LESSON

파이썬의 모듈

모듈(Module)의 정의

- 모듈은 파이썬 코드를 논리적으로 묶어서 관리하고 사용할 수 있도록 하는 것
- 하나의 파이썬 .py 파일이 하나의 모듈
- 모듈의 종류

표준 모듈

사용자 정의 모듈

외부 모듈 (3rd party)

모듈의 사용

- import 모듈명으로 모듈을 가져 올 수 있음
- as (alias)를 활용해 긴 모듈명을 줄일 수 있음
- 모듈은 하나의 새로운 이름 공간을 확보하며 정의됨

```
import keyword as k

print(k.kwlist)
```

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

keyword 모듈은 파이썬 기본 모듈로 파이썬에서 이미 예약된 키워드들을 정의. 모듈 내 kwlist 함수로 키워드 목록 확인 가능

모듈의 사용

- **함수와 모듈의 차이**
  - -함수 : 파일 내의 일부 코드를 묶어 사용하는 것
  - -모듈 : 파일 단위로 코드들을 묶어 사용하는 것
  - ※ 비슷하거나 관련된 일을 하는 함수 등의 코드를 하나의 파일에 저장하고 추후 사용하는 단위

```
print(kwlist)
```

```
----------------------------------------
NameError
<ipython-input-7-43df03ada53d> in <module>
----> 1 print(kwlist)

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

모듈을 호출(import)하지 않은 채 모듈 내 코드를 사용하면 오류 발생

모듈의 장점

- 중복된 코드를 줄이고 재사용성을 높임
- 전체 코드를 관련된 모듈들로 분리하여 설계함으로써 구조적 프로그래밍 가능
- 별도의 이름공간을 제공함으로써 동일한 이름의 여러 함수나 변수들을 모듈마다 정의할 수 있음

모듈의 검색경로

- 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

파이썬이 제공하는 표준 라이브러리 모듈들

- 파이썬은 기본적으로 상당히 많은 표준 라이브러리 모듈들을 제공하고 있음

파이썬에서 제공되는 모듈 목록 확인

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

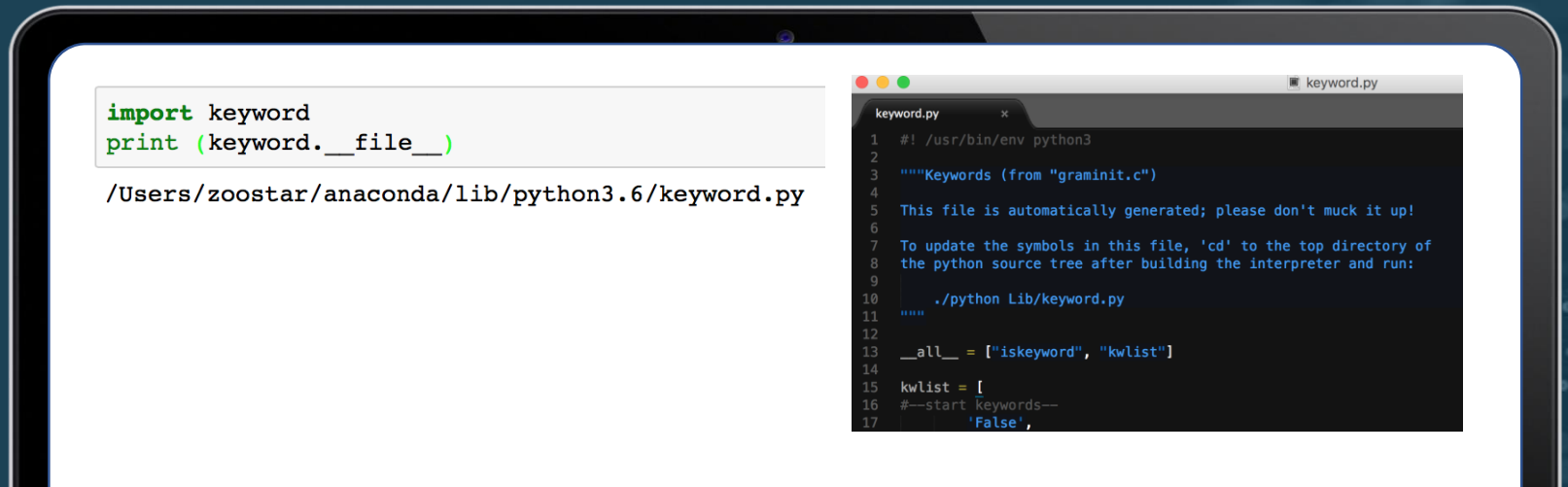
Python Module Index

_ | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | z

_	
__future__	*Future statement definitions*
__main__	*The environment where the top-level script is run.*
_dummy_thread	*Drop-in replacement for the _thread module.*
_thread	*Low-level threading API.*
a	
abc	*Abstract base classes according to :pep:`3119`.*
aifc	*Read and write audio files in AIFF or AIFC format.*
argparse	*Command-line option and argument parsing library.*
array	*Space efficient arrays of uniformly typed numeric values.*
ast	*Abstract Syntax Tree classes and manipulation.*
asynchat	*Support for asynchronous command/response protocols.*
asyncio	*Asynchronous I/O.*
asyncore	*A base class for developing asynchronous socket handling services.*
atexit	*Register and execute cleanup functions.*
audioop	*Manipulate raw audio data.*

파이썬이 제공하는 표준 라이브러리 모듈들

- help("modules") 명령어로 현재 사용 할 수 있는 (설치된) 모듈의 목록 확인 가능
- Anaconda로 파이썬을 설치한 경우 아나콘다 폴더 내 lib 폴더에 모듈을 직접 확인 가능
  - 해당 파일을 직접 찾아서 해당 모듈의 내용을 수정할 수 있지만 권장하지 않음



파이썬 표준 모듈 예시

- os 모듈: 운영체제와 상호작용하기 위한 수십 가지 함수들을 제공
- time 모듈: 시간과 관련된 여러 함수들을 제공

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

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

```
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_isd
st=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_isd
st=0)
```

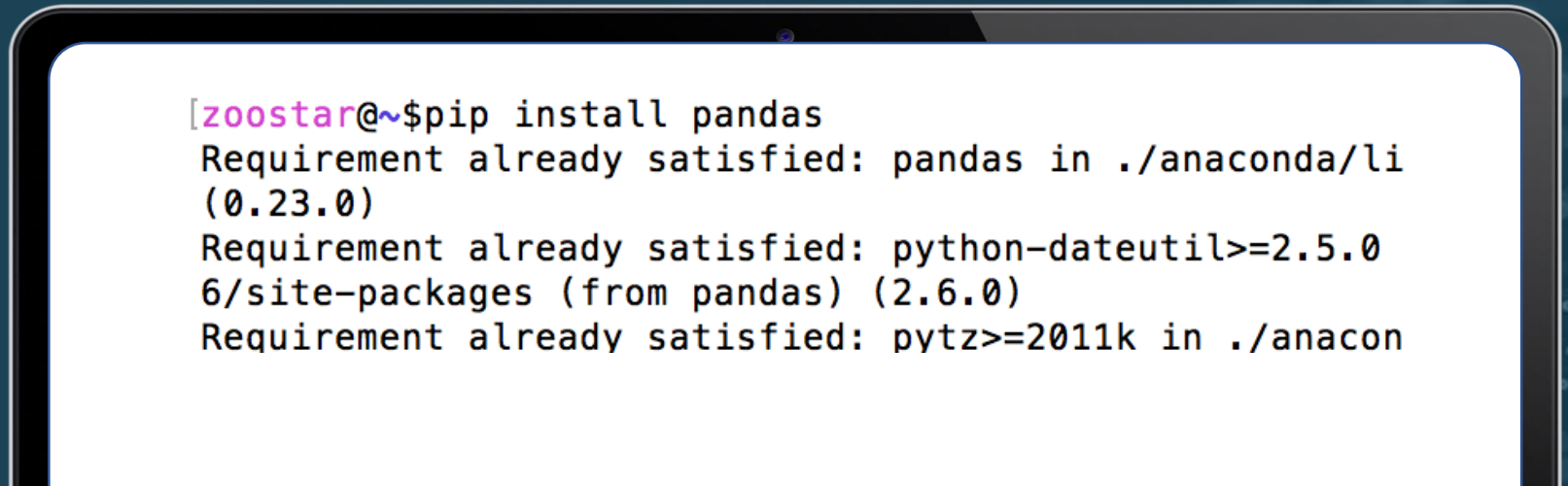
외부 모듈이란?

- 최근 4차산업혁명과 관련해 IT 분야에서 파이썬의 중요성이 높아지고 있는 이유 중 하나는 파이썬의 외부 모듈이 해당 분야의 프로그래밍에 도움되는 **외부 모듈을 다양하게 제공**하기 때문

데이터 분석/통계	인공지능	웹 크롤링
numpy, pandas, matplotlib …	Tensorflow, PyTorch, Keras …	BeautifulSoup, selenium …

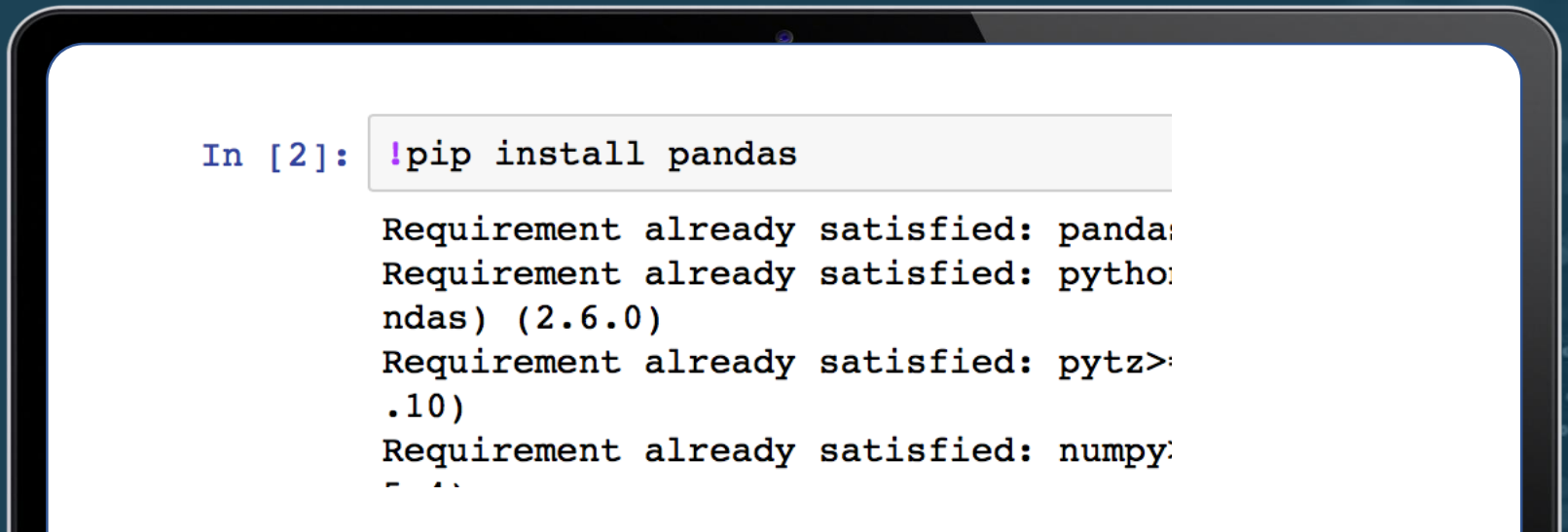
외부 모듈 : 설치하기(1/3)

- 패키지 관리자 사용하기(pip)
  - 파이썬 3.4 이후 버전은 기본적으로 pip 를 포함하고 있으며, 손쉽게 외부 모듈을 설치할 수 있음
- 콘솔 창에서 pip install 패키지명 한 줄로 외부 모듈 설치 가능



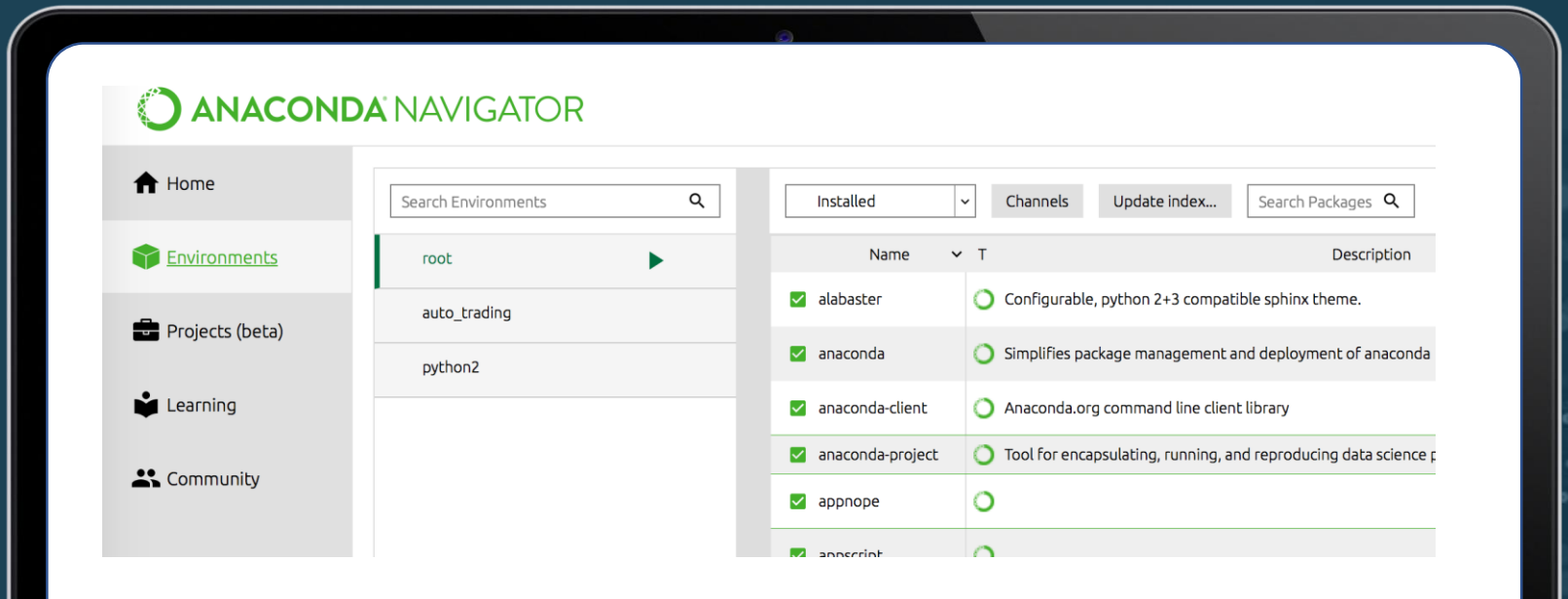
외부 모듈 : 설치하기(2/3)

- Jupyter Notebook 환경에서 명령어 앞에 '!'를 붙여 설치 가능
  - 느낌표(!)를 붙이면 콘솔 창에서 입력하는 것과 같은 역할
  - 삭제: pip uninstall 모듈명



외부 모듈 : 설치하기(3/3)

- Anaconda를 설치했다면 Anaconda Navigator에서 설치 가능
  - -환경을 달리 정의할 수 있고, 환경마다 패키지(외부 모듈)를 설치할 수 있음

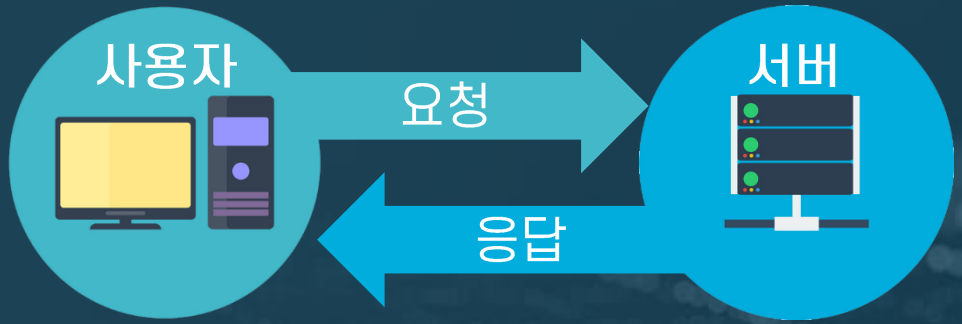


LESSON

requests 모듈

HTTP 요청/응답 구조

- **HTTP는 요청과 응답으로 이루어져 있음**
  - 사용자가 원하는 정보를 요청하게 되면,
    서버는 해당 요청을 확인 후 적절한 응답을 해주는 구조



HTTP 요청/응답 구조 : 요청

- 사용자가 서버에 요청을 할 때는 크게 네 개로 구분하여 요청할 수 있음
  - URL로 요청하며 요청할 때 메서드를 변경하여 기능을 구분

메서드	설명
GET	정보를 가져오기 위해 요청
POST	새로운 정보를 보내기 위해 요청
PUT	수정할 정보를 보내기 위해 요청
DELETE	정보를 삭제하기 위해 요청

HTTP 요청/응답 구조 : 응답

- 서버가 사용자에게 요청에 대한 응답을 보낼 때 크게 5개의 경우가 있음
  - 응답 코드로 요청의 진행 상황과 서버의 상태를 예측 가능

응답 코드	설명
1XX	요청을 받았고, 작업 진행 중
2XX	사용자의 요청이 성공적으로 수행 됨
3XX	요청은 완료 되었으나, 리다이렉션이 필요
4XX	사용자의 요청이 잘못됨
5XX	서버에 오류가 발생함

requests 모듈 설치

- pip install requests 명령어로 설치
  - Anaconda를 설치했다면 기본으로 함께 설치가 되어 있음
- import requests로 모듈 호출 후 사용

능력개발교육원 홈페이지 요청 후 응답 코드 출력 예

```
import requests

URL = 'http://hrdi.koreatech.ac.kr'
response = requests.get(URL)
print(response.status_code)
```

```
200
```

※ Naver 메인 페이지 정보 가져오기

requests 모듈 활용 예시1

```
import requests

URL = 'https://www.naver.com'
response = requests.get(URL)
print(response.status_code)
print(response.text)
```

get 메서드로 Naver 서버에 정보를 가져오기 위한 GET 요청하면, 서버가 응답한 정보에서 text 메서드로 html 출력

```
<meta name="Referrer" content="origin">
<meta http-equiv="Content-Script-Type" content="text/javascript">
<meta http-equiv="Content-Style-Type" content="text/css">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=1100">
<meta name="apple-mobile-web-app-title" content="NAVER" />
<meta name="robots" content="index,nofollow"/>
<meta name="description" content="네이버 메인에서 다양한 정보와 유용한 컨텐츠를 만나 보세요"/>
<meta property="og:title" content="네이버">
<meta property="og:url" content="https://www.naver.com/">
<meta property="og:image" content="https://s.pstatic.net/static/www/mobile/edit/2016/0705/m
<meta property="og:description" content="네이버 메인에서 다양한 정보와 유용한 컨텐츠를 만나 보세요"/>
<meta name="twitter:card" content="summary">
<meta name="twitter:title" content="">
<meta name="twitter:url" content="https://www.naver.com/">
<meta name="twitter:image" content="https://s.pstatic.net/static/www/mobile/edit/2016/0705/
<meta name="twitter:description" content="네이버 메인에서 다양한 정보와 유용한 컨텐츠를 만나 보세요"/>
```

※ Naver 검색 결과 정보 가져오기(1)

requests 모듈 활용 예시2

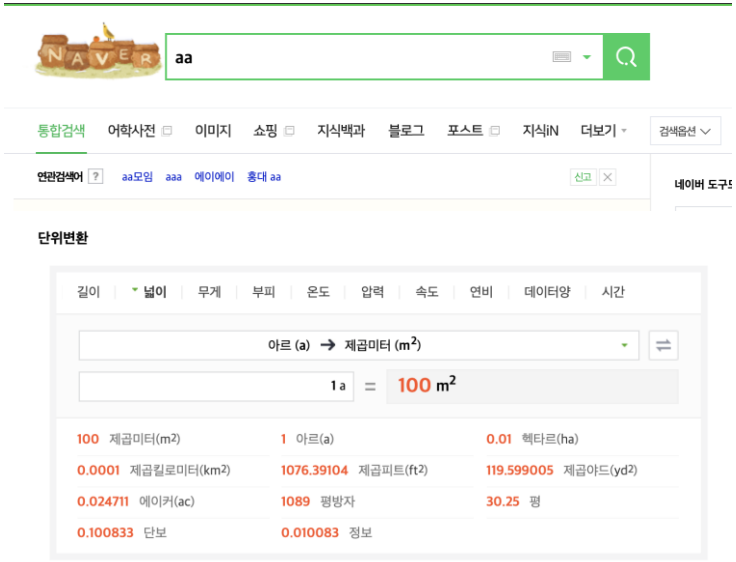
```
import requests

URL = 'https://search.naver.com/search.naver'
params = {'query' : 'aa'}
response = requests.get(URL,params=params)
print(response.status_code)
print(response.text)
```

Naver에서 'aa'를 검색

- get 메서드에 params 인자에 값을 넣어 함께 GET 요청
- Naver의 검색 결과 주소는 'https://search.naver.com/search.naver'
- 파라미터: query

※ Naver 검색 결과 정보 가져오기(1)

requests 모듈 활용 예시2




```
7%9C%EA%B3%84%EC%82%B0" class="_a">날짜계산</a></li> <li class="_li"><a nocr=1 onclick="ja
te_clicklog(this.className);" href="#" class="star_ico _star">전역일 계산 즐겨찾기 등록</a><a
R(this, 'u='+urlencode(this.href)+'&r=1&a=nco_x8a*1.name&i='+'80143911_0000001D1728');"
etc&query=%EC%A0%84%EC%97%AD%EC%9D%BC%EA%B3%84%EC%82%B0" class="_a">전역일 계산</a></li> <
ck="javascript:_dss_util_set_favorite_clicklog(this.className);" href="#" class="star_ic
</a><a nocr onclick="return goOtherCR(this, 'u='+urlencode(this.href)+'&r=1&a=nco_x8a*1
28');" href="?where=nexearch&sm=tab_etc&query=%EB%A7%8C%EB%82%98%EC%9D%B4%EA%B3%84%EC%8
</li> </ul> </li> </ul> <ul class="main_tlst last_lst"> <li class="_onTarget"> <a nocr
s, 'r=1&a=nco_x8a*1.trans&i='+'80143911_0000001D1728');" href="#" class="tlst_btn _tlst_
</span><em>단위변환</em><span class="gt_ico _onTarget"></span></a> <ul class="sub_tlst _su
ocr=1 onclick="javascript:_dss_util_set_favorite_clicklog(this.className);" href="#" cla
기 등록</a><a nocr onclick="return goOtherCR(this, 'u='+urlencode(this.href)+'&r=1&a=nco_
001D1728');" href="?where=nexearch&sm=tab_etc&query=%EA%B8%B8%EC%9D%B4%EB%B3%80%ED%99%9
이변환">길이</a></li> <li class="_li"><a nocr=1 onclick="javascript:_dss_util_set_favorite
ef="#" class="star_ico _star">넓이 즐겨찾기 등록</a><a nocr onclick="return goOtherCR(this,
=1&a=nco_x8a*1.name&i='+'80143911_0000001D1728');" href="?where=nexearch&sm=tab_etc&que
0%ED%99%98" class="_a" data-my-name="넓이변환">넓이</a></li> <li class="_li"><a nocr=1 oncl
_favorite_clicklog(this.className);" href="#" class="star_ico _star">무게 즐겨찾기 등록</a><
rCR(this, 'u='+urlencode(this.href)+'&r=1&a=nco_x8a*1.name&i='+'80143911_0000001D1728')
```

※ Naver 검색 결과 정보 가져오기(2)

requests 모듈 활용 예시3

```
import requests

URL = 'https://comic.naver.com/webtoon/detail.nhn'
params = {'titleId' : 687915,'no' : 157}
response = requests.get(URL,params=params)
print(response.text)
```

https://comic.naver.com/webtoon/detail.nhn?titleId=687915&no=157

파라미터를 두 개 이상 사용하여 정보 요청 가능 (특정 날짜의 네이버 웹툰)

※ Naver 검색 결과 정보 가져오기(2)

requests 모듈 활용 예시3


```
                                                    <div data-ad-banner="premium_ba
                                                    <iframe id="da_webtoon_end" dat
ttps://veta.naver.com/fxshow?su=SU10316" marginheight="0" marginwidth="0" borde
lign="center" width="694" height="0"></iframe>

                                                </div>


                                                <iframe id="commentIframe" width="100%"
="꿈의 기업 - 4부 14화 댓글영역" style="min-height:500px"></iframe>

                                                <!-- 최신 업데이트 만화  -->
                                <div class="new_comic">

                                                    <h4><img src="https://s
OMICWEB/NAVER/img/detail/h_tit_new.gif" width="113" height="15" alt="최신 업데이트
                                                    <ul class="lst_new">
```

requests 모듈 활용 Tip

json 형식의 데이터 가져오기

-json을 파이썬의 사전 자료형으로 활용 가능

```
import requests

response = requests.get('https://raw.githubusercontent.com/naver/naver-openapi-guide/draft/naver-openapi-swagger.json')
result = response.json()
print(type(result))
```

```
<class 'dict'>
```

```
print(dic.items())
```

```
dict_items([('swagger', '2.0'), ('info', {'description': 'Nave
Naver Open API at [https://developers.naver.com](https://devel
Open API', 'termsOfService': 'https://developers.naver.com/prc
th', '/'), ('tags', [{'name': 'Clova', 'description': 'Naver A
'Find out more', 'url': 'https://developers.naver.com/products
Machine Learning Translation APIs'}, {'name': 'Naver Login', '
ervices', 'description': 'Naver data trend, search, shorten ur
er Mpas JS, geocode, static map APIs'}]), ('schemes', ['https'
lova'], 'summary': 'Clova Face Recognition (얼굴감지)', 'descrip
ationId': 'vision.face', 'consumes': ['multipart/form-data'],
```

requests 모듈 활용 Tip

- 그 밖의 활용법
  - python requests 문서: https://3.python-requests.org/
- 데이터와 함께 POST 요청
  - post(url, data=XX) , post(url, files=XXX)
- 헤더, 쿠키와 함께 GET 요청
  - get(url, headers=XXX, cookies=XXX)
- 인코딩 설정
  - encoding = 'UTF-8'