"Python 기초부터 Flask mini프로젝트까지

실습중심 Python 30일 완전정복!!!"



**목차**

파이썬 소개  
1.1 파이썬이란?  
1.2 파이썬의 특징  
1.3 파이썬과 컴파일러 언어(C, C++, Java)의 비교

파이썬 기초 문법  
2.1 자료형  
2.2 입출력  
2.3 들여쓰기  
2.4 조건문  
2.5 반복문  
2.6 함수  
2.7 모듈  
2.8 슬라이싱  
2.9 map 함수  
2.10 예외 처리  
2.11 클래스

라이브러리 활용법  
3.1 OpenCV와 Pillow 기본 활용법  
3.2 이미지 처리 관련 활용 예제 5가지  
3.3 실행 결과 설명

심화 과정: Flask 프로젝트  
4.1 설치해야 하는 라이브러리 및 설치 이유  
4.2 Flask 기본 코드와 상세 해설  
4.3 MySQL 연동 게시판 구현  
4.4 Flask 서버 디렉토리 구조와 설정 - HTML, CSS, SASS, JS와 Python 통합  
4.5 통합구현 : Flask를 활용한 게시판 프로젝트

**파이썬 소개**

**1.1 파이썬이란?**

파이썬은 1991년 귀도 반 로섬(Guido van Rossum)이 개발한 범용 프로그래밍 언어로, 간결하고 직관적인 문법으로 유명합니다. 초보자와 전문가 모두 사용하며 데이터 분석, 웹 개발, 머신러닝 등 다양한 분야에서 활용됩니다.

**1.2 파이썬의 특징**

**간결한 문법**: 영어처럼 읽히는 코드 작성이 가능.

**인터프리터 기반**: 코드가 한 줄씩 실행되며 테스트 및 디버깅이 쉬움.

**동적 타이핑**: 변수의 자료형을 미리 선언할 필요 없음.

**광범위한 라이브러리 지원**: 복잡한 작업을 간단하게 처리 가능.

**1.3 파이썬과 컴파일러 기반 언어의 비교**

**파이썬 기초 문법**

**2.1 자료형**

- 주요 자료형

- 숫자형: int, float, complex

- 문자열: str

- 리스트: list

- 튜플: tuple

- 딕셔너리: dict

- 불리언: bool

**2.1.1 숫자형(Number)**

**설명**

파이썬에서 숫자형은 정수(int), 실수(float), 복소수(complex)로 나뉩니다.

int: 정수를 나타냅니다. 예: 5, -3, 0

float: 실수를 나타냅니다. 예: 3.14, -0.5, 1.0

complex: 복소수를 나타냅니다. 예: 3+4j

예제

# 정수형

a = 10

print(type(a)) # <class 'int'>

# 실수형

b = 3.14

print(type(b)) # <class 'float'>

# 복소수형

c = 2 + 3j

print(type(c)) # <class 'complex'>

실습 문제

문제 1

아래 코드의 결과를 예측하세요.

x = 10

y = 3.0

result = x \* y

print(type(result))

문제 2

사용자로부터 두 개의 정수를 입력받아 곱셈 결과를 출력하는 프로그램을 작성하세요.

**정답 코드 및 풀이**

**문제 1 정답**

x = 10

y = 3.0

result = x \* y

print(type(result)) # <class 'float'>

**풀이**:  
x는 정수형(int), y는 실수형(float)입니다. 정수와 실수를 곱하면 결과는 실수형(float)으로 반환됩니다.

**문제 2 정답**

# 두 정수를 입력받아 곱셈 결과 출력

num1 = int(input("첫 번째 정수를 입력하세요: "))

num2 = int(input("두 번째 정수를 입력하세요: "))

result = num1 \* num2

print(f"곱셈 결과: {result}")

**풀이**

input() 함수로 문자열을 입력받습니다.

int()로 문자열을 정수로 변환합니다.

두 정수를 곱한 결과를 result에 저장하고 출력합니다.

2.1.2 문자열형(string)

**설명**

파이썬의 문자열은 작은따옴표( ‘ )나 큰따옴표( “ )로 묶어 표현하며, 문자들의 연속된 데이터입니다.

예제

s = "Hello, Python"

print(type(s)) # <class 'str'>

# 문자열 길이

print(len(s)) # 13

# 슬라이싱

print(s[0:5]) # Hello

실습문제

문제 1

"Learning Python" 문자열의 길이를 출력하세요.

문제 2

사용자가 입력한 문자열을 거꾸로 출력하는 프로그램을 작성하세요.

**정답 코드 및 풀이**

**문제 1 정답**

s = "Learning Python"

print(len(s)) # 15

**풀이**

문자열의 길이는 len() 함수를 사용하여 구할 수 있습니다.

**문제 2 정답**

# 사용자 입력 문자열 거꾸로 출력

user\_input = input("문자열을 입력하세요: ")

reversed\_string = user\_input[::-1]

print(f"거꾸로 출력: {reversed\_string}")

**풀이**

input()으로 문자열을 입력받습니다.

문자열 슬라이싱을 이용하여 [::-1]로 역순 문자열을 생성합니다.

2.1.3 리스트(List)

설명

리스트는 대괄호( [ ] )로 감싸서 표현하며, 여러 자료형을 저장할 수 있습니다.

예제

lst = [1, 2, 3, "Python", 3.14]

print(type(lst)) # <class 'list'>

# 요소 접근

print(lst[0]) # 1

# 요소 변경

lst[0] = 100

print(lst) # [100, 2, 3, 'Python', 3.14]

실습문제

문제 1

아래 리스트에서 ‘Python’을 ‘Programming’으로 변경하세요.

languages = ["C", "Java", "Python"]

문제 2

사용자로부터 5개의 숫자를 입력받아 리스트에 저장하고, 그 합을 출력하는 프로그램을 작성하세요.

**정답 코드 및 풀이**

**문제 1 정답**

languages = ["C", "Java", "Python"]

languages[2] = "Programming"

print(languages) # ["C", "Java", "Programming"]

**풀이**

리스트의 요소는 인덱스를 통해 접근하여 값을 변경할 수 있습니다.

**문제 2 정답**

# 5개의 숫자 입력받기

numbers = []

for i in range(5):

num = int(input(f"{i+1}번째 숫자를 입력하세요: "))

numbers.append(num)

# 합계 출력

total = sum(numbers)

print(f"입력한 숫자의 합: {total}")

**풀이**

for 반복문을 사용하여 5개의 숫자를 입력받습니다.

append( )로 리스트에 숫자를 추가합니다.

sum( ) 함수를 사용해 합계를 구합니다.

2.1.4 튜플(Tuple)

설명

튜플은 리스트와 유사하지만, 변경할 수 없는 자료형입니다.

선언: 소괄호 ( )를 사용하여 선언합니다.

읽기 전용 데이터가 필요할 때 사용합니다.

예제

# 튜플 선언

tpl = (1, 2, 3, "Python")

print(type(tpl)) # <class 'tuple'>

# 요소 접근

print(tpl[0]) # 1

# 요소 변경 불가

# tpl[0] = 100 # 오류 발생!

실습 문제

문제 1

다음 튜플에서 "Python" 값을 출력하세요.

languages = ("C", "Java", "Python", "Ruby")

문제 2

사용자로부터 3개의 숫자를 입력받아 튜플로 저장하고, 그 합계를 출력하는 프로그램을 작성하세요.

**정답 코드 및 풀이**

**문제 1 정답**

languages = ("C", "Java", "Python", "Ruby")

# "Python" 출력

print(languages[2]) # Python

**풀이**:

튜플의 요소는 리스트와 동일하게 **인덱스를 통해 접근**할 수 있습니다.

"Python"은 튜플의 2번 인덱스에 위치하므로 languages[2]를 출력하면 됩니다.

**문제 2 정답**

# 사용자 입력 숫자를 튜플에 저장

num1 = int(input("첫 번째 숫자를 입력하세요: "))

num2 = int(input("두 번째 숫자를 입력하세요: "))

num3 = int(input("세 번째 숫자를 입력하세요: "))

# 튜플 생성

numbers = (num1, num2, num3)

# 합계 출력

total = sum(numbers)

print(f"입력한 숫자의 합: {total}")

**풀이**:

input()으로 입력받은 문자열을 int()로 변환해 정수로 만듭니다.

입력된 숫자를 튜플로 묶어 numbers에 저장합니다.

sum() 함수로 튜플의 합을 구하고 출력합니다.

2.1.5 딕셔너리(Dictionary)

설명

딕셔너리는 {key: value} 쌍으로 데이터를 저장하며, 키 (key)를 이용해 값을 (value) 접근하거나 수정할 수 있습니다.

**키(key)**: 유일해야 하며, 수정할 수 없는 값(예: 문자열, 숫자, 튜플)만 사용 가능합니다.

**값(value)**: 어떤 자료형도 가능합니다.

딕셔너리는 빠른 데이터 검색과 수정이 필요한 경우 유용합니다.

**딕셔너리 주요 메서드**

dict.get(key, default) : 키에 해당하는 값을 가져옵니다. 키가 없으면 기본값(default)을 반환합니다.

dict.keys() : 딕셔너리의 모든 키를 반환합니다.

dict.values() : 딕셔너리의 모든 값을 반환합니다.

dict.items() : 키-값 쌍을 튜플로 반환합니다.

예제 1

딕셔너리 생성 및 값 접근

# 딕셔너리 생성

person = {"name": "Alice", "age": 25, "city": "Seoul"}

# 값 접근

print(person["name"]) # Alice

print(person.get("age")) # 25

print(person.get("country", "Unknown")) # Unknown (키가 없을 때 기본값)

예제 2

딕셔너리 수정 및 삭제

# 값 수정

person["age"] = 26

# 키-값 추가

person["job"] = "Engineer"

# 키-값 삭제

del person["city"]

print(person)

# 결과: {'name': 'Alice', 'age': 26, 'job': 'Engineer'}

실습 문제

문제 1

다음 딕셔너리에서 “age” 키의 값을 출력하세요.

person = {"name": "Alice", "age": 25, "city": "Seoul"}

문제 2

사용자로부터 키와 값을 입력받아 딕셔너리에 추가하고, 딕셔너리 전체를 출력하는 프로그램을 작성하세요.

**정답 코드 및 풀이**

**문제 1 정답**

person = {"name": "Alice", "age": 25, "city": "Seoul"}

# "age" 키의 값 출력

print(person["age"]) # 25

풀이

딕셔너리의 값은 키를 이용해 접근할 수 있습니다.

“age” 키에 연결된 값 25가 출력됩니다.

**문제 2 정답**

# 빈 딕셔너리 생성

user\_dict = {}

# 키와 값 입력받기

key = input("추가할 키를 입력하세요: ")

value = input("추가할 값을 입력하세요: ")

# 딕셔너리에 추가

user\_dict[key] = value

# 딕셔너리 출력

print(f"현재 딕셔너리: {user\_dict}")

**풀이**:

빈 딕셔너리를 생성합니다.

input()으로 키와 값을 입력받습니다.

딕셔너리[키] = 값 형식으로 딕셔너리에 항목을 추가합니다.

추가된 딕셔너리를 출력합니다.

2.1.6 불리언(Boolean)

불리언 자료형은 True (참) 또는 False (거짓)의 두 가지 값만 가집니다.

주로 조건만이나 비교 연산에서 사용됩니다.

Bool( ) 함수로 값을 불리언으로 변환할 수 있습니다.

불리언 주요 특징

참(Truthy) : 비어있지 않은 데이터(예: 숫자 1, 문자열 “Hello”, 리스트 [1, 2] 등).

거짓(Falsy) : 비어있는 데이터(예: 0, 빈 문자열 “”, 빈 리스트 [ ] 등).

예제 1

비교 연산과 불리언 값

x = 10

y = 5

print(x > y) # True

print(x == y) # False

print(x != y) # True

예제 2

Bool( ) 함수 사용

print(bool(0)) # False

print(bool(1)) # True

print(bool("")) # False

print(bool("Python")) # True

실습 문제

문제 1

아래 코드를 실행 했을 때 결과를 예측하세요.

x = 10

y = 5

print(x > y)

print(x < y)

문제 2

사용자로부터 나이를 입력받아, 입력된 나이가 18세 이상이면 “성인입니다.”를 출력하고 그렇지 않으면 “미성년자입니다.”를 출력하세요.

**정답 코드 및 풀이**

**문제 1 정답**

**풀이**

1. x > y는 10이 5보다 크므로 True입니다.

2. x < y는 10이 5보다 작지 않으므로 False입니다.

**문제 2 정답**

# 문제 2

age = int(input("나이를 입력하세요: "))

if age >= 18:

print("성인입니다")

else:

print("미성년자입니다")

**풀이**

입력된 나이를 if 조건문으로 비교합니다. 18 이상이면 "성인입니다"를, 그렇지 않으면 "미성년자입니다"를 출력합니다.

2.2 입출력 (print, input)

설명

2.2.1 출력 - print( )

print( ) 함수는 화면에 데이터를 출력하는데 사용됩니다.

여러 값을 출력할 때는 쉼표(,)로 구분하며, 기본적으로 띄어쓰기가 포함됩니다.

문자열을 출력하려면 작은따옴표(')나 큰따옴표(")로 감쌉니다.

sep : 출력할 값들 사이에 삽입할 구분자를 설정합니다. 기본값은 공백입니다.

end : 출력 마지막에 삽입할 문자열을 설정합니다. 기본값은 줄바꿈(\n)입니다.

2.2.2 입력 – input( )

input( ) 함수는 사용자로부터 데이터를 입력받는데 사용됩니다.

입력받은 데이터는 기본적으로 **문자열**로 처리됩니다.

숫자 입력이 필요하면 int() 또는 float()로 변환해야 합니다.

name = input("이름을 입력하세요: ") # 사용자로부터 문자열 입력받기

age = int(input("나이를 입력하세요: ")) # 정수 입력받기

print(f"안녕하세요, {name}님. 당신의 나이는 {age}세입니다.")

예제 1

print("Hello, World!") # 문자열 출력

print(10, 20, 30) # 여러 값 출력

print("Python", "is", "fun", sep="-") # 구분자 설정

print("Hello", end=" ") # 줄바꿈 대신 공백 추가

print("World!")

예제 2 – 기본출력

print("안녕하세요!")

print("오늘은", "좋은 날입니다.")

print("Python", "is", "great!", sep=" | ", end="!!!\n")

**출력 결과**

안녕하세요!

오늘은 좋은 날입니다.

Python | is | great!!!!

예제 3 – 사용자 입력 처리

name = input("이름을 입력하세요: ")

age = int(input("나이를 입력하세요: "))

print(f"안녕하세요, {name}님. {age}살 이군요!")

**실행 흐름(예제 입력)**

이름을 입력하세요: Alice

나이를 입력하세요: 25

**출력 결과**

안녕하세요, Alice. 25살 이군요!

실습 문제

문제 1

사용자로부터 두 개의 숫자를 입력받아, 두 숫자의 합을 출력하세요.

문제 2

사용자에게 이름과 직업을 입력받아 다음과 같은 문장을 출력하세요.

<이름>님의 직업은 <직업>입니다.

**정답 코드 및 풀이**

**문제 1 정답**

# 사용자로부터 숫자 입력받기

num1 = int(input("첫 번째 숫자를 입력하세요: "))

num2 = int(input("두 번째 숫자를 입력하세요: "))

# 두 숫자의 합 계산

result = num1 + num2

# 결과 출력

print(f"두 숫자의 합은 {result}입니다.")

**풀이**

input() 함수로 두 개의 숫자를 입력받습니다.

int()로 문자열을 정수로 변환합니다.

입력받은 숫자를 더한 후 결과를 출력합니다.

**문제 2 정답**

# 사용자로부터 이름과 직업 입력받기

name = input("이름을 입력하세요: ")

job = input("직업을 입력하세요: ")

# 출력 문장 생성 및 출력

print(f"{name}님의 직업은 {job}입니다.")

**풀이**

input() 함수로 이름과 직업을 각각 입력받습니다.

f-문자열을 사용해 입력받은 값을 출력 문장에 삽입합니다.

최종 출력 문장을 출력합니다.

2.3 들여쓰기 (Indentation)

설명

파이썬은 들여쓰기를 사용하여 코드의 블록을 구분합니다.

일반적으로 4칸의 공백 또는 탭을 사용하며, 같은 수준의 들여쓰기를 유지해야합니다.

잘못된 들여쓰기는 IndentationError를 발생시킵니다.

들여쓰기는 함수, 조건문, 반복문, 클래스 등에서 사용됩니다.

예제

2.3.1 올바른 들여쓰기

if True:

print("올바른 들여쓰기입니다.")

2.3.2 잘못된 들여쓰기 (IndentationError 발생)

# if True:

# print("잘못된 들여쓰기입니다.")

실습 문제

문제 1

아래 코드의 잘못된 부분을 수정하세요.

x = 10

if x > 5:

print("x는 5보다 큽니다.")

문제 2

다음 조건을 만족하는 코드를 작성하세요.

변수 num이 0보다 크면 "양수"를 출력.

그렇지 않으면 "0 또는 음수"를 출력.

**정답 코드 및 풀이**

**문제 1 정답**

# 정답 코드

x = 10

if x > 5:

print("x는 5보다 큽니다.")

**풀이**

Print 함수 앞에 들여쓰기(스페이스 4칸)를 추가해 블록 구조를 맞췄습니다.

**문제 2 정답**

# 정답 코드

num = -3

if num > 0:

print("양수")

else:

print("0 또는 음수")

풀이

조건 if num > 0에서 True일 경우 "양수"를 출력하고, 그렇지 않을 경우 else에서 "0 또는 음수"를 출력합니다.

2.4 조건문(Conditional statements)

2.4.1 기본조건문

설명

파이썬은 if, elif, else를 사용합니다.

조건은 **참(True)** 또는 \*\*거짓(False)\*\*으로 평가됩니다.

2.4.2 2항 연산자

설명

문법: 참값 if 조건 else 거짓값

x = 5

result = "양수" if x > 0 else "음수"

2.4.3 3항 연산자

설명

파이썬에서는 3항 연산자 구조를 기본조건문으로 사용합니다.

예제

x = 10

if x > 0:

print("x는 양수입니다.")

elif x == 0:

print("x는 0입니다.")

else:

print("x는 음수입니다.")

2.4.4 2항 연산자 예제

y = -5

result = "양수" if y > 0 else "음수 또는 0"

print(result)

실습 문제

문제 1

다음 조건을 만족하는 코드를 작성하세요:

숫자 x가 10보다 크면 "10보다 큼" 출력.

그렇지 않으면 "10 이하" 출력.

문제 2

다음 코드를 2항 연산자로 변환하세요.

if x > 5:

result = "크다"

else:

result = "작다"

**정답 코드 및 풀이**

**문제 1 정답**

x = 8

if x > 10:

print("10보다 큼")

else:

print("10 이하")

**풀이**

조건 if x> 10으로 x의 값을 비교한 뒤, 조건에 따라 메시지를 출력합니다.

**문제 2 정답**

x = 3

result = "크다" if x > 5 else "작다"

print(result)

**풀이**

2항 연산자를 사용해 간결하게 조건을 표현했습니다.

2.5 반복문(Loop)

설명

2.5.1 for 반복문

시퀀스(리스트, 문자열 등)를 순회합니다.

for 변수 in 시퀀스:

실행문

2.5.2 while 반복문

조건이 참인 동안 실행합니다.

while 조건:

실행문

예제

2.5.3 for 반복문 예제

for i in range(1, 6):

print(f"{i}번째 반복입니다.")

2.5.4 while 반복문 예제

count = 5

while count > 0:

print(f"남은 횟수: {count}")

count -= 1

실습 문제

문제 1

1부터 10까지의 숫자를 출력하세요.

문제 2

while을 사용해 숫자 3의 배수를 15이하에서 출력하세요.

**정답 코드 및 풀이**

**문제 1 정답**

for i in range(1, 11):

print(i)

**풀이**

Range(1, 11)은 1부터 10까지의 숫자를 생성하며, for 반복문으로 출력합니다.

**문제 2 정답**

n = 3

while n <= 15:

print(n)

n += 3

**풀이**

변수 n을 3으로 초기화하고, 반복문 안에서 3씩 증가하며 조건에 따라 숫자를 출력합니다.

2.6 함수(Function)

설명

함수는 코드 재사용을 위해 사용하는 블록입니다.

Def 키워드를 사용해 정의합니다.

2.6.1 사용법

def 함수이름(매개변수1, 매개변수2):

실행문

return 반환값

예제

2.6.2 함수 정의

def add(a, b):

return a + b

2.6.3 함수 호출

result = add(3, 5)

print(f"덧셈 결과: {result}")

실습 문제

두 수를 곱하는 함수를 작성하세요.

정수를 입력받아 짝수인지 확인하는 함수를 작성하세요.

**정답 코드 및 풀이**

**문제 1 정답**

def multiply(a, b):

return a \* b

result = multiply(4, 7)

print(f"곱셈 결과: {result}")

풀이

매개변수 a, b를 곱한 결과를 반환합니다.

**문제 2 정답**

def is\_even(n):

return n % 2 == 0

number = 10

if is\_even(number):

print("짝수입니다.")

else:

print("홀수입니다.")

풀이

나머지 연산( % )으로 2로 나눈 나머지가 0인지 확인해 짝수를 판단합니다.

2.7 모듈(Module)

설명

모듈은 파이썬 파일 단위로 구성된 코드 집합입니다.

import 키워드로 가져옵니다.

예제

2.7.1 math 모듈 사용

import math

print(f"원의 넓이: {math.pi \* 3\*\*2}")

실습 문제

문제 1

math 모듈을 사용해 16의 제곱근을 계산하세요.

문제 2

Random 모듈을 사용해 1에서 10까지의 난수를 생성하세요.

**정답 코드 및 풀이**

**문제 1 정답**

import math

result = math.sqrt(16)

print(f"16의 제곱근: {result}")

**풀이**

Math.sqrt( )는 제곱근을 계산하는 함수입니다.

**문제 2 정답**

import random

num = random.randint(1, 10)

print(f"생성된 난수: {num}")

**풀이**

random.randint(1, 10)은 1과 10 사이의 정수를 무작위로 반환합니다.

2.8 슬라이싱(Slicing)

설명

슬라이싱은 리스트, 문자열 등 시퀀스 데이터에서 특정 부분을 추출합니다.

예제

2.8.1 문자열 슬라이싱

text = "Python Programming"

print(text[0:6]) # 출력: Python

2.8.2 리스트 슬라이싱

numbers = [1, 2, 3, 4, 5]

print(numbers[1:4]) # 출력: [2, 3, 4]

실습 문제

문제 1

문자열 "Hello World"에서 "World"만 추출하세요.

문제 2

리스트 [10, 20, 30, 40, 50]에서 짝수 번째 요소를 추출하세요.

**정답 코드 및 풀이**

**문제 1 정답**

text = "Hello World"

result = text[6:]

print(result)

풀이

슬라이싱 [6:]으로 문자열의 6번째 인덱스부터 끝까지 추출했습니다.

**문제 2 정답**

numbers = [10, 20, 30, 40, 50]

result = numbers[1::2]

print(result)

풀이

슬라이싱 [1::2]은 인덱스 1부터 간격 2로 요소를 추출합니다.

2.9 map 함수

설명

map(함수, 시퀀스)는 시퀀스의 모든 요소에 함수를 적용합니다.

예제

2.9.1 map 예제

numbers = [1, 2, 3, 4]

squared = map(lambda x: x\*\*2, numbers)

print(list(squared)) # 출력: [1, 4, 9, 16]

실습 문제

문제 1

리스트 [5, 10, 15]의 각 요소를 2배로 만드는 코드를 작성하세요.

문제 2

리스트 [-1, -2, 3, 4]의 절댓값을 계산하세요.

**정답 코드 및 풀이**

**문제 1 정답**numbers = [5, 10, 15]

doubled = map(lambda x: x \* 2, numbers)

print(list(doubled))

**풀이**

lambda를 사용해 각 요소를 2배로 만들었습니다.

**문제 2 정답**

numbers = [-1, -2, 3, 4]

absolute = map(abs, numbers)

print(list(absolute))

풀이

내장 함수 abs( )를 사용해 절댓값을 계산했습니다.

2.10 예외 처리

설명 try, except 블록을 사용해 에러를 처리합니다.

예제

try:

num = int(input("숫자를 입력하세요: "))

print(f"입력한 숫자는 {num}")

except ValueError:

print("유효한 숫자가 아닙니다.")

실습 문제

문제 1

숫자를 입력받고 5로 나눈 결과를 출력하세요. 잘못된 입력이 들어오면 "숫자를 입력하세요"를 출력하세요.

문제 2

리스트에서 인덱스를 입력받아 해당 요소를 출력하는 코드를 작성하세요. 인덱스가 잘못되면 "잘못된 인덱스입니다"를 출력하세요.

**정답 코드 및 풀이**

**문제 1 정답**

try:

num = int(input("숫자를 입력하세요: "))

print(f"결과: {num / 5}")

except ValueError:

print("숫자를 입력하세요.")

풀이

ValueError를 처리해 숫자 외의 입력을 방지했습니다.

**문제 2 정답**

data = [10, 20, 30]

try:

index = int(input("인덱스를 입력하세요: "))

print(f"요소: {data[index]}")

except (ValueError, IndexError):

print("잘못된 인덱스입니다.")

풀이

IndexError와 ValueError를 모두 처리했습니다.

2.11 클래스

설명

객체 지향 프로그래밍(OOP)의 기본 단위로, 속성과 메서드를 가집니다.

2.11.1 사용법

class 클래스이름:

def \_\_init\_\_(self, 속성):

self.속성 = 속성

예제

class Person:

def \_\_init\_\_(self, name):

self.name = name

def greet(self):

print(f"안녕하세요, 저는 {self.name}입니다.")

p = Person("홍길동")

p.greet()

실습 문제

문제 1

이름과 나이를 속성으로 가지는 Student 클래스를 작성하고, 소개 메서드를 추가하세요.

문제 2

길이와 너비를 받아 면적을 계산하는 Rectangle 클래스를 작성하세요.

**정답 코드 및 풀이**

**문제 1 정답**

class Student:

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

def introduce(self):

print(f"저는 {self.name}, {self.age}살입니다.")

s = Student("김철수", 20)

s.introduce()

풀이

\_\_init\_\_ 메서드로 속성을 초기화하고, introduce 메서드로 소개합니다.

**문제 2 정답**

class Rectangle:

def \_\_init\_\_(self, length, width):

self.length = length

self.width = width

def area(self):

return self.length \* self.width

r = Rectangle(5, 10)

print(f"면적: {r.area()}")

풀이

길이와 너비를 속성으로 받아 area 메서드에서 계산합니다.

라이브러리 활용법

3.1 OpenCV와 Pillow 기본 활용법

- 포트폴리오 할 때 많이 쓰일 것 같은 라이브러리라 특정 라이브러리를 선택하여 간단한 사용법을 정리한 것입니다.

설명

OpenCV는 이미지 처리와 컴퓨터 비전 작업을 위한 라이브러리입니다.

cv2.imread()로 이미지를 불러오고, cv2.imshow()로 이미지를 화면에 띄웁니다.

OpenCV: 컴퓨터 비전 라이브러리를 사용할 때 초심자들은 지금은 시간이 있다면 Full Package를 설치해보자. 샘플예제 라이브러리가 많아서 OpenCV 오픈 소스 라이브러리를 이용한 자율주행자동차, 안면인식, 장애물인식 등의 미니 프로젝트 만들기에 수월합니다.

3.1.1 사용법

설치

pip install 패키지명 – 원하는 Python 패키지를 설치할 때 사용하는 명령어 잘설치 됐나 확인하기!

터미널 -> Python

Import 패키지명

아무거도 나오지않고 프롬프트가 밑으로 떨어지고 커서가 깜빡이면 import가 잘 된 것입니다.

예제 1

Python은 설치할 때 패키지명과 사용할 때 (import) 이름이 다를 수도있습니다.

OpenCV패키지 간단설치 - pip install opencv-python

OpenCV패키지 사용 - 이미지 불러오기

import cv2

image = cv2.imread('image.jpg')

cv2.imshow('Image', image)

cv2.waitKey(0)

예제 2

import cv2

# 이미지 불러오기

image = cv2.imread('image.jpg') # 'image.jpg'는 파일 경로

cv2.imshow('Loaded Image', image) # 이미지 출력

cv2.waitKey(0) # 키 입력 대기

cv2.destroyAllWindows() # 창 닫기

3.1.2 Pillow 기본 활용법

설명

Pillow는 Python Imaging Library(PIL)의 업그레이드 버전입니다.

Image.open()으로 이미지를 불러오고, .show()로 간단히 표시합니다.

Pillow 라이브러리는 실제로 코드에서 import해서 사용할 때 PIL로 사용 됩니다.

3.1.3 사용법 – 모든 패키지(라이브러리)가 설치와 사용법이 비슷합니다. 다만 패키지의 설치 시 이름과 import할 때 이름이 다를 수도 있다는점 잊지 마세요!

설치

pip install 패키지명 – 원하는 Python 패키지를 설치할 때 사용하는 명령어 잘설치 됐나 확인하기!

터미널 -> Python

Import 패키지명

아무거도 나오지않고 프롬프트가 밑으로 떨어지고 커서가 깜빡이면 import가 잘 된 것입니다.

예제

from PIL import Image

# 이미지 불러오기

image = Image.open('image.jpg') # 'image.jpg'는 파일 경로

image.show() # 이미지 표시

3.2 이미지 처리 관련 활용 예제 5가지 (OpenCV 사용)

3.2.1 이미지 크기 조정(image resize)

import cv2

image = cv2.imread('image.jpg')

resized = cv2.resize(image, (200, 200)) # 이미지 크기 200x200으로 조정

cv2.imshow('Resized Image', resized)

cv2.waitKey(0)

cv2.destroyAllWindows()

**실행 결과**:  
이미지가 200x200 크기로 조정되어 출력됩니다.

3.2.2 이미지 그레이스케일 변환(gray scale)

import cv2

image = cv2.imread('image.jpg')

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY) # 그레이스케일로 변환

cv2.imshow('Grayscale Image', gray)

cv2.waitKey(0)

cv2.destroyAllWindows()

**실행 결과**:  
컬러 이미지가 흑백으로 변환되어 출력됩니다.

그레이스케일로 변환 이유 (정말 중요!!)

데이터 간소화

**컬러 이미지의 구조**: 컬러 이미지는 보통 3개의 채널(Red, Green, Blue)로 구성되어 있으며, 각 픽셀당 3개의 값을 가집니다.

**그레이스케일 이미지의 구조**: 그레이스케일 이미지는 단일 채널로, 각 픽셀당 하나의 밝기 값만을 가집니다.

처리 데이터의 양이 1/3로 줄어들어 연산 속도가 빨라지고, 메모리 사용량이 감소합니다.

연산 단순화

많은 컴퓨터 비전 알고리즘(에지 검출, 객체 감지 등)은 색상 정보보다는 **명암(밝기)** 차이에 의존합니다.

컬러 이미지는 각 채널에 대해 별도로 계산해야 하지만, 그레이스케일은 단일 채널만 계산하므로 알고리즘이 더 간단하고 효율적입니다.

특정 알고리즘 요구

OpenCV에서 사용하는 여러 알고리즘(예: Canny 에지 검출, Hough 선 검출 등)은 입력 데이터를 그레이스케일로 처리하도록 설계되었습니다.

이 알고리즘들은 색상보다는 **명도 변화**에 민감하게 반응하기 때문입니다.

객체 감지 및 특징 추출 최적화

특징 점 검출(예: SIFT, ORB 등) 또는 템플릿 매칭에서는 색상 정보보다 이미지의 구조적 패턴이 더 중요합니다.

그레이스케일 이미지를 사용하면 불필요한 색상 정보로 인한 복잡성이 줄어들고 특징 검출 성능이 향상됩니다.

불필요한 정보 제거

특정 작업에서는 색상 정보가 필요 없거나 오히려 방해가 될 수 있습니다.

예를 들어, 문서 스캔에서 텍스트를 인식하려면 텍스트 대비를 강조하기 위해 그레이스케일이 더 유용합니다.

예제 1

에지 검출(Canny)

edges = cv2.Canny(gray\_image, 100, 200)

cv2.imshow("Edges", edges)

예제 2

이진화 처리

\_, binary\_image = cv2.threshold(gray\_image, 127, 255, cv2.THRESH\_BINARY)

cv2.imshow("Binary Image", binary\_image)

코드 참고만 하시고 혼자 적용해보기 힘드신분들은 제가 학생들과 수업을 진행 뒤 깃허브에 참고 자료 업로드 해놓겠습니다.

3.2.3 이미지에 텍스트 추가

import cv2

image = cv2.imread('image.jpg')

text\_image = cv2.putText(image, 'Hello, OpenCV!', (50, 50),

cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 255, 0), 2)

cv2.imshow('Image with Text', text\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

**실행 결과**:  
이미지 상단에 "Hello, OpenCV!" 텍스트가 초록색으로 표시됩니다.

3.2.4 이미지 블러링

import cv2

image = cv2.imread('image.jpg')

blurred = cv2.GaussianBlur(image, (15, 15), 0) # 15x15 블러링

cv2.imshow('Blurred Image', blurred)

cv2.waitKey(0)

cv2.destroyAllWindows()

**실행 결과**:  
이미지가 부드럽고 흐릿하게 출력됩니다.

3.2.5 이미지 엣지 검출

import cv2

image = cv2.imread('image.jpg')

edges = cv2.Canny(image, 100, 200) # 엣지 검출

cv2.imshow('Edge Detected Image', edges)

cv2.waitKey(0)

cv2.destroyAllWindows()

**실행 결과**:  
이미지에서 윤곽선(엣지)만 검출되어 출력됩니다.

3.3 실행 결과 설명

3.3.1. **크기 조정**

cv2.resize()를 사용해 원본 이미지를 원하는 크기로 변경합니다.  
실행 후, 새로운 창에 크기가 변경된 이미지가 표시됩니다.

3.3.2. **그레이스케일 변환**

cv2.cvtColor()는 이미지를 특정 색상 공간으로 변환합니다.  
여기서는 cv2.COLOR\_BGR2GRAY를 사용해 흑백 이미지를 생성합니다.

3.3.3. **텍스트 추가**

cv2.putText()를 사용해 이미지 위에 텍스트를 표시합니다.  
폰트, 크기, 색상, 두께 등을 조절할 수 있습니다.

3.3.4. **블러링**

cv2.GaussianBlur()로 이미지를 흐리게 처리합니다.  
커널 크기 (15, 15)는 블러링 정도를 나타냅니다.

3.3.5. **엣지 검출**

cv2.Canny()는 이미지를 분석하여 엣지를 추출합니다.  
두 개의 임계값(100, 200)은 엣지 검출 기준을 설정합니다.

심화 과정: Flask 프로젝트

여기서 부터는 Flask프레임워크와 데이터베이스 공부가 부족하다고 생각하시는 분들은 개별적으로 공부하시거나 찾아보면서 하시는걸 추천드립니다. 이제 Python 기초 문법과 기본 활용법은 끝났습니다. 미니 프로젝트 시작해봅시다.

4.1 설치해야 하는 라이브러리 및 설치 이유

4.1.1 설치해야 하는 라이브러리

Flask와 PyMySQL

Flask 설치 명령어 – pip install flask

Flask 설치를 통해 웹 서버를 구축하고 클라이언트-서버 간 통신과 템플릿 렌더링을 구현합니다.

PyMySQL 설치 명령어 – pip install pymysql

PyMySQL 설치를 통해 Python과 MySQL 데이터베이스를 연결하고, 데이터 저장, 조회 수정, 삭제 등의 작업을 수행합니다.

**Flask와 PyMySQL의 협업**

**Flask**는 클라이언트 요청을 처리하고 HTML, CSS, JS 파일을 렌더링합니다.

**PyMySQL**은 Flask에서 처리한 요청을 기반으로 데이터베이스와상호작용합니다.

결론적으로, **Flask**는 서버와 UI를 담당하고, **PyMySQL**은 데이터 관리를담당합니다. 이 둘을 함께 사용하면 사용자의 요청(예: 게시판 글 등록/조회/수정/삭제)이 서버와 데이터베이스에서 효율적으로 처리됩니다.

4.1.2 Flask 설치 이유

Flask는 파이썬으로 웹 애플리케이션을 개발하기 위한 **경량 웹프레임워크**입니다.  
아래의 이유로 프로젝트에서 사용됩니다.

**웹 서버 구축**

Flask는 간단한 코드로 웹 서버를 구축할 수 있도록 지원합니다.

HTTP 요청을 처리하고, 클라이언트(브라우저)와 서버 간의 데이터를주고받는 데 적합합니다.

**템플릿 렌더링 지원**

Flask는 Jinja2라는 템플릿 엔진을 사용하여 동적으로 HTML 페이지를 생성할 수 있습니다.

이 프로젝트에서는 HTML, CSS, JS로 작성한 클라이언트 UI를 Flask 서버와 연동하기 위해 사용합니다.

**확장성**

Flask는 경량이지만, 필요한 기능을 확장할 수 있는 플러그인(예: 인증, 데이터베이스 연동)이 많아 유연합니다.

**초보자 친화적**

간단한 프로젝트부터 대규모 애플리케이션까지 적합하며, 코드가 비교적 단순해서 배우기 쉽습니다.

4.1.3 PyMySQL 설치 이유

PyMySQL은 Python에서 **MySQL 데이터베이스와 연동**하기 위해 사용됩니다.  
이 프로젝트에서 PyMySQL을 사용하는 이유는 다음과 같습니다.

**MySQL 데이터베이스 연결**

MySQL은 데이터를 저장하고 관리하기 위한 강력한 관계형데이터베이스 관리 시스템(RDBMS)입니다.

PyMySQL을 통해 Python 코드로 MySQL에 연결하고, 데이터베이스 작업(등록, 조회, 수정, 삭제)을 실행할 수 있습니다.

**SQL 쿼리 실행 지원**

PyMySQL은 Python 코드에서 SQL 명령을 실행할 수 있게 합니다.

예를 들어, Flask 애플리케이션에서 사용자 입력 데이터를데이터베이스에 저장하거나 가져올 때 사용합니다.

**Flask와의 궁합**

Flask는 데이터베이스 연동 작업을 효율적으로 처리할 수 있도록PyMySQL과 호환됩니다.

PyMySQL은 간단한 구문으로 MySQL 연결과 쿼리를 실행할 수 있어 Flask와의 통합 작업이 쉽습니다.

**Python 표준 DB API 준수**

PyMySQL은 Python의 데이터베이스 API 표준을 준수하여 다른 데이터베이스 라이브러리와 유사한 방식으로 사용 가능합니다.

4.2 Flask 기본 코드와 상세 해설



Flask를 공부할때는 공식문서를 참고해서 공부하시는 것이 가장 좋다고 생각해서 공식문서 기준으로 설명을 하겠습니다. 공식문서에 Flask 기본코드와 듀토리얼까지 있으니 아래 공식문서 주소를 참고해주세요.

**공식문서 주소**

<https://flask-docs-kr.readthedocs.io/ko/latest/index.htm>

4.2.1 Flask 기본 코드

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route('/')

def home():

return "Hello, Flask!"

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

실행시켜보세요 – 이 문서에서 개발환경은 windows10 + vscode입니다.

4.2.2 **코드 상세해설**

From flask import Flask

Flask 모듈 가져오기(flask를 우선 설치하셔야 import가 가능합니다)

Flask 클래스는 애플리케이션을 생성하고 실행하는 핵심 객체입니다.

이 코드를 통해 Flask 라이브러리에서 필요한 기능을 불러옵니다.

app = Flask(\_\_name\_\_)

Flask 애플리케이션 객체 생성

Flask( ) 클래스의 인스턴스를 app으로 생성합니다.

\_\_name\_\_은 현재 실행 중인 모듈의 이름을 나타냅니다.

\_\_name\_\_을 전달하는 이유

Flask는 실행되는 애플리케이션의 이름을 기준으로 정적 파일, 템플릿 등을 로드합니다.

실행되는 모듈의 경로를 내부적으로 활용합니다.

@app.route('/')

def home( ):

return "Hello, Flask!"

라우팅 설정

@app.route(‘/’) : 데코레이터로 라우트를 설정합니다.

‘/’은 웹 애플리케이션의 루트 경로를 나타냅니다.

예를 들어, <http://localhost:5000/> 로 접근하면 이 경로와 매핑됩니다.

def home( ) : 이 경로에 대한 요청이 들어올 때 실행되는 뷰 함수입니다.

함수 이름은 자유롭게 정할 수 있습니다.

여기서는 home으로 이름을지정했습니다.

Return “Hello, Flask!”:

브라우저에 반환할 데이터를 지정합니다.

여기서는 문자열 “hello, Flask!”를 반환하며, 브라우저에 그대로 출력됩니다.

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

메인 프로그램 실행 조건:

If \_\_name\_\_ == ‘\_\_main\_\_’:

현재 스크립트가 직접 실행될 때만 아래 코드를 실행합니다.

이 조건이 없으면 다른 모듈에서 이 코드를 가져왔을 때도 Flask 서버가 실행되는 문제가 발생할 수 있습니다.

app.run(debug=True)

Flask 애플리케이션을 실행하는 함수입니다.

주요 옵션

debug=True : 디버그 모드를 활성화합니다.

애플리케이션 코드 변경 시 자동으로 서버를 다시 시작합니다.

오류 발생 시 디버그 페이지(인터랙티브 콘솔 포함)를 제공합니다.

기본적으로 localhost:5000 에서 서버가 실행됩니다.

4.3 MySQL 연동 게시판 구현

Pythonr기반 vscode환경에서 개발 중 -> pymysql 사용합니다.

Pymysql에 대한 설치방법 및 설명은 위에서 언급했기에 생략하겠습니다.

4.3.1 Pymysql 사용방법

설명

아래 코드는 **Python의 pymysql 라이브러리**를 사용하여 MySQL데이터베이스에 연결하고, 데이터베이스에 게시글을 등록하는 예제입니다. 각 줄과 기능을 상세히 설명하겠습니다.

예제

import pymysql

connection = pymysql.connect(

host='localhost',

user='root',

password='password',

database='test\_db'

)

cursor = connection.cursor()

# 게시글 등록

query = "INSERT INTO posts (title, content) VALUES (%s, %s)"

cursor.execute(query, ('제목', '내용'))

connection.commit()

4.3.2 **코드 상세해설**

import pymysql

**pymysql 라이브러리**를 임포트합니다.

pymysql은 Python에서 MySQL 데이터베이스와 상호작용할 수 있는 모듈입니다.

이를 통해 MySQL에 접속, 쿼리 실행, 결과 처리 등이 가능합니다.

connection = pymysql.connect(

host='localhost',

user='root',

password='password',

database='test\_db'

)

**MySQL 연결**

pymysql.connect()를 사용하여 MySQL 데이터베이스와 연결합니다.

주요 매개변수

**host='localhost'**

데이터베이스 서버의 주소를 지정합니다.

localhost는 로컬 컴퓨터를 의미합니다.

(원격 서버의 경우 IP나 도메인을 입력)

**user='root'**

데이터베이스에 접속할 사용자 이름.

**password='password'**

해당 사용자 계정의 비밀번호.

**database='test\_db'**

연결할 데이터베이스 이름.

결과적으로 **MySQL 연결 객체**(connection)가 생성됩니다.

cursor = connection.cursor()

**커서 객체 생성**

데이터베이스와의 상호작용(쿼리 실행 및 결과 반환)을 위해 커서 객체를 생성합니다.

cursor는 MySQL 서버로 명령(쿼리)을 보내고 결과를 가져오는 역할을 합니다.

query = "INSERT INTO posts (title, content) VALUES (%s, %s)"

**SQL 쿼리 정의**

SQL의 **INSERT INTO** 명령을 사용하여 posts 테이블에 데이터를 삽입하는 쿼리를 작성합니다.

**구성 요소**

INSERT INTO posts: posts라는 테이블에 데이터를 삽입합니다.

(title, content): 데이터가 들어갈 열(컬럼)의 이름입니다.

VALUES (%s, %s): 삽입할 데이터를 나타냅니다. %s는 파라미터로 전달되는 값을 위한 자리 표시자입니다.

SQL 인젝션을 방지하기 위해 값은 직접 삽입하지 않고 자리 표시자로 처리합니다.

cursor.execute(query, ('제목', '내용'))

**쿼리 실행**

cursor.execute(query, ('제목', '내용'))를 통해 SQL 쿼리를 실행합니다.

첫 번째 매개변수 query는 실행할 SQL 명령입니다.

두 번째 매개변수 ('제목', '내용')은 쿼리에 전달될 실제 값들입니다.

첫 번째 %s에는 '제목'이, 두 번째 %s에는 '내용'이 들어갑니다.

실행 결과로 posts 테이블에 새 행이 추가됩니다.

connection.commit()

**변경사항 적용**

connection.commit()를 호출하여 **INSERT 작업을 실제로 데이터베이스에 반영**합니다.

MySQL은 기본적으로 **트랜잭션**을 사용합니다.

데이터 삽입, 업데이트, 삭제 등 변경 작업은 커밋(commit)해야데이터베이스에 반영됩니다.

커밋하지 않으면 변경사항이 무효화됩니다.

코드 실행 후 데이터베이스 상태

**posts 테이블에 새 데이터가 삽입됩니다**

title 컬럼에 '제목'.

content 컬럼에 '내용'.

전체 흐름 요약

MySQL 서버와 연결을 설정합니다.

데이터베이스 커서를 생성합니다.

SQL 쿼리를 작성하고, 파라미터를 사용해 데이터를 삽입합니다.

변경사항을 커밋하여 데이터베이스에 반영합니다.

4.3.3 확장

게시글 삽입 코드의 개선

**예외 처리 추가**

데이터베이스 작업은 실패할 가능성이 있으므로 예외 처리가 필요합니다.

try:

connection = pymysql.connect(

host='localhost',

user='root',

password='password',

database='test\_db'

)

cursor = connection.cursor()

query = "INSERT INTO posts (title, content) VALUES (%s, %s)"

cursor.execute(query, ('제목', '내용'))

connection.commit()

except Exception as e:

print(f"Error: {e}")

finally:

cursor.close()

connection.close()

**동적 데이터 삽입**

사용자 입력을 받아 데이터를 삽입:

title = input("Enter title: ")

content = input("Enter content: ")

cursor.execute(query, (title, content))

4.4 Flask 서버 디렉토리 구조와 설정

- HTML, CSS, SASS, JS와 Python 통합

4.4.1 Flask 프로젝트 구조(중요!!) – 각각 파일의 코드 첨부(참고)



**Flask 서버 코드(app.py)**

from flask import Flask, render\_template

app = Flask(\_\_name\_\_)

@app.route('/')

def home():

return render\_template('index.html') # HTML 파일 렌더링

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

**HTML 템플릿 (templates/index.html)**

HTML에서 Flask와 함께 CSS, JS 파일을 포함하는 방법입니다.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Flask with HTML, CSS, JS</title>

<!-- CSS 파일 포함 -->

<link rel="stylesheet" href="{{ url\_for('static', filename='css/style.css') }}">

</head>

<body>

<h1>Welcome to Flask</h1>

<button id="clickButton">Click Me</button>

<!-- JS 파일 포함 -->

<script src="{{ url\_for('static', filename='js/script.js') }}"></script>

</body>

</html>

**CSS 파일 (static/css/style.css)**

body {

font-family: Arial, sans-serif;

background-color: #f0f0f0;

color: #333;

text-align: center;

margin: 0;

padding: 20px;

}

button {

background-color: #007BFF;

color: white;

border: none;

padding: 10px 20px;

border-radius: 5px;

cursor: pointer;

}

button:hover {

background-color: #0056b3;

}

**SASS 파일 (static/css/style.scss) – 필수는 아님(이쁘게 꾸밀려면 필요함)**

SASS 파일을 작성하고 CSS로 변환합니다.

$primary-color: #007BFF;

$hover-color: #0056b3;

body {

font-family: Arial, sans-serif;

background-color: #f0f0f0;

color: #333;

text-align: center;

margin: 0;

padding: 20px;

}

button {

background-color: $primary-color;

color: white;

border: none;

padding: 10px 20px;

border-radius: 5px;

cursor: pointer;

&:hover {

background-color: $hover-color;

}

}

SASS 컴파일

SASS 설치 – 터미널에서 진행

npm install -g sass – 에러발생시 node.js가 설치 되어 있지 않아서 그런것입니다.

**해결방법**

Node.js 설치 확인

Node.js가 설치되어 있어야 npm을 사용할 수 있습니다.

터미널에서 다음 명령어로 Node.js가 설치되어 있는지 확인하세요.

node -v

npm -v

**Node.js** 버전과 **npm** 버전이 출력되면 설치된 상태입니다.

설치되어 있지 않다면 [Node.js 공식 사이트](https://nodejs.org/)에서 다운로드 후 설치하세요.

<https://nodejs.org/en>

**VSCode에서 터미널 열기**:

VSCode에서 상단 메뉴 \*\*View → Terminal\*\*을 선택하거나, 단축키 \*\*Ctrl + ~\*\*를 눌러 터미널을 엽니다.

**npm 명령 실행**:

터미널에서 프로젝트 폴더로 이동:

cd /path/to/your/project

npm install 명령어 실행:

npm install -g sass

SASS처럼 글로벌 설치 시 -g 플래그를 사용하며, 프로젝트 별 설치는 -g를 제외합니다.

sass설치 완료 후

**SASS 컴파일**:

sass static/css/style.scss static/css/style.css

style.scss 파일이 style.css로 변환됩니다.

**JavaScript 파일 (**static/js/script.js**)**

**– 대부분의 반응형 웹페이지 제작시 사용합니다.**

javascript

document.getElementById('clickButton').addEventListener('click', function () {

alert('Button Clicked!');

});

**실행 방법**

Flask 서버를 실행:

python app.py

브라우저에서 http://127.0.0.1:5000/로 접속합니다.

**실행 결과**

CSS 스타일이 적용된 페이지가 나타납니다.

버튼을 클릭하면 JavaScript 동작이 실행됩니다.

여기까지 빠르게 Python 기본 문법과 Flask서버을 구축하여 Pymysql과 HTML, CSS, JS, SASS를 활용하는 법까지 알아보았습니다. 마지막으로 모든 것을 한번에 활용해볼 수 있는 게시판 만들기 미니 프로젝트 통합구현을 한번 해보겠습니다.

4.5 통합구현 : Flask를 활용한 게시판 프로젝트

미니 프로젝트 : 게시판 제작

디렉토리 구조



데이터베이스 초기화

**database.sql**

CREATE DATABASE flask\_board;

USE flask\_board;

CREATE TABLE posts (

id INT AUTO\_INCREMENT PRIMARY KEY,

title VARCHAR(255) NOT NULL,

content TEXT NOT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

Flask 애플리케이션 메인 파일

**app.py**

from flask import Flask, render\_template, request, redirect, url\_for

import pymysql

# Flask 애플리케이션 생성

app = Flask(\_\_name\_\_)

# MySQL 데이터베이스 연결 설정

def get\_db\_connection():

return pymysql.connect(

host='localhost',

user='root',

password='password',

database='flask\_board',

cursorclass=pymysql.cursors.DictCursor

)

# 메인 페이지 - 게시글 목록

@app.route('/')

def index():

connection = get\_db\_connection()

with connection.cursor() as cursor:

# 모든 게시글 조회

cursor.execute("SELECT \* FROM posts ORDER BY created\_at DESC")

posts = cursor.fetchall()

connection.close()

return render\_template('index.html', posts=posts)

# 게시글 작성 페이지

@app.route('/create', methods=('GET', 'POST'))

def create():

if request.method == 'POST':

title = request.form['title']

content = request.form['content']

connection = get\_db\_connection()

with connection.cursor() as cursor:

# 새로운 게시글 추가

cursor.execute("INSERT INTO posts (title, content) VALUES (%s, %s)", (title, content))

connection.commit()

connection.close()

return redirect(url\_for('index'))

return render\_template('create.html')

# 게시글 수정 페이지

@app.route('/<int:id>/edit', methods=('GET', 'POST'))

def edit(id):

connection = get\_db\_connection()

with connection.cursor() as cursor:

# 수정할 게시글 조회

cursor.execute("SELECT \* FROM posts WHERE id = %s", (id,))

post = cursor.fetchone()

if request.method == 'POST':

title = request.form['title']

content = request.form['content']

with connection.cursor() as cursor:

# 게시글 업데이트

cursor.execute("UPDATE posts SET title = %s, content = %s WHERE id = %s", (title, content, id))

connection.commit()

connection.close()

return redirect(url\_for('index'))

connection.close()

return render\_template('edit.html', post=post)

# 게시글 삭제

@app.route('/<int:id>/delete', methods=('POST',))

def delete(id):

connection = get\_db\_connection()

with connection.cursor() as cursor:

# 게시글 삭제

cursor.execute("DELETE FROM posts WHERE id = %s", (id,))

connection.commit()

connection.close()

return redirect(url\_for('index'))

# 애플리케이션 실행

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

**HTML 템플릿 파일**

**templates/base.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Flask 게시판</title>

<link rel="stylesheet" href="{{ url\_for('static', filename='css/styles.css') }}">

</head>

<body>

<header>

<h1><a href="{{ url\_for('index') }}">Flask 게시판</a></h1>

</header>

<main>

{% block content %}{% endblock %}

</main>

</body>

</html>

**templates/index.html**

{% extends 'base.html' %}

{% block content %}

<a href="{{ url\_for('create') }}">새 글 작성</a>

<ul>

{% for post in posts %}

<li>

<h2>{{ post.title }}</h2>

<p>{{ post.content }}</p>

<p>작성일: {{ post.created\_at }}</p>

<a href="{{ url\_for('edit', id=post.id) }}">수정</a>

<form action="{{ url\_for('delete', id=post.id) }}" method="post" style="display:inline;">

<button type="submit">삭제</button>

</form>

</li>

{% endfor %}

</ul>

{% endblock %}

**templates/create.html**

{% extends 'base.html' %}

{% block content %}

<h2>새 글 작성</h2>

<form method="post">

<label for="title">제목</label>

<input type="text" id="title" name="title" required>

<label for="content">내용</label>

<textarea id="content" name="content" required></textarea>

<button type="submit">작성</button>

</form>

{% endblock %}

**templates/edit.html**

{% extends 'base.html' %}

{% block content %}

<h2>게시글 수정</h2>

<form method="post">

<label for="title">제목</label>

<input type="text" id="title" name="title" value="{{ post.title }}" required>

<label for="content">내용</label>

<textarea id="content" name="content" required>{{ post.content }}</textarea>

<button type="submit">수정</button>

</form>

{% endblock %}

**CSS 및 JavaScript**

**static/css/styles.css**

body {

font-family: Arial, sans-serif;

line-height: 1.6;

margin: 0;

padding: 0;

}

header {

background: #333;

color: #fff;

padding: 1rem 0;

text-align: center;

}

main {

padding: 1rem;

}

a {

color: #333;

text-decoration: none;

}

form {

margin: 1rem 0;

}

button {

background: #333;

color: #fff;

border: none;

padding: 0.5rem 1rem;

cursor: pointer;

}

지금가지 **Flask, MySQL, HTML, CSS, JavaScript**를 통합하여 게시판을 구축하는 프로젝트의 완전한 예제입니다. **주석을 통해 각 코드의 역할을 설명**했으며, 이 프로젝트를 실행하면 게시글 등록, 수정, 삭제를 포함한 기본 CRUD 기능을 구현할 수 있습니다. 추가적으로 문의 사항이나 확장이 필요하면 알려주세요!

추후 학생들과의 해당 과정의 수업 후 저의 github에 업로드를 지속적으로 할 예정입니다. 참고하세요.

[www.github.com/moojueng](http://www.github.com/moojueng)

**마무리: 함께한 여정의 끝, 새로운 시작**

이 전자책을 통해 Python의 기초 문법에서부터 실습 문제 풀이, OpenCV와 Pillow를 활용한 이미지 처리(프로젝트와는 관계없음), Flask와 PyMySQL을 이용한 CRUD 게시판 구현까지 폭넓은 내용을 학습했습니다. 여러분은 이제 Python을 사용해 단순한 스크립트 작성부터 데이터베이스와 연동된 웹 애플리케이션까지 개발할 수 있는 기초를 탄탄히 다졌습니다.

**여기까지 완료했다면, 여러분은 이미 성장했습니다.**

Python의 기초 문법과 핵심 개념을 이해하고 활용할 수 있습니다.

이미지 처리와 같은 Python의 강력한 라이브러리 활용 능력을 익혔습니다.

Flask와 PyMySQL로 데이터베이스 연동 및 CRUD 기능을 구축하는 데성공했습니다.

HTML, CSS, JavaScript와 Python의 조화를 통해 웹 애플리케이션 개발을 경험했습니다.

**다음에 도전해볼 만한 과제**

배운 내용을 기반으로 더 흥미롭고 도전적인 프로젝트를 진행해보세요. 아래는 추천 과제들입니다.

**회원 관리 시스템 구축**: 로그인/회원가입 기능 추가

**API 개발**: Flask로 JSON 기반 API를 구현하고 프론트엔드에서 활용

**머신러닝 모델 연동**: Flask와 OpenCV를 활용해 실시간 얼굴 인식 웹앱 구축

**클라우드 배포**: 프로젝트를 AWS, Azure, 또는 Heroku 같은 클라우드 플랫폼에 배포

* **머신러닝 모델 연동, 클라우드 배포 관련 과제는 관련 예제로 기초를 탄탄히 다진 후 도전하는 것을 추천드립니다.**

**감사의 말**

이 전자책은 Python을 배우고자 하는 열정적인 학습자를 위해 만들어졌습니다.  
코드 작성과 실행 중 어려움을 겪었더라도 끝까지 따라와 주셔서감사드립니다.  
프로그래밍은 끝없는 탐구와 학습의 여정입니다. 앞으로도 도전과 배움을 즐기시길 바랍니다.

**여러분의 도전은 끝이 아닙니다.**

"하루를 낭비하는 가장 확실한 방법은 배움 없는 하루를 보내는 것입니다."  
여러분이 이 여정을 통해 성장했듯이, 다음 프로젝트에서도 무한한 가능성을 펼쳐가시길 응원합니다!