

파이썬 자료형을 작업해야하는 영역과 연결해줘

좋습니다! 파이썬의 **자료형(Data Type)**은 다양한 작업(예: 데이터 분석, 웹 개발, 자동화 등)에서 필수적으로 사용되며, 각 자료형은 특정 작업에 적합한 특징을 가지고 있습니다. 아래 표는 **파이썬 자료형별로 주로 사용되는 작업 영역**을 연결한 것입니다.

🧠 파이썬 자료형과 작업영역 연결 표

자료형	예시	주요 특징	활용되는 작업 영역
int (정수)	10, -3	수학 계산, 반복문 제어 등에 사용	수치 분석, 반복제어, 알고리즘
float (실수)	3.14, -0.5	소수점 연산 가능	통계 계산, 머신러닝 수치 처리
bool (불린)	True, False	조건문, 필터링	조건 분기, 판다스 필터링
str (문자열)	"hello"	텍스트 처리 가능	데이터 클렌징, 텍스트 마이닝, 웹 크롤링
list (리스트)	[1, 2, 3]	순서 있음, 변경 가능	반복 처리, 데이터 그룹
tuple (튜플)	(1, 2)	순서 있음, 변경 불가	함수 반환값, 좌표 저장
set (집합)	{1, 2, 3}	중복 제거, 교집합/합집합 등	중복 제거, 데이터 비교
dict (딕셔너리)	{'a': 1}	키-값 구조, 빠른 탐색	JSON 처리, API 응답 저장, DB 조회 결과
None	None	값이 없음 표현	값 초기화, 누락값 처리

父 작업 영역별 자료형 예시

1. 데이터 분석 (pandas, numpy)

• float, int: 수치 데이터 (예: 연봉, 점수)

• str: 이름, 지역, ID

• bool: 조건 필터링 (ex: df['나이'] > 20)

• list, dict: 한 셀에 리스트/딕셔너리 저장 가능성 (멀티레이블 등)

2. 웹 크롤링 / 텍스트 처리

• str: HTML 내용, 키워드, 기사 본문 등

• dict: JSON 형태 데이터 저장

• list: 크롤링한 여러 기사 목록 저장

3. 자동화 스크립트 / GUI 프로그래밍

• str: 파일 이름, 입력값

• int: 반복 횟수, 좌표값

• dict: 설정 저장

• bool: 체크박스 상태, 조건 분기

4. AI/머신러닝

• float, int: 모델 학습 데이터

• list, numpy.ndarray: 입력 데이터 세트

• dict: 하이퍼파라미터, 결과 저장

🕶 참고: 자료형 확장 개념

확장 자료형	설명	관련 패키지
numpy.ndarray	수치 계산용 배열	numpy
pandas.Series / DataFrame	열/표 형태의 데이터	pandas
datetime, timedelta	날짜, 시간 자료형	datetime, pandas

필요하시면 각 자료형을 실제 코드 예제로 보여드릴 수도 있어요. 어느 영역을 집중하고 싶으신가요? (예: 데이터 분석, 자동화, 웹개발 등)



패키지내에서의 튜플이 나오는 예제를 알려줘, 간단히

좋아요! 패키지 내에서 **tuple(튜플)**이 실제로 자주 등장하는 예제 중 하나는 enumerate **함수**, pandas, OpenCV, matplotlib 등에서 찾을 수 있습니다. 아래는 각 패키지에서 **간단하게 튜플이 나오는 예제**를 보여드립니다.

☑ 1. enumerate() — 내장 함수지만 실무에서 자주 사용

```
fruits = ['사과', '바나나', '딸기']
for idx, fruit in enumerate(fruits):
    print((idx, fruit)) # → 튜플로 반환됨
```

출력 예시:

```
bash
(0, '사과')
(1, '바나나')
(2, '딸기')
```

✓ 2. pandas에서 DataFrame.itertuples()

```
import pandas as pd

df = pd.DataFrame({'이름': ['철수', '영희'], '점수': [90, 85]})
for row in df.itertuples():
    print(row) # 튜플 형태로 한 행씩 반환
```

출력 예시:

```
SCSS

Pandas(Index=0, 이름='철수', 점수=90)
Pandas(Index=1, 이름='영희', 점수=85)
```

✓ 3. OpenCV에서 이미지 크기 — tuple 형태

```
import cv2
img = cv2.imread('example.jpg') # 이미지 불러오기
print(img.shape) # (높이, 너비, 채널) → 튜플로 출력
```

출력 예시 (예: 640x480 RGB 이미지):

```
scss
(480, 640, 3)
```

✓ 4. matplotlib에서 좌표 입력

```
import matplotlib.pyplot as plt

point = (2, 3) # 튜플로 좌표 설정
plt.plot(point[0], point[1], 'ro')
plt.show()
```

🧠 요약

패키지	튜플 예시	사용 목적
enumerate	(인덱스, 값)	리스트 반복
pandas	itertuples()	행 데이터를 튜플로 반환
cv2	img.shape	이미지 정보 반환
matplotlib	(x, y)	좌표 지정

필요하면 특정 패키지에 대해 더 심화된 예제도 만들어드릴 수 있어요. 어느 쪽을 더 해볼까요?



리스트 예제도 보여줘

물론이죠! 실무에서 **리스트(list)** 가 자주 사용되는 **패키지별 간단 예제**를 아래에 정리해드릴게요. 리스트는 **데이터 모음**, **여러 값 저장**, **반복 처리** 등 다양한 작업에 매우 유용합니다.

✓ 1. pandas — 열을 리스트로 변환

```
import pandas as pd

df = pd.DataFrame({'이름': ['철수', '영희', '민수'], '점수': [90, 85, 78]})
이름리스트 = df['이름'].tolist()
print(이름리스트) # ['철수', '영희', '민수']
```

✓ 2. matplotlib — 리스트로 x, y값 전달

```
python

import matplotlib.pyplot as plt

x = [1, 2, 3]
y = [10, 20, 15]

plt.plot(x, y)
plt.title("리스트 기반 그래프")
plt.show()
```

✓ 3. random — 리스트에서 무작위 추출

```
import random

names = ['철수', '영희', '민수', '수지']

print(random.choice(names)) # 리스트에서 하나 랜덤 선택
```

✓ 4. json — 리스트를 JSON으로 변환 (API에서 자주 사용)

```
python

import json

data = {
    "학생목록": ["철수", "영희", "민수"]
}
```



```
json_str = json.dumps(data, ensure_ascii=False)
print(json_str) # {"학생목록": ["철수", "영희", "민수"]}
```

🔽 5. openpyxl — 엑셀에 리스트 쓰기

```
python

from openpyxl import Workbook

wb = Workbook()
ws = wb.active

score_list = [90, 85, 78]
ws.append(score_list) # 리스트 한 줄씩 쓰기
wb.save("sample.xlsx")
```

🧠 리스트 요약

패키지	리스트 예시	사용 목적
pandas	.tolist()	열을 리스트로 변환
matplotlib	[x], [y]	그래프 그리기
random	choice(list)	랜덤 추출
json	list → JSON	데이터 교환
openpyxl	append(list)	엑셀에 쓰기

필요하면 **리스트와 반복문, 조건문** 등을 연결한 예제도 함께 보여드릴게요. 혹시 다음으로 궁금한 자료형이나 패키지가 있을까요?



스칼라,벡터,어레이,텐서 의 개념도 차근히 알려줘

좋습니다! 이번에는 많은 사람들이 혼동하는 스칼라(Scalar), 벡터(Vector), 어레이(Array), 텐서(Tensor) 개념을 쉽게, 차근차근, 비교 중심으로 알려드릴게요.

✓ 1. 스칼라 (Scalar)

- 정의: 하나의 숫자 (크기만 있음, 방향 없음)
- 예시: 3, -7, 5.5
- **차원**: 0차원 (0D)
- 파이썬 예시:

```
python
x = 5 # 스칼라
```

• 일상 비유:

체온: 36.5도 점수: 100점 → 그냥 '값' 하나!

✓ 2. 벡터 (Vector)

- 정의: 여러 숫자가 일렬로 나열된 것 (크기 + 방향 존재)
- **예시**: [1, 2, 3] (x축, y축, z축 방향으로 각각의 성분을 가짐)
- **차원**: 1차원 (1D)
- 파이썬 예시:

```
import numpy as np
v = np.array([1, 2, 3])
```

• 일상 비유:

오늘의 성적: 국어 90, 수학 80, 영어 85 → 한 사람의 점수 목록 (순서 있는 여러 값)

✓ 3. 배열 (Array)

- 정의: 스칼라/벡터를 확장한 모양 있는 숫자 묶음
- 차원에 따라:
 - 0차원: 스칼라
 - 1차원: 벡터
 - 2차원: 행렬
 - 3차원 이상: 텐서
- 파이썬 예시:

```
python
a = np.array([[1, 2], [3, 4]]) # 2D 배열
```

핵심: array는 데이터를 담는 그릇
 (넘파이, 텐서플로우 등에서 핵심 구조)



☑ 4. 텐서 (Tensor)

- 정의: 다차원 배열 (배열의 일반화 버전)
- 차원 예시:
 - 0D 텐서: 스칼라 (5)
 - 1D 텐서: 벡터 ([1, 2, 3])
 - 2D 텐서: 행렬 ([[1, 2], [3, 4]])
 - 3D 텐서: 여러 장의 행렬 (예: 컬러 이미지 3채널)
 - nD 텐서: 고차원 구조 (딥러닝 모델에서 사용)
- 파이썬 예시:

```
python

import numpy as np
t = np.array([[[1], [2]], [[3], [4]]]) # 3차원 텐서
print(t.shape) # (2, 2, 1)
```

• AI에서 중요: 이미지, 음성, 텍스트 데이터를 수치로 표현할 때 다 텐서 사용

📊 비교표

개념	정의	차원	예시	사용처
스칼라	하나의 숫자	0D	3.5	온도, 나이 등
벡터	숫자의 나열	1D	[3, 4, 5]	점수 목록, 위치 등
배열	데이터를 담는 구조체	1D~	[[1,2], [3,4]]	넘파이, 이미지 등
텐서	다차원 수치 데이터 구조	nD	shape: (32, 32, 3)	딥러닝 이미지, 음성, 텍스트

🖈 정리 한 줄 요약

스칼라는 숫자 하나,

벡터는 숫자 줄,

배열은 숫자 표,

텐서는 배열의 확장판 (모든 걸 포함)!

원하시면 시각적으로 그림으로도 설명해드릴 수 있어요. 또는, 직접 텐서 구조를 시각화하는 코드도 제공 가능해요.

원하시는 방식이 있을까요?



이 자료형과 리스트,튜플,딕셔너리등과의 연계도 부탁

훌륭한 질문이에요! 이제 방금 배운 **스칼라, 벡터, 배열, 텐서** 개념과 파이선의 기본 자료형들 (리스트, 튜플, 딕셔너리) 사이의 연결 관계를 **구조와 특징 중심**으로 정리해드릴게요.

🧠 개념 연결: "자료 구조(Structure) vs. 수학 개념(형태)"

파이썬 구조	설명	수학/AI적 개념	예시
int, float	숫자 하나	스칼라 (0D)	x = 10
list	순서 있는 가변형 데이터 모음	벡터 (1D)	x = [1, 2, 3]
tuple	순서 있는 불변형 데이터 모음	벡터 (1D)	x = (1, 2, 3)
list of lists	리스트 안에 리스트가 있는 구조	행렬 (2D)	[[1, 2], [3, 4]]
dict	키-값 쌍의 집합	매핑 구조	{"이름": "철수", "나이": 20}
numpy.array	수학적 다차원 배열	배열/텐서 (nD)	np.array([[1, 2], [3, 4]])

✔ 연결 핵심 요약

수학 개념	파이썬 자료형	형태 구조	예시
스칼라	int, float	단일 값	3, 5.5
벡터	list, tuple	일렬 데이터	[1, 2, 3], (1, 2, 3)
배열	list of lists, numpy.array	표 형태	[[1, 2], [3, 4]]
텐서	numpy.array, torch.Tensor	고차원 데이터	np.zeros((2, 3, 4))
매핑 구조	dict	키-값 구조	{"x": 1, "y": 2}

🦈 실무 연계 예시

1. 데이터프레임 만들기 전

```
python
# 튜플로 행을 구성, 리스트로 묶음
data = [("철수", 90), ("영희", 85)] # → 2D 벡터 구조
```

2. 딕셔너리를 이용한 피처-값 매핑

```
python
student = {"이름": "철수", "국어": 90, "수학": 85} # → 매핑 구조
```

3. 텐서로 바꾸기 (넘파이)



```
import numpy as np tensor = np.array([[1, 2], [3, 4]]) # → 2D 텐서
```

🔄 개념 도식 요약

```
SCSS

[스칼라]
└─ int, float

[벡터 (1D)]
└─ list, tuple → 순서만 있음

[행렬/배열 (2D~)]
└─ list of list, numpy.array

[텐서 (nD)]
└─ numpy.array, torch.Tensor (3D 이상 포함)

[매핑]
└─ dict → 좌표 또는 속성값 표현
```

🢡 왜 이 연결이 중요할까?

- **머신러닝/딥러닝**: 모든 데이터는 텐서로 표현됨 → 결국 리스트/딕셔너리에서 변환
- 데이터 분석: 리스트 = 표의 행, 딕셔너리 = 열 기준 정보, 튜플 = 고정된 정보 묶음
- 자동화/웹 작업: 딕셔너리로 JSON 통신, 리스트로 결과 목록

필요하다면 이걸 **시각화 도식**, 또는 **실습용 예제 코드**로도 확장해드릴 수 있어요. 어떤 방식으로 확장하면 더 좋을까요? (예: "딥러닝 이미지 예제로 텐서 보여줘" 또는 "dict → DataFrame 변환 보여줘" 등)



판다스에서 제일 중요한 자료형은

훌륭한 질문이에요!

pandas에서 가장 중요한 자료형은 단연 Series와 DataFrame입니다.

이 둘이 pandas의 핵심 구조이자 자료형이며, 모든 분석은 이 두 가지를 중심으로 이뤄집니다.

🧱 1. Series (1차원 자료형)

- 정의: 하나의 열(column) 혹은 1차원 배열
- 형태: 인덱스를 가진 리스트

```
python

import pandas as pd

s = pd.Series([90, 85, 78], index=['철수', '영희', '민수'])
print(s)
```

출력:

```
go
철수 90
영희 85
민수 78
dtype: int64
```

• 특징:

- 리스트처럼 인덱스로 접근 가능
- 넘파이처럼 수치 계산 가능
- 하나의 열(column)로 사용됨

🧱 2. DataFrame (2차원 자료형)

- 정의: Series들이 가로로 모인 2차원 표 형태
- **형태**: 행(row)과 열(column)을 가진 테이블

```
python

df = pd.DataFrame({
    '이름': ['철수', '영희', '민수'],
    '점수': [90, 85, 78]
})
print(df)
```

출력:

```
이름 점수
0 철수 90
1 영희 85
2 민수 78
```

• 특징:

- Excel, CSV, DB 테이블 구조와 동일
- 행과 열로 데이터를 처리
- 정렬, 필터링, 그룹화, 시각화의 기본 구조



자료형	차원	설명	예시
Series	1D	하나의 열, 인덱스 있음	s = pd.Series([])
DataFrame	2D	여러 열로 구성된 테이블	<pre>df = pd.DataFrame({})</pre>

III 실무 기준으로 가장 중요한 이유

작업 유형	사용하는 자료형
엑셀/CSV 불러오기	DataFrame
열 필터링	Series
그룹 분석	DataFrame
시각화 전 처리	Series & DataFrame
머신러닝 데이터 준비	DataFrame

☑ 한 문장 요약

Series는 열, DataFrame은 표.

판다스는 결국 이 둘을 다루는 도구!

원하시면, 이 두 자료형의 관계나 변환(예: Series → DataFrame), 시각화 예제까지 확장해 드릴 수 있어요. 혹시 실습을 원하시나요? 또는 데이터 분석 목표에 맞게 활용법을 정리해드릴까요?



자료형을확인하는 방법과 자료형 변환, 그리고 자료형 변환이 필요한 경우는 언제?

아주 중요한 주제예요!

데이터 분석을 하다 보면 **자료형 확인** → **필요시 변환**은 필수 작업입니다.

pandas에서는 **자료형(type)**을 제대로 이해하고 다루는 것이 **정확한 분석**과 **오류 방지**의 핵심입니다.

☑ 1. 자료형 확인 방법

① dtype (단일 열의 자료형)

python

df['점수'].dtype

② dtypes (전체 열의 자료형)

```
python

df.dtypes
```

③ info() (전체 구조 + 결측치 + 자료형 등 요약)

```
python

df.info()
```

☑ 2. 자료형 변환 방법

방법	설명	예시
astype()	원하는 자료형으로 변환	df['나이'].astype(int)
<pre>pd.to_numeric()</pre>	숫자로 강제 변환, 오류 처리 가능	pd.to_numeric(df['점수'], errors='coerce')
<pre>pd.to_datetime()</pre>	날짜형 변환	pd.to_datetime(df['생년월일'])
<pre>pd.to_string()</pre>	문자열로 변환	df['ID'].astype(str)

예시 코드:

```
import pandas as pd

df = pd.DataFrame({
    '나이': ['20', '30', '40'], # 문자열
    '점수': ['90', '85', '78'], # 문자열
    '생일': ['2000-01-01', '1995-03-22', '1985-07-09']
})

# 문자열 → 숫자

df['나이'] = df['나이'].astype(int)

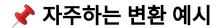
# 문자열 → 날짜

df['생일'] = pd.to_datetime(df['생일'])
```

▌ 3. 왜 자료형 변환이 필요한가요?

상황	이유	예시
숫자형인데 문자열로 저장됨	연산 불가, 정렬 오류	'85' + 5→오류
날짜 데이터를 분석	시계열 분석 불가	문자열 상태에선 .dt.year 사용 불가
범주형 데이터를 정리	시각화, 그룹 분석 오류	성별: '남', '여' → category
문자열 데이터를 숫자로 변환	머신러닝 학습이 안됨	"Male" \rightarrow 0, "Female" \rightarrow 1





원본 컬럼 예	변환 후 자료형	사용 이유
'100'	int	평균 계산, 정렬 등 수치연산
'2022-01-01'	datetime64[ns]	날짜 간 차이 계산
'남', '여'	category	메모리 절약, 그룹 연산 빠름
'True', 'False'	bool	필터링, 조건문 사용

☑ 한눈에 보기

```
python
# 전체 자료형 확인
print(df.dtypes)
# '점수'를 숫자로 변환
df['점수'] = df['점수'].astype(int)
# '생일'을 날짜로 변환
df['생일'] = pd.to datetime(df['생일'])
# '성별'을 범주형으로 변환
df['성별'] = df['성별'].astype('category')
```

필요하다면 변환 후 분석/시각화 예제나

실제 오류 발생하는 사례도 만들어드릴 수 있어요.

실습하고 싶은 방향 있으실까요? (예: 날짜 분석, 그룹화, 머신러닝용 변환 등)



판다스가 아닌. 일반 파이썬에서 자료형 때문에 고생하는 경우를 미리 알려줘,

훌륭한 질문입니다.

pandas가 아닌 **일반 파이썬(Pure Python)**에서도 자료형 때문에 오류나 고생하는 경우는 **초보자뿐 아니라 중급자** 도 자주 겪는 문제입니다.

아래는 자주 발생하는 사례 + 그 원인 + 해결법을 차근히 정리해드릴게요.



🔪 1. 숫자형인데 문자열로 받아서 연산이 안 됨

```
python
x = input("숫자를 입력하세요: ") # 사용자가 "10" 입력 → 문자열
print(x + 5) # ★ TypeError 발생
```



```
python

TypeError: can only concatenate str (not "int") to str
```

☑ 해결:

```
python

x = int(input("숫자를 입력하세요: ")) # 형변환 필수
print(x + 5) # → 15
```

📏 2. 리스트 복사 시 같은 객체를 공유해서 생기는 문제

```
python

a = [1, 2, 3]
b = a # 얕은 복사 (주소 공유)

b.append(4)
print(a) # 【 [1, 2, 3, 4] → 원하지 않는 결과
```

☑ 해결:

```
python
b = a.copy() # 또는 b = list(a)
```

📏 3. None, 0, '', False를 헷갈려 조건문이 이상해짐

```
python

user_input = ''

if user_input:
    print("입력 있음") # ★ 실행 안 됨 → 빈 문자열은 False로 취급

else:
    print("입력 없음") # ✔ 실행됨
```

☑ 해결:

```
python

if user_input != '':
    print("입력 있음")
```

igwedge 4. \mathtt{dict} 키에 리스트 사용 ightarrow igwedge TypeError

```
python

key = [1, 2, 3]

my_dict = {key: "값"} # 🗙 리스트는 unhashable
```

☑ 해결:

```
python

key = (1, 2, 3) # 튜플은 해시 가능

my_dict = {key: "값"}
```

📏 5. 정수 나눗셈 결과가 예상과 다름 (Python 2 vs Python 3)

```
python
result = 5 / 2 # Python 3: 2.5 / Python 2: 2
```

☑ 해결:

- Python 3에서는 항상 float 결과
- // 연산자는 정수 나눗셈:

```
python
result = 5 // 2 # 2
```

📏 6. is vs == 사용 헷갈림

```
python

a = [1, 2]
b = [1, 2]

print(a == b) # ✓ True (값이 같음)
print(a is b) # ✗ False (주소가 다름)
```

☑ 요약:

비교	의미
==	값(value) 비교
is	객체 주소(id) 비교

📏 7. 자료형 혼용으로 정렬 이상



```
python

data = [10, '20', 5]
print(sorted(data)) # X TypeError
```

☑ 해결:

```
python
# 모든 요소를 문자열로 변환 후 정렬
print(sorted(map(str, data)))
```

◎ 요약: 일반 파이썬에서 자주 고생하는 자료형 사례

상황	문제	해결
사용자 입력	input()은 문자열	int(), float() 등으로 변환
리스트 복사	같은 객체 공유	.copy(),list()
조건문	'', None , 0 = False	값 존재 여부 명시적 비교
딕셔너리 키	리스트 사용 🗙	튜플로 변경
== vs is	객체 vs 값 비교	값 비교는 == 사용
정렬	숫자/문자 혼합	모두 변환해서 정렬
나눗셈	5/2 결과 다름	//, / 명확히 사용

필요하다면 위 사례들을 코드 실습으로 구성한 워크북, 또는 문제-풀이 퀴즈 형태로도 구성해드릴 수 있어요. 원하시는 학습 방식이 있으실까요?



워크북형식으로

좋습니다!

초보자도 이해하기 쉬운 **"파이썬 자료형 워크북"** 형식으로 문제-예상 출력-해결 코드 구조로 구성해드릴게요.

🧠 개념을 확인하고, 🚣 실습하고, 🛠 고치면서 이해하는 워크북입니다.

🔟 파이썬 자료형 워크북 (입문자용)

📔 [1장] 입력 자료형 오류: 문자열을 숫자로 착각

🧈 문제

```
python

x = input("숫자를 입력하세요: ") # "10" 입력
print(x + 5)
```

🤍 예상 출력

```
python

TypeError: can only concatenate str (not "int") to str
```

火 해결 코드

```
python

x = int(input("숫자를 입력하세요: ")) # 문자열 → 정수
print(x + 5)
```

📔 [2장] 리스트 복사의 함정

達 문제

```
python

a = [1, 2, 3]
b = a
b.append(4)
print(a)
```

🤍 예상 출력

```
csharp
[1, 2, 3, 4] ← 예상과 다름
```

❤ 해결 코드

```
python

b = a.copy()
b.append(4)
print(a) # [1, 2, 3]
print(b) # [1, 2, 3, 4]
```

[3장] 조건문에서 '', None, 0 헷갈림

🚅 문제

```
python

name = ''
if name:
    print("이름이 입력되었습니다.")
else:
    print("입력 없음")
```

____ 출력

```
입력 없음
```

父 해결 코드

```
python

if name != '':
   print("이름이 입력되었습니다.")
```

📔 [4장] 딕셔너리에서 리스트를 키로 사용

🧈 문제

```
python

key = [1, 2]

my_dict = {key: "값"} # 🗙
```

◯ 에러

```
TypeError: unhashable type: 'list'
```

❤️ 해결 코드

```
python

key = (1, 2) # 튜플은 키로 사용 가능

my_dict = {key: "값"}
```

📔 [5장] is 와 == 혼동

🧈 문제

```
python

a = [1, 2]
b = [1, 2]
print(a == b) # ?
print(a is b) # ?
```

출력 출력

```
graphql

True ← 값은 같다

False ← 메모리 주소는 다르다
```

💡 개념 정리

연산자	의미	
==	값이 같은가?	
is	같은 객체인가? (메모리 주소)	

📔 [6장] 서로 다른 자료형 정렬

🧈 문제

```
python

data = [10, '20', 5]
print(sorted(data))
```

○ 에러

```
javascript

TypeError: '<' not supported between instances of 'str' and 'int'</pre>
```

🛠 해결 코드

```
python

# 숫자로 바꾸거나 문자열로 통일
print(sorted(map(int, data))) # [5, 10, 20]
```

📔 [7장] 날짜 문자열을 그대로 사용

🧈 문제

```
python
birth = "2000-01-01"
print(birth.year) # X
```

의 에러

```
pgsql
AttributeError: 'str' object has no attribute 'year'
```

父 해결 코드

```
from datetime import datetime
birth = datetime.strptime("2000-01-01", "%Y-%m-%d")
print(birth.year) # 2000
```

📙 [마무리 요약]

문제 상황	문제 원인	해결 방법 예시
input 입력 값 오류	문자열 → 숫자 변환 누락	<pre>int(input())</pre>
리스트 복사 공유	얕은 복사	.copy(),list()
딕셔너리 키 오류	리스트는 키 불가	튜플 사용 (1, 2)
is vs ==	객체 vs 값 비교 혼동	== 사용
정렬 불가	문자열+숫자 혼합	map(int, list)
날짜 오류	문자열 → 날짜 변환 누락	<pre>datetime.strptime()</pre>

필요하시면 이 워크북을 PDF, Jupyter Notebook, **퀴즈용 버전**, 또는 **실습용 데이터셋과 함께 구성**해드릴 수 있습니다.

어떤 형식이 가장 학습에 도움이 될까요?



퀴즈용 버전

좋아요!

아래는 학습 효과를 높이기 위한 **[파이썬 자료형 퀴즈형 워크북]** 버전입니다. 각 문항은 선택형 또는 빈칸형으로 구성되었고, 마지막에 정답과 해설을 드리겠습니다.

◎ 파이썬 자료형 퀴즈 워크북 (기초편)

🔢 [Q1] 입력 자료형 오류

```
python
x = input("숫자를 입력하세요: ")
print(x + 10)
```

- Q. 위 코드 실행 시 사용자에게 "10"을 입력받으면 어떤 일이 벌어질까?
- a) 20이 출력된다
- b) "1010"이 출력된다
- c) TypeError가 발생한다
- d) 10.0이 출력된다

📋 [Q2] 리스트 복사

```
python

a = [1, 2, 3]
b = a
b.append(4)
print(a)
```

Q. 출력 결과는?

- a) [1, 2, 3]
- b) [1, 2, 3, 4]
- c) 오류 발생
- d) [4]

🤐 [Q3] 딕셔너리 키 관련

```
python

key = [1, 2]
d = {key: "武"}
```

Q. 위 코드의 실행 결과는?

- a) 정상 실행됨
- b) 딕셔너리 키가 [1, 2]가 됨
- c) TypeError 발생
- d) 키가 튜플로 자동 변환됨

🧠 [Q4] == 와 is의 차이

```
python
a = [1, 2]

b = [1, 2]
```

Q. 다음 중 올바른 설명은?

- a) a == b 는 False다
- b) a is b는 True다
- c) a == b는 True고 a is b는 False다
- d) 두 비교는 항상 동일한 결과를 준다



👛 [Q5] 자료형 확인

```
python
x = "2024-01-01"
```

Q. x의 자료형은 무엇인가?

- a) datetime
- b) str
- c) int
- d) date



<u>↑</u> [Q6] 정렬 문제

```
python
data = [10, "5", 7]
sorted_data = sorted(data)
```

Q. 위 코드를 실행하면?

- a) [5, 7, 10]
- b) ["10", "5", "7"]
- c) TypeError 발생
- d) [10, 5, 7]

📆 [Q7] 날짜 속성 사용

```
python
birth = "2000-01-01"
print(birth.year)
```

Q. 어떤 문제가 발생하는가?

- a) 2000이 출력된다
- b) AttributeError 발생
- c) 문자열로 "year"가 출력됨
- d) 타입 변환 없이도 사용 가능

☑ 정답 및 해설

문항	정답	해설	
Q1	С	input() 은 문자열 반환 → 문자열 + 숫자 → TypeError	
Q2	b	$b = a$ 는 같은 리스트를 참조 $\rightarrow b \cdot append()$ 는 a 도 바꿈	
Q3	С	리스트는 해시 불가능 → 딕셔너리 키로 사용 불가	
Q4	С	==는 값 비교, is는 메모리 비교	
Q5	b	문자열로 저장된 날짜	
Q6	С	숫자와 문자열 혼합은 정렬 불가	
Q7	b	문자열에는 .year 속성 없음 $ ightarrow$ datetime으로 변환 필요	

四 다음 단계 아이디어

- ■ 틀린 문제 복습용 미니 실습 제공
- 🔲 Jupyter Notebook용 퀴즈버전 제작
- □ 점수 자동 채점 기능 구현
- □ 실전형 "자료형 실수 찾아서 고치기" 과제 모음

필요하다면 이 퀴즈를 실습용 .ipynb, .pdf, 또는 Streamlit 인터랙티브 웹버전으로 제작해드릴 수 있어요. 어떤 형식이 학습에 더 도움이 될까요?