



표준 모듈

- 모듈 (module)
 - 코드를 분리하고 공유하는 기능
 - 표준 모듈
 - 파이썬에 기본적으로 내장된 모듈
 - 외부 모듈
 - 사람들이 만들어 공개한 모듈

```
import 모듈 이름
```

모듈 사용의 기본 : math 모듈

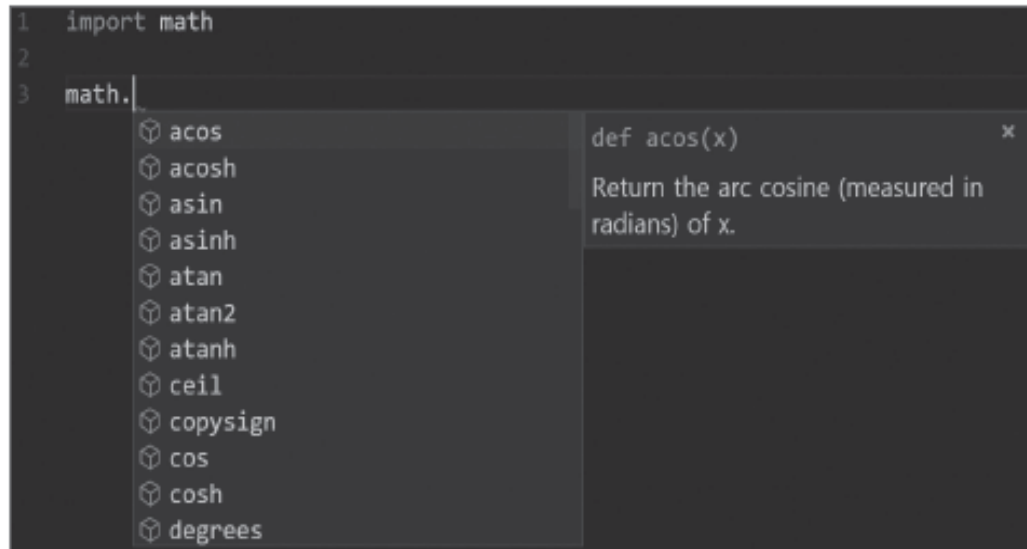
- math 모듈

- 수학과 관련된 기능

```
import math
```

- 여러 변수와 함수를 가진 패키지

자동 완성 기능으로 살펴보는 math 모듈의 변수와 함수



모듈 사용의 기본 : math 모듈

- math 모듈을 사용하는 코드

```
>>> import math
```

- 수학/삼각함수

```
>>> math.sin(1)      # 사인
0.8414709848078965
>>> math.cos(1)      # 코사인
0.5403023058681398
>>> math.tan(1)      # 탄젠트
1.5574077246549023
>>>
>>> math.floor(2.5)   # 내림
2
>>> math.ceil(2.5)    # 올림
3
```

모듈 사용의 기본 : math 모듈

- from 구문

- 다양한 함수를 계속해서 입력하는 것의 비효율성

```
from 모듈 이름 import 가져오고 싶은 변수 또는 함수
```

- ‘가져오고 싶은 변수 또는 함수’에 여러 개의 변수 또는 함수 입력 가능
- 이를 통해 가져온 기능은 math 붙이지 않고도 사용할 수 있음

```
>>> from math import sin, cos, tan, floor, ceil
>>> sin(1)
0.8414709848078965
>>> cos(1)
0.5403023058681398
>>> tan(1)
1.5574077246549023
>>> floor(2.5)
2
>>> ceil(2.5)
3
```

모듈 사용의 기본 : math 모듈

- as 구문

- 모듈의 이름이 너무 길어 짧게 줄여 사용하고 싶은 경우

```
import 모듈 as 사용하고 싶은 식별자
```

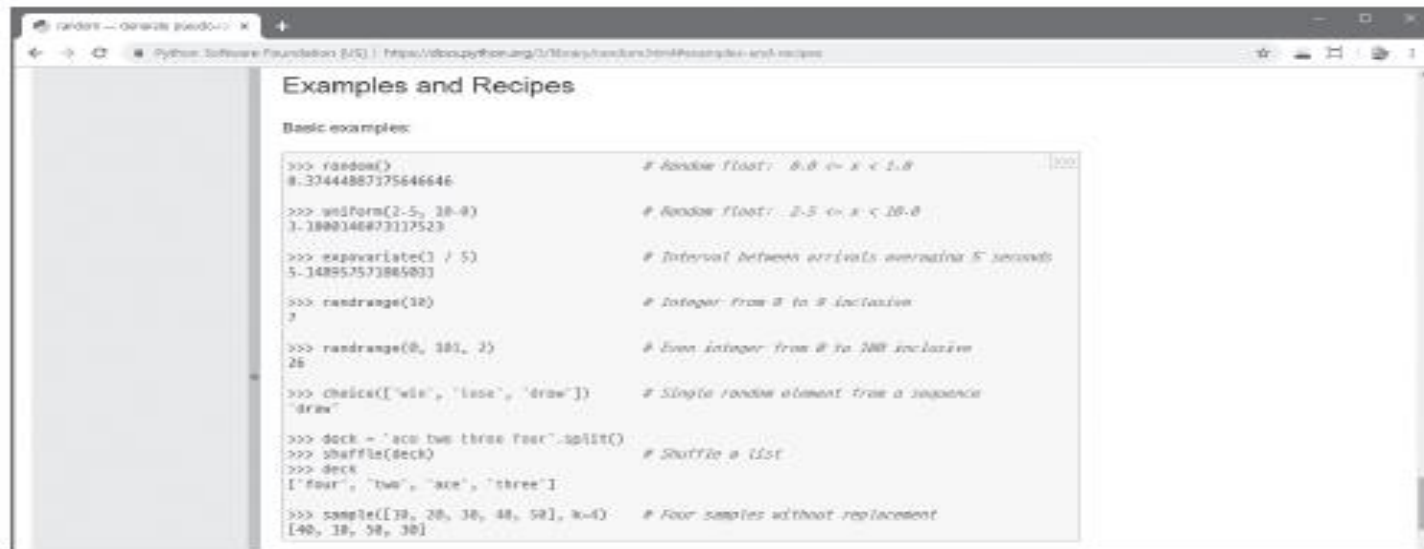
```
>>> import math as m
>>> m.sin(1)
0.8414709848078965
>>> m.cos(1)
0.5403023058681398
>>> m.tan(1)
1.5574077246549023
>>> m.floor(2.5)
2
>>> m.ceil(2.5)
3
```

random 모듈

- random 모듈

```
import random
```

- random 모듈 문서
 - <http://docs.python.org/3/library/random.html#examples-and-recipes>



random 모듈

- 예시

```
01 import random
02 print("# random 모듈")
03
04 # random(): 0.0 <= x < 1.0 사이의 float를 리턴합니다.
05 print("- random():", random.random())
06
07 # uniform(min, max): 지정된 범위 사이의 float를 리턴합니다.
08 print("- uniform(10, 20):", random.uniform(10, 20))
09
10 # randrange(): 지정된 범위의 int를 리턴합니다.
11 # - randrange(max): 0부터 max 사이의 값을 리턴합니다.
12 # - randrange(min, max): min부터 max 사이의 값을 리턴합니다.
13 print("- randrange(10)", random.randrange(10))
```


random 모듈

```
14
15 # choice(list): 리스트 내부에 있는 요소를 랜덤하게 선택합니다.
16 print("- choice([1, 2, 3, 4, 5]):", random.choice([1, 2, 3, 4, 5]))
17
18 # shuffle(list): 리스트의 요소들을 랜덤하게 섞습니다.
19 print("- shuffle([1, 2, 3, 4, 5]):", random.shuffle([1, 2, 3, 4, 5]))
20
21 # sample(list, k=<숫자>): 리스트의 요소 중에 k개를 뽑습니다.
22 print("- sample([1, 2, 3, 4, 5], k=2):", random.sample([1, 2, 3, 4, 5], k=2))
```

실행결과

```
# random 모듈
- random(): 0.5671614057098718
- uniform(10, 20): 18.627114055572356
- randrange(10) 6
- choice([1, 2, 3, 4, 5]): 2
- shuffle([1, 2, 3, 4, 5]): None
- sample([1, 2, 3, 4, 5], k=2): [5, 4]
```

random 모듈

- 5행의 random.random()처럼 random을 계속 입력하는 것은 효율적이지 못하므로 from 구문 활용해서 임포트

```
from time import random, randrange, choice
```

- sys 모듈

- 시스템과 관련된 정보 가진 모듈
- 명령 매개변수 받을 때 많이 사용

```
01  # 모듈을 읽어 들입니다.  
02  import sys  
03  
04  # 명령 매개변수를 출력합니다.  
05  print(sys.argv)
```

```
06  print("----")
07
08  # 컴퓨터 환경과 관련된 정보를 출력합니다.
09  print("getwindowsversion()", sys.getwindowsversion())
10  print("----")
11  print("copyright:", sys.copyright)
12  print("----")
13  print("version:", sys.version)
14
15  # 프로그램을 강제로 종료합니다.
16  sys.exit()
```

-
- 5행 sys.argv
 - 아래와 같이 실행하면 ['module_sys.py', '10', '20', '30'] 리스트 들어옴

```
> python module_sys.py 10 20 30
```

`['module_sys.py', '10', '20', '30']` → 명령 매개변수입니다. 입력한 명령어에 따라 달라집니다.

```
getwindowsversion:() sys.getwindowsversion(major=10, minor=0, build=14393,
platform=2, service_pack='')

```

```
copyright: Copyright (c) 2001-2019 Python Software Foundation.
All Rights Reserved.

```

...생략...

```
version: 3.7.3 (v3.7.3:ef4ecbed12, Mar 21 2019, 17:54:52) [MSC v.1916 32
bit (Intel)]

```

- os 모듈

- 운영체제와 관련된 기능 가진 모듈
- 새로운 폴더 만들거나 폴더 내부 파일 목록 보는 등

```
01  # 모듈을 읽어 들입니다.
02  import os
03
04  # 기본 정보를 몇 개 출력해봅시다.
05  print("현재 운영체제:", os.name)
06  print("현재 폴더:", os.getcwd())
07  print("현재 폴더 내부의 요소:", os.listdir())
08
09  # 폴더를 만들고 제거합니다[폴더가 비어있을 때만 제거 가능].
10  os.mkdir("hello")
11  os.rmdir("hello")
12
```

```
13  # 파일을 생성하고 + 파일 이름을 변경합니다.
14  with open("original.txt", "w") as file:
15      file.write("hello")
16      os.rename("original.txt", "new.txt")
17
18  # 파일을 제거합니다.
19  os.remove("new.txt")
20  # os.unlink("new.txt")
21
22  # 시스템 명령어 실행
23  os.system("dir")
```

현재 운영체제: nt

현재 폴더: C:\Users\hasat\sample

현재 폴더 내부의 요소: ['.vscode', 'beaut.py', 'download-png1.py', 'file.txt', 'freq.json', 'ghostdriver.log', 'iris.csv', 'lang-plot.png', 'mnist', 'mtest.py', 'newFile.xlsx', 'output.png', 'proj', 'rint.py', 'stats_104102.xlsx', 'test', 'test.csv', 'test.html', 'test.png', 'test.py', 'test.rb', 'test.txt', 'test_a.txt', 'train', 'underscore.js', 'Website.png', 'Website_B.png', 'Website_C.png', 'Website_D.png', '__pycache__']

C 드라이브의 볼륨: BOOTCAMP

볼륨 일련 번호: FCCF-6067

C:\Users\hasat\sample 디렉터리

2019-05-01	오전 12:18	<DIR>	.
2019-05-01	오전 12:18	<DIR>	..
...생략...			
2019-05-28	오전 04:49	<DIR>	__pycache__
		24개 파일	1,908,017 바이트
		8개 디렉터리	16,895,188,992 바이트 남음

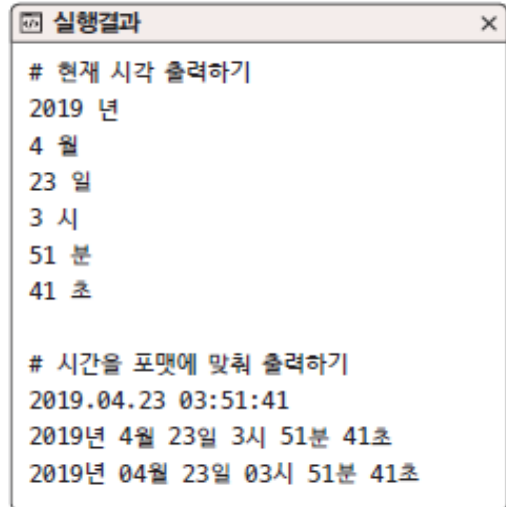
→ 명령 프롬프트에서
그냥 dir을 입력했을 때의
결과와 동일합니다.
단지 파이썬에서
dir 명령어를
호출했을 뿐입니다.

datetime 모듈

- datetime 모듈

- date(날짜) 및 time(시간)과 관련된 모듈로, 날짜 형식 만들 때 자주 사용되는 코드로 구성

```
01  # 모듈을 읽어 들입니다.  
02  import datetime  
03  
04  # 현재 시각을 구하고 출력하기  
05  print("# 현재 시각 출력하기")  
06  now = datetime.datetime.now()  
07  print(now.year, "년")  
08  print(now.month, "월")  
09  print(now.day, "일")  
10  print(now.hour, "시")  
11  print(now.minute, "분")  
12  print(now.second, "초")  
13  print()  
14
```



```
# 현재 시각 출력하기  
2019 년  
4 월  
23 일  
3 시  
51 분  
41 초  
  
# 시간을 포맷에 맞춰 출력하기  
2019.04.23 03:51:41  
2019년 4월 23일 3시 51분 41초  
2019년 04월 23일 03시 51분 41초
```

datetime 모듈

```
15  # 시간 출력 방법
16  print("# 시간을 포맷에 맞춰 출력하기")
17  output_a = now.strftime("%Y.%m.%d %H:%M:%S")
18  output_b = "{}년 {}월 {}일 {}시 {}분 {}초".format(now.year,\
19      now.month,\
20      now.day,\
21      now.hour,\
22      now.minute,\
23      now.second)
24  output_c = now.strftime("%Y{} %m{} %d{} %H{} %M{} %S{}").format(*"년월일시분초")
25  print(output_a)
26  print(output_b)
27  print(output_c)
28  print()
```

문자열, 리스트 등 앞에 *을
요소 하나하나가 매개변수로

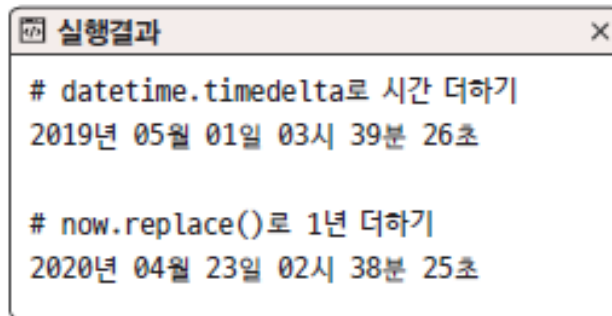
datetime 모듈

- output_a처럼 strftime() 함수 사용하면 시간을 형식에 맞춰 출력 가능
- 그 외 다양한 시간 처리 기능

```
01  # 모듈을 읽어 들입니다.
02  import datetime
03  now = datetime.datetime.now()
04
05  # 특정 시간 이후의 시간 구하기
06  print("# datetime.timedelta로 시간 더하기")
07  after = now + datetime.timedelta(\
08      weeks=1,\
09      days=1,\
10      hours=1,\
11      minutes=1,\
12      seconds=1)
```

datetime 모듈

```
13 print(after.strftime("%Y{} %m{} %d{} %H{} %M{} %S{}").format(*"년월일시분초"))
14 print()
15
16 # 특정 시간 요소 교체하기
17 print("# now.replace()로 1년 더하기")
18 output = now.replace(year=(now.year + 1))
19 print(output.strftime("%Y{} %m{} %d{} %H{} %M{} %S{}").format(*"년월일시분초"))
```



실행결과

```
# datetime.timedelta로 시간 더하기
2019년 05월 01일 03시 39분 26초

# now.replace()로 1년 더하기
2020년 04월 23일 02시 38분 25초
```

- timedelta() 함수 사용하면 특정한 시간의 이전 또는 이후 구함
 - “1년 후” 구할 때는 replace() 함수 사용해 날짜 값을 교체

time 모듈

- time 모듈

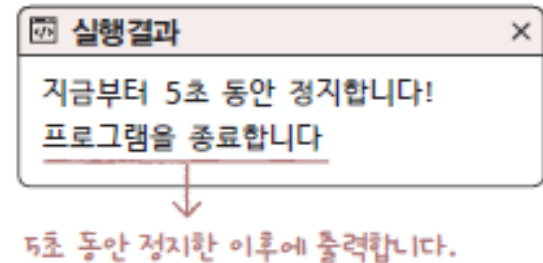
- 시간과 관련된 기능

```
import time
```

- `time.sleep()` □

- 특정 시간 동안 코드 진행을 정지
- 정지하고 싶을 시간을 초 단위로 입력

```
01 import time
02
03 print("지금부터 5초 동안 정지합니다!")
04 time.sleep(5)
05 print("프로그램을 종료합니다")
```



urllib 모듈

- urllib 모듈

- URL 다루는 라이브러리

```
01  # 모듈을 읽어 들입니다.
02  from urllib import request
03
04  # urlopen() 함수로 구글의 메인 페이지를 읽습니다.
05  target = request.urlopen("https://google.com")
06  output = target.read()
07
08  # 출력합니다.
09  print(output)
```

- urlopen() 함수 : URL 주소의 페이지 열기

```
b'<!doctype html><html itemscope="" itemtype="http://Schema.org/WebPage"
lang="ko"><head><meta content="text/html; charset=UTF-8" http-equiv="Content-
Type"><meta content="/logos/doodles/2019/amy-johnsons-114th-birthday-
5154304993263616.2-law.gif" itemprop="image">
...생략...
```

외부 모듈

목차

- 모듈 설치하기
- 모듈 찾아보기
- BeautifulSoup 모듈
- Flask 모듈
- 라이브러리와 프레임워크

모듈 설치하기

- 외부 모듈 설치

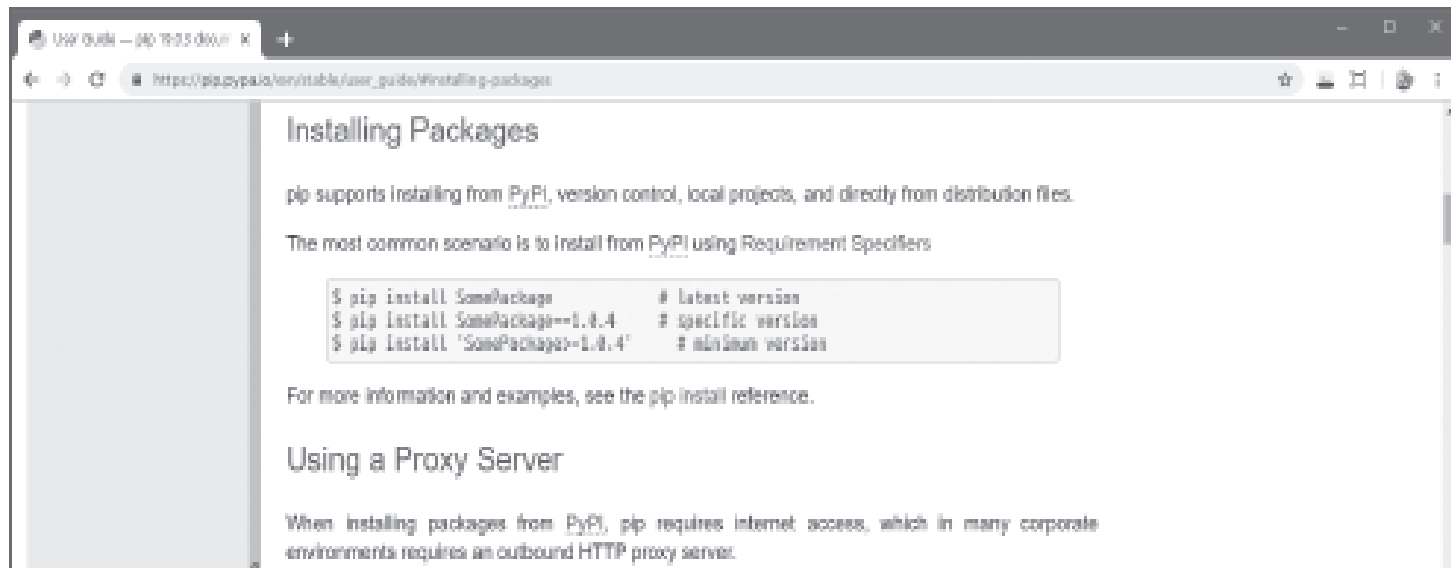
- [Windows] + [R] 클릭하여 프로그램 실행창 띄우고 [cmd] 입력하면 명령 프롬프트 창 나타남

```
pip install 모듈 이름
```

```
> pip install beautifulsoup4
Collecting beautifulsoup4
  Downloading beautifulsoup4-4.6.0-py3-none-any.whl (86kB)
    100% |#####~####| 92kB 422kB/s
Installing collected packages: beautifulsoup4
Successfully installed beautifulsoup4-4.6.0
```

모듈 설치하기

- pip
 - 특정 버전 모듈 설치 및 제거 등
 - http://pip.pypa.io/en/stable/user_guide/#installing-packages



- 주피터 노트북(jupyter notebook)에서 라이브러리 설치하기: **!pip install**

모듈 찾아보기

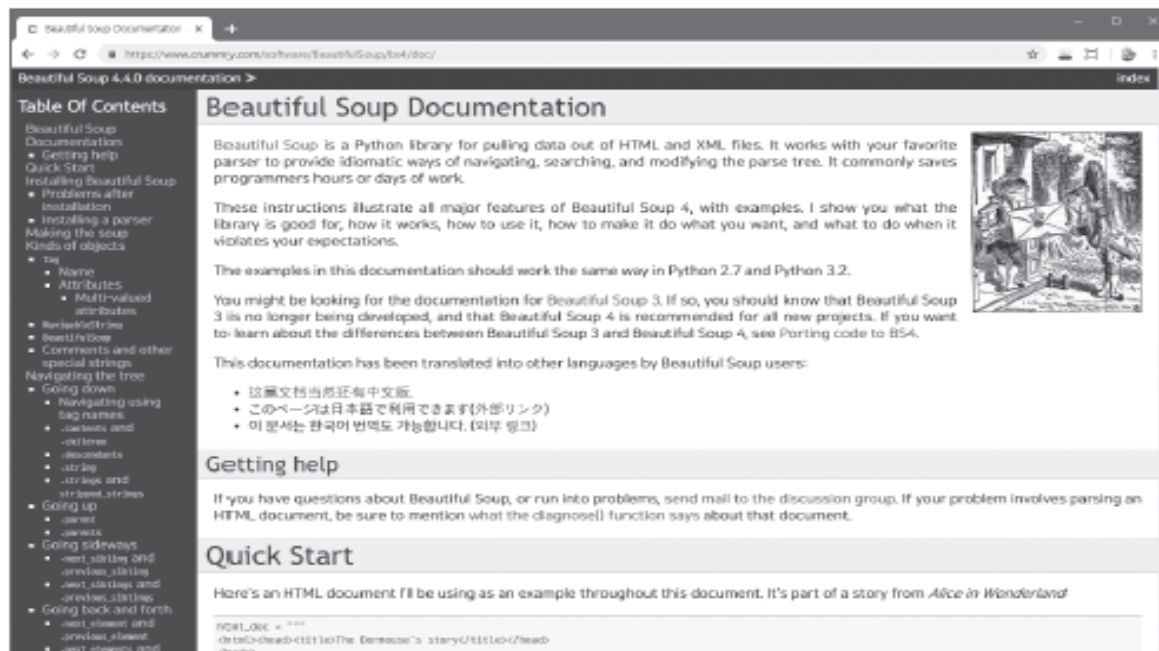
- 파이썬 관련 도서 구매할 경우 추천 받는 모듈
- 인터넷 커뮤니티 정보로 접하는 모듈
- 직접 구글에서 검색하여 찾는 모듈
 - Python 키워드 옆에 내가 원하는 것을 더하여 검색

BeautifulSoup 모듈

- Beautiful Soup 모듈

- 웹 페이지 분석 모듈

- <http://www.crummy.com/software/BeautifulSoup/bs4/doc/>



BeautifulSoup 모듈

```
01  # 모듈을 읽어 들입니다.
02  from urllib import request
03  from bs4 import BeautifulSoup
04
05  # urlopen() 함수로 기상청의 전국 날씨를 읽습니다.
06  target = request.urlopen("http://www.kma.go.kr/weather/forecast/mid-term-
    rss3.jsp?stnId=108") → 이 코드는 한 줄 코드이니 이어서 입력해야 합니다.
07
08  # BeautifulSoup을 사용해 웹 페이지를 분석합니다.
09  soup = BeautifulSoup(target, "html.parser")
10
11  # location 태그를 찾습니다.
12  for location in soup.select("location"):
13      # 내부의 city, wf, tmn, tmx 태그를 찾아 출력합니다.
14      print("도시:", location.select_one("city").string)
15      print("날씨:", location.select_one("wf").string)
16      print("최저기온:", location.select_one("tmn").string)
17      print("최고기온:", location.select_one("tmx").string)
18      print()
```

```
<rss version="2.0">
  <channel>
    <title>기상청 육상 중기예보</title>
    <!-- 생략 -->
    <item>
      <author>기상청</author>
      <!-- 생략 -->
      <description>
        <header>
          <title>전국 육상중기예보</title>
          <tm>201904221800</tm>
          <wf><![CDATA[기압골의 영향으로 25일 오후부터 26일 오전 사이 제주도를 제외한 전국에,
29일에는 충청도와 남부지방, 제주도에 비가 오겠습니다.<br />그 밖의 날은 고기압의 가장자리에 들어 가
끔 구름많겠습니다.<br />기온은 평년(최저기온: 4~13℃, 최고기온: 18~24℃)보다 전반에는 조금 낮겠
고, 후반에는 비슷하겠습니다.<br />강수량은 평년(1~7mm)보다 많겠습니다.]]></wf>
        </header>
        <body>
          <location wl_ver="3"> → location마다 지역이 표기되어 있으므로 이를 모두 추출합니다.
          <province>서울 · 인천 · 경기도</province>
          <city>서울</city>
```

BeautifulSoup 모듈

```
<data>
  <mode>A02</mode>
  <tmEf>2019-04-25 00:00</tmEf>
  <wf>구름많음</wf>
  <tmn>14</tmn>
  <tmx>20</tmx>
  <reliability>보통</reliability>
</data>
<!-- 생략 -->
</location>
<!-- 생략 -->
</body>
</description>
</item>
</channel>
</rss>
```

→ 내부에 날씨가 적혀 있으므로 이것들을 가져옵니다.

BeautifulSoup 모듈

- 여기에서 지역 표기된 location 찾고,
location 내부에 있는 city, wf, tmn, tmx 내용 추출

도시: 서울

날씨: 구름많음

최저기온: 14

최고기온: 20

도시: 인천

날씨: 구름많음

최저기온: 13

최고기온: 18

도시: 수원

날씨: 구름많음

최저기온: 14

최고기온: 20

...생략...

- Django 모듈

- 다양한 기능 제공하는 웹 개발 프레임워크

- Flask

- 작은 기능만을 제공하는 웹 개발 프레임워크
- [Window] + [R] 눌러 프로그램 실행 창 띄우고 [cmd] 입력하여 명령 프롬프트 창

```
pip install flask
```

- 예시 – Flask 모듈 사용하기

```
01  from flask import Flask
02  app = Flask(__name__)
03
04  @app.route("/")
05  def hello():
06      return "<h1>Hello World!</h1>"
```

- 데코레이터 (decorator)
 - @app.route()

Flask 모듈

- Flask에서 코드 실행

```
set FLASK_APP=파일 이름.py  
flask run
```

```
> set FLASK_APP=flask_basic.py  
> flask run  
* Serving Flask app "flask_basic.py"  
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

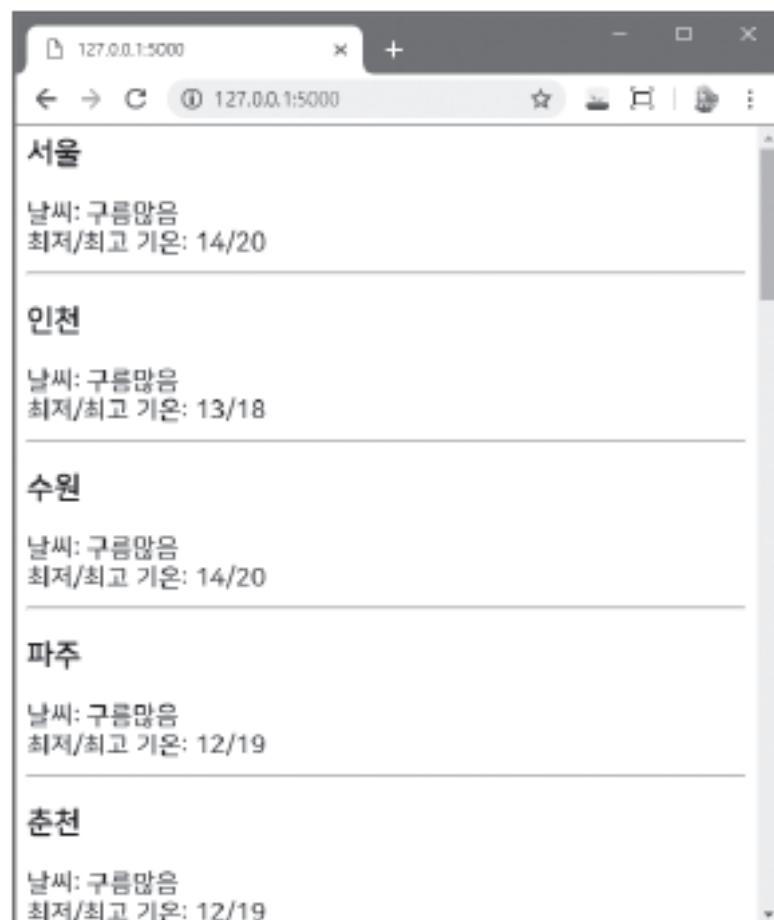
- 종료
 - [Ctrl] + [c]

- 예시 - BeautifulSoup 스크레이핑 실행하기

```
01  # 모듈을 읽어 들입니다.
02  from flask import Flask
03  from urllib import request
04  from bs4 import BeautifulSoup
05
06  # 웹 서버를 생성합니다.
07  app = Flask(__name__)
08  @app.route("/")
09
10  def hello():
11      # urlopen() 함수로 기상청의 전국 날씨를 읽습니다.
12      target = request.urlopen("http://www.kma.go.kr/weather/forecast/mid-term-rss3.jsp?stnId=108") → 이 코드는 한 줄 코드이니 이어서 입력해야 합니다.
13
14      # BeautifulSoup를 사용해 웹 페이지를 분석합니다.
15      soup = BeautifulSoup(target, "html.parser")
```

```
16
17     # location 태그를 찾습니다.
18     output = ""
19     for location in soup.select("location"):
20         # 내부의 city, wf, tmn, tmx 태그를 찾아 출력합니다.
21         output += "<h3>{}</h3>".format(location.select_one("city").string)
22         output += "날씨: {}<br/>".format(location.select_one("wf").string)
23         output += "최저/최고 기온: {}/{}"\
24             .format(\
25                 location.select_one("tmn").string,\
26                 location.select_one("tmx").string\
27             )
28         output += "<hr/>"
29     return output
```

접속할 때마다 날씨 정보를 보여주는 웹 서버



정리합시다

- 모듈 사용의 기본 : math 모듈
- random 모듈
- os 모듈
- datetime 모듈
- time 모듈
- urllib 모듈

모듈 만들기

- module_basic 디렉터리 만든 후 아래 두 파일 넣기



main.py



test_module.py

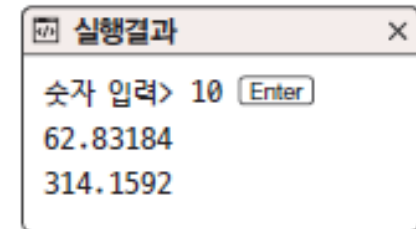
모듈 만들기

- module_basic 디렉터리 만든 후 아래 두 파일 저장하고
main.py 파일 실행

```
01  # test_module.py 파일
02  PI = 3.141592
03
04  def number_input():
05      output = input("숫자 입력> ")
06      return float(output)
07
08  def get_circumference(radius):
09      return 2 * PI * radius
10
11  def get_circle_area(radius):
12      return PI * radius * radius
```

모듈 만들기

```
01  # main.py 파일
02  import test_module as test
03
04  radius = test.number_input()
05  print(test.get_circumference(radius))
06  print(test.get_circle_area(radius))
```



- 패키지 (package)
 - 구조화된 모듈 만들 때 사용하는 기능

__name__=="__main__"

- `__name__`
 - 엔트리 포인트 (entry point) / 메인 (main)
 - 프로그램의 진입점
 - 메인 내부에서의 `__name__`은 “`__main__`”

```
>>> __name__  
'__main__'
```

- 모듈의 `__name__`
 - 엔트리 포인트 아니지만 엔트리 포인트 파일 내에서 import 되었기 때문에 모듈 내 코드가 실행
 - 모듈 내부에서 `__name__` 출력하면 모듈의 이름 나타냄

__name__=="__main__"

- 예시 - 모듈 이름을 출력하는 모듈 만들기

```
01  # main.py 파일
02  import test_module
03
04  print("# 메인의 __name__ 출력하기")
05  print(__name__)
06  print()
```

```
01  # test_module.py 파일
02  print("# 모듈의 __name__ 출력하기")

03  print(__name__)
04  print()
```

실행결과

```
# 모듈의 __name__ 출력하기
test_module

# 메인의 __name__ 출력하기
__main__
```

__name__=="__main__"

- __name__ 활용하기
 - 엔트리 포인트 파일 내부에서 __name__이 "__main__" 값을 가짐을 활용하여 현재 파일이 모듈로 실행되는지 엔트리 포인트로 실행되는지 확인
 - 예시 - test_module.py (모듈 활용하기)

```
01  PI = 3.141592
02
03  def number_input():
04      output = input("숫자 입력> ")
05      return float(output)
06
07  def get_circumference(radius):
08      return 2 * PI * radius
09
10  def get_circle_area(radius):
```

__name__=="__main__"

```
11     return PI * radius * radius
```

```
12
```

```
13 # 활용 예
```

```
14 print("get_circumference(10):", get_circumference(10))
```

```
15 print("get_circle_area(10): ", get_circle_area(10))
```

"이런식으로 동작해요!"를 알려주는
활용 예를 넣었습니다.



```
01 import test_module as test → 위 모듈을 읽어 들입니다.
```

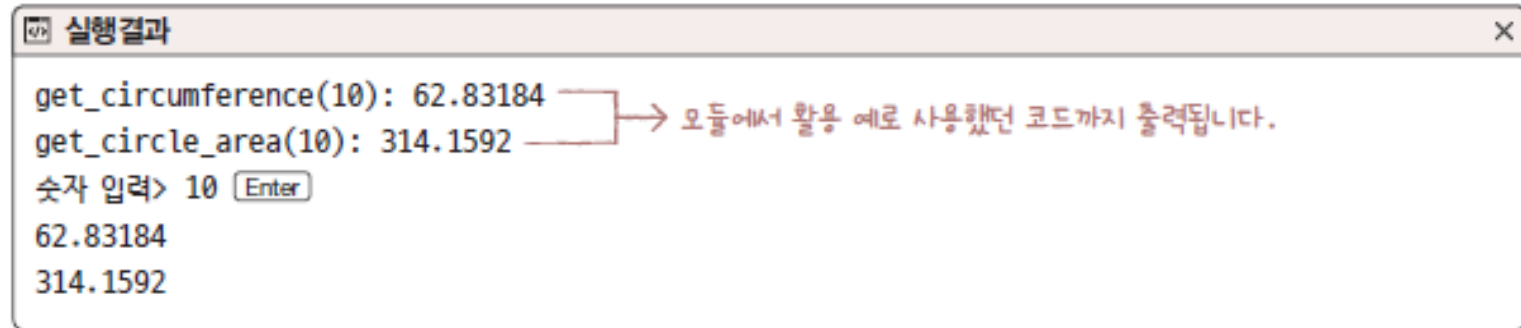
```
02
```

```
03 radius = test.number_input()
```

```
04 print(test.get_circumference(radius))
```

```
05 print(test.get_circle_area(radius))
```

`__name__ == "__main__"`



```
실행결과
get_circumference(10): 62.83184
get_circle_area(10): 314.1592
숫자 입력> 10 [Enter]
62.83184
314.1592
```

→ 모듈에서 활용 예로 사용했던 코드까지 출력됩니다.

- 현재 test_module.py 파일에는 동작 설명을 위해 추가한 활용 예시 부분 존재
- 모듈로 사용하고 있는데 내부에서 출력 발생하여 문제
- 현재 파일이 엔트리 포인트인지 구분하는 코드 활용
- 조건문으로 `__name__`이 `"__main__"`인지 확인

`__name__ == "__main__"`

```
06
07 def get_circumference(radius):
08     return 2 * PI * radius
09
10 def get_circle_area(radius):
11     return PI * radius * radius
```

```
12 # 활용 예
```

```
13 if __name__ == "__main__":
14     print("get_circumference(10):", get_circumference(10))
15     print("get_circle_area(10): ", get_circle_area(10))
```

현재 파일이 엔트리 포인트인지 확인하고,
엔트리 포인트일 때만 실행합니다.



__name__=="__main__"

```
01  import test_module as test
02
03  radius = test.number_input()
04  print(test.get_circumference(radius))
05  print(test.get_circle_area(radius))
```

실행결과

X

숫자 입력> 10

62.83184

314.1592

패키지

- 모듈 (module)
- 패키지 관리 시스템 (Package Management System)
 - pip
 - 모듈이 모여서 구조 이루면 패키지
- 패키지 만들기
 - main.py 파일은 엔트리 포인트로, test_package 폴더는 패키지로 사용



패키지

- test_package 폴더 내부에 module_a.py 파일과 module_b.py 파일 생성



module_a.py



module_b.py

- 두 파일에 아래와 같이 입력

```
01  # ./test_package/module_a.py의 내용
02  variable_a = "a 모듈의 변수"
```

```
01  # ./test_package/module_b.py의 내용
02  variable_b = "b 모듈의 변수"
```

패키지

```
01  # 패키지 내부의 모듈을 읽어 들입니다.  
02  import test_package.module_a as a  
03  import test_package.module_b as b  
04  
05  # 모듈 내부의 변수를 출력합니다.  
06  print(a.variable_a)  
07  print(b.variable_b)
```

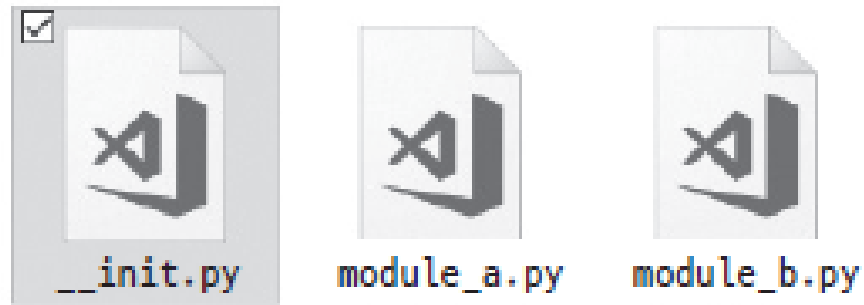
실행결과

a 모듈의 변수
b 모듈의 변수

패키지

- `__init__.py` 파일

- 패키지 읽을 때 어떤 처리를 수행해야 하거나 패키지 내부의 모듈들을 한꺼번에 가져오고 싶을 때 사용
- `test_package` 폴더 내부에 `__init__.py` 파일 추가



- 패키지 읽어들이는 때 `__init__.py`를 가장 먼저 실행
- 패키지와 관련된 초기화 처리 등 할 수 있음

패키지

```
01  # "from test_package import *"로
02  # 모듈을 읽어 들일 때 가져올 모듈
03  __all__ = ["module_a", "module_b"] → *사용 시 읽어들일 모듈의 목록
04
05  # 패키지를 읽어 들일 때 처리를 작성할 수도 있습니다.
06  print("test_package를 읽어 들였습니다.")
```

```
01  # 패키지 내부의 모듈을 모두 읽어 들입니다.
02  from test_package import *
03
04  # 모듈 내부의 변수를 출력합니다.
05  print(module_a.variable_a)
06  print(module_b.variable_b)
```

실행결과

```
test_package를 읽어 들였습니다.
a 모듈의 변수
b 모듈의 변수
```


모듈을 분석하는 방법 ()

- pip list 명령어 사용하여 설치된 명령어 확인하고
- pip show <설치된 모듈> 입력하여 모듈 설치된 위치 확인

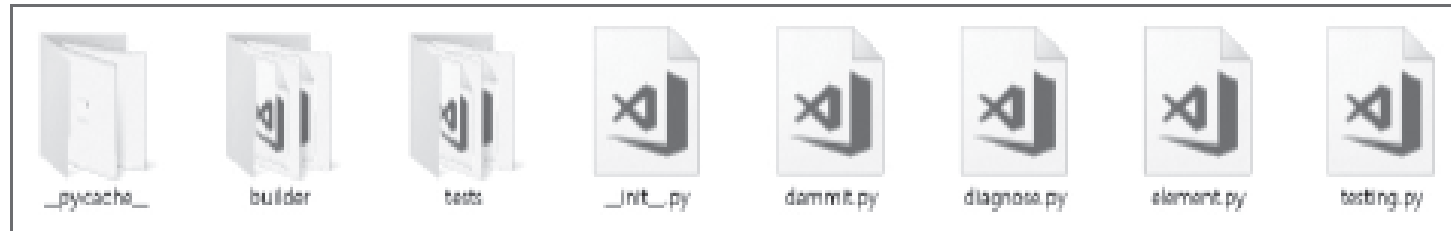
```
> pip list
astroid (1.5.2)
beautifulsoup4 (4.6.0)
certifi (2017.4.17)
chardet (3.0.4)
...생략...
> pip show beautifulsoup4
Name: beautifulsoup4
Version: 4.6.0
Summary: Screen-scraping library
Home-page: http://www.crummy.com/software/BeautifulSoup/bs4/
...생략...
Location: c:\users\hasat\appdata\local\programs\python\python36-32\lib\site-
packages

Requires:
```

모듈을 분석하는 방법 ()

- 탐색기 사용하여 Location 폴더로 들어가 여러 모듈 설치된 것 확인

BeautifulSoup 모듈의 파일



- 파일을 하나하나 열어보며 찬찬히 분석해보기

정리합시다

- 모듈 만들기
- `__name__=="__main__"`
- 패키지