

파이썬 프로그래밍

10-1: 함수 (2) / 모듈

2022.5.3



인하대학교
INHA UNIVERSITY

산업경영공학과 임세실

Overview

- 고급함수 사용하기
- 모듈 이해하기
- 패키지 이해하기



고급 함수 사용하기

- 지역변수와 전역변수
- 함수를 리스트에 할당해 사용하기
- 함수를 매개변수로 전달하기



함수 내 변수의 효력 범위

- 다음 코드를 실행했을 때 print(a)의 출력값은?

```
a=1
def var_test(a):
    a=a+1

var_test(a)
print(a)
```

1) 1

2) 2

- ➡ 함수 내 변수 a는 지역변수 (local variable)
- ➡ 함수 밖 변수 a는 전역변수 (global variable)

변수의 구분

지역(local) 변수

- 함수, 반복문, 분기문 등의 내부에서 생성
- 생성된 범위(코드블록) 안에서만 유효
- 코드블록 블록 끝나면 소멸

전역(global) 변수

- 모든 함수, 반복문, 분기문 등의 외부에 선언되는 변수
- 프로그램 전체에서 사용가능
- 프로그램 종료시까지 남아있음

지역변수와 전역변수의 예시

- 사각형의 넓이 구하는 함수

```
width = 10  
height = 20
```

```
def area(w,h):  
    result=w*h  
    return result
```

```
print(area(width,height))
```

➡ 함수 밖에 있으므로 전역변수

➡ 함수 안에 있으므로 지역변수

함수 내에서 전역변수의 사용

- global 키워드를 써서 전역변수로 선언해야함

```
a=1
def var_test():
    global a
    a=a+1

var_test()
print(a)
```

➡ 매개변수 없이 함수 밖의 전역 변수 `a`를 직접 불러옴

함수를 리스트에 할당해 사용하기

- 함수를 리스트에 할당 가능

```
def plus(a,b):  
    return a+b  
  
def minus(a,b):  
    return a-b  
  
whole = [plus, minus]  
  
print(whole[0](1,2))  
print(whole[1](1,2))
```

[12] ✓ 0.1s Python

... 3
-1

함수를 매개변수로 전달하기: map() 함수

- 리스트의 각 요소에 함수를 적용한 반환값으로 새 리스트 구성

map(함수, 리스트)

- 리스트 [1,2,3,4,5]의 각 요소를 제공해 새 리스트 [1,4,8,16,25]를 만듦

```
def power(item):  
    return item*item  
  
list1 = [1,2,3,4,5]  
list2 = map(power, list1)  
print(list2)  
print(list(list2))  
[6] ✓ 0.6s  
... <map object at 0x109f711b0>  
[1, 4, 9, 16, 25]
```

함수를 매개변수로 전달하기: filter() 함수

- 리스트의 각 요소에 함수를 적용한 반환값이 True인 요소로 새 리스트 구성

filter(함수, 리스트)

- 리스트 [1,2,3,4,5] 중 3보다 작은 요소만 모아서 새 리스트 [1,2]를 만듦

```
def under_3(item):  
    return item<3  
  
list1=[1,2,3,4,5]  
list2=filter(under_3, list1)  
print(list2)  
print(list(list2))  
[7] ✓ 0.7s  
... <filter object at 0x109f72710>  
[1, 2]
```

함수를 매개변수로 전달하기: 람다 (lambda) (1)

- 이름없는 함수라고 불림
- 함수를 한줄로 간결하게 생성할 때 사용

lambda 매개변수: 리턴값

- def를 사용해야할 정도로 복잡하지 않거나, def 사용할 수 없는 곳에 사용
- 함수를 매개변수로 전달할 때 유용하게 사용가능

함수를 매개변수로 전달하기: 람다 (lambda) (2)

- map 함수를 매개변수로 변환

def 사용

```
def power(item):  
    return item*item  
  
list1 = [1,2,3,4,5]  
list2 = map(power, list1)  
print(list2)  
print(list(list2))
```

람다 사용 (1)

```
power = lambda item: item*item  
  
list1 = [1,2,3,4,5]  
list2 = map(power, list1)
```

람다 사용 (2)

```
list1 = [1,2,3,4,5]  
list2 = map(lambda item: item*item, list1)
```

모듈 이해하기

- 사용자 생성 모듈 만들기
- 모듈 사용하기
- 메인 모듈과 하위모듈



Python의 모듈 (Module)

- 함수나 변수들을 모아놓은 파일
- 다른 파이썬 프로그램에서 불러와 사용할 수 있도록 만들어진 파이썬 파일
- 각각의 소스파일 (.py)을 일컬어 모듈이라고 함
- 하나 또는 여러 개의 모듈을 모아놓은 것을 패키지(Package)



모듈의 분류

- 표준 모듈
 - 파이썬에 기본적으로 내장되어 있는 모듈
- 외부 모듈 (서드파티 모듈; 3rd party module)
 - 다른 프로그래머 혹은 업체가 만들어서 공개한 모듈
- 사용자 생성 모듈
 - 프로그래머가 직접 작성한 모듈



사용자 생성 모듈 만들기 (1)

- 덧셈과 뺄셈 함수만 있는 모듈 만들기
 - 텍스트 편집기 활용하기: **모듈명.py**로 저장

cal.py

```
cal.py ×  
  
cal.py > ...  
1  # cal.py  
2  def add(a,b):  
3      return a+b  
4  
5  def sub(a,b):  
6      return a-b  
7
```

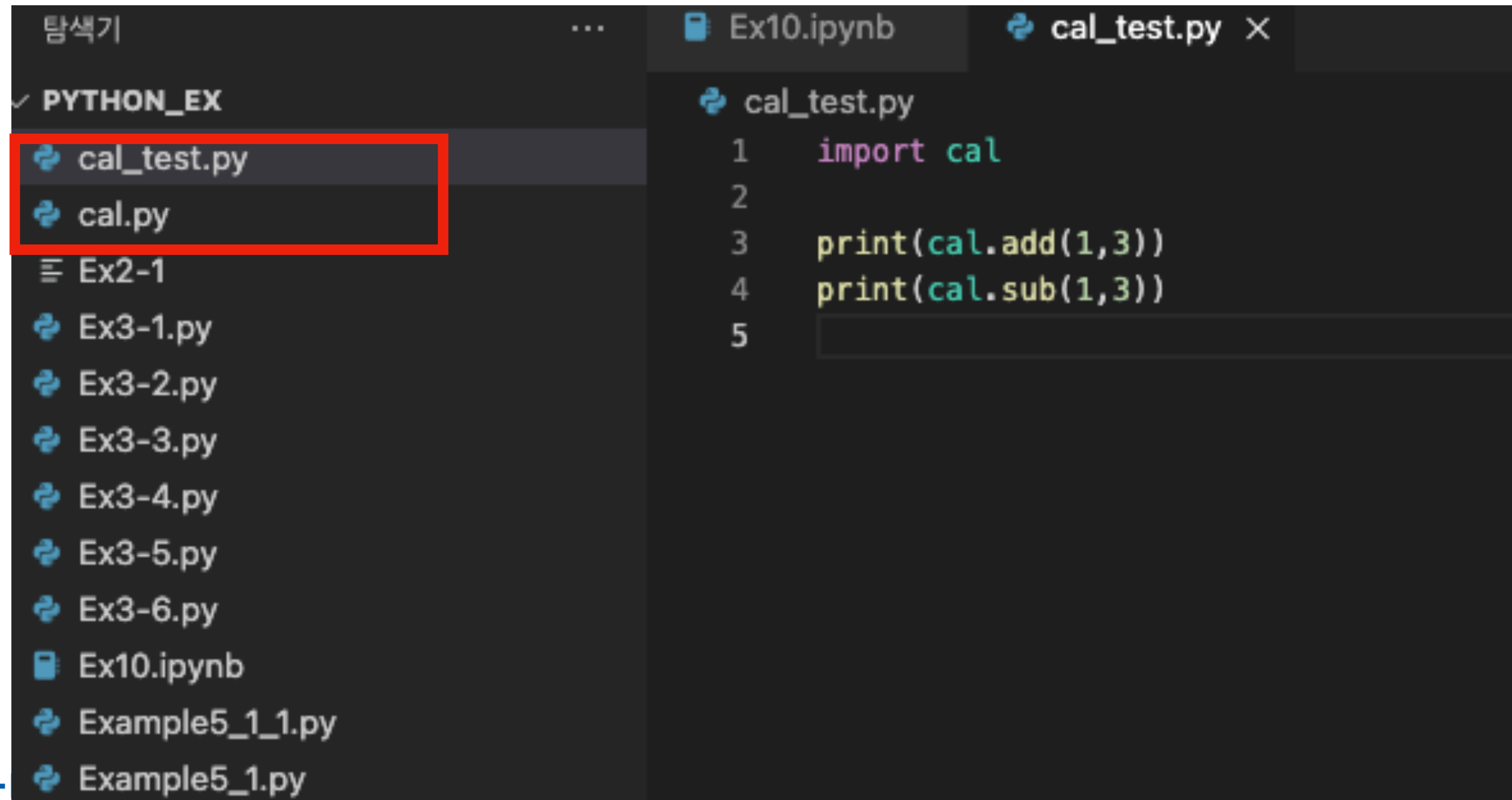

사용자 생성 모듈 만들기 (2)

- 덧셈과 뺄셈 함수만 있는 모듈 만들기
 - 주피터 노트북 / Google Colab 활용하기: %%writefile 키워드 사용

```
▶ ▼  
%%writefile cal.py  
def add(a,b):  
    return a+b  
  
def sub(a,b):  
    return a-b  
[6] ✓ 0.1s  
... Writing cal.py
```

모듈 사용하기 (1)

- 텍스트 편집기 사용할 경우, 소스파일들이 모두 같은 저장위치에 있어야 함



모듈 사용하기 (2)

- import 키워드를 통해 다른 모듈 내의 코드에 대해 접근 가능
 - 모듈 내 모든 코드 불러오기

```
import 모듈명
```

- 모듈 내 일부 함수 및 변수만 불러오기

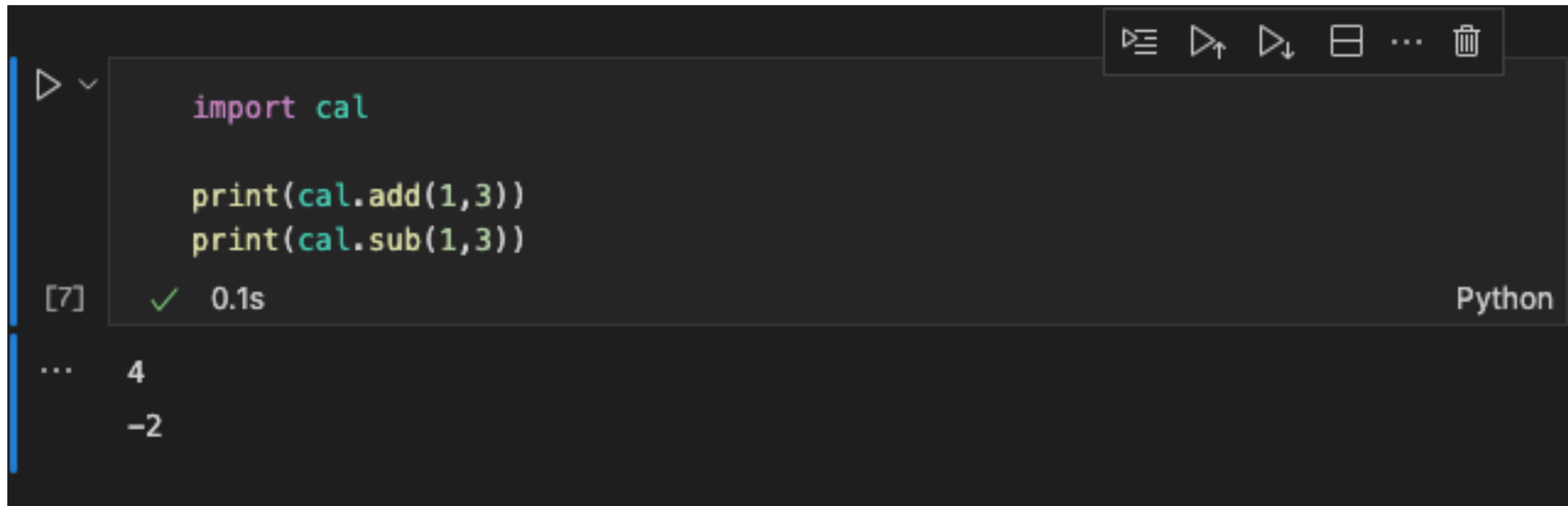
```
From 모듈명 import 함수/변수
```

- 모듈 이름을 짧게 줄여서 가져오기

```
import 모듈명 as 별명
```

모듈 사용하기 (3)

- 모듈 내 모든 코드 불러오기
- 모듈 내의 함수 혹은 변수 불러올 때 **모듈명.함수명**의 식으로 써야함



The screenshot shows a Python IDE with a dark theme. The main editor window contains the following code:

```
import cal

print(cal.add(1,3))
print(cal.sub(1,3))
```

Below the code, the output is displayed:

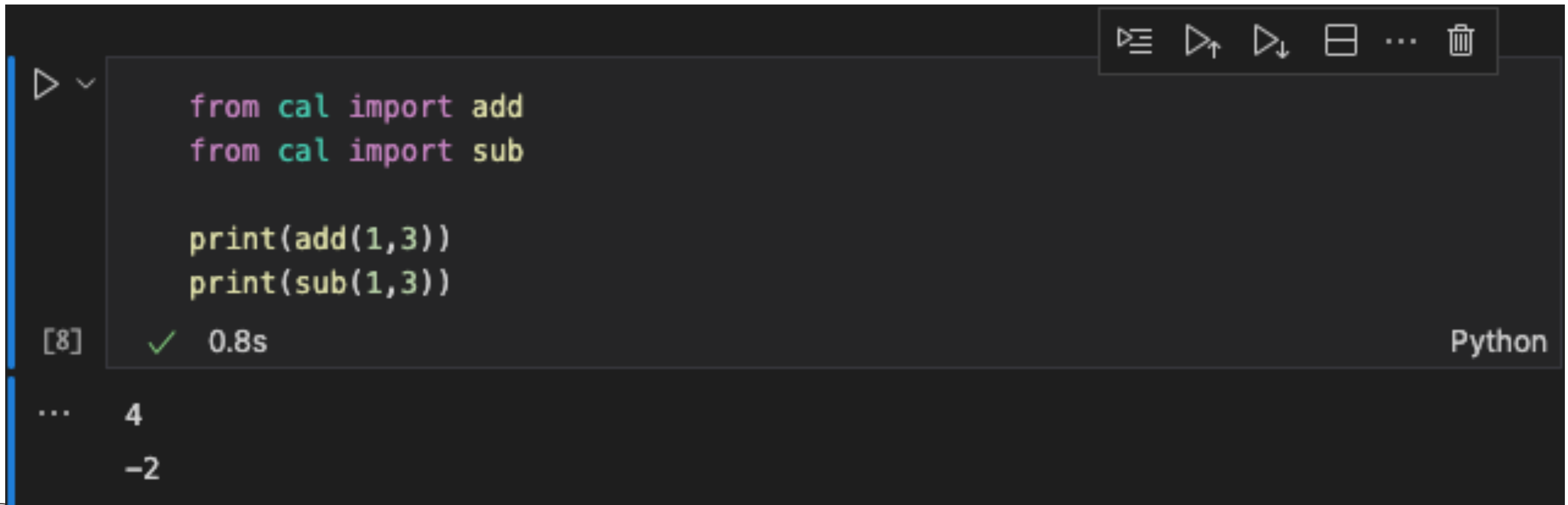
```
[7] ✓ 0.1s

... 4
    -2
```

The word "Python" is visible in the bottom right corner of the IDE window.

모듈 사용하기 (4)

- 모듈 내 일부 함수 및 변수만 불러오기
 - 모듈 내 필요한 함수 혹은 변수를 따로 import해서 호출시 함수명만 씀



The screenshot shows a Python IDE interface. The code editor contains the following Python code:

```
from cal import add
from cal import sub

print(add(1,3))
print(sub(1,3))
```

Below the code editor, the console shows the execution results:

```
[8] ✓ 0.8s Python
... 4
    -2
```

모듈 사용하기 (5)

- 참고: 모듈 내의 모든 함수 및 변수 가져오면서, 모듈명 붙이기 없기를 원할 때
- 와일드카드(*) 사용

```
From 모듈명 import *
```

- 함수 및 변수들끼리 이름 충돌이 발생할 수 있음
- 가급적 사용 안하는 것이 좋음

모듈 사용하기 (6)

- 모듈 이름을 짧게 줄여서 가져오기
- 모듈명 대신 별명 사용 가능

```
import cal as c
```

```
print(c.add(1,3))
```

```
print(c.sub(1,3))
```

[9]

✓ 0.1s

Python

...

4

-2



메인 모듈과 하위 모듈 (1)

- 메인 모듈: 파이썬에서 제일 먼저 실행되는 파일
- 하위 모듈: 메인 모듈에 import 되어서 실행되는 소스 파일
- 메인 모듈은 `__name__` 변수를 통해 확인 가능
 - 실행 모듈의 이름을 저장하는 전역변수
 - 메인 모듈의 경우 “`__main__`”이 저장됨
 - 하위 모듈은 모듈명이 저장됨

메인 모듈과 하위 모듈 (2)

- `__name__` 변수 확인하기

```
%%writefile test_module.py

print("모듈의 __name__ 출력하기")
print(__name__)
print()

[11] ✓ 0.1s

... Writing test_module.py

import test_module

print("메인의 __name__ 출력하기")
print(__name__)
print()

[12] ✓ 0.6s Python

... 모듈의 __name__ 출력하기
test_module

메인의 __name__ 출력하기
__main__
```

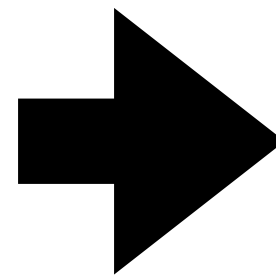
메인 모듈과 하위 모듈 (3)

- `__name__` 변수를 메인 모듈인지 확인해서 분기점 만드는데 활용 가능
- 예: cal1.py 모듈에 활용 예시 추가

```
%%writefile cal1.py
def add(a,b):
    return a+b

def sub(a,b):
    return a-b

print("add(1,3):", add(1,3))
print("sub(1,3):", sub(1,3))
```



```
import cal1 as c
print(c.add(1,2))
print(c.sub(1,2))

[15] ✓ 0.7s

... add(1,3): 4
      sub(1,3): -2
      3
      -1
```

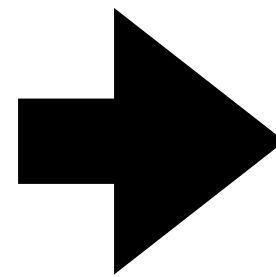
메인 모듈과 하위 모듈 (4)

- `__name__` 변수를 메인 모듈인지 확인해서 분기점 만드는데 활용 가능
- 예: cal1.py 모듈에 활용 예시 추가

```
%%writefile cal1.py
def add(a,b):
    return a+b

def sub(a,b):
    return a-b

if __name__ == "__main__":
    print("add(1,3):", add(1,3))
    print("sub(1,3):", sub(1,3))
```



```
import cal1 as c
print(c.add(1,2))
print(c.sub(1,2))

[17] ✓ 0.1s

... 3
    -1
```

대표적 표준 모듈: random 모듈

- 랜덤한 값을 생성할 때 사용

함수명	기능
random()	0.0과 1.0 사이의 랜덤한 float 반환
uniform(a, b)	a와 b 사이의 랜덤한 float 반환
randrange(b)	0과 b 사이의 랜덤한 int 반환
randrange(a, b)	a와 b 사이의 랜덤한 int 반환
choice(list)	리스트 내부의 요소 하나를 랜덤하게 선택
shuffle(list)	리스트의 요소를 랜덤하게 섞음
sample(list, k=n)	리스트의 요소 중 n개를 랜덤하게 뽑음

대표적인 외부모듈: 넘파이 (NumPy)

- 고성능 과학계산 및 데이터 분석을 위한 기본적 모듈
 - 고차원 배열의 계산에 특화되어 있음
- 외부모듈의 설치
 - 텍스트편집기 사용할 경우 명령 프롬프트창에 다음 키워드 실행

```
pip install 모듈명
```

- 주피터 노트북이나 Google Colab의 경우

```
!pip install 모듈명
```

패키지 이해하기

- 패키지 만들기
- 패키지 내부의 모듈 한꺼번에 가져오기



Python의 패키지 (Package)

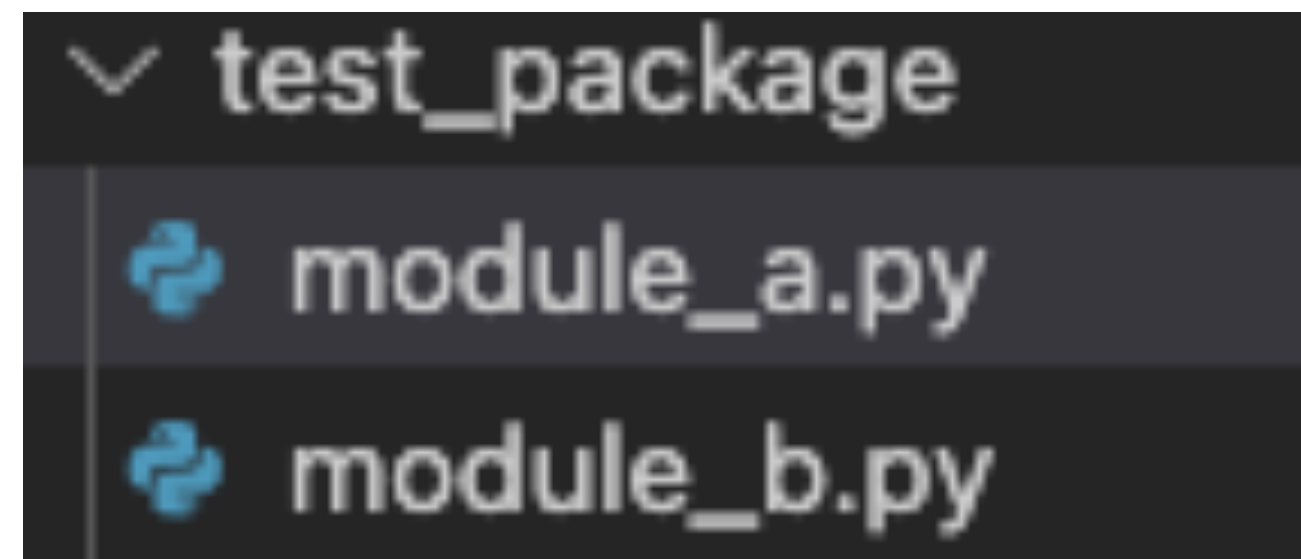
- 모듈이 디렉토리(폴더)에 정리되어 계층적으로 모여있는 것
- 패키지명.모듈명으로 관리됨

```
game/  
    __init__.py  
    sound/  
        __init__.py  
        echo.py  
        wav.py  
    graphic/  
        __init__.py  
        screen.py  
        render.py  
    play/  
        __init__.py  
        run.py  
        test.py
```



패키지 만들기 (1)

- module_a와 module_b 가지고 있는 패키지 만들기
 - test_package 폴더 생성
 - 폴더 안에 module_a와 module_b 만들기



패키지 만들기 (2)

- 모듈 채워넣기

```
%%writefile test_package/module_a.py  
variable_a = "a 모듈의 변수"
```

[25] ✓ 0.9s

... Overwriting test_package/module_a.py

```
%%writefile test_package/module_b.py  
variable_b = "b 모듈의 변수"
```

[26] ✓ 0.9s

... Overwriting test_package/module_b.py

패키지 만들기 (3)

- 패키지 내부의 모듈 읽어오기

```
import test_package.module_a as a
import test_package.module_b as b

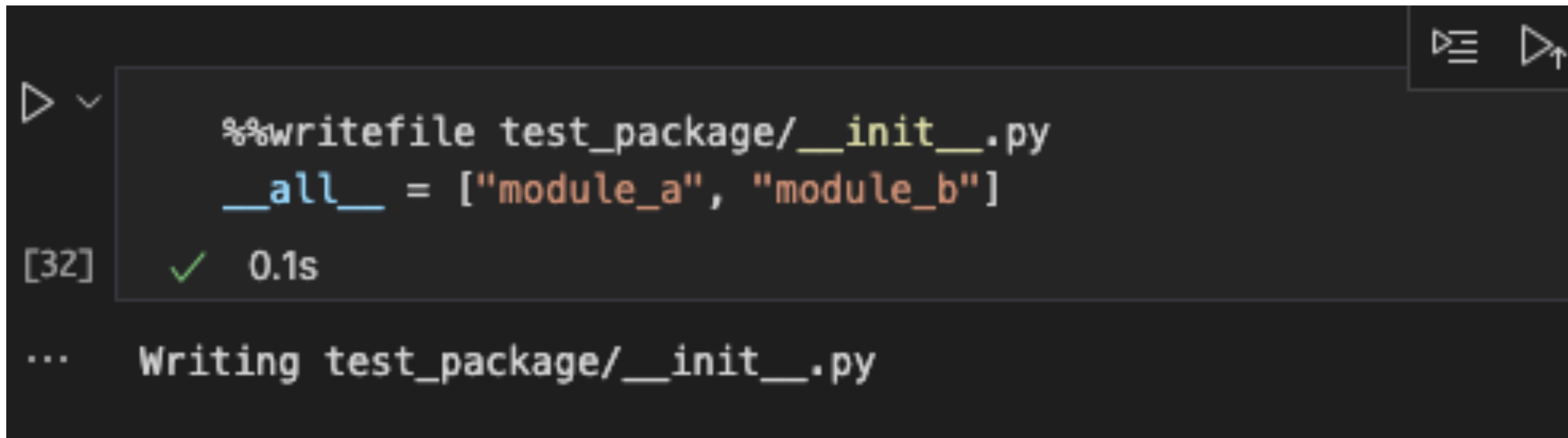
print(a.variable_a)
print(b.variable_b)
```

[30] ✓ 0.6s

... a 모듈의 변수
b 모듈의 변수

패키지 내부의 모듈 한꺼번에 가져오기 (1)

- `__init__.py`
 - `__all__` 리스트 변수를 설정
 - `from 패키지 import *` 사용할 때 포함될 모듈의 목록을 선언



A screenshot of a Jupyter Notebook interface. The top part shows a code cell with the following Python code: `%%writefile test_package/__init__.py` and `__all__ = ["module_a", "module_b"]`. Below the code cell, the output shows `[32] ✓ 0.1s`. At the bottom, a status bar indicates `... Writing test_package/__init__.py`. The interface includes standard Jupyter icons for running, saving, and navigating.

```
%%writefile test_package/__init__.py
__all__ = ["module_a", "module_b"]

[32] ✓ 0.1s

... Writing test_package/__init__.py
```

패키지 내부의 모듈 한꺼번에 가져오기 (2)

- from 패키지 import * 사용

```
from test_package import *

print(module_a.variable_a)
print(module_b.variable_b)
```

[33] ✓ 0.9s

... a 모듈의 변수
b 모듈의 변수

실습 (5/5)

- 각자 집에서 실습 수행
 - 과제 제출 여부로 출석 확인
- 실습과제는 실습시간 정시에 iClass에 업로드될 예정
- 5월 6일 자정까지 제출해야 과제 점수 인정

