2026.01.09.월

생성일	@2026년 1월 19일 오전 9:03
태그	Python

목차

▼ 1) 파이썬 기초

- 개발 환경 구축
  - vs code (code runnder + python environments)
  - pyenv
  - venv

💡

- pyenv와 venv의 차이
  - pyenv: 런타임 차이, 버전 분리
  - venv: 패키지 분리
- venv는 한 번에 하나만 실행 가능 → 패키지 충돌 불가능
- 버전 분리는 왜 필요한가?
  - 버전마다 문법, 표준 라이브러리, 패키지 호환성은 버전 마다 다름.
  - 그런데 서버, 로컬, ML인프라 등 버전이 다 다를 수 있음. 모두가 최신 버전을 맞출 수 없는게 현실
- Docker가 있는데 pyenv를 사용하는 이유?
  - 도커는 실행 환경을 제공하지 개발 환경에 적합하진 않음. 닭 잡는데 소 잡는 칼 쓰는 상황 발생. 파이썬 언어의 사용하는 이유가 서비스 개발이 전부가 아니기 때문 → 데이터 분석을 배포해서 실행시킬 이유는 없다...
- 실행 순서
  - docker 위에 pyenv 위에 venv 위치
  - pyenv가 선택한 파이썬 버전 위에 venv 패키지 실행

```
# 프로젝트 버전 관리 명령어 (자동으로 .python-version 파일 생성)
pyenv local 3.10.13

# pyenv에 의해 설치된 버전 확인
pyenv --versions
```

- Python Environments, Code Runnder Extention의 역할
  - Python Environments: 파이썬을 vs code가 언어로 인식할 수 있게 해줌. 없다면 파이썬은 일반 텍스트랑 다를게 없음
  - Code Runner: 설정된 인터프리터로 실행 시켜줌. 명령어 입력 생략 없다면 'python main.py' 같은 명령어 필요

항목	Python Extension	Code Runner
인터프리터 인식	⭕	❌
venv 연동	⭕	❌
디버깅	⭕	❌
테스트	⭕	❌
실무 적합성	높음	낮음

  - 터미널에선 되는데 vs code 실행에서 안되는 이유 → 바라보는 인터프리터가 다르다
  - 실행 버튼으로 안된다면 venv 실행 이후 `python main.py` 명령 입력 하는게 정석
  - 아님 파일 저장을 안 했거나...

▼ 2) 변수

```
# 단일 변수 선언
name = "홍길동"
age = 25
height= 175.5
is_student = True

# 변수 선언 시 주의 사항
# 1. 변수명은 의미있게 지어야함
student_name = "김철수" # good
x = "김철수"          # bad
```

```
# 2. 변수 선언 시 초기값 설정이 좋음
score = 0     # good
# total_score   # bad

# 여러 변수 동시 선언
# 방법 1
x, y, z = 1, 2, 3

# 방법 2
a = b = c = 10

# 방법 3
numbers = [1, 2, 3]
first, second, third = numbers

# 변수 값 교환
# 일방적인 방법
a = 1
b = 2
temp = a
a = b
b = temp

# 파이썬 방식
a, b = 1, 2
a, b = b, a

# 변수 타입 특징
x = 10
print(type(x))  # <class 'int'>

x = "hello"
print(type(x))  # <class 'str'>

x = True
print(type(x))  # <class 'bool'>

# 파이썬 변수 저장 방법
# 가변 타입의 참조는 메모리 공간을 공유하는 깊은 참조에 해당한다.
list1 = [1,2 ,3]
list2 = list1
list2 = 10
print(list2)  # 10

# 변수 스코프 (전역 변수)
global_var = "전역 변수"

def my_function():
  print(global_var) # 전역 변수 접근

print(global_var) # 전역 변수 접근

# 변수 스코프 (지역 변수)
def my_function2():
  local_var = "지역 변수"
  print(local_var)

#print(local_var) # 오류: 정의되지 않음
```

```
# 변수 스코프
count = 0

def increment():
    global count # 없다면 에러
    count += 1

increment()
print(count)  # 1

student_name = "김철수"
student_age = 20
student_height = 175.5
is_enrolled = True

# 정보 출력
print("=== 학생 정보 ===")
print(f"이름: {student_name}")
print(f"나이: {student_age}세")
print(f"키: {student_height}cm")
print(f"등록 여부: {'등록' if is_enrolled else '미등록'}")

# 실습 2: 수학 연산
x = 10
y = 3

# 기본 연산
addition = x + y
subtraction = x - y
multiplication = x * y
division = x / y
remainder = x % y

print("\n=== 수학 연산 ===")
print(f"덧셈: {x} + {y} = {addition}")
print(f"뺄셈: {x} - {y} = {subtraction}")
print(f"곱셈: {x} * {y} = {multiplication}")
print(f"나눗셈: {x} / {y} = {division}")
print(f"나머지: {x} % {y} = {remainder}")
```

▼ 3) 숫자형

```
# 작은 수
small = 10
print(type(small))  # <class 'int'>

# 큰 수
large = 12345678901234567890123456789O
print(type(large))  # <class 'int'>

# 정밀도 제한 예시
a = 0.1 + 0.2
print(a) # 0.30000000000000004 (예상과 다른 결과)

# 정밀도 제한으로 인한 비교 문제
print(0.1 + 0.2 == 0.3) # False

# 해결 방법: 반올림 사용
print(round(0.1 + 0.2, 1) == 0.3) # True
```

```
# 1. 반올림 사용
result = round(0.1 + 0.2, 1) # 0.3

# 2. decimal 모듈 사용 (정확한 10진수 연산)
from decimal import Decimal
a = Decimal('0.1')
b = Decimal('0.2')
print(a + b) # 0.3

# 3. math 모듈의 isclose 함수 사용
import math
print(math.isclose(0.1 + 0.2, 0.3)) # True

# 복소수 생성 방법
z1 = 3 + 4j         # 직접 리터럴로 생성
z2 = complex(3, 4)  # complex() 함수로 생성

# 복소수 속성
print(z1.real)      # 3.0 (실수부)
print(z1.imag)      # 4.0 (허수부)

# 켤레복소수 (허수부의 부호를 바꾼 복소수)
print(z1.conjugate()) # (3-4j)
```

▼ 4) 문자열

```
# 문자열 생성 방법
str1 = "Hello"
# str2 = `World` → 백틱은 더이상 지원하지 않음
str3 = """여러 줄 문자열"""
# str4 = ```여러 줄 문자열``` 백틱은 더이상 지원하지 않음

# 문자열 불변성
text = "Hello"
text[0] = "3"   # 오류 발생

text = "Hello"
new_text = text.replace("H", "3") # ."Hello" 는 그대로, "Jello" 라는 새 문자열 생성
text += " World" # 실제로는 "Hello World" 라는 새 문자열 생성

# 문자열 연산
concatenation = "Hello" + " " + "World"  # "Hello World"
repetition = "Hi" * 3  # "HiHiHi"

# 문자열 인덱싱
text = "Python"
first_char = text   # 'P'
last_char = text[-1]   # 'n' 음수 인덱스는 문자열의 끝에서부터 셈

# 문자열 슬라이싱
substring = text[1:4]  # 'yth' 시작 인덱스 포함, 끝 인덱스는 포함하지 않음

# 문자열 변환 메서드
text = "Python Programming"
print(text.upper())     # "PYTHON PROGRAMMING"
print(text.lower())     # "python programming"
print(text.title())     # "Python Programming"
print(text.swapcase())  # "pYTHON pROGRAMMING"
```

```
# 검색 메서드
print(text.find("Pro")) # 7 (첫 번째 위치 반환, 없으면 -1)
print(text.count("m"))  # 2 (문자 출현 횟수)
print("Pro" in text)    # True (포함 여부)

# 변형 메서드
print(text.replace("Python", "Java"))  # "Java Programming"
print(" Hello ".strip())  # "Hello" (양쪽 공백 제거)
print(text.split(" "))    # ["Python", "Programming"] (분할)
print("-".join(["a", "b", "c"])) # "a-b-c" (결합)

# 검증 메서드
print("12345".isdigit())  # True (모두 숫자인지)
print("abcde".isalpha())  # True (모두 알파벳인지)
print("Python1".isalnum()) # True (알파벳 또는 숫자인지)

# f-string (Python 3.6+)
name = "홍길동"
age = 25
message = f"이름: {name}, 나이: {age}"

# format 메서드
message = "이름: {}, 나이: {}".format(name, age)
```

▼ 5) 불리언

```
print(bool(0))       # False
print(bool(1))       # True
print(bool(-1))      # True (0이 아닌 모든 숫자는 True)
print(bool(""))      # False (빈 문자열)
print(bool("Hello"))  # True (비어있지 않은 문자열)
print(bool([]))      # False (빈 리스트)
print(bool([1, 2]))   # True (비어있지 않은 리스트)
print(bool(None))     # False

# 논리 연산자
x = True
y = False

print(x and y)  # False (둘 다 True일 때만 True)
print(x or y)   # True (둘 중 하나라도 True면 True)
print(not x)    # False (부정)

# 단축 평가(Short-circuit evaluation)
# and 연산자: 첫 값이 False면 두 번째 값은 평가하지 않고 바로 False 반환
print(False and print("확인"))  # False (print 함수 실행되지 않음)

# or 연산자: 첫 값이 True면 두 번째 값은 평가하지 않고 바로 True 반환
print(True or print("확인"))    # True (print 함수 실행되지 않음)

# 실제 값 반환
print(0 and 5)   # 0 (첫 번째 값이 False이므로 0 반환)
print(2 and 5)   # 5 (첫 번째 값이 True이므로 두 번째 값 반환)
print(0 or 5)    # 5 (첫 번째 값이 False이므로 두 번째 값 반환)
print(2 or 5)    # 2 (첫 번째 값이 True이므로 첫 번째 값 반환)

# 비교 연산
equal = (5 == 5)       # True
```

```
not_equal = (5 != 3)    # True
greater = (5 > 3)       # True
less = (5 < 3)          # False

# 복합 비교 연산
x = 10
complex_check = (5 < x < 15)  # True (5 < 10 and 10 < 15)

# 객체 비교
list1 = [1, 2, 3]
list2 = [1, 2, 3]
list3 = list1

print(list1 == list2)   # True (내용이 같음)
print(list1 is list2)   # False (다른 객체)
print(list1 is list3)   # True (같은 객체)

# 정수로 변환
int("123")    # 123
int(3.14)     # 3
int(True)     # 1

# 실수로 변환
float("3.14") # 3.14
float(5)      # 5.0
float(True)   # 1.0

# 문자열로 변환
str(123)      # "123"
str(3.14)     # "3.14"
str(True)     # "True"

# 불리언으로 변환
bool(1)       # True
bool(0)       # False
bool("Hello") # True
bool("")      # False

# 문제 1: 다음 코드의 출력 결과는 무엇인가요?
a = 0.1 + 0.1 + 0.1
b = 0.3
print(a == b)

# 문제 2: 다음 중 False로 평가되지 않는 것은 무엇인가요?
# a) 0
# b) ""
# c)
# d) None
# e) False

# 문제 3: 다음 코드의 실행 결과와 그 이유를 설명하세요.
text = "Python"
text = "J"
print(text)
```

▼ 6) 컬렉션 타입

```
# 1. 대괄호로 생성
fruits = ["사과", "바나나", "체리"]
```

```
# 2. list() 함수 사용
numbers = list(range(1, 6))  # [1-3, 5, 6]

# 3. 빈 리스트 생성
empty_list1 = []
empty_list2 = list()

# 4. 다양한 데이터 타입 혼합
mixed_list = [1, "안녕", 3.14, True, [1-3]]

# 5. 리스트 컴프리헨션(List Comprehension)
squares = [x**2 for x in range(1, 6)]  # [1, 5, 7-9]

# 리스트 수정
fruits = ["사과", "바나나", "체리"]

# 요소 변경
fruits[1] = "블루베리"  # ['사과', '블루베리', '체리']

# 요소 추가
fruits.append("딸기")   # ['사과', '블루베리', '체리', '딸기']

# 특정 위치에 삽입
fruits.insert(1, "포도") # ['사과', '포도', '블루베리', '체리', '딸기']

# 리스트 확장
more_fruits = ["키위", "망고"]
fruits.extend(more_fruits) # ['사과', '포도', '블루베리', '체리', '딸기', '키위', '망고']

# 요소 제거 (첫 번째 일치하는 항목)
fruits.remove("체리")   # ['사과', '포도', '블루베리', '딸기', '키위', '망고']

# 특정 위치 요소 제거 및 반환
removed = fruits.pop(2) # '블루베리' 제거 및 반환
print(removed)  # '블루베리'
print(fruits)   # ['사과', '포도', '딸기', '키위', '망고']

# 리스트 정렬
fruits.sort()   # ['망고', '딸기', '사과', '키위', '포도'] (알파벳 순)

# 리스트 역순으로 뒤집기
fruits.reverse() # ['포도', '키위', '사과', '딸기', '망고']

# 리스트 길이
print(len(fruits))  # 5

# 요소 개수 세기
fruits.append("사과")
print(fruits.count("사과")) # 2 ('사과'가 2번 등장)

# 요소 위치 찾기
print(fruits.index("키위")) # 3 ('키위'의 위치)

# 리스트 내용 비우기
fruits.clear()  # []

# enumerate 객체 변환 예시
letters = ['a', 'b', 'c']
enum_obj = enumerate(letters)
```

```
print(list(enum_obj)) # [(0, `a`), (1, `b`), (2, `c`)]

# enumerate 함수 활용하기
# 리스트 요소와 인덱스를 함께 순회할 때 유용
fruits = ["사과", "바나나", "체리", "딸기"]
for index, fruit in enumerate(fruits):
    print(f"{index}번: {fruit}") # '0번: 사과', '1번: 바나나' 등 출력

# 시작 인덱스 지정
for index, fruit in enumerate(fruits, start=1):
    print(f"{index}번: {fruit}") # '1번: 사과', '2번: 바나나' 등 출력

# 리스트의 특정 값 위치 찾기
colors = ["빨강", "파랑", "초록", "파랑", "노랑"]
blue_indices = [i for i, color in enumerate(colors) if color == "파랑"]
print(f"파랑의 위치: {blue_indices}") # [1, 3]

# enumerate를 사용하지 않으면?
fruits = ["사과", "바나나", "체리"]
for i in range(len(fruits)):
    print(f"{i}번: {fruits[i]}")
```

▼ 7) 투두리스트 실습

```
# 할 일 목록 조회
tasks = []
def view_tasks():
    if tasks:
        print("\n===== 할 일 목록 =====")
        for i, task in enumerate(tasks):
            print(f"{i+1}. {task}")
        print("======================")
    else:
        print("할 일이 없습니다.")

def add_task(newTask):
    tasks.append(newTask)
    print(f"새로운 할 일 {newTask}이(가) 추가되었습니다.")

def complete_task(idx):
    if (0 <= idx < len(tasks)):
      tasks.remove(tasks[idx-1])
    else :
      print("존재하지 않는 일입니다.")

# 프로그램 실행 예시
add_task("파이썬 공부하기")
add_task("장보기")
add_task("운동하기")
view_tasks()
complete_task(1) # "장보기" 완료
view_tasks()
```

▼ 8) 학생 관리 리스트 실습

```
# 학생, 점수, 정보, 리스트, 이름
# 학생들의 이름과 점수 정보를 리스트로 관리하는 코드 구현
# 다음 기능을 구현하세요:
```

```
  # 학생 추가: 이름과 점수를 입력 받아 목록에 추가
  # 학생 삭제: 이름을 입력 받아 해당 학생 정보 삭제
  # 성적 수정: 이름을 입력 받아 해당 학생의 점수 수정
  # 전체 목록 출력: 모든 학생의 이름과 점수 출력
  # 통계 출력: 최고 점수, 최저 점수, 평균 점수 계산 및 출력

class Student:
  def __init__(self, name, score) :
    self.name = name
    self.score = score

  def editScore(self, newScore) :
    self.score = newScore

students = []

def print_all_students() :
  print("=== 전체 학생 목록 ===")

  if len(students) == 0 :
    print("학생이 없습니다.")
    return

  for i, student in enumerate(students) :
    print(f"{i+1}번 학생: {student.name} - {student.score}점")

def add_student (name, score) :
  students.append(Student(name, score))
  print(f"{name} 학생 추가")

def delete_student (name) :
  for student in students :
    if student.name == name :
      students.remove(student)
      print(f"{name} 학생 삭제")
      return
  print("학생을 찾을 수 없습니다.")

def edit_student_score (name, score) :
  for student in students :
    if student.name == name :
      student.editScore(score)
      print(f"{name} 학생 점수 {score}점으로 수정 완료")
      return
  print("학생을 찾을 수 없습니다.")

def print_statistics() :
  highScore = 0
  lowScore = 100
  avgScore = 0

  if len(students) == 0 :
    print("학생이 없습니다.")
    return

  for i, student in enumerate(students) :
    if (highScore < student.score) :
      highScore = student.score
    if (lowScore > student.score) :
      lowScore = student.score
```

```
    avgScore += student.score

  avgScore /= len(students)

  print(f"최고 점수: {highScore} | 최저 점수: {lowScore} | 평균 점수: {avgScore:.1f}")

# 테스트 코드
add_student("홍길동", 30)
add_student("전우치", 60)

print_all_students()
print_statistics()

edit_student_score("홍길동", 70)
edit_student_score("각시탈", 100)

print_all_students()
print_statistics()

delete_student("홍길동")

print_all_students()
print_statistics()

delete_student("전우치")

print_all_students()
print_statistics()
```