AI활용프로그래밍

Week 1. 프로그래밍 언어 구조와 LLM

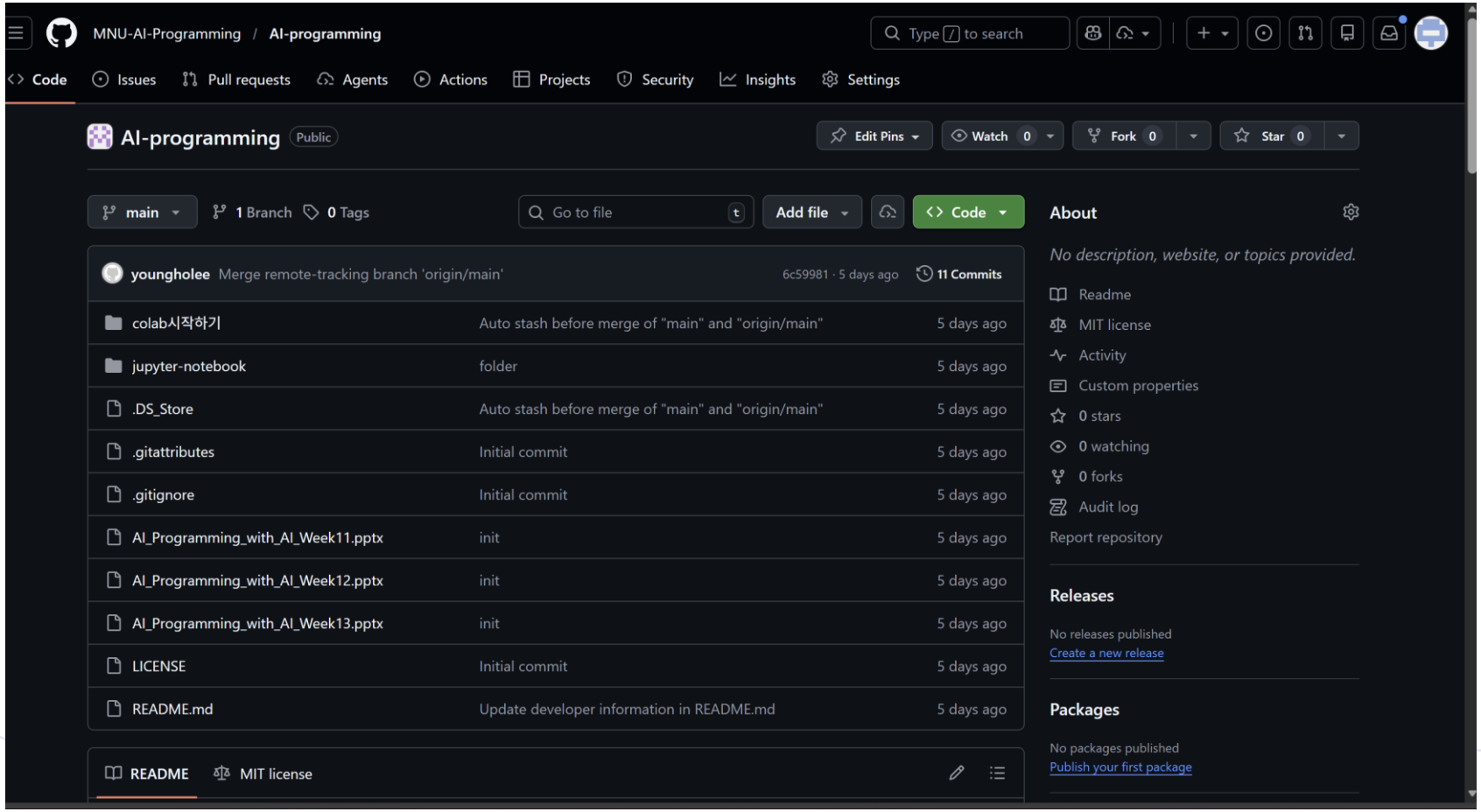
프로그래밍 기본 구조 + With AI 코딩 시작하기

학습 목표

- 프로그래밍 언어의 구성요소(문법·의미·실행)를 설명할 수 있다.
- 컴파일 방식과 인터프리트 방식의 차이를 비교할 수 있다.
- LLM을 코딩 학습에 활용하는 기본 원칙과 프롬프트 템플릿을 적용할 수 있다.

Github 저장소

- 저장소: https://github.com/MNU-AI-Programming/AI-programming



오늘의 구성

- 과목 운영(평가/출석) 빠르게 확인
- 프로그래밍 언어 구조: 문법·의미·실행 흐름
- 오류 유형과 디버깅 루틴
- LLM 개념과 With AI 코딩 방법
- 프롬프트 실습 & 미니 퀴즈

수업 운영(평가·출석)

성적 평가(총 100점)

출석 기준(중요)

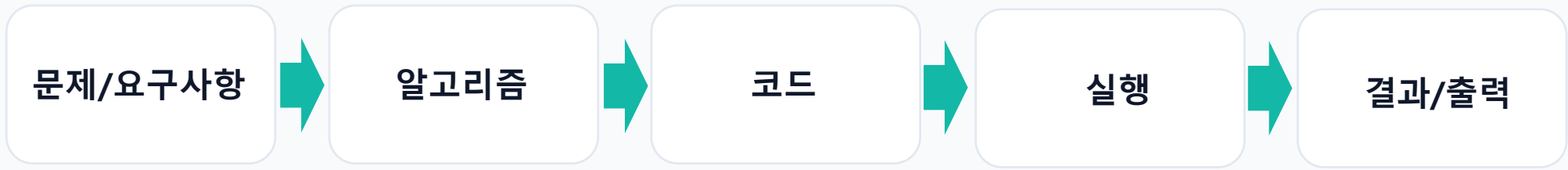
- 출석률이 수업시간의 3/4 미만이면 해당 교과목 성적은 F 처리
- 공결(출석처리 특례) 사유가 있을 경우: 결석계 제출 및 학과 확인 필요

※ 세부 기준은 강의계획서 및 학사 규정을 따름

프로그래밍 언어란?

- 사람의 문제 해결 절차(알고리즘)를 컴퓨터가 실행 가능한 형태로 표현하는 규칙의 집합
- 문법(Syntax): “어떻게 쓰는가” | 의미(Semantics): “무슨 뜻인가”
- 예: Python, Java, C, JavaScript …
- 핵심: 언어는 “사람 ↔ 컴퓨터” 사이의 약속(표현 규칙)이다.

언어가 하는 일: 번역의 흐름



문제를 "정확한 명령 + 데이터"로 바꾸는 과정이 프로그래밍입니다.

컴파일 vs 인터프리트

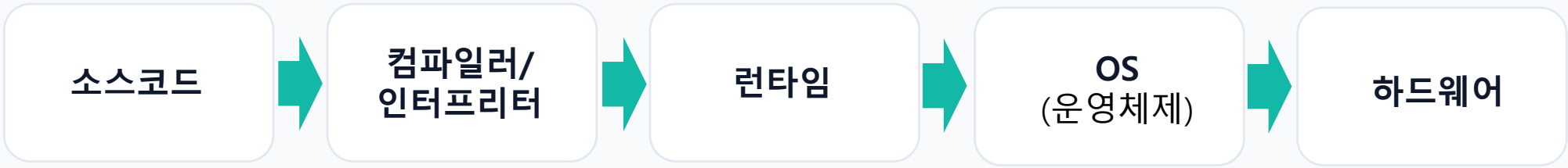
컴파일(Compile)

- 실행 전에 소스코드를 기계어(또는 중간 코드)로 변환
- 오류(문법/타입) 일부를 미리 발견하기 쉬움
- 실행 속도가 빠른 편(환경에 따라)
- 예: C/C++, (전통적) Java

인터프리트(Interpret)

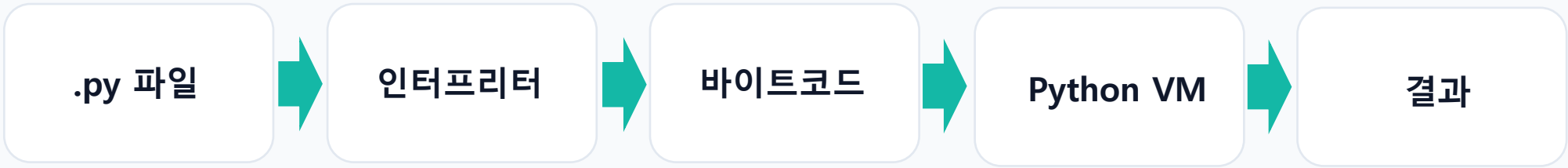
- 실행 시점에 한 줄씩 해석(또는 VM에서 실행)
- REPL/빠른 실험에 유리
- 런타임 에러를 실행 중에 만날 수 있음
- 예: Python, JavaScript

프로그램 실행 구조(개념도)



언어/도구에 따라 "어디서 변환하고 어디서 실행하는지"가 달라집니다.

파이썬 실행 흐름



우리는 "실행→관찰→수정"을 빠르게 반복하며 학습합니다.

15주 로드맵

1~8주

- 1주: 프로그래밍 언어 구조와 LLM
- 2주: 파이썬 환경 구축 & Hello World
- 3주: 변수·자료형·연산자 + LLM 연습문제
- 4주: 조건문·반복문 + LLM 연습문제
- 5주: 함수·라이브러리 + LLM 연습문제
- 6주: 리스트 자료구조 + LLM 연습
- 7주: 튜플·딕셔너리·집합 + LLM 연습
- 8주: 시험

9~15주

- 9주: 파일 입출력 + LLM 연습
- 10주: NumPy·Pandas + LLM 연습
- 11주: Matplotlib(그래프) + LLM 연습
- 12주: 필터링·정렬 + LLM 연습
- 13주: 쓰레드·미니게임 + LLM 연습
- 14주: 개인별 창의적 프로그램 과제 발표
- 15주: 시험

문법(Syntax) vs 의미(Semantics)

```
# (1) 문법 오류: 실행 전에 멈춤
print("Hello"

# (2) 의미/논리 오류: 실행은 되지만 결과가 틀릴 수 있음
scores = [80, 90, 100]
avg = sum(scores) / 2  # <- 의도는 3으로 나누기!
print(avg)
```

핵심 포인트

- 문법 오류: "규칙대로 안 써서" 실행 자체가 불가
- 논리 오류: "의도와 다르게" 계산/처리가 진행됨
- 좋은 습관: 작은 단위로 실행하고, 결과를 자주 확인

오류의 3종류

1) Syntax Error (문법)

- 문법 규칙 위반
- 예: 괄호/따옴표 누락
- 대부분 "실행 전"에 발견

2) Runtime(런타임) /
3) Logic(기능)

- Runtime Error(Exception): 실행 중 예외 발생
- 예: 0으로 나누기, 인덱스 범위 오류
- Logic Error: 실행은 되나 답이 틀림
- 예: 평균 계산에서 잘못 나누기

디버깅 기본 루틴

- 증상 재현: 입력/환경을 고정하고 “언제, 어떻게” 오류가 나는지 확인
- Traceback 읽기: 에러 종류(Error Type)와 줄 번호(Line)를 먼저 본다
- 최소 재현 코드(MRE, Minimal Reproducible Example) 만들기
  : 문제를 가장 작은 형태로 줄인다
- 가설 → 한 번에 한 가지 변경 → 다시 실행
- 수정 후 테스트: 정상 케이스 + 경계(엣지) 케이스

With AI 코딩: 역할 분담

사람(나)의 역할

- 문제 정의 & 요구사항 정리
- 입력/출력 예시 만들기
- 최종 판단(정답/품질/윤리)
- 실행·테스트·검증

AI(LLM)의 역할

- 초안 코드/여러 접근 제시
- 개념 설명/요약/예시 생성
- 오류 원인 후보 제안
- 가독성 개선(리팩터링) 제안

LLM(대규모 언어모델)이란?

- 대량의 텍스트/코드 데이터를 학습해 “다음 단어(토큰)”를 예측하는 모델
- 질문에 대해 그럴듯한 답을 만들어내지만, 항상 사실/정답을 보장하지는 않는다
- 따라서: AI 답변은 “초안”이며, 실행/검증이 필수
- 좋은 입력(프롬프트) → 좋은 출력(답변/코드) 확률이 올라간다

LLM이 잘하는 것 / 조심할 것

잘하는 것(활용 추천)

- 예제 코드/템플릿 생성
- 개념을 쉬운 말로 설명
- 코드 주석/문서화 초안
- 리팩터링 아이디어

조심할 것(검증 필요)

- 틀린 내용도 확신 있게 말할 수 있음(환각)
- 환경/버전에 따라 코드는 바로 안 될 수 있음
- 보안/개인정보/저작권 이슈
- "정답처럼 보이는 오답"에 주의

안전하고 똑똑한 사용 원칙

- 맥락을 충분히 제공: 목표, 입력/출력 예시, 제한 조건
- 출력 형식을 요구: 표/목록/코드/단계별 절차 등
- 검증은 필수: 실행, 테스트, 공식 문서 확인
- 개인정보/민감정보는 입력하지 않기
- AI 사용 내역을 기록(프롬프트/수정 사항)

ChatGPT는 무엇을 해주는가? (코딩 관점)

ChatGPT = 코딩을 돕는 LLM 기반 대화형 도구

- 자연어(한국어/영어 등)로 **요구사항을 설명**하면
  → 코드/설계/설명/수정안을 만들어준다
- 개발 흐름에서 하는 역할
  - **생성**: 초안 코드, 함수 템플릿
  - **설명**: 코드 해석, 개념 정리
  - **개선**: 리팩터링/가독성/스타일
  - **디버깅**: 에러 원인 추정, 해결 절차 제안

코딩 관점에서 ChatGPT는 "검색기"보다 "초안 생성+코치"에 가깝다

ChatGPT에게 '좋게' 물어보는 법

- 나쁜 질문 예: "파이썬 코드 짜줘"
- 좋은 질문 템플릿
  - 목표: 무엇을 만들고 싶은지
  - 입력/출력: 예시 1~2개
  - 제약: 라이브러리/성능/금지 조건
  - 형식: "코드만", "설명 포함", "단계별"

예시

"정수 리스트가 주어지면 평균을 반환하는 함수를 작성해줘.
입력 예: [1,2,3] → 출력: 2.0
예외: 빈 리스트면 None 반환. 파이썬 함수로."

ChatGPT 답변은 항상 검증한다 (수업 규칙)

- 실행해보기: 그대로 돌려서 동작 확인
- 테스트 추가: 정상/엣지 케이스(빈값, 0, 큰 값)
- 에러 메시지 공유: 에러 전문을 그대로 붙여넣기
- 요구사항 재확인: 출력 형식/예외 처리/제약 조건

- 자주 하는 실수들
  - 경로/파일 위치(작업 폴더) 문제
  - 입력 형식 가정이 틀림
  - 예외 처리 누락

프롬프트 템플릿

```
역할: 너는 친절한 파이썬 튜터야.
목표: 아래 코드를 초보가 이해하도록 설명해줘.
맥락: 나는 1학년이고 변수/반복문을 막 배우기 시작했어.
제약: 어려운 용어는 최소화하고, 핵심을 5줄로 요약해줘.
출력 형식: (1) 한 줄 요약 (2) 줄별 설명 (3) 실수하기 쉬운 포인트

코드:
```python
# 여기에 코드 붙여넣기
```
```

핵심 포인트

- 역할(Role): 어떤 전문가처럼 답할지
- 목표(Goal): 무엇을 얻고 싶은지
- 맥락(Context): 내 수준/상황/전제
- 제약(Constraints): 분량, 금지사항, 기준
- 형식(Format): 표/목록/코드/JSON 등

실습 1: 코드 설명받기

```
1 total = 0
2 for i in range(1, 6):
3     total += i
4 print(total)
```

핵심 포인트

- 프롬프트에 "줄별 설명 + 쉬운 비유 1개"를 요구해보기
- 같은 동작을 sum()로 다시 작성하도록 요청해보기
- 출력 결과(15)가 왜 나오는지 스스로 말할 수 있으면 성공

실습 2: 오류 메시지로 디버깅

```python
nums = [1, 2, 3]
print(nums[3])

# Traceback(예시)
# IndexError: list index out of range
```

핵심 포인트

- AI에게 "원인 1줄 + 해결 2가지 + 예방 팁 1개" 형태로 답하게 해보기
- 해결안이 여러 개면: "어느 경우에 어떤 해결안이 좋은지"까지 질문
- 최종 답은 내가 이해한 뒤 선택한다

실습 3: 리팩터링(가독성 개선)

```
a=10;b=20
c=a+b
print("sum=",c)
```

핵심 포인트

- AI에게 "변수 이름 개선 + 주석 2줄 + 출력 동일"을 요구
- 코드 스타일(띄어쓰기, 줄바꿈)을 정돈해달라고 요청
- 리팩터링 후에도 동작이 같다는 걸 확인(테스트)

AI 답변 검증 체크리스트

- 코드를 직접 실행했는가?
- 요구사항/제약을 모두 만족하는가?
- 경계(엣지) 케이스를 생각해봤는가?
- 필요한 라이브러리/버전이 맞는가?
- 내가 "왜 이렇게 동작하는지" 설명할 수 있는가?

검증하지 않은 AI 답변은 "참고 의견"일 뿐입니다.

미니 퀴즈(5문항)

문항

1) 문법 오류(Syntax Error)는 보통 언제 발견될까?
A. 실행 중 B. 실행 전 C. 테스트 후

2) LLM 답변에 대한 올바른 태도는?
A. 무조건 믿는다 B. 참고하고 검증한다 C. 읽지 않는다

3) 디버깅의 첫 단계로 가장 좋은 것은?
A. 랜덤 수정 B. 증상 재현 C. 코드 전체 삭제

4) 좋은 프롬프트에 포함되면 좋은 요소가 아닌 것은?
A. 목표 B. 출력 형식 C. 개인정보

5) 논리 오류(Logic Error)의 특징은?
A. 실행 불가 B. 실행 가능하지만 결과가 틀림 C. 컴파일이 느림

정답: 1-B, 2-B, 3-B, 4-C, 5-B

과제/연습문제

- 프롬프트 3종 작성: (1) 설명 요청 (2) 디버깅 요청 (3) 리팩터링 요청
- 각 프롬프트에 대한 AI 답변을 받고, "검증/수정"한 내용을 함께 정리
- 최종 제출물에는 다음을 포함: 프롬프트, AI 답변 요약, 내가 수정한 점, 배운 점
- 개인정보/민감정보는 포함하지 않기

목표는 "AI를 써서 더 빨리"가 아니라 "AI와 함께 더 정확히"입니다.

요약 & 다음 주 예고

- 프로그래밍 언어는 문법·의미·실행 구조로 이해할 수 있다.
- 오류는 문법/런타임/논리로 나뉘며, 디버깅은 “재현→분리→검증”이다.
- LLM은 강력한 도우미지만, 답변 검증과 기록이 필수다.
- 다음 주: 파이썬 개발환경 구축 + Hello World

다음 주 수업 전: 노트북 준비(가능하면 Python/VS Code 설치 시도).