웹 페이지의 콘텐츠 읽어오기

한국기술교육대학교
온라인평생교육원

학습목표

- urllib 패키지를 활용하여 GET 방식 요청과 POST 방식 요청의 구현 방법을 설명하고, Query 문자열 또는 요청 파라미터를 포함하여 요청할 수 있다.
- requests 패키지를 활용하여 GET 방식 요청과 POST 방식 요청의 구현 방법을 설명하고, Query 문자열 또는 요청 파라미터를 포함하여 요청할 수 있다.

학습내용

- urllib 패키지를 활용한 웹 페이지 요청
- requests 패키지를 활용한 웹 페이지 요청

urllib 패키지를 활용한 웹 페이지 요청

1 주요 모듈 활용

urllib	▪ URL 작업을 위한 여러 모듈을 모은 패키지 ▪ 파이썬의 표준 라이브러리

URL 문자열과 웹 요청에 관련된 모듈 5개 제공

- urllib.request
  — URL 문자열을 가지고 요청 기능 제공
- urllib.response
  — urllib 모듈에 의해 사용되는 응답 클래스들 제공
- urllib.parse
  — URL 문자열을 파싱하여 해석하는 기능 제공
- urllib.error
  — urllib.request에 의해 발생하는 예외 클래스들 제공
- urllib.robotparser
  — robots.txt 파일을 구문 분석하는 기능 제공

urllib 패키지를 활용한 웹 페이지 요청

1 주요 모듈 활용

URL 문자열과 웹 요청에 관련된 모듈 5개 제공

- urllib.request
  — URL 문자열을 가지고 요청 기능 제공
- urllib.response
  — urllib 모듈에 의해 사용되는 응답 클래스들 제공
- urllib.parse
  — URL 문자열을 파싱하여 해석하는 기능 제공
- urllib.error
  — urllib.request에 의해 발생하는 예외 클래스들 제공
- urllib.robotparser
  — robots.txt 파일을 구문 분석하는 기능 제공

URL 문자열을 가지고 HTTP 요청을 수행하는 urllib.request 모듈

URL 문자열(주소)을 해석하는 urllib.parse 모듈

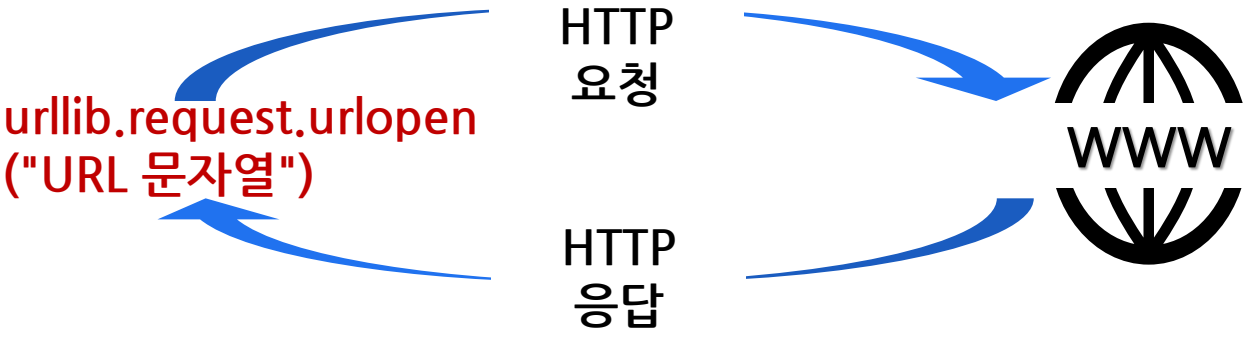
urllib 패키지를 활용한 웹 페이지 요청

1 주요 모듈 활용

1 urllib.request 모듈

- URL 문자열을 가지고 HTTP 요청을 수행
- urlopen() 함수를 사용하여 웹 서버에 페이지를 요청하고, 서버로부터 받은 응답을 저장하여 응답 객체(http.client.HTTPResponse)를 반환

```
res = urllib.request.urlopen
("요청하려는 페이지의 URL 문자열")
```



urllib 패키지를 활용한 웹 페이지 요청

1 주요 모듈 활용

2 http.client.HTTPResponse 클래스

- 웹 서버로부터 받은 응답을 래핑하는 객체
- 응답 헤더나 응답 바디의 내용을 추출하는 메서드 제공

- HTTPResponse.read([amt])
- HTTPResponse.readinto(b)
- HTTPResponse.getheader(name, default=None)
- HTTPResponse.getheaders()
- HTTPResponse.msg
- HTTPResponse.version
- HTTPResponse.status
- HTTPResponse.reason
- HTTPResponse.closed

urllib 패키지를 활용한 웹 페이지 요청

1 주요 모듈 활용

3 http.client.HTTPResponse 객체의 read() 메서드

- read() 메서드를 실행하면 웹 서버가 전달한 데이터(응답 바디)를 바이트열로 읽어 들임

바이트열	
	▪ 16진수로 이루어진 수열이기 때문에 읽기 어려우므로 웹 서버가 보낸 한글을 포함한 텍스트 형식의 HTML 문서의 내용을 읽을 때는 텍스트 형식으로 변환함 ▪ 바이트열(bytes)의 decode('문자 셋') 메서드를 실행하여 응답된 문자 셋에 알맞은 문자로 변환함

urllib 패키지를 활용한 웹 페이지 요청

1 주요 모듈 활용

3 http.client.HTTPResponse 객체의 read() 메서드

res.read()

```
<body>₩r₩n<h1>₩xea₩
xb0₩x80₩xeb₩x82₩x98
₩xeb₩x8b₩xa4ABC</h1>
₩r₩n</body>
```

res.read().decode('utf-8')

```
<body>
<h1>가나다ABC</h1>
</body>
```

urllib 패키지를 활용한 웹 페이지 요청

1 주요 모듈 활용

4 웹 페이지 인코딩 체크(1)

- 웹 크롤링하려는 웹 페이지가 어떠한 문자 셋으로 작성되었는지 파악하는 것이 필수

페이지의 문자 셋 정보

- 페이지의 소스 내용에서 <meta> 태그의 charset 정보를 체크하면 파악 가능

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<h1>블럭스타일 태그</h1>
```

```
<!doctype html>
<!--[if lt IE 7]>   <html class="n
<!--[if IE 7]>      <html class="n
<!--[if IE 8]>      <html class="n
<!--[if gt IE 8]><!--><html class=

<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compati
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<html>
  <head>
      <title id="Title">알라딘</title>
      <meta http-equiv="Content-Type" content="text/html; charset=euc-kr">
      <meta content="Microsoft Visual Studio .NET 7.1" name="GENERATOR">
      <meta content="C#" name="CODE_LANGUAGE">
      <meta content="JavaScript" name="vs_defaultClientScript">
```

urllib 패키지를 활용한 웹 페이지 요청

1 주요 모듈 활용

5 웹 페이지 인코딩 체크(2)

- 웹 페이지의 문자 셋 정보를 파이썬 프로그램으로도 파악할 수 있음

사용되는 API	▪ http.client.HTTPMessage 객체의 get_content_charset() 메서드

urllib.request.urlopen() 함수의 리턴 값인 http.client.HTTPResponse 객체의 info() 메서드 호출

↓

http.client.HTTPMessage 객체가 리턴 됨

↓

get_content_charset() 메서드 호출

↓

문자 셋 정보를 문자열로 리턴 받음

urllib 패키지를 활용한 웹 페이지 요청

1 주요 모듈 활용

5 웹 페이지 인코딩 체크(2)

웹 서버로부터 응답될 때 전달되는
Content-Type이라는 응답 헤더 정보를 읽고
해당 페이지의 문자 셋 정보를 추출해 줌

```
url = 'http://www.python.org/'
f = urllib.request.urlopen(url)
encoding = f.info().get_content_charset()
```

urllib 패키지를 활용한 웹 페이지 요청

1 주요 모듈 활용

6 urllib.parse 모듈

- 웹 서버에 페이지 또는 정보를 요청할 때 함께 전달하는 데이터
  - GET 방식 요청 : Query 문자열
  - POST 방식 요청 : 요청 파라미터

name=value&name=value&name=value&.....

- 영문과 숫자는 그대로 전달되지만 한글은 %기호와 함께 16진수 코드 값으로 전달되어야 함
- 웹 크롤링을 할 때 요구되는 Query 문자열을 함께 전달해야 하는 경우, 직접 Query 문자열을 구성해서 전달해야 함

urllib.parse 모듈 사용

- urllib.parse.urlparse()
- urllib.parse.urlencode()

urllib 패키지를 활용한 웹 페이지 요청

1 주요 모듈 활용

6 urllib.parse 모듈

1 urllib.parse.urlparse("URL문자열")

urlparse() 함수	▪ 아규먼트에 지정된 URL 문자열의 정보를 파싱하고 각각의 정보를 정해진 속성으로 저장하여 urllib.parse.ParseResult 객체를 리턴 함 ▪ 각 속성들을 이용하여 필요한 정보만 추출할 수 있음

```
url1 =urlparse
('https://movie.daum.net/moviedb/main?movieId=93252')
```

```
ParseResult(scheme='https',
netloc='movie.daum.net',
            path='/moviedb/main', params='',
            query='movieId=93252', fragment='')
```

```
url1.netloc, url1.path, url1.query, url1.scheme,
url1.port, url1.fragment, url1.geturl()
```

urllib 패키지를 활용한 웹 페이지 요청

1 주요 모듈 활용

6 urllib.parse 모듈

2 urllib.parse.urlencode()

urlencode() 함수	▪ 메서드의 아규먼트로 지정된 name과 value로 구성된 딕셔너리 정보를 정해진 규격의 Query 문자열 또는 요청 파라미터 문자열로 리턴 함

```
urlencode({'number': 12524, 'type': 'issue', 'action': 'show'})
```

```
number=12524&type=issue&action=show
```

```
urlencode({'addr': '서울시 강남구 역삼동'})
```

```
addr=%EC%84%9C%EC%9A%B8%EC%8B%9C+%EA%B0%95%EB%82%A8%EA%B5%AC+%EC%97%AD%EC%82%BC%EB%8F%99
```

urllib 패키지를 활용한 웹 페이지 요청

1 주요 모듈 활용

7 Query 문자열을 포함하여 요청

- Query 문자열을 포함하여 요청하는 것
  ➡ GET 방식 요청

- urllib.parse.urlencode 함수로 name과 value로 구성되는 Query 문자열을 만듦
- URL 문자열의 뒤에 '?' 기호를 추가하여 요청 URL로 사용

```
params = urllib.parse.urlencode({'name': '유니코',
'age': 10})
url ="http://unico2013.dothome.co.kr/
crawling/get.php?%s" % params
```

⬇

```
http://unico2013.dothome.co.kr/crawling/get.php?n
ame=%EC%9C%A0%EB%8B%88%EC%BD%94&
age=10
```

```
urllib.request.urlopen(url)
```

urllib 패키지를 활용한 웹 페이지 요청

1 주요 모듈 활용

8 요청 파라미터를 포함하여 요청(1)

- 요청 바디안에 요청 파라미터를 포함하여 요청하는 것
  ➡ POST 방식 요청

- GET 방식과 같이 name과 value로 구성되는 문자열을 만듦
- POST 방식 요청에서는 바이트 형식의 문자열로 전달해야 하므로, encode('ascii') 메서드를 호출하여 바이트 형식의 문자열로 변경
- urllib.request.urlopen() 호출 시 바이트 형식의 문자열로 변경된 데이터를 두 번째 아규먼트로 지정

```
data = urllib.parse.urlencode({'name': '유니코', 'age': 10})
data = data.encode('ascii')
```

```
b'name=%EC%9C%A0+%EB%8B%88%EC%BD%94&age=10'
```

```
url="http://unico2013.dothome.co.kr/crawling/post.php"
urllib.request.urlopen(url, data)
```

urllib 패키지를 활용한 웹 페이지 요청

1 주요 모듈 활용

9 요청 파라미터를 포함하여 요청(2)

- URL 문자열과 요청 파라미터 문자열을 지정한 urllib.request.Request 객체 생성
- urllib.request.urlopen() 함수 호출 시 URL 문자열 대신 urllib.request.Request 객체 지정

```
data = urllib.parse.urlencode({'name': '유니코', 'age': 10})
postdata = data.encode('ascii')
req = urllib.request.Request(url='http://unico2013.dothome.co.kr/crawling/post.php',data=postdata)
urllib.request.urlopen(req)
```

urllib 패키지를 활용한 웹 페이지 요청

2 소스 분석

1 소스 분석(1)

```
#파일명 : exam3_1.py
import urllib.request
res = urllib.request.urlopen("http://www.naver.com/")
print(type(res))
print(res.status)
print(res.version)
print(res.msg)
res_header = res.getheaders()
print("[ header 정보 ]----------")
for s in res_header :
    print(s)
```

```
PS C:\example\myvscode> & C:/Users/Samsung/Anaconda3/python.exe c:/example/myvscode/unit3/exam3_1.py
<class 'http.client.HTTPResponse'>
200
11
OK
[ header 정보 ]----------
('Server', 'NWS')
('Date', 'Sat, 11 May 2019 03:41:33 GMT')
('Content-Type', 'text/html; charset=UTF-8')
('Transfer-Encoding', 'chunked')
('Connection', 'close')
('Set-Cookie', 'PM_CK_loc=0045c9b9eb46ab09fa603466d76ebc7675cf46d75f88e82c62791b7f7ec93108; Expires=Sun, 12 May 2019
03:41:33 GMT; Path=/; HttpOnly')
('Cache-Control', 'no-cache, no-store, must-revalidate')
('Pragma', 'no-cache')
('P3P', 'CP="CAO DSP CURa ADMa TAIa PSAa OUR LAW STP PHY ONL UNI PUR FIN COM NAV INT DEM STA PRE"')
('X-Frame-Options', 'DENY')
('X-XSS-Protection', '1; mode=block')
('Strict-Transport-Security', 'max-age=63072000; includeSubdomains')
('Referrer-Policy', 'unsafe-url')
```

urllib 패키지를 활용한 웹 페이지 요청

2 소스 분석

2 소스 분석(2)

```
#파일명 : exam3_2.py
import urllib.request
res = urllib.request.urlopen

("http://unico2013.dothome.co.kr/crawling/tagstyle.html")
print(res)
print("[ header 정보 ]----------")
res_header = res.getheaders()
for s in res_header :
    print(s)
print("[ body 내용 ]----------")
print(res.read())
#print(res.read().decode('utf-8'))
```

urllib 패키지를 활용한 웹 페이지 요청

2 소스 분석

2 소스 분석(2)

```
PS C:\example\myvscode> & C:/Users/Samsung/Anaconda3/python.exe c:/example/myvscode/unit3/exam3_2.py
<http.client.HTTPResponse object at 0x0000027DECE3AB00>
[ header 정보 ]----------
('Date', 'Sat, 11 May 2019 04:09:56 GMT')
('Server', 'Microsoft-IIS/5.0')
('Content-Length', '461')
('Connection', 'close')
('Content-Type', 'text/html')
[ body 내용 ]-----------
b'<!DOCTYPE html>\r\n<html>\r\n<head>\r\n<meta charset="UTF-8">\r\n<title>Insert title here</title>\r\n</
head>\r\n<body>\r\n<h1>\xeb\xb8\x94\xeb\x9f\xad\xec\x8a\xa4\xed\x83\x80\xec\x9d\xbc \xed\x83\x9c\xea\xb7\
xb8</h1>\r\n<div style="background-color:yellow">\xed\x85\x8c\xec\x8a\xa4\xed\x8a\xb8\xec\x9e\x85\xeb\x8b
\x88\xeb\x8b\xa41</div>\r\n<div>\xed\x85\x8c\xec\x8a\xa4\xed\x8a\xb8\xec\x9e\x85\xeb\x8b\x88\xeb\x8b\xa42
</div>\r\n<div>\xed\x85\x8c\xec\x8a\xa4\xed\x8a\xb8\xec\x9e\x85\xeb\x8b\x88\xeb\x8b\xa43</div>\r\n<hr>\r\
n<h1>\xec\x9d\xb8\xeb\x9d\xbc\xec\x9d\xb8\xec\x8a\xa4\xed\x83\x80\xec\x9d\xbc \xed\x83\x9c\xea\xb7\xb8</h
1>\r\n<span style="background-color:yellow">\xed\x85\x8c\xec\x8a\xa4\xed\x8a\xb8\xec\x9e\x85\xeb\x8b\x88\
xeb\x8b\xa41</span>\r\n<span>\xed\x85\x8c\xec\x8a\xa4\xed\x8a\xb8\xec\x9e\x85\xeb\x8b\x88\xeb\x8b\xa42</s
pan>\r\n<span>\xed\x85\x8c\xec\x8a\xa4\xed\x8a\xb8\xec\x9e\x85\xeb\x8b\x88\xeb\x8b\xa43</span>\r\n</body>
\r\n</html>'
```

```
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
<h1>블럭스타일 태그</h1>
<div style="background-color:yellow">테스트입니다1</div>
<div>테스트입니다2</div>
<div>테스트입니다3</div>
<hr>
<h1>인라인스타일 태그</h1>
<span style="background-color:yellow">테스트입니다1</span>
<span>테스트입니다2</span>
<span>테스트입니다3</span>
</body>
</html>
```

urllib 패키지를 활용한 웹 페이지 요청

2 소스 분석

3 소스 분석(3)

```
#파일명 : exam3_3.py
import urllib.request
print("================================")
url = 'https://www.python.org/'
f = urllib.request.urlopen(url)
print(type(f))
print(type(f.info()))
encoding = f.info().get_content_charset()
print(url, ' 페이지의 인코딩 정보 :', encoding)
text = f.read(500).decode(encoding)
print(text)
print("================================")
```

urllib 패키지를 활용한 웹 페이지 요청

2 소스 분석

3 소스 분석(3)

```
url = 'https://www.daum.net/'
f = urllib.request.urlopen(url)
encoding = f.info().get_content_charset()
print(url, ' 페이지의 인코딩 정보 :', encoding)
text = f.read(500).decode(encoding)
print(text)
print("==================================")

url = 'https://www.aladin.co.kr/home/welcome.aspx'
f = urllib.request.urlopen(url)
encoding = f.info().get_content_charset()
print(url, ' 페이지의 인코딩 정보 :', encoding)
text = f.read(500).decode(encoding)
print(text)
print("==================================")
```

urllib 패키지를 활용한 웹 페이지 요청

2 소스 분석

3 소스 분석(3)

```
PS C:\example\myvscode> & C:/Users/Samsung/Anaconda3/python.exe c:/example/myvscode/unit3/exam3_3.py
============================================================
<class 'http.client.HTTPResponse'>
<class 'http.client.HTTPMessage'>
https://www.python.org/  페이지의 인코딩 정보 : utf-8
<!doctype html>
<!--[if lt IE 7]>   <html class="no-js ie6 lt-ie7 lt-ie8 lt-ie9">   <![endif]-->
<!--[if IE 7]>      <html class="no-js ie7 lt-ie8 lt-ie9">          <![endif]-->
<!--[if IE 8]>      <html class="no-js ie8 lt-ie9">                 <![endif]-->
<!--[if gt IE 8]><!--><html class="no-js" lang="en" dir="ltr">  <!--<![endif]-->

<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <link rel="prefetch" href="//ajax.googleapis.com/ajax/libs/jqu
============================================================
https://www.daum.net/  페이지의 인코딩 정보 : utf-8
<!DOCTYPE html>
<html lang="ko" class="">
<head>
<meta charset="utf-8"/>
<title>Daum</title>
<meta property="og:url" content="https://www.daum.net/">
```

```
<meta property="og:type" content="website">
<meta property="og:title" content="Daum">
<meta property="og:image" content="//i1.daumcdn.net/svc/image/U03/common_icon/5587C4E4012FCD0001">
<meta property="og:description" content="나의 관심 콘텐츠를 가장 즐겁게 볼 수 있는 Daum">
<meta name="msapplication-task" content="name=Daum;action-
============================================================
https://www.aladin.co.kr/home/welcome.aspx  페이지의 인코딩 정보 : ks_c_5601-1987

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN" >
<html>
  <head>
      <title id="Title">알라딘</title>
      <meta http-equiv="Content-Type" content="text/html; charset=euc-kr">
          <meta content="Microsoft Visual Studio .NET 7.1" name="GENERATOR">
          <meta content="C#" name="CODE_LANGUAGE">
          <meta content="JavaScript" name="vs_defaultClientScript">
          <meta content="http://schemas.microsoft.com/intellisense/ie5" name="vs_targetSchema">
      <meta http-equiv="X-U
============================================================
```

requests 패키지를 활용한 웹 페이지 요청

1 requests 패키지 소개

1 requests 패키지란?

requests 패키지	▪ Kenneth Reitz에 의해 개발된 파이썬 라이브러리 ▪ HTTP 프로토콜과 관련된 기능 지원

requests 패키지의 공식 홈페이지 소개

Requests is an elegant and simple HTTP library for Python, built for human beings. You are currently looking at the documentation of the development release.

〈 https://2.python-requests.org/en/master/ 〉

requests 패키지를 활용한 웹 페이지 요청

1 requests 패키지 소개

1 requests 패키지란?

- 아나콘다에는 requests 패키지가 site-packages로 설치되어 있음
- 만일 설치를 해야 한다면 pipenv 명령으로 설치

```
C:\Users\Samsung>pip show requests
Name: requests
Version: 2.21.0
Summary: Python HTTP for Humans.
Home-page: http://python-requests.org
Author: Kenneth Reitz
Author-email: me@kennethreitz.org
License: Apache 2.0
Location: c:\users\samsung\anaconda3\lib\site-packages
Requires: urllib3, certifi, chardet, idna
Required-by: Sphinx, conda, conda-build, anaconda-project, anaconda-client
```

pipenv install requests

urllib 패키지	requests 패키지
인코딩하여 바이너리 형태로 데이터 전송	딕셔너리 형태로 데이터 전송
데이터 전달 방식에 따라 GET 요청, POST 요청을 구분	요청 메서드(GET, POST)를 명시하여 요청

requests 패키지를 활용한 웹 페이지 요청

1 requests 패키지 소개

2 requests.request() 함수

- requests 패키지의 대표 함수
- HTTP 요청을 서버에 보내고 응답을 받아오는 기능 지원

requests.request(method, url, **kwargs)

- method : 요청 방식 지정(GET, POST, HEAD, PUT, DELETE, OPTIONS)
- url : 요청할 대상 URL 문자열 지정
- params : [선택적] 요청 시 전달할 Query 문자열 지정 (딕셔너리, 튜플리스트, 바이트열 가능)
- data : [선택적] 요청 시 바디에 담아서 전달할 요청 파라미터 지정 (딕셔너리, 튜플리스트, 바이트열 가능)
- json : [선택적] 요청 시 바디에 담아서 전달할 JSON 타입의 객체 지정
- auth : [선택적] 인증처리(로그인)에 사용할 튜플 지정

requests 패키지를 활용한 웹 페이지 요청

1 requests 패키지 소개

3 HTTP 요청 방식을 지원하는 함수

- requests.request() 함수 외에 각각의 요청 방식에 따른 메서드들도 제공
- requests.request() 함수에 요청 방식을 지정하여 호출하는 것과 동일

- requests.get(url, params=None, **kwargs)
- requests.post(url, data=None, json=None, **kwargs)
- requests.head(url, **kwargs)
- requests.put(url, data=None, **kwargs)
- requests.patch(url, data=None, **kwargs)
- requests.delete(url, **kwargs)

requests 패키지를 활용한 웹 페이지 요청

1 requests 패키지 소개

3 HTTP 요청 방식을 지원하는 함수

- HTTP 프로토콜에서 정의한 GET, POST, HEAD, PUT, PATCH, DELETE 등의 요청 방식을 처리하는 메서드들을 모두 지원
  - ➡ GET, HEAD, POST만 학습

GET 방식

- 요청한 페이지의 헤더, 바디를 모두 받아오는 요청
- Query 문자열을 추가하여 요청할 수도 있음

HEAD 방식

- 콘텐츠 없이 요청 헤더만을 받아오는 방식
- 요청 바디에 요청 파라미터 데이터를 추가하여 요청
- 헤더와 바디를 모두 받아옴

requests 패키지를 활용한 웹 페이지 요청

1 requests 패키지 소개

4 GET 방식 요청

- GET 방식 요청은 다음 두 가지 함수 중 하나를 호출하여 처리 가능

```
requests.request('GET', url, **kwargs)

requests.get(url, **kwargs)
        [ kwargs ]
        params - (선택적) 요청 시 전달할 Query 문자열을 지정
```

- Query 문자열을 포함하여 요청 : params 매개변수에 딕셔너리, 튜플리스트, 바이트열(bytes) 형식으로 전달
- Query 문자열을 포함하지 않는 요청
  : params 매개변수의 설정 생략

requests 패키지를 활용한 웹 페이지 요청

1 requests 패키지 소개

5 POST 방식 요청

- POST 방식 요청은 다음 두 가지 함수 중 하나를 호출하여 처리 가능

```
requests.request('POST', url, **kwargs)

requests.post(url, **kwargs)
```

[kwargs]
data - (선택적) 요청 시 바디에 담아서 전달할 요청 + 파라미터를 지정
딕셔너리, 튜플리스트 형식, 바이트열(bytes) 형식
json - (선택적) 요청 시 바디에 담아서 전달할 JSON 타입의 객체를 지정
JSON 형식

requests 패키지를 활용한 웹 페이지 요청

1 requests 패키지 소개

6 응답 처리

- requests.request(), requests.get(), requests.head(), requests.post() 함수 모두 리턴 값은 requests.models.Response 객체임

text

- 문자열 형식으로 응답 콘텐츠 추출
- 추출 시 사용되는 문자 셋은 'ISO-8859-1'이므로 'utf-8'이나 'euc-kr' 문자 셋으로 작성된 콘텐츠 추출 시 한글이 깨지는 현상 발생
- 추출 전 응답되는 콘텐츠의 문자 셋 정보를 파악하여 Response 객체의 encoding 속성에 문자 셋 정보를 설정한 후 추출

content

- 바이트열 형식으로 응답 콘텐츠 추출
- 응답 콘텐츠가 이미지와 같은 바이너리 형식인 경우 사용
- 한글이 들어간 문자열 형식인 경우 r.content.decode('utf-8')를 사용해서 디코드 해야 함

requests 패키지를 활용한 웹 페이지 요청

2 소스 분석

1 소스 분석(1)

```
#파일명 : exam3_8.py
import requests
r = requests.request('get',
'http://unico2013.dothome.co.kr/crawling/exam.html')
r.encoding = 'utf-8'
print(type(r))
if r.text :
    print(r.text)
else :
    print('응답된 콘텐츠가 없어요')
print('-------------------------------------------------------')
r = requests.request('head',
'http://unico2013.dothome.co.kr/crawling/exam.html')
r.encoding = 'utf-8'
print(type(r))
if r.text :
    print(r.text)
else :
    print('응답된 콘텐츠가 없어요')
print('-------------------------------------------------------')
r = requests.request('post',
'http://unico2013.dothome.co.kr/crawling/post.php',
data= {'name':'백도', 'age' : 12})
r.encoding = 'utf-8'
print(type(r))
if r.text :
    print(r.text)
else :
    print('응답된 콘텐츠가 없어요')
```

requests 패키지를 활용한 웹 페이지 요청

2 소스 분석

1 소스 분석(1)

```
<class 'requests.models.Response'>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>테스트</title>
</head>
<body>
<h1>웹 크롤링을 테스트 합니다.</h1>
</body>
</html>
------------------------------------------------------------
<class 'requests.models.Response'>
응답된 콘텐츠가 없어요
------------------------------------------------------------
<class 'requests.models.Response'>
 <!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>POST TEST</title>
  </head>
  <body>
   <h1>POST : 이름은 백도이고 나이는 12이네요!!</h1>   </body>
</html>
```

requests 패키지를 활용한 웹 페이지 요청

2 소스 분석

2 소스 분석(2)

```
#파일명 : exam3_9.py
import requests

r = requests.get('http://unico2013.dothome.co.kr/
crawling/exam.html')
r.encoding = 'utf-8'
print(type(r))
print(r.headers)
if r.text :
    print(r.text)
else :
    print('응답된 콘텐츠가 없어요')
```

```
<class 'requests.models.Response'>
{'Date': 'Sat, 11 May 2019 19:40:59 GMT', 'Server': 'Microsoft-IIS/5.0', 'Content-Length': '164', 'Conne
ction': 'close', 'Content-Type': 'text/html'}
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>테스트</title>
</head>
<body>
<h1>웹 크롤링을 테스트 합니다.</h1>
</body>
</html>
```

requests 패키지를 활용한 웹 페이지 요청

2 소스 분석

3 소스 분석(3)

```
#파일명 : exam3_10.py
import requests

r = requests.head('http://unico2013.dothome.co.kr
/crawling/exam.html')
print(type(r))
print(r.headers)
if r.text :
    print(r.text)
else :
    print('응답된 콘텐츠가 없어요')
```

```
<class 'requests.models.Response'>
{'Date': 'Thu, 15 Aug 2019 06:01:39 GMT', 'Server': 'Microsoft-IIS/5.0', 'Connection':
 'close', 'Content-Type': 'text/html'}
응답된 콘텐츠가 없어요
```

학습정리

1. urllib 패키지를 활용한 웹 페이지 요청

- urllib 패키지에서는 URL 문자열(주소)을 해석해주는 urllib.parse 모듈과 HTTP 요청을 수행하는 urllib.request 모듈이 주로 쓰임
- urlencode() 함수는 name과 value로 구성된 딕셔너리 정보를 정해진 규격의 Query 문자열 또는 요청 파라미터 문자열로 리턴함
- POST 방식 요청에서는 바이트 형식의 문자열로 전달해야 하므로 encode('ascii') 메서드를 호출하여 바이트 형식의 문자열로 변경해야 함

학습정리

2. requests 패키지를 활용한 웹 페이지 요청

- requests 패키지는 Kenneth Reitz에 의해 개발된 파이썬 라이브러리로, HTTP와 관련된 기능을 지원함
- requests.request() 함수는 requests 패키지의 대표 함수로서 HTTP 요청을 서버에 보내고 응답을 받아오는 기능을 지원하는 함수임
- Query 문자열을 포함하여 GET 방식으로 요청할 때는 params 매개변수에 딕셔너리, 튜플리스트, 바이트열(bytes) 형식으로 설정함
- POST방식 요청 시 data 매개변수에 딕셔너리, 튜플리스트 형식, 바이트열(bytes)로 지정 가능하며, json 매개변수에는 JSON 객체 형식을 지정함