AI활용프로그래밍

Week 5. 함수와 라이브러리 + LLM 사용 연습

함수로 코드 구조화 · 모듈/라이브러리 활용 · With AI 설계/리팩터링

학습 목표

- 함수의 개념(입력→처리→출력)과 장점을 설명할 수 있다.
- def, 매개변수/인자, return을 사용해 함수를 구현할 수 있다.
- import를 이해하고 표준/외부 라이브러리를 활용할 수 있다.
- LLM을 활용해 함수 설계·리팩터링·디버깅을 수행할 수 있다.

오늘의 구성

- 지난 주(조건문/반복문) 복습 → 함수를 사용해 코드 구조화
- 함수 문법 & 호출 흐름 (def, parameter/argument, return)
- 모듈/라이브러리 사용법 (import, 표준 라이브러리, pip 개요)
- With AI 실습 3개 + 미니 퀴즈 + 과제 안내

왜 함수가 필요한가?

```
1 # 함수 없이: 같은 로직이 여러 번 반복
2 score = int(input('점수 입력: '))
3 if score >= 60:
4     print('PASS')
5 else:
6     print('FAIL')
7
8 score2 = int(input('점수 입력: '))
9 if score2 >= 60:
10     print('PASS')
11 else:
12     print('FAIL')
```

함수의 이점

- 중복 제거(DRY): 같은 코드를 한 곳에서만 관리
- 가독성 향상: '무엇을 하는지' 이름으로 설명 가능
- 재사용/확장: 다른 곳에서도 호출해서 사용
- 테스트 용이: 함수 단위로 검증(assert/테스트케이스)
- 협업에 유리: 역할(입력/처리/출력) 분리

함수 기본 문법(def)

```
1 def greet(name):
2     """이름을 받아 인사 메시지를 반환"""
3     msg = f"안녕하세요, {name}님!"
4     return msg
5
6 result = greet('민수')
7 print(result)
```

핵심 포인트

- 함수는 '작은 프로그램'(입력→처리→출력)이다.
- return은 '값'을 돌려준다. (print와 역할이 다름)
- 함수 이름은 동사로: greet, calculate, convert ...
- Docstring(""" """)은 사용법/의도를 기록한다.

매개변수·인자·반환값

```
1 def add(a, b):  # a,b는 매개변수(parameter)
2     return a + b
3
4 x = add(2, 3)   # 2,3은 인자(argument)
5 print(x)        # 5
6
7 # return이 없으면 None이 반환됨
```

정리

- 정의할 때: 매개변수(parameter) / 호출할 때: 인자(argument)
- return은 값을 '돌려주고' 함수는 종료된다.
- print는 화면 출력(부수효과)이고 값 반환과는 별개다.
- 초보 실수: return 대신 print만 해서 결과를 못 쓰는 경우

기본값·키워드 인자·가변 인자

```
1 def greet(name, msg='안녕'):
2     return f"{msg}, {name}!"
3
4 print(greet('철수'))
5 print(greet('영희', msg='반가워'))
6
7 def total(*nums):
8     return sum(nums)
9 print(total(1,2,3,4))
```

핵심 포인트

- 기본값(default)은 인자를 생략했을 때 사용된다.
- 키워드 인자로 '의도'를 드러낼 수 있다: msg='반가워'
- *args는 '여러 개의 위치 인자'를 튜플로 받는다(맛보기).
- **kwargs는 '여러 개의 키워드 인자'를 딕셔너리로 받는다(맛보기).

변수 범위(Scope)와 주의점

```
x = 10  # 전역 변수

def f():
    x = 3   # 지역 변수(함수 안)
    return x

print(f())  # 3
print(x)    # 10

# 전역 변경은 가급적 피하기
```

기억할 것

- 함수 안에서 만든 변수는 기본적으로 '지역'이다.
- 이름이 같아도 지역/전역은 서로 다른 변수일 수 있다.
- 전역을 변경해야 한다면 설계를 다시 생각해보자(부작용).
- LLM에게 '전역 변수 없이' 구현하라고 제약을 주면 안전하다.

함수 품질 체크

- 이름: 무엇을 하는지 한 눈에 보이게 (예: calc_avg, load_csv)
- 입력/출력: 함수가 받는 것과 돌려주는 것을 명확히
- 길이: 20~30줄 이상 길어지면 분해를 고려
- Docstring: 사용법/예시/예외 상황을 기록
- 타입 힌트(선택): 의도를 표현해 디버깅을 돕는다

Tip: '한 함수는 한 가지 일' 원칙을 연습해봅시다.

표준 라이브러리 맛보기

```python
import math
import random
from datetime import datetime

print(math.sqrt(16))          # 4.0
print(random.randint(1, 10))  # 1~10
print(datetime.now())
```

오늘은 '표준'만으로도 충분

- 표준 라이브러리 = 파이썬 설치에 기본 포함
- math: 수학 계산 / random: 난수 / datetime: 날짜·시간
- 실습에서 random으로 미니게임(숫자맞추기/로또)을 만들어볼 것
- LLM에게 '표준 라이브러리만 사용' 조건을 주면 과해지지 않는다

모듈과 import

```python
# my_utils.py 파일이 있다고 가정
def to_f(c):
    return c * 9/5 + 32

# main.py
import my_utils
print(my_utils.to_f(25))

# 또는
from my_utils import to_f
print(to_f(25))
```

핵심 포인트

- 모듈(module) = .py 파일 1개
- import는 '가져와서 쓰기'(중복 없이 재사용)
- from ... import ... 는 필요한 것만 가져오기
- 파일 이름과 모듈 이름이 연결된다 (my_utils.py → my_utils)

외부 라이브러리 설치(pip) 개요

- 외부 라이브러리 = 파이썬에 기본 포함되지 않는 추가 기능
- 설치: pip installl 패키지이름
- 권장: 가상환경(venv) 안에서 설치 (프로젝트별 독립)
- 협업/제출: requirements.txt로 설치 목록 공유
- 문제 해결: ModuleNotFoundError → (1) venv 활성화 (2) pip 설치 확인

라이브러리 선택 기준

- 문서/예제가 충분한가? (공식 문서 우선)
- 최근 업데이트가 있는가? (유지보수)
- 사용자/커뮤니티가 큰가? (검색/질문 가능)
- 라이선스/보안 이슈는 없는가?
- 수업에서는: '필요한 만큼만' 가져와 쓰기

함수 설계 5단계(추천 루틴)

1. **입력/출력** 정의
2. **예시 I/O** 작성
3. **예외/경계값** 정리
4. **작은 함수로** 분해
5. **테스트** 작성

With AI 팁: 위 5단계를 '프롬프트 요구사항'으로 그대로 적으면, 함수 품질이 급상승합니다.

With AI: 함수 설계 프롬프트

```
[역할] 너는 친절한 파이썬 튜터야.
[목표] 아래 요구사항을 만족하는 함수를 설계/구현해줘.
[요구사항]
- 입력: (예) 점수 리스트(list[int])
- 출력: 평균(float) + 등급(str) 반환
- 예외: 빈 리스트면 (0.0, 'N/A')
- 제약: 전역 변수 사용 금지
[출력형식]
(1) 함수 시그니처
(2) 코드
(3) 테스트 3개(assert)
```

사용법

- 요구사항을 '입력/출력/예외/제약'으로 쪼개서 쓰기
- 원하는 출력 형식을 지정하면 답변 품질이 좋아진다
- 테스트(assert)를 같이 요구하면 검증이 쉬워진다
- 코드를 받으면 직접 실행해서 결과를 확인하기

With AI: 함수로 리팩터링

```
아래 코드를 '함수 3개'로 분해해줘.
- 함수1: 입력 처리
- 함수2: 핵심 계산(반환값 필수)
- 함수3: 출력/표시

추가 조건:
- 중복 제거
- 함수 이름은 의미 있게
- 예외 상황(잘못된 입력) 처리

출력은 (설계 → 코드 → 테스트) 순서로.
```

체크 포인트

- 함수 분해 기준: '역할(입력/처리/출력)'로 나누기
- 반환값이 있는 함수와 없는 함수를 구분하기
- 입력 검증(try/except)도 함수화하면 깔끔해진다
- 리팩터링 후 동작이 동일한지 테스트로 확인

실습 1: 코드 분해하기

```
# 목표: 아래 코드를 함수 3개로 분해
# - read_score()
# - get_grade(score) -> str
# - print_report(name, score, grade)

name = input('이름: ')
score = int(input('점수: '))
if score >= 90: grade='A'
elif score >= 80: grade='B'
elif score >= 70: grade='C'
else: grade='F'
print(name, score, grade)
```

완료 기준

- 함수 3개 이상으로 분해되어 있다
- get_grade()는 return으로 등급을 돌려준다
- 입력값이 숫자가 아닐 때를 처리한다(선택)
- assert 테스트 2개 이상 작성(선택)

실습 2: random 미션(로또)

```python
import random

def lotto_numbers():
    # TODO: 1~45 중 중복 없이 6개 뽑기
    pass

def main():
    nums = lotto_numbers()
    print('이번 주 번호:', nums)

if __name__ == '__main__':
    main()
```

완료 기준

- 중복 없는 번호 6개를 생성한다
- 정렬해서 보기 좋게 출력한다(선택)
- 함수를 최소 2개 이상 사용한다
- LLM에게 '중복 없는 랜덤' 구현을 질문해본다

실습 3: 미니 프로그램(택1)

- A) 단위 변환기(섭→화, km→mile 등): 함수 4개 이상
- B) 간단 계산기(+,-,*,/): 입력 검증 포함
- C) 메뉴 추천기(random.choice 활용): 중복 추천 방지(선택)
- 공통 조건: 표준 라이브러리 1개 이상 사용
- README에 'AI에게 요청한 프롬프트/수정 내용' 기록

자주 만나는 오류(함수/모듈)

```
1 def add(a, b):
2     return a + b
3
4 add(1)  # TypeError: missing 1 required
positional argument
5
6 print(to_f(10))  # NameError: name 'to_f' is not
defined
7
8 import numpy  # ModuleNotFoundError(설치/venv 확
인)
```

빠른 해결 루틴

- TypeError → 인자 개수/순서/기본값 확인
- NameError → 함수 정의 위치/오타/import 여부 확인
- ModuleNotFoundError → venv 활성화 → pip install 확인
- 에러 메시지 전체를 LLM에 전달 + 내가 한 시도도 함께 적기

빠른 테스트: assert

```python
def add(a, b):
    return a + b

assert add(2, 3) == 5
assert add(-1, 1) == 0

# assert가 실패하면 AssertionError 발생
# → 즉시 버그 위치를 좁힐 수 있음
```

팁

- 테스트는 '작게, 자주' 작성하면 디버깅이 빨라진다
- 경계값 테스트: 0, 음수, 빈 입력 등
- LLM에게 '테스트 케이스 5개 생성' 요청해보기
- 통과/실패 결과를 보고 코드 수정 → 다시 실행

AI 답변 검증 체크리스트

- 요구사항(입력/출력/예외)을 모두 만족하는가?
- return/print가 목적에 맞게 사용되었는가?
- 전역 변수/복잡한 의존성(불필요한 라이브러리)이 없는가?
- 내 컴퓨터에서 그대로 실행되는가? (직접 실행 필수)
- 엣지 케이스 테스트를 통과하는가?

미니 퀴즈(5문항)

문항

- 1) 매개변수(parameter)와 인자(argument)의 차이는?
- 2) return과 print의 차이를 한 문장으로 설명해보자.
- 3) 기본값 매개변수는 언제 유용한가?
- 4) import 모듈 as 별칭 을 쓰는 이유는?
- 5) 함수 안에서 만든 변수는 기본적으로 (지역/전역) 중 무엇인가?

정답 예시: 1) 정의/호출 시점, 2) 값 반환 vs 화면 출력, 3) 자주 쓰는 옵션, 4) 긴 이름 단축, 5) 지역

과제/연습문제

- 필수: 함수 5개 이상 사용하는 프로그램 1개 제출(실습 3 택1 확장 가능)
- 조건: 입력 검증(try/except) 또는 assert 테스트 3개 중 1개 이상 포함
- 표준 라이브러리 1개 이상 사용 (random/math/datetime 등)
- 제출물: .py 파일 + README(실행 방법, AI 활용 내역)
- 권장: 'AI가 만든 코드' 그대로 제출 금지 → 본인이 이해하고 수정한 흔적 남기기

요약 & 다음 주 예고

- 오늘: 함수로 코드 구조화(def/return/scope) + 모듈/라이브러리 활용
- 핵심: 작은 함수 + 명확한 입력/출력 + 간단한 테스트(assert)
- With AI: 설계→코드→테스트를 한 번에 요구하면 품질이 좋아진다
- 다음 주(6주): 리스트 자료구조 + LLM 사용 연습