



파이썬기초

5주차 - 2교시

표준 모듈과
사용자 정의 모듈





학습내용

- 표준 모듈
- 사용자 정의 모듈



학습목표

- 파이썬 표준 모듈의 종류에 대해 이해하고, 표준 모듈을 이용해서 코드를 작성할 수 있다.
- 사용자 정의 모듈에 대해 이해하고, 직접 모듈을 만들고 생성한 모듈을 이용해서 코드를 작성할 수 있다.



생각해 봅시다

자주 사용하는 코드들을 모듈로
만들 수 있는 방법에 대해 생각해보고
이미 만들어진 모듈이 있다면
어떤 것들이 있을지 생각해 봅시다.



01



표준 모듈



1 | 표준 모듈

01 정의

표준 모듈

프로그래밍을 개발하기 위해 기본적으로 사용해야 하는
문자 처리, 웹, 수학과 관련된 다양한 내장 모듈을 말함

→ 추가 설치 없이 import문 한 줄로 사용함

1 | 표준 모듈

02 sys 모듈

sys 모듈

시스템과 관련된 정보 가진 모듈로 파이썬 인터프리터가 제공하는 변수나 함수를 제어할 수 있는 방법을 제공함



1 | 표준 모듈

02 sys 모듈

In [1]: # 모듈을 불러옴.

```
import sys
```

실행 컴퓨터 환경에 관련한 정보 출력

```
print("sys.getwindowsversion()Wn",sys.getwindowsversion())
```

```
print("-----")
```

```
print("sys.copyrightWn",sys.copyright)
```

```
print("-----")
```

```
print("sys.versionWn",sys.version)
```

1 | 표준 모듈

02 sys 모듈

결과

```
sys.getwindowsversion()
sys.getwindowsversion(major=10, minor=0, build=19044, platform=2, service_pack='')
-----
sys.copyright
Copyright (c) 2001-2022 Python Software Foundation.
All Rights Reserved.

Copyright (c) 2000 BeOpen.com.
All Rights Reserved.

Copyright (c) 1995-2001 Corporation for National Research Initiatives.
All Rights Reserved.

Copyright (c) 1991-1995 Stichting Mathematisch Centrum, Amsterdam.
All Rights Reserved.
-----
sys.version
3.10.9 | packaged by Anaconda, Inc. | (main, Mar 1 2023, 18:18:15) [MSC v.1916 64
bit (AMD64)]
```


1 | 표준 모듈

03 OS 모듈

OS 모듈

운영체제에서 제공되는 **여러 기능을 실행할 수 있는 방법을 제공**하는 모듈로, 새로운 폴더를 만들거나 폴더 내부 파일 목록 등을 확인하는 기능 등을 제공함



1 | 표준 모듈

03 os 모듈

```
In [2]: import os
# 운영체제의 기본 정보 출력
print("os.name : ",os.name)
print("os.getcwd() : ",os.getcwd())
print("os.listdir()Wn",os.listdir())
# 디렉터리 생성 및 삭제
# 디렉터리 내부가 비어 있어야 삭제가능
os.mkdir("newDir")
os.rmdir("newDir")
```

1 | 표준 모듈

03 os 모듈

결과

os.name : nt

os.getcwd() : C:\Users\Wjin

os.listdir()

**['.android', '.astropy', '.bash_history', '.cache', '.conda',
'condarc', '.config', '.continuum', '.docker', '.eclipse', '.gitconfig',
'gradle', '.idlerc', '.ipynb_checkpoints', '.ipython', '.jdk',
'jupyter', -- 생략 --]**

1 | 표준 모듈

04 math 모듈

math 모듈

수학과 관련된 다양한 함수들과 상수들을 제공하는 모듈



1 | 표준 모듈

04 math 모듈

In [3]: **import** math

사인

print("math.sin(1) : ",math.sin(1))

내림

print("math.floor(3.14) : ",math.floor(3.14))

올림

print("math.ceil(3.14) : ",math.ceil(3.14))

1 | 표준 모듈

04 math 모듈

결과

`math.sin(1) : 0.8414709848078965`

`math.floor(3.14) : 3`

`math.ceil(3.14) : 4`

1 | 표준 모듈

05 random 모듈

random 모듈

난수 생성 모듈로, 정수 모듈을 생성하는 **randint()** 함수와 임의의 난수를 생성하는 **random()** 함수가 있음



1 | 표준 모듈

05 random 모듈

In [4]: **import** random

random() -> 0.0 <= r < 1.0 사이의 float 숫자 반환

print("random.random() : ",random.random())

uniform(start,end) -> start~end 범위의 float 반환

print("random.uniform(1,10) : ",random.uniform(1,10))

1 | 표준 모듈

05 random 모듈

```
In [4]: # random.randrange(end) : 0~end 범위의 int 반환
# random.randrange(start,end) -> start~end 범위의 int 반환
print("random.randrange(10) : ",random.randrange(10))
print("random.randrange(1,45) : ",random.randrange(1,45))

# random.choice(list) -> list 내부의 요소를 랜덤하게 반환
print("random.choice([10,20,30,40]) :",random.choice([10,20,30,40]))
```

1 | 표준 모듈

05 random 모듈

```
In [4]: list_temp = [10,20,30,40]
# random.shuffle(list) -> list의 요소를 랜덤에서 섞음.
random.shuffle(list_temp)
print("list_temp : ",list_temp)

# random.sample(list, k=개수) -> 리스트 중에 개수만큼 추출
print("random.sample([10,20,30,40], k=2) : ",
random.sample([10,20,30,40], k=2))
```

1 | 표준 모듈

05 random 모듈

결과

```
random.random() : 0.638717202582311  
random.uniform(1,10) : 1.3906001517400615  
random.randrange(10) : 5  
random.randrange(1,45) : 27  
random.choice([10,20,30,40]) : 30  
list_temp : [10, 20, 30, 40]  
random.sample([10,20,30,40], k=2) : [10, 20]
```

1 | 표준 모듈

06 datetime 모듈

datetime 모듈

date(날짜) 및 time(시간)과 관련된 모듈로,
날짜 형식을 만들 때 자주 사용함



1 | 표준 모듈

06 datetime 모듈

```
In [5]: import datetime
# 현재 날짜-시간 출력
now = datetime.datetime.now()
print(now.year, "년", end=" ")
print(now.month, "월", end=" ")
print(now.day, "일", end=" ")
print(now.hour, "시", end=" ")
print(now.minute, "분", end=" ")
print(now.second, "초")
```

1 | 표준 모듈

06 datetime 모듈

```
In [5]: # 시간 포매팅  
print(now.strftime("%Y. %m. %d %H:%M:%S"))
```

결과

```
2023 년 5 월 7 일 2 시 40 분 24 초  
2023. 05. 07 02:40:24
```

1 | 표준 모듈

07 urllib 모듈

urllib 모듈

웹과 관련된 모듈로, **웹 주소의 정보**를 불러옴

→ 대표적으로 urllib의 request 모듈을 사용하면
특정 URL의 정보를 불러올 수 있음

1 | 표준 모듈

07 urllib 모듈

In [5]: **import** urllib.request as req

urlopen() 함수로 웹 페이지의 코드를 읽어옴.

target = req.urlopen("https://dept.sjcu.ac.kr/computer/index.do");

output = target.read()

print(output)

1 | 표준 모듈

07 urllib 모듈

결과

```
b'WnWtWtWtWrWn<!doctype html>Wn<html
lang="ko">WnWt<head>WnWtWt<title>WxecWx84Wxb8WxecWx
a2Wx85WxecWx82WxacWxecWx9dWxb4WxebWxb2Wx84Wxeb
Wx8cWx80WxedWx95Wx99WxeaWxb5Wx90
WxecWx86Wx8cWxedWx94Wx84WxedWx8aWxb8WxecWx9bWx
a8WxecWx96Wxb4WxeaWxb3Wxb5WxedWx95Wx99WxeaWxb3
Wxbc</title>WnWtWt<meta http-equiv="Content-Type"
content="text/html; charset=utf-8" />WnWtWt<meta http-
equiv="X-UA-Compatible" content="IE=edge" />WnWtWt<meta
name="viewport" content="width=device-width,initi
-- 생략 --
```

02



사용자 정의 모듈



1 | 사용자 정의 모듈

01 정의

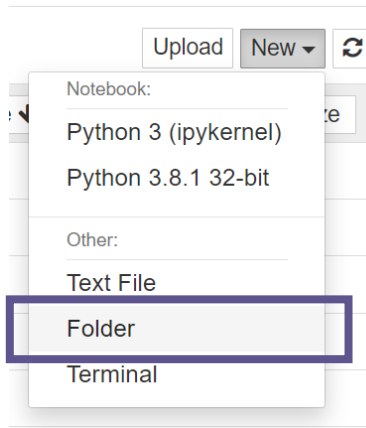
- 파이썬에서는 **.py 파일 자체가 모듈임**
- Jupyter Notebook 환경의 경우 같은 경로에 .py 파일로 코드 작성

1 | 사용자 정의 모듈

01 정의

- 폴더 생성

➤ new 버튼으로 Folder를 선택하고 폴더 생성

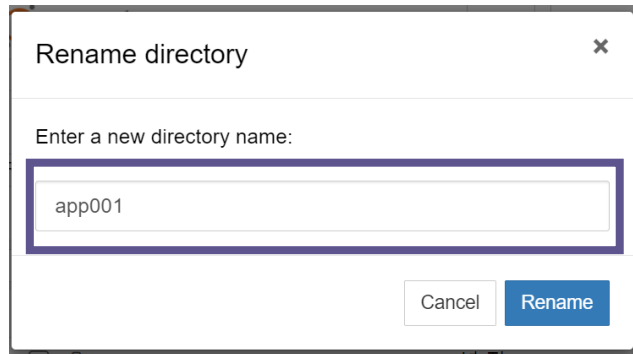
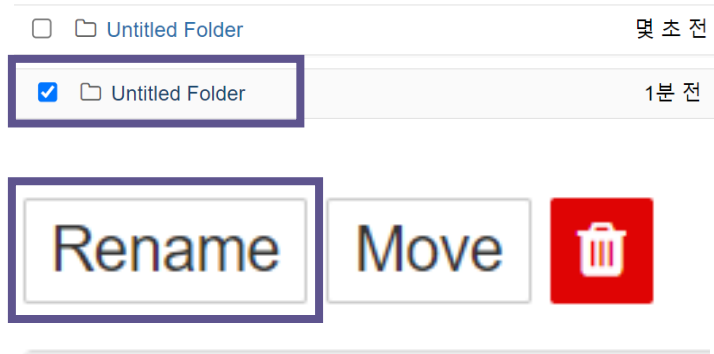


1 | 사용자 정의 모듈

01 정의

• 폴더 생성

➤ 생성된 폴더를 선택하고 좌측 상단의 Rename 버튼을 눌러 Folder 이름 변경

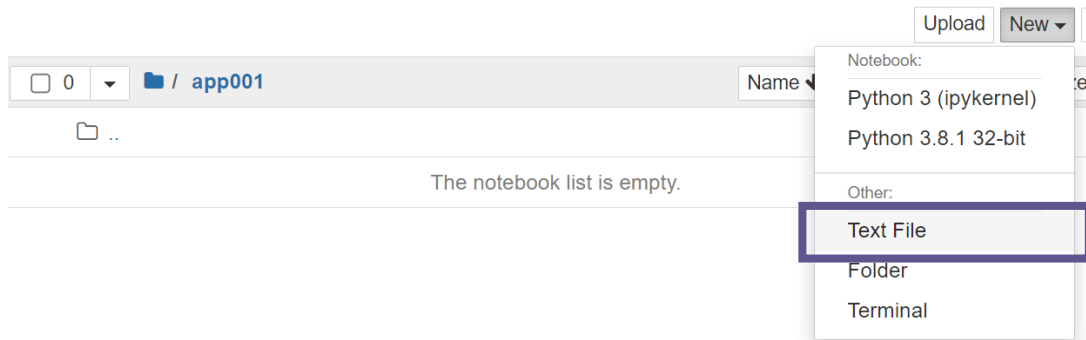


1 | 사용자 정의 모듈

01 정의

- 생성된 “app001” 폴더 안에 test.py 파일로 코드 작성

➤ Text File 생성



1 | 사용자 정의 모듈

01 정의

- 생성된 “app001” 폴더 안에 test.py 파일로 코드 작성

➤ 코드를 작성하고 저장

```
File      Edit      View      Language

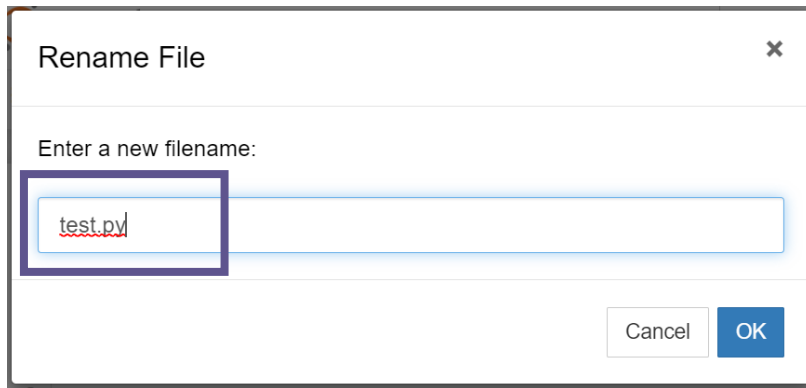
1  print('모듈 생성 테스트를 위한 테스트모듈 파일입니다.')
2
3  def test_print() :
4      print('test print .....')
5
6  def add(n1, n2) :
7      return n1+n2
8
9  pi = 3.14
```

1 | 사용자 정의 모듈

01 정의

- 생성된 “app001” 폴더 안에 test.py 파일로 코드 작성

➤ “test.py”로 이름변경



1 | 사용자 정의 모듈

01 정의

- 생성된 “app001” 폴더 안에 test.py 파일로 코드 작성

➤ “test.py”로 이름변경

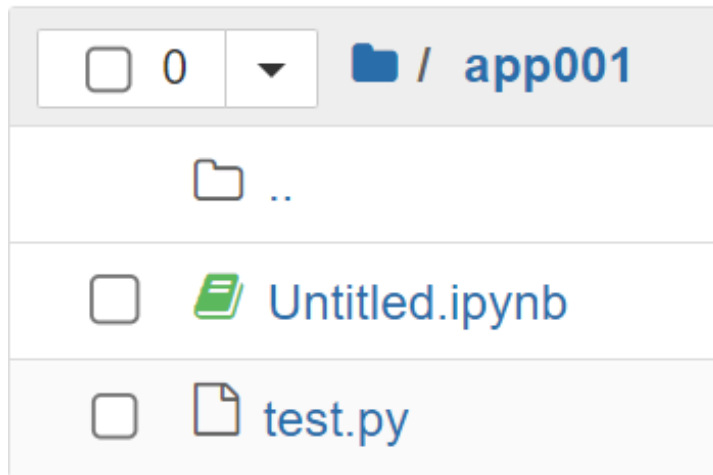
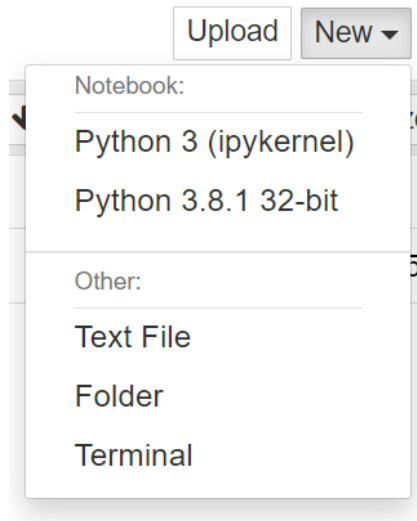


```
jupyter test.py✓ 몇 초 전
File Edit View Language
1 print('모듈 생성 테스트를 위한 테스트모듈 파일입니다.')
2
3 def test_print() :
4     print('test print .....')
5
6 def add(n1, n2) :
7     return n1+n2
8
9 pi = 3.14
```

1 | 사용자 정의 모듈

01 정의

- “app001” Folder에 새로운 notebook 파일 생성



1 | 사용자 정의 모듈

01 정의

- 코드 작성

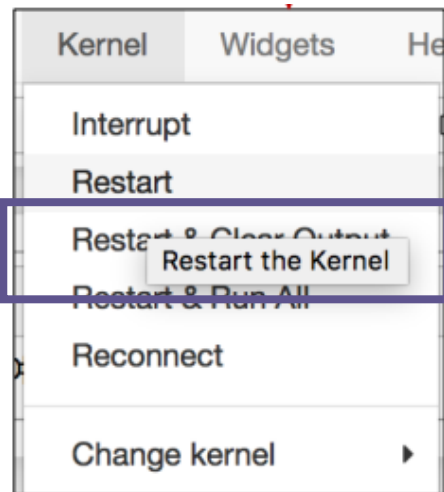
In [1]: `import test`

모듈 생성 테스트를 위한 테스트모듈 파일입니다.

1 | 사용자 정의 모듈

01 정의

- import 오류 발생 시 커널 재시작



1 | 사용자 정의 모듈

01 정의

- 모듈 멤버 사용

➤ test. 후 tap 키를 누르면 모듈 내부 멤버 확인

```
In [ ]: test.  
test.add  
In [ ]: test.pi  
test.test_print  
In [ ]: test.py
```

1 | 사용자 정의 모듈

01 정의

- 모듈 멤버 사용

```
In [2]: test.test_print()
```

```
test print .....
```

```
In [3]: print(test.add(10,20))
```

```
30
```

```
In [4]: print(test.pi)
```

```
3.14
```

1 | 사용자 정의 모듈

02 __name__ 사용

 `_name_`

모듈의 이름이 저장되는 변수, 현재 모듈이
최상위 모듈로 수행되는지 여부 확인 가능

1 | 사용자 정의 모듈

02 __name__ 사용

```
In [5]: print(_name_)
```

```
_main_
```

```
In [6]: print(test._name_)
```

```
test
```


1 | 사용자 정의 모듈

02 __name__ 사용



print(_name_)

현재 **수행**되는 파이썬 파일의 이름

→ 최상위 모듈은 __main__을 반환



print(test._name_)

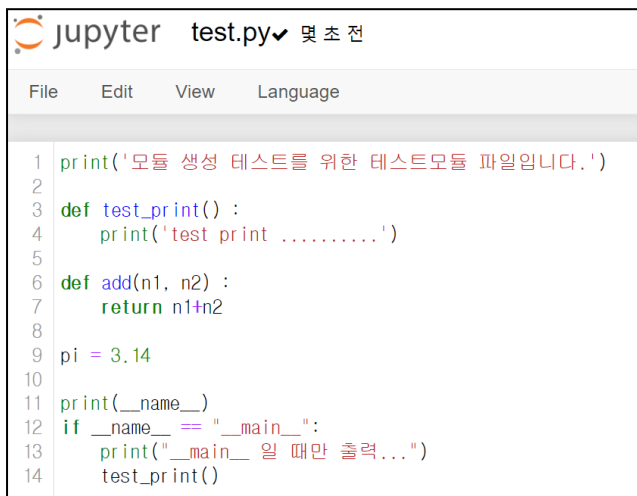
test 모듈은 현재 이 파일에선 모듈로 호출한 것

→ test라는 이름을 반환

1 | 사용자 정의 모듈

02 `__name__` 사용

- 사용자 정의 모듈에 **name이 main일 때의 조건문**을 적어 테스트 코드로 사용 가능



```
1 print('모듈 생성 테스트를 위한 테스트모듈 파일입니다.')
```

```
2
```

```
3 def test_print() :
```

```
4     print('test print .....')
```

```
5
```

```
6 def add(n1, n2) :
```

```
7     return n1+n2
```

```
8
```

```
9 pi = 3.14
```

```
10
```

```
11 print(__name__)
```

```
12 if __name__ == "__main__":
```

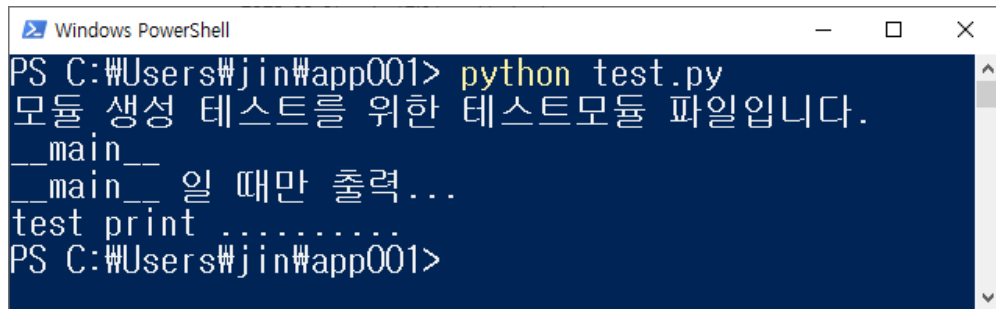
```
13     print("__main__ 일 때만 출력...")
```

```
14     test_print()
```

1 | 사용자 정의 모듈

02 `__name__` 사용

- 해당 조건문 내 코드는 **test.py가 최상위 모듈로** 사용될 때만 실행
 - 모듈로 활용될 때는 무시



```
Windows PowerShell
PS C:\Users\jin\app001> python test.py
모듈 생성 테스트를 위한 테스트모듈 파일입니다.
__main__
__main__ 일 때만 출력...
test print .....
PS C:\Users\jin\app001>
```



파이썬기초

NEXT
외부 모듈

