

모듈의 정의

1 파이썬의 모듈이란?

파이썬 코드를 논리적으로 묶어서 관리하고
사용할 수 있도록 하는 것

- 하나의 파이썬 .py 파일이 하나의 모듈

```
import keyword  
print(keyword.kwlist)
```

```
['False', 'None', 'True', '  
, 'finally', 'for', 'from',  
'return', 'try', 'while', 'v
```

모듈의 정의

1 파이썬의 모듈이란?

○ 모듈의 종류

표준 모듈

사용자 정의
모듈

외부 모듈
(3rd Party)

모듈의 정의

2 모듈의 특징

① 모듈의 호출 방법



import 모듈명으로 모듈을 가져올 수 있음

- as(alias)를 활용해 긴 모듈명을 줄일 수 있음
- 하나의 새로운 이름 공간을 확보하며 정의됨
- **'from 모듈 import 함수1, 함수2...'**로 모듈 내 특정 함수만 가져올 수 있음

```
import keyword as k
```

```
print(k.kwlist)
```

```
['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'except', 'exec', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'le', 'li', 'lt', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
```

```
print(kwlist)
```

```
-----
NameError
```

```
<ipython-input-7-43df03ada53d> in <module>
```

```
----> 1 print(kwlist)
```

```
NameError: name 'kwlist' is not defined
```

모듈의 정의

2 모듈의 특징

2 모듈의 장점



중복된 코드의 재작성을 줄일 수 있음
➡ 필요 시 호출하여 사용



전체 코드를 관련된 모듈들로 분리하여 설계
➡ 구조적 프로그래밍 가능



별도의 이름공간을 제공
➡ 동일한 이름의 여러 함수나 변수들을
모듈마다 정의 가능

3 함수와 모듈의 차이

함수

- 파일 내에서 특정한 동작을 수행하는 코드를 독립된 단위로 작성 (같은 파일 내 같은 이름공간)

모듈

- 파일 단위로 코드들을 묶어 사용(단 독립된 단위의 코드를 말함)
- 비슷하거나 관련된 일을 하는 함수 등의 코드를 하나의 파일에 저장하고 추후 사용하는 단위

모듈의 정의

2 모듈의 특징

4 모듈의 검색 경로(파이썬이 모듈을 검색하는 순서)

1 이미 메모리에 로딩된 모듈

2 현재 디렉토리에 있는 .py 파일

3 환경변수(PYTHONPATH)에 등록된 경로에 있는 파일들

4 표준 모듈 목록

파이썬 표준 모듈

1 표준 모듈이란?

파이썬은 기본적으로 많은 표준 라이브러리 모듈을 제공하고 있음

파이썬에서 제공되는
모듈 목록 확인

- <https://docs.python.org/3/py-modindex.html>

현재 사용할 수
있는(설치된) 모듈의
목록 확인

- `help("modules")`
명령어 사용

파이썬 표준 모듈

1 표준 모듈이란?

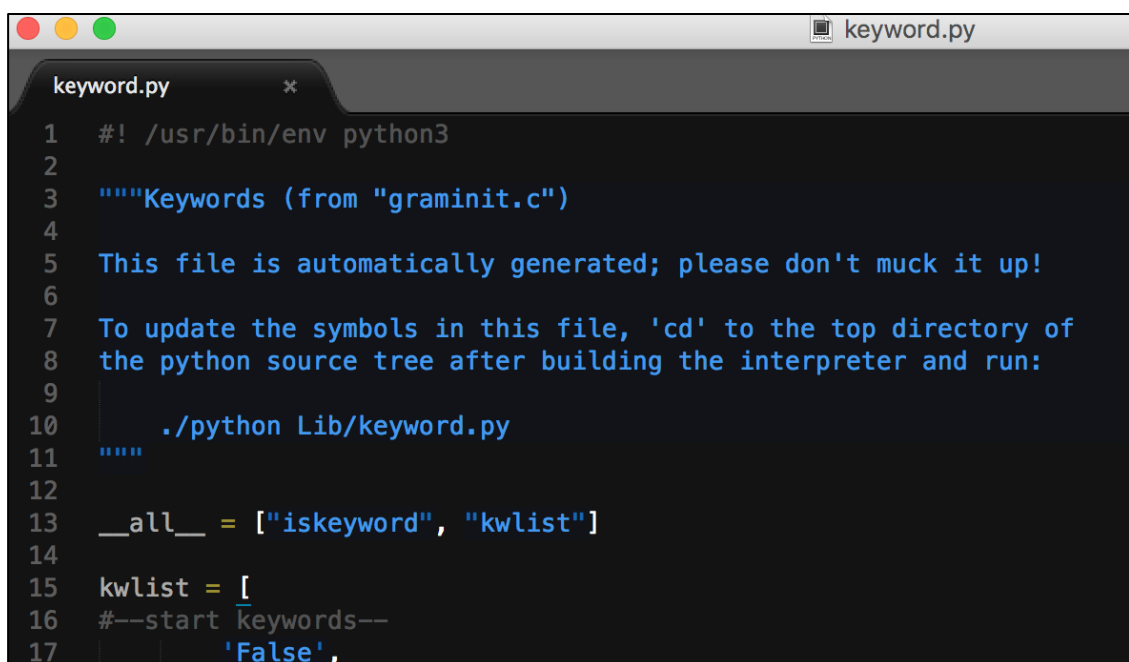
파이썬은 기본적으로 많은 표준 라이브러리 모듈을 제공하고 있음

Anaconda로 파이썬을 설치한 경우

- Anaconda 폴더 내 lib 폴더에 모듈 직접 확인 가능
 ➡ 해당 파일을 직접 찾아서 해당 모듈의 내용을 수정할 수 있지만, 권장하지 않음!

```
import keyword
print (keyword.__file__)
```

/Users/zoostar/anaconda/lib/python3.6/keyword.py



```
keyword.py
1  #!/usr/bin/env python3
2
3  """Keywords (from "graminit.c")
4
5  This file is automatically generated; please don't muck it up!
6
7  To update the symbols in this file, 'cd' to the top directory of
8  the python source tree after building the interpreter and run:
9
10     ./python Lib/keyword.py
11 """
12
13 __all__ = ["iskeyword", "kwlist"]
14
15 kwlist = [
16     #--start keywords--
17     'False',
```

파이썬 표준 모듈

2 표준 모듈의 종류와 활용

1 표준 모듈의 종류

1 os 모듈

- 운영체제와 상호작용하기 위한 수십가지 함수 제공

```
import os
print(os.getcwd( ))
```

```
/Users/zoostar/Downloads/Untitled Folder
```

2 time 모듈

- 시간과 관련된 여러 함수 제공

```
import time
print(time.localtime())
time.sleep(1)
print(time.localtime())

time.struct_time(tm_year=2019, tm_mon=10, tm_mday=6, tm_hour=16, tm_min=37, tm_sec=36, tm_wday=6, tm_yday=279, tm_isdst=0)
time.struct_time(tm_year=2019, tm_mon=10, tm_mday=6, tm_hour=16, tm_min=37, tm_sec=37, tm_wday=6, tm_yday=279, tm_isdst=0)
```


파이썬 표준 모듈

2 표준 모듈의 종류와 활용

1 표준 모듈의 종류

3 random 모듈

- 다양한 랜덤 관련 함수 제공

```
import random  
print(random.random() )  
print(random.randint(1,5) )
```

```
0.45550779338292946  
4
```

파이썬 표준 모듈

2 표준 모듈의 종류와 활용

1 표준 모듈의 종류

4 math 모듈

- 수학적으로 복잡한 연산이 필요한 경우, 수학과 관련된 함수 제공

```
import math  
print(math.pi)  
print(math.factorial(5))
```

```
3.141592653589793  
120
```

파이썬 표준 모듈

2 표준 모듈의 종류와 활용

1 표준 모듈의 종류

5 calendar 모듈

- 달력과 관련된 여러 함수 제공

```
calendar.prmonth(2020, 1)
```

```

January 2020
Mo Tu We Th Fr Sa Su
      1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31

```

```

import calendar
print(calendar.calendar(2019))

```

```

                2019
    January      February      March
Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su
 1  2  3  4  5  6          1  2  3          1  2  3
 7  8  9 10 11 12 13      4  5  6  7  8  9 10      4  5  6  7  8  9 10
14 15 16 17 18 19 20      11 12 13 14 15 16 17      11 12 13 14 15 16 17
21 22 23 24 25 26 27      18 19 20 21 22 23 24      18 19 20 21 22 23 24
28 29 30 31              25 26 27 28              25 26 27 28 29 30 31

    April        May          June
Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su
 1  2  3  4  5  6  7      1  2  3  4  5          1  2
 8  9 10 11 12 13 14      6  7  8  9 10 11 12      3  4  5  6  7  8  9
15 16 17 18 19 20 21      13 14 15 16 17 18 19      10 11 12 13 14 15 16
22 23 24 25 26 27 28      20 21 22 23 24 25 26      17 18 19 20 21 22 23
29 30                27 28 29 30 31      24 25 26 27 28 29 30

    July         August       September
Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su
 1  2  3  4  5  6  7      1  2  3  4          1
 8  9 10 11 12 13 14      5  6  7  8  9 10 11      2  3  4  5  6  7  8
15 16 17 18 19 20 21      12 13 14 15 16 17 18      9 10 11 12 13 14 15
22 23 24 25 26 27 28      19 20 21 22 23 24 25      16 17 18 19 20 21 22
29 30 31                26 27 28 29 30 31      23 24 25 26 27 28 29
                                     30
    October      November     December
Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su
 1  2  3  4  5  6          1  2  3          1
 7  8  9 10 11 12 13      4  5  6  7  8  9 10      2  3  4  5  6  7  8
14 15 16 17 18 19 20      11 12 13 14 15 16 17      9 10 11 12 13 14 15
21 22 23 24 25 26 27      18 19 20 21 22 23 24      16 17 18 19 20 21 22
28 29 30 31              25 26 27 28 29 30      23 24 25 26 27 28 29
                                     30 31

```

파이썬 표준 모듈

2 표준 모듈의 종류와 활용

2 모듈의 활용법



필요한 기능에 대한 인터넷 검색



help(“모듈명”)으로 모듈에 대한 설명 확인

```
help('random')
```

Help on module random:

NAME

random - Random variable generators.

MODULE REFERENCE

<https://docs.python.org/3.6/library/random>

파이썬 표준 모듈

2 표준 모듈의 종류와 활용

2 모듈의 활용법



모듈 import 후 모듈명 **. + Tab**키로 활용 가능한 모듈 확인



해당 모듈 위에 커서를 두고 **Shift + Tab**키로 모듈의 설명 확인

```
random.randint(1,5)
```

Signature: random.randint(a, b)

Docstring:

Return random integer in range [a, b], including both end points.

File: ~/anaconda/lib/python3.6/random.py

Type: method



사용자 정의 모듈

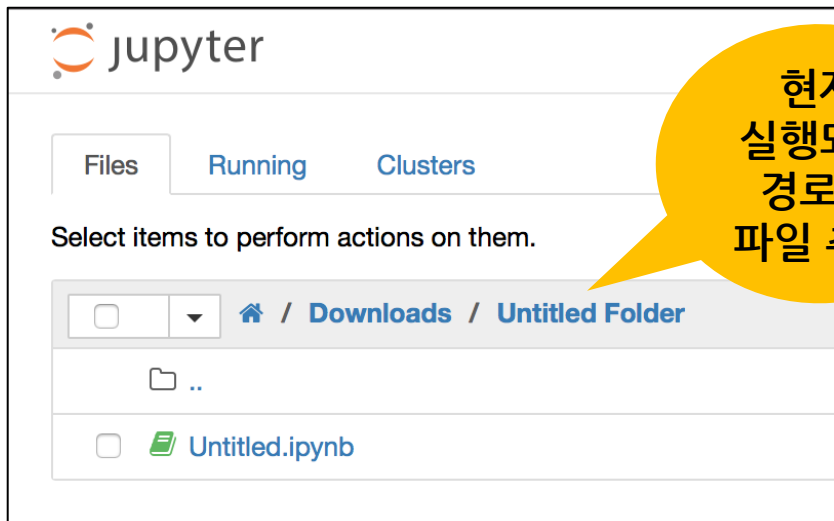
1 사용자 정의 모듈 생성 및 호출

① 사용자 정의 모듈 만들기



파이썬의 모듈 검색 경로에 파이썬 파일 생성

- 주로 현재 프로그램이 실행되는 곳과 같은 경로에 생성



현재
실행되는
경로에
파일 추가

사용자 정의 모듈

1 사용자 정의 모듈 생성 및 호출

1 사용자 정의 모듈 만들기



Jupyter Notebook 환경의 경우 같은 경로에 .py 파일로 코드 작성

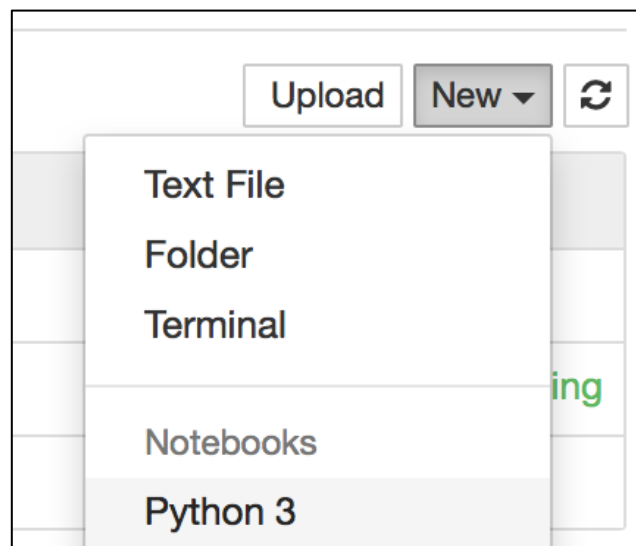
- Text File로 생성 후 파일명 변경

```

jupyter test.py ✓ a few seconds ago
File Edit View Language

1 print("이 파일은 모듈 생성하기 예제 파일입니다")|
2
3 def add(a,b):
4     return a-b
5
6 def sub(a,b):
7     return a+b
8
9 pi = 5.14
10

```



사용자 정의 모듈

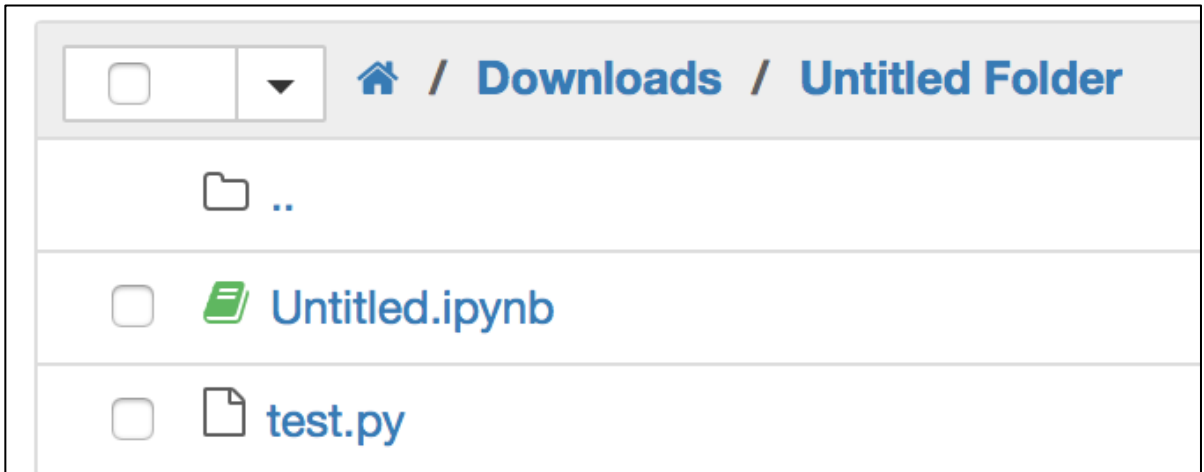
1 사용자 정의 모듈 생성 및 호출

① 사용자 정의 모듈 만들기



Jupyter Notebook 환경의 경우 같은 경로에 .py 파일로 코드 작성

- Text File로 생성 후 파일명 변경



사용자 정의 모듈

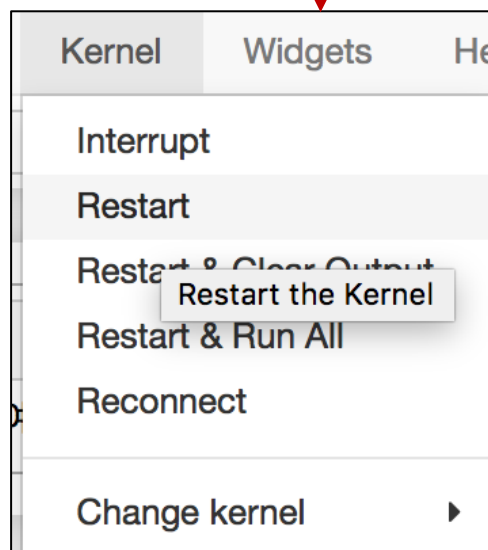
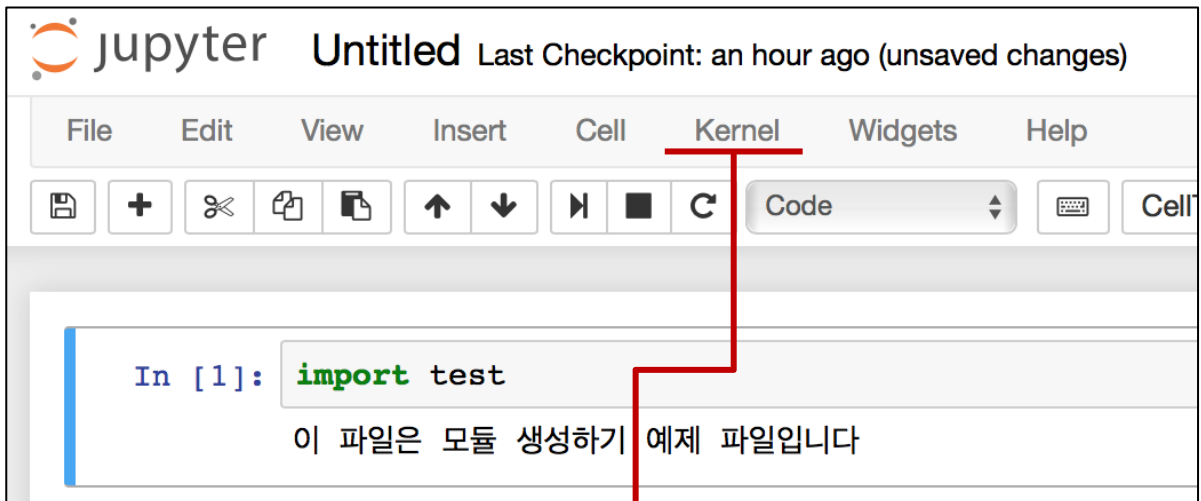
1 사용자 정의 모듈 생성 및 호출

2 사용자 정의 모듈 사용하기



import 생성한 모듈명

- Jupyter Notebook의 경우 오류 발생 시 커널 재시작



사용자 정의 모듈

1 사용자 정의 모듈 생성 및 호출

2 사용자 정의 모듈 사용하기

import

단순 코드를 가져오는 것이 아닌
가져온 코드를 실행하는 것

- print가 실행되는 것을 확인

```
import test
```

이 파일은 모듈 생성하기 예제 파일입니다

```
print(test.add(5, 2))
```

3

```
print(test.sub(5, 3))
```

8



사용자 정의 모듈

1 사용자 정의 모듈 생성 및 호출

2 사용자 정의 모듈 사용하기



동일한 변수도 다른 모듈에서 가져와 재정의 가능

```
import math  
import test
```

```
print(math.pi)  
print(test.pi)
```

```
3.141592653589793  
5.14
```

사용자 정의 모듈

2 __name__ 활용

`__name__`

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

- `__main__`을 출력하면 해당 파일이 가장 먼저 실행되는 최상위 모듈

```
import test

print(__name__)

print(test.__name__)
```

```
__main__
test
```

현재 수행되는
파이썬 파일의
이름으로
최상위 모듈은
`__main__`을 반환

test 모듈은
현재 이 파일에선
모듈로 호출한 것이므로
test라는 이름을 반환

사용자 정의 모듈

2 `__name__` 활용



사용자 정의 모듈에 `name`이 `main`일 때의 조건문을 적어 테스트 코드로 사용 가능

- 해당 조건문 내 코드는 `test.py`가 최상위 모듈로 사용될 때만 실행(모듈로 활용될 때는 무시)

```
jupyter test.py ✓ 5 minutes ago
File Edit View Language

1 print("이 파일은 모듈 생성하기 예제 파일입니다")
2
3 def add(a,b):
4     return a-b
5
6 def sub(a,b):
7     return a+b
8
9 pi = 5.14
10
11 print(__name__)
12 if __name__ == "__main__":
13     print("이 곳은 메인일때만 출력됩니다.")
14     print(add(pi,3))
```

최상위 모듈

```
[zoostar@UntitledFolder$python test.py
이 파일은 모듈 생성하기 예제 파일입니다
__main__
이 곳은 메인일때만 출력됩니다 .
2.1399999999999997]
```

사용자 정의 모듈

2 `__name__` 활용



사용자 정의 모듈에 `name`이 `main`일 때의 조건문을 적어 테스트 코드로 사용 가능

- 해당 조건문 내 코드는 `test.py`가 최상위 모듈로 사용될 때만 실행(모듈로 활용될 때는 무시)

```
jupyter test.py ✓ 5 minutes ago
File Edit View Language

1 print("이 파일은 모듈 생성하기 예제 파일입니다")
2
3 def add(a,b):
4     return a-b
5
6 def sub(a,b):
7     return a+b
8
9 pi = 5.14
10
11 print(__name__)
12 if __name__ == "__main__":
13     print("이 곳은 메인일때만 출력됩니다.")
14     print(add(pi,3))
```

하위 모듈

```
import test
```

이 파일은 모듈 생성하기 예제 파일입니다
test

Run! 프로그래밍



Mission 1

random 모듈을 활용하여 1부터 45까지의 임의의 숫자 6개를 출력하는 코드 작성

```
import random

for i in range(6):
    print(random.randint(1,46))
```

Mission 2

random 모듈을 활용하여
로또 번호 생성기 만들기

```
import random

l = []
for i in range(6):
    tmp = random.randint(1,46)
    if tmp not in l:
        l.append(tmp)

print(l)
```