

Python

CH07: 모듈

전종준 교수

서울시립대학교 통계학과/통계데이터사이언스대학원

Contents

- 모듈의 개념과 사용법
- 사용자 정의 모듈
- 중급: 가계부 프로그램 만들기

모듈의 개념과 사용법

파이썬 모듈 기초

- ▶ 모듈(Module): 파이썬 코드가 저장된 파일
- ▶ 함수, 변수, 클래스 등을 담아두고 재사용 가능
- ▶ 코드의 재사용성과 유지보수성을 높임

예시:

```
1 import math
2 print(math.sqrt(16))
```

학습 목표

- ▶ 파이썬 모듈의 개념을 이해하고 사용할 수 있다.
- ▶ 표준 모듈과 사용자 정의 모듈을 구분할 수 있다.
- ▶ 모듈을 사용하는 다양한 문법을 익힌다.

모듈이란?

- ▶ 다른 파이썬 파일에 정의된 기능을 가져와 사용하는 방식
- ▶ 대표적인 표준 모듈: `math`, `random`, `datetime`
- ▶ 사용자 정의 모듈 만들기

import 기본 사용법

- ▶ import 모듈이름 형태로 불러옴
- ▶ 모듈 이름을 붙여서 함수/변수를 사용

예시:

```
1 import math
2 print(math.pi)
3 print(math.sqrt(9))
```

import ... as ... 사용

- ▶ 모듈에 별명을 붙이면 코드가 더 짧아짐

예시:

```
1 import math as m
2 print(m.sin(0))
3 print(m.log(10))
```

sin 함수의 개형 시각화 (matplotlib)

- ▶ matplotlib을 사용해 sin 함수 그래프를 그리기
- ▶ -3부터 0.1씩 증가하여 600개의 x 값을 만들기

예시 코드:

```
1 import math
2 import matplotlib.pyplot as plt
3 x_values = []
4 sin_values = []
5 x = -3
6 for _ in range(600):
7     x_values.append(x)
8     sin_values.append(math.sin(x))
9     x += 0.1
10 plt.plot(x_values, sin_values)
11 plt.title("y = sin(x)")
12 plt.xlabel("x")
13 plt.ylabel("sin(x)")
14 plt.grid(True)
15 plt.show()
```

from 모듈 import 이름

▶ 모듈 이름 없이 직접 사용 가능

예시:

```
1 from math import pi, sqrt
2 print(pi)
3 print(sqrt(49))
```

여러 import 방식 비교

- ▶ `import math`: 전체 모듈
- ▶ `import math as m`: 별칭 사용
- ▶ `from math import sqrt`: 필요한 것만

비교 예시:

```
1 import math
2 print(math.sqrt(4))
3
4 from math import sqrt
5 print(sqrt(4))
```

math 모듈 실습

- ▶ 수학 관련 기능 제공
- ▶ 삼각함수, 로그, 제곱근 등

예시:

```
1 import math
2 print(math.factorial(5))
3 print(math.cos(math.radians(60)))
```

random 모듈 실습

▶ 난수 생성 관련 함수 제공

예시:

```
1 import random
2 print(random.randint(1, 6))    # 주사위
3 print(random.choice([" ", " ", " "]))
```

datetime 모듈 실습

▶ 날짜와 시간 관련 기능

예시:

```
1 from datetime import datetime
2 now = datetime.now()
3 print("현재 시각:", now)
4 print("연도:", now.year, "월:", now.month)
```

두 날짜 사이의 일 수 계산하기

- ▶ `datetime.date` 객체를 사용해 날짜 차이를 계산할 수 있음
- ▶ 날짜를 직접 지정해서 함수에 넣으면 일 수를 알려줌

예시 코드:

```
1 from datetime import date
2
3 def days_between(d1, d2):
4     return abs((d2 - d1).days)
5
6 # 사용 예시
7 day1 = datetime.date.today()
8 day2 = date(2025, 12, 25)
9 print("크리스마스까지 남은 날:", days_between(day1, day2))
```

사용자 정의 모듈

Colab 파일 구조 살펴보기

- ▶ /content/는 Colab의 작업 디렉토리.
- ▶ sample_data는 Colab이 제공하는 샘플 데이터 폴더

예시 코드:

```
1 import os
2
3 # 현재 디렉토리 확인
4 print(os.getcwd())
5
6 # 폴더 안 파일 목록 보기
7 print(os.listdir("/content"))
8 print(os.listdir("/content/sample_data"))
```

사용자 정의 모듈 만들기

- ▶ 함수나 변수를 별도 파일에 저장
- ▶ 파일 이름: mytools.py

mytools.py 예시:

```
1  def add(a, b):  
2      return a + b  
3  
4  def greet(name):  
5      print(f"{name}, 안녕하세요!")
```

모듈 파일 사용하기

- ▶ 같은 폴더에 있는 모듈을 불러와 사용

main.py 예시:

```
1  import mytools
2
3  print(mytools.add(2, 3))
4  mytools.greet("지민")
```

```
if __name__ == "__main__"
```

- ▶ 모듈 실행 시에만 특정 코드 실행
- ▶ import 시엔 실행되지 않음

예시:

```
1  def hello():  
2      print("안녕하세요!")  
3  
4  if __name__ == "__main__":  
5      hello()
```

모듈 연습문제 1

▶ `math`, `random`, `datetime`을 활용해 보세요.

연습:

1. 오늘 날짜를 출력하세요.
2. 1부터 45 사이의 번호 6개를 무작위로 뽑아 출력하세요.
3. 3.141592의 제곱근을 소수 둘째 자리까지 출력하세요.

모듈 연습문제 1 - 풀이 예시

```
1  from datetime import date
2  import random
3  import math
4
5  print("오늘:", date.today())
6  print("로또 번호:", random.sample(range(1, 46), 6))
7  print("제곱근:", round(math.sqrt(3.141592), 2))
```

모듈 연습문제 2 - 사용자 정의

- ▶ 직접 모듈 파일을 만들어보세요.

목표:

1. `calculator.py` 모듈 생성
2. 덧셈, 뺄셈 함수 정의
3. `main.py`에서 모듈 사용

사용자 정의 모듈 연습 - 정답 예시

calculator.py

```
1  def add(a, b):  
2      return a + b  
3  
4  def sub(a, b):  
5      return a - b
```

main.py

```
1  import calculator  
2  print(calculator.add(10, 5))  
3  print(calculator.sub(10, 5))
```


모듈 관련 주의할 점

- ▶ 모듈 이름과 파일 이름이 동일해야 함
- ▶ 폴더 구조에 따라 경로 설정 필요
- ▶ 이름 충돌 방지를 위해 별칭 사용 고려

모듈 이름과 파일 이름은 같아야 함

- ▶ 모듈 이름은 실제 파일 이름(확장자 제외)과 동일해야 함
- ▶ 예: `myutils.py` → `import myutils`
- ▶ 다른 이름으로 import하면 오류 발생

예시 (올바른 경우):

```
1 # 파일명: calculator.py
2 def add(a, b):
3     return a + b
4
5 # main.py에서
6 import calculator
7 print(calculator.add(3, 4))
```

정리 퀴즈

1. `random.randint()` 함수의 역할은?
2. `from 모듈 import 함수` 구문의 장점은?
3. 사용자 정의 모듈은 어떻게 만들고 사용하는가?

중급: 가계부 프로그램 만들기

오늘의 목표

- ▶ 기존에 만들었던 가계부 프로그램을 **GUI**로 바꿔 보자
- ▶ Gradio를 이용해 웹 브라우저에서 동작하도록 만든다
- ▶ 사용자 입력을 받는 인터페이스와 버튼으로 동작을 제어한다

Gradio란?

- ▶ 파이썬 코드에 GUI를 쉽게 추가할 수 있게 도와주는 도구
- ▶ `pip install gradio`만 하면 바로 사용 가능
- ▶ 웹 브라우저에서 실행되는 깔끔한 인터페이스 제공

예:

```
import gradio as gr
def greet(name):
    return f"{name}님, 안녕하세요!"
gr.Interface(fn=greet, inputs="text", outputs="text").launch()
```

Gradio 앱의 구조

- ▶ `gr.Blocks()`: 여러 입력창, 버튼, 출력 박스를 구성하는 컨테이너
- ▶ Textbox, Button, Row 등을 배치할 수 있음
- ▶ `click()`: 버튼을 눌렀을 때 호출할 함수를 지정

예시 구조:

```
with gr.Blocks() as demo:
```

```
    ├── 입력창들
    ├── 출력창
    └── 버튼들
```

가계부 프로그램 기존 함수 재활용하기

- ▶ 기존에 만든 `add_record`, `show_stats` 함수들을 그대로 사용 가능
- ▶ 입력값은 Textbox → 매개변수
- ▶ 출력은 Textbox → return 문자열

예시:

```
def add_record(date, category, desc, amount):  
    ledger.append({...})  
    return "추가 완료"
```


버튼 클릭과 함수 연결

- ▶ `btn.click(함수, inputs=..., outputs=...)` 형식
- ▶ `inputs` 의 형식은
- ▶ 버튼을 누르면 해당 함수가 호출되고 결과가 출력됩니다

예시:

```
btn_add.click(add_record,  
               inputs=[date, category, desc, amount],  
               outputs=output)
```

Gradio 인터페이스 전체 구조

- ▶ `gr.Blocks()`를 사용하여 여러 입력창과 버튼을 하나의 앱으로 구성합니다.
- ▶ 입력(날짜, 분류, 내용, 금액) → 처리 함수 → 출력 결과로 연결됩니다.
- ▶ 각 버튼은 서로 다른 동작(add, view, stats 등)을 수행합니다.

구성 요소:

1. 입력창 4개
2. 출력창 1개
3. 버튼 5개
4. 함수 연결 (click)

입력창 구성: 날짜 / 분류 / 내용 / 금액

- ▶ `gr.Row()`로 한 줄에 입력창 4개를 배치
- ▶ 각 입력값은 이후 함수의 매개변수로 전달됩니다.

코드 예시:

```
1 with gr.Row():
2     date = gr.Textbox(label="날짜")
3     category = gr.Textbox(label="분류")
4     desc = gr.Textbox(label="내용")
5     amount = gr.Textbox(label="금액")
```

출력창과 버튼 구성

- ▶ `output = gr.Textbox(...)`: 결과 메시지를 표시할 출력 박스
- ▶ `gr.Button(...)`: 기능별로 5개의 버튼을 만든다.

코드 예시:

```
1 output = gr.Textbox(label="결과 출력", lines=10)
2
3 with gr.Row():
4     btn_add = gr.Button("추가")
5     btn_view = gr.Button("보기")
6     btn_stats = gr.Button("통계")
7     btn_save = gr.Button("저장")
8     btn_load = gr.Button("불러오기")
```

버튼 클릭 → 함수 실행 연결

- ▶ 각 버튼의 `click()` 메서드는 함수와 연결됩니다.
- ▶ 입력값을 받아 처리한 후 결과를 출력창에 표시합니다.

예시:

```
1 btn_add.click(add_record,  
2             inputs=[date, category, desc, amount],  
3             outputs=output)  
4  
5 btn_view.click(view_ledger, outputs=output)  
6 btn_stats.click(show_stats, outputs=output)  
7 btn_save.click(save_file, outputs=output)  
8 btn_load.click(load_file, outputs=output)
```

입력창 UI 구성: 날짜, 분류, 내용, 금액

- ▶ 사용자로부터 하나의 가계부 항목 정보를 입력받기 위한 4개의 필드를 구성:

코드:

```
1 with gr.Row():
2     date = gr.Textbox(label="날짜")
3     category = gr.Dropdown(
4         choices=["식비", "교통", "문화", "의료", "기타"],
5         label="분류",
6         value="식비" # 기본 선택값
7     )
8     desc = gr.Textbox(label="내용")
9     amount = gr.Textbox(label="금액")
```

- ▶ Textbox: 사용자가 문자열을 자유롭게 입력
- ▶ Dropdown: 미리 정해진 선택지 중 하나만 선택 가능 → 오타 방지
- ▶ 기본 분류는 식비로 지정되어 있어 편리