```

replace(): 문자열에서 특정 문자를 다른 문자로 교체

```
text = "I love Java"
new_text = text.replace("Java", "Python")
print(new_text)  # I love Python
```

```
I love Python
```

문자열 "Hello, Python!"에서 Python을 World로 변경하기

```
Hello, World!
```

```python
# 문자열 정의
text = "Hello, Python!"

# 문자열 변경 (replace 사용)
new_text = text.replace("Python", "World")

# 결과 출력
print(new_text)
```

코드 작성 시 발생할 수 있는 예기치 않은 상황들을 처리
프로그램의 안정성과 신뢰성을 높이기 위해 사용

try : 오류가 발생할 수 있는 코드
except : 오류 발생 시 처리할 코드
finally : 오류 발생 여부와 관계 없이 항상 실행

```
try:
    number = int(input("숫자를 입력하세요: "))
    print(10 / number)
except ZeroDivisionError:
    print("0으로 나눌 수 없습니다.")
except ValueError:
    print("숫자가 아닌 값을 입력했습니다.")
finally:
    print("프로그램 종료")
```

```
try:
    with open("nonexistent.txt", "r") as file:
        content = file.read()
except FileNotFoundError:
    print("파일이 존재하지 않습니다.")
```

사용자에게 두 개의 숫자를 입력 받아 나눗셈을 수행하세요.
단, 0으로 나누려고 하면 오류 메시지를 출력하세요.

```
첫 번째 숫자를 입력하세요:  7
두 번째 숫자를 입력하세요:  0
오류: 0으로 나눌 수 없습니다.
```

```python
try:
    num1 = int(input("첫 번째 숫자를 입력하세요: "))
    num2 = int(input("두 번째 숫자를 입력하세요: "))
    result = num1 / num2
    print(f"결과: {result}")
except ZeroDivisionError:
    print("오류: 0으로 나눌 수 없습니다.")
```

리스트에서 존재하지 않는 인덱스에 접근할 때 오류 처리하기

```
인덱스를 입력하세요 (0~4):  10
오류: 존재하지 않는 인덱스입니다.
```

```python
my_list = [10, 20, 30, 40, 50]
try:
    index = int(input("인덱스를 입력하세요 (0~4): "))
    print(f"선택한 값: {my_list[index]}")
except IndexError:
    print("오류: 존재하지 않는 인덱스입니다.")
```

숫자를 입력받아 int()로 변환할 때 문자열을 입력하면 오류 처리하기

```
정수를 입력하세요:  ㅋ
오류: 정수를 입력해야 합니다.
```

```python
try:
    number = int(input("정수를 입력하세요: "))
    print(f"입력한 숫자: {number}")

except ValueError:
    print("오류: 정수를 입력해야 합니다.")
```

python os(Operating System)

- 파일 시스템, 디렉토리, 환경 변수, 경로 등
  운영체제와 상호작용할 수 있는 기능 제공
- 파일과 폴더를 생성, 삭제, 이동, 이름 변경 등의 작업 자동화 가능

```
import os
```

getcwd() → Current Working Directory 확인
현재 파일이나 폴더가 어디에 있는지 경로를 알려줌.
* 파일을 읽고 쓸 때 기준이 됨

```python
import os

# 현재 작업 디렉토리 확인
print("현재 작업 디렉토리:", os.getcwd())
```

```
현재 작업 디렉토리: /Users/yehona/Desktop/workspace2
```

폴더 생성: mkdir() → 새로운 폴더 생성
폴더 삭제: rmdir() → 빈 폴더 삭제
단, rmdir()은 빈 폴더만 삭제 가능

```
# 폴더 생성
os.mkdir("new_folder")  # "new_folder"라는 폴더 생성
print("폴더 생성 완료:", os.listdir("."))

# 폴더 삭제
os.rmdir("new_folder")  # 빈 폴더만 삭제 가능
print("폴더 삭제 완료:", os.listdir("."))
```

listdir() → 특정 폴더 내의 파일 및 폴더 목록 반환

```
# 현재 폴더의 파일 목록 출력
print("현재 폴더의 파일 목록:", os.listdir("."))

# 특정 폴더 내 목록 출력
print("Documents 폴더 목록:", os.listdir("Documents"))
```

파일 이름 및 폴더 이름 변경

```
# 파일 이름 변경
os.rename("old_name.txt", "new_name.txt")

# 폴더 이름 변경
os.rename("old_folder", "new_folder")
```

practice_folder라는 이름의 새로운 폴더를 생성하시오

```
폴더 'practice_folder' 생성 완료!
```

```python
# 폴더 생성
folder_name = "practice_folder"

# 폴더가 이미 존재하는지 확인하고 생성
if not os.path.exists(folder_name):
    os.mkdir(folder_name)
    print(f"폴더 '{folder_name}' 생성 완료!")
else:
    print(f"폴더 '{folder_name}'가 이미 존재합니다.")
```

practice_folder 안에 test.txt 파일을 만들고 폴더 내 파일 목록 출력하기

* os.path.join() → 운영체제에 맞게 경로 결합

```
파일 'practice_folder/test.txt' 생성 완료!
'practice_folder' 폴더 내 파일 목록: ['test.txt']
```

python os 실습

대한상공회의소
서울기술교육센터

```python
# 파일 경로 설정
file_path = os.path.join(folder_name, "test.txt")

# 파일 생성 및 내용 작성
with open(file_path, "w") as file:
    file.write("이것은 테스트 파일입니다.\nPython으로 파일 자동화 연습 중입니다.")

print(f"파일 '{file_path}' 생성 완료!")

# 폴더 내 파일 목록 출력
files_in_folder = os.listdir(folder_name)
print(f"'{folder_name}' 폴더 내 파일 목록:", files_in_folder)
```

test.txt의 이름을 renamed_test.txt로 변경하기

```
파일 이름이 'practice_folder/test.txt' → 'practice_folder/renamed_test.txt'로 변경되었습니다.
```

```python
# 원본 파일과 새 파일 경로 설정
original_file = os.path.join(folder_name, "test.txt")
renamed_file = os.path.join(folder_name, "renamed_test.txt")

# 파일 이름 변경
if os.path.exists(original_file):
    os.rename(original_file, renamed_file)
    print(f"파일 이름이 '{original_file}' → '{renamed_file}'로 변경되었습니다.")
else:
    print("변경할 파일이 존재하지 않습니다.")
```