

# 11장. 정규 표현식 및 웹 크롤링



# 정규식 – Regular Expression

## ❖ 정규표현식이란?

- 문자열에서 특정 패턴을 찾거나, 검사하거나, 바꾸는 도구이다

+ 파이썬의 정규 표현식에 대해 알려줘



## ❖ 활용 분야

- 이메일 형식인지 확인
- 전화번호만 추출
- 숫자만 골라내기
- 특정 단어 치환

# 정규식 – Regular Expression

- 자주 사용하는 정규 표현식

표현식	설 명
<code>^</code>	정규식 시작
<code>\$</code>	정규식 끝
<code>^[0-9]*\$</code>	숫자
<code>^[a-zA-Z]*\$</code>	영문 대, 소문자
<code>^[가-힣]*\$</code>	한글
<code>^010[-](d{3} \d{4})[-]\d{4}\$</code>	휴대폰
<code>^\d{6}[-][1-4]{6}\$</code>	주민등록번호

# 정규식 – Regular Expression

- 정규표현식에 사용되는 메타문자

메타문자	설 명(사용 예)
[ ]	대괄호는 [ ]사이의 문자들과 일치함, [x]
-	문자의 범위를 지정하는 하이픈(-), [1-4]
^	부정을 나타내는 캐럿, [^0-9]
*	0번 이상 반복, 1번 이상 반복(+)
{m}	m은 반복횟수, {3,4} – 3개 또는 4개
( )	소괄호는 서브 클래스. 그룹을 만들 때 사용
\d	숫자 – [0-9]
\w	알파벳 + 숫자
\s	공백

## 정규표현식 지원 – re 모듈

- 자주 사용하는 함수

re 모듈 가져오기

```
import re
```

메서드	기 능
re.match()	문자열 <b>처음부터</b> 패턴이 맞는지
re.serarch()	문자열 <b>어디든</b> 패턴이 있으면.
re.findall()	패턴에 맞는 <b>모든 값 리스트</b>
re.finditer()	패턴에 맞는 <b>반복자(iterator)</b>
re.sub()	패턴을 찾아 <b>다른 문자열로 치환</b>

# 정규식을 사용한 문자열 검색

- 정규 표현식 활용

```
import re

# search 예제
text = "내 전화번호는 010-1234-5678 입니다"
result = re.search(r"\d{3}-\d{4}-\d{4}", text)

print(result.group())

# findall 예제
text = "사과 3개, 배 5개, 감 10개"
numbers = re.findall(r"\d+", text)
print(numbers)

# sub 예제
text = "비밀번호는 1234입니다"
masked = re.sub(r"\d", "*", text)
print(masked)
```

# 정규표현식 지원 – re 모듈

- 정규 표현식 활용

1. `re.compile('[a-z]+')` : 정규 표현식을 컴파일 한다.
2. `match("korea")` : 문자열의 시작 부분에서 정규 표현식과 일치하는 부분을 찾음

<match 객체의 주요 메서드>

메서드	기 능
<code>group()</code>	매치된 문자열을 돌려준다.
<code>start()</code>	매치된 문자열의 시작위치를 돌려준다.
<code>end()</code>	매치된 문자열의 끝위치를 돌려준다
<code>span()</code>	매치된 문자열의 (시작, 끝)에 해당하는 튜플 반환.

# 정규식을 사용한 문자열 검색

- 정규 표현식 활용

```
pat = re.compile("[a-z]+") #정규 표현식
mat = pat.match("korea")   #조사할 문자열
print(mat)
print(mat.group()) #korea

if mat:
    print('문자열 있음: ', mat.group())
else:
    print('문자열 없음')

# *은 0개이상, +는 1개 이상
pat = re.compile("a+b")
mat = pat.match("b") #aaab
# print(mat)
if mat:
    print('문자열 있음: ', mat.group())
else:
    print('문자열 없음')
```



# 정규식을 사용한 문자열 검색

- 유효성 검사

- ✓ fullmatch() 함수 – 문자열 전체가 정규 표현식과 일치하는지를 찾음

```
# 전화번호 검증
# phone_pat = re.compile('010-\d{3,4}-\d{4}')
phone_pat = re.compile("010-[0-9]{3,4}-[0-9]{4}")
mat = phone_pat.fullmatch("010-12-5678")
print(bool(mat)) #False

# 한글과 전화번호 패턴 검사
name_pat = "제갈수연";
pat = re.compile("[가-힣]{2,5}")
mat = pat.fullmatch(name_pat)
print(bool(mat)) #True
```

# 정규식을 사용한 문자열 검색

## ● 유효성 검사 예제

```
# 전화번호 패턴 유효성 검사
def validate_phone_number(phone):
    """전화번호 유효성 검사 (010-XXXX-XXXX 형식)"""
    phone_pat = re.compile("010-\d{3,4}-\d{4}")
    return bool(phone_pat.fullmatch(phone))

phone_list = [
    "010-1234-5678", # 유효
    "010-123-4567",  # 유효
    "010-12-5678",   # 무효
    "012-1234-5678", # 무효
    "01012345678",   # 무효
    "010-1234-567"   # 무효
]

print("=== 전화번호 검증 결과 ===")
for phone in phone_list:
    print(f"{phone}: {validate_phone_number(phone)}")
```

# 정규식을 사용한 문자열 검색

- 유효성 검사 예제

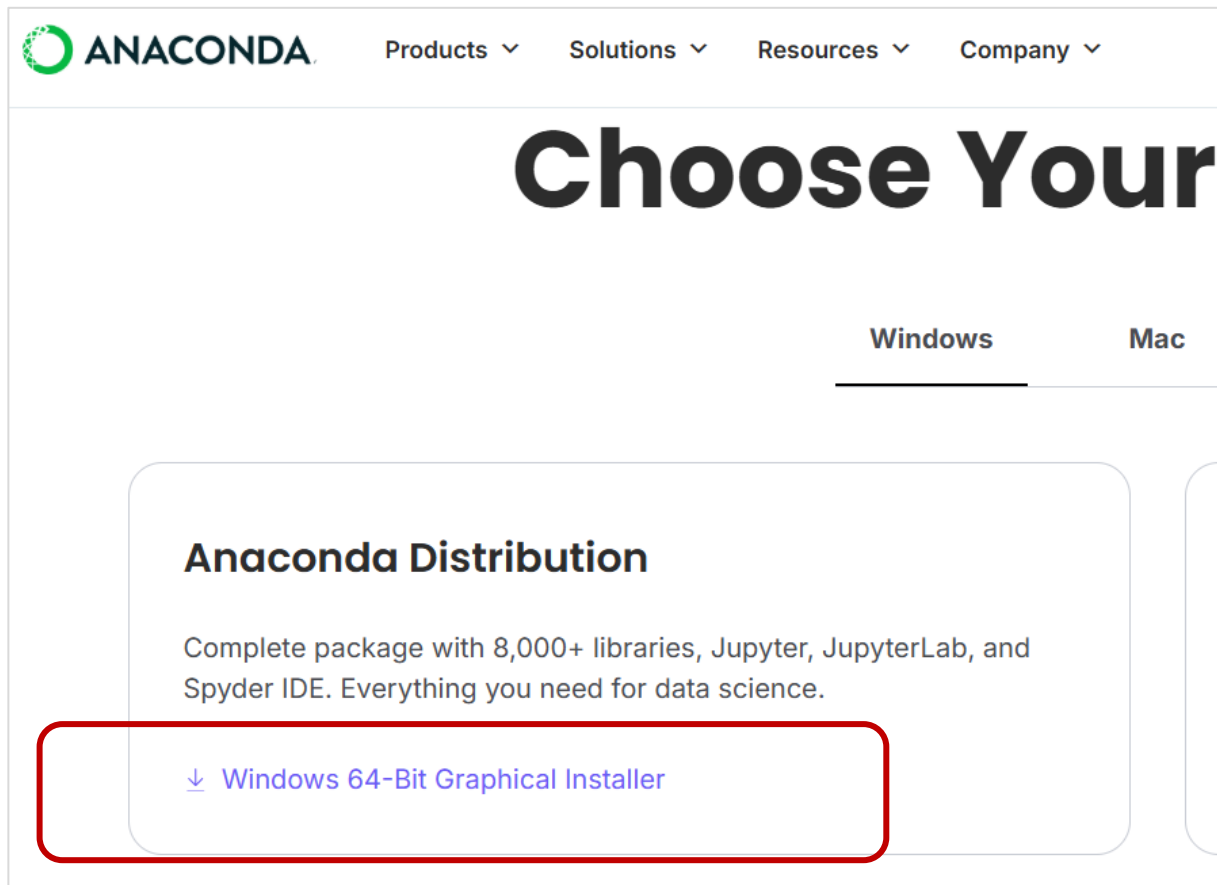
```
# 한글이름 패턴 유효성 검사
def validate_name(user_name):
    pattern = re.compile("[가-힣]{2,5}$")
    return bool(pattern.fullmatch(user_name))

while True:
    user_name = input("한글 이름 입력 (2~5자): ")

    if validate_name(user_name):
        print(f"이름: {user_name}")
        break
    else:
        print("올바른 한글 이름이 아닙니다. 다시 입력하세요")
```

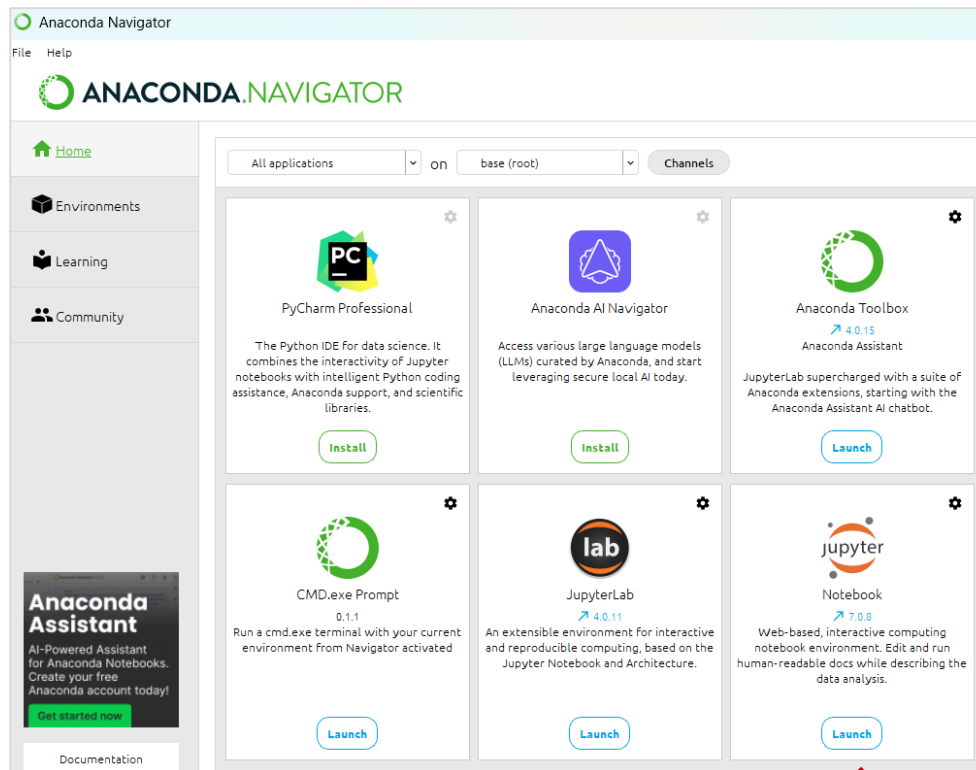
# 아나콘다(Anaconda) 플랫폼

## ❖ *Anaconda Download*



# 아나콘다(Anaconda) 플랫폼

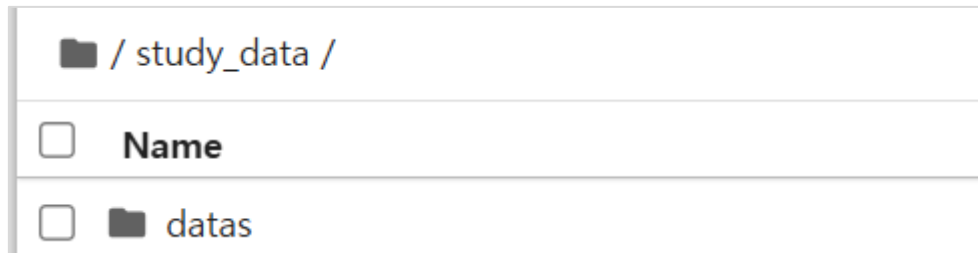
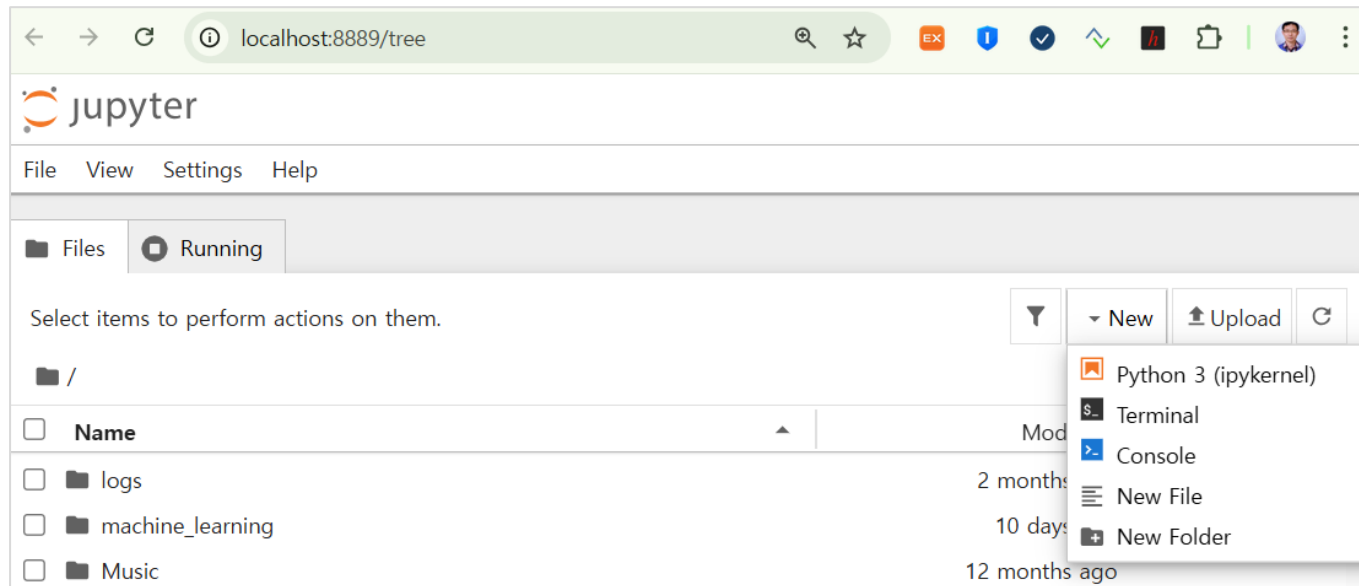
## ❖ 주피터 노트북(Jupyter Notebook) IDE 실행



# jupyter notebook ^사용

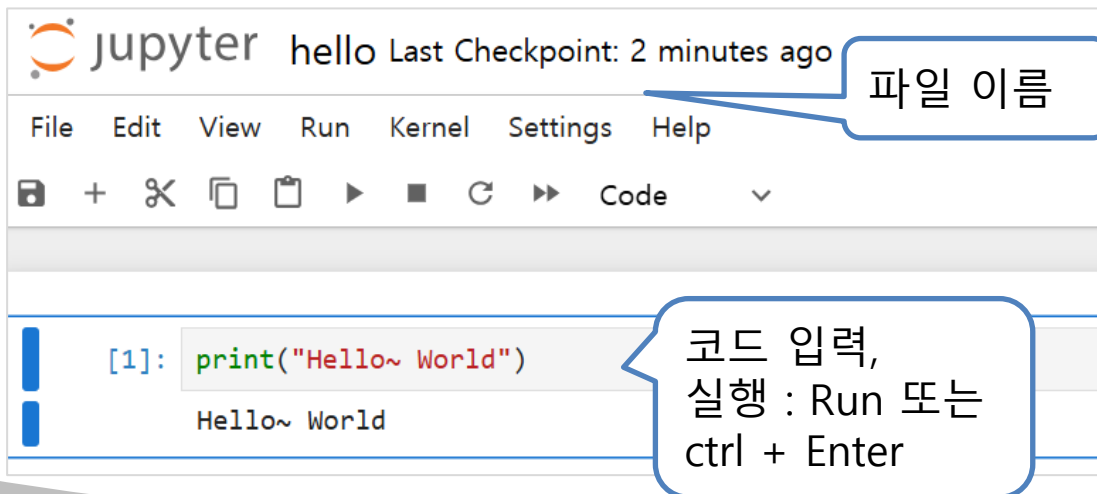
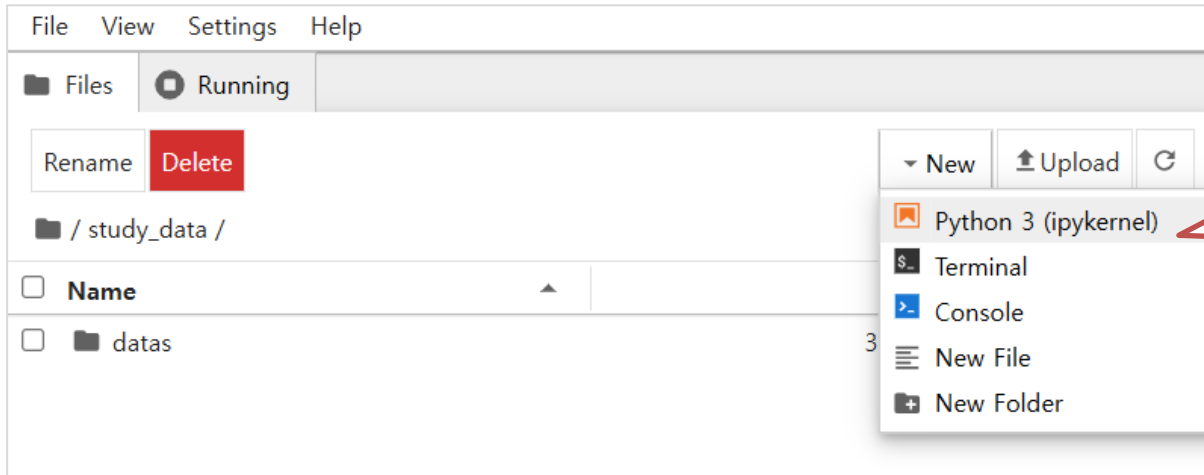
## ❖ 작업 디렉터리 생성

- study\_data 폴더 생성 > 하위에 data 폴더 생성



# jupyter notebook ^사용

- 파일 만들기(basic.ipynb)



# Markdown 문서 이해하기

- 파이썬 코딩

```
[1]: print("Hello~ World")
```

Hello~ World

```
[2]: print("안녕~ 세계")
```

안녕~ 세계

```
[7]: 10 + 20  
      10 - 20
```

```
[7]: -10
```

```
[6]: n1 = 20  
      n2 = 10
```

```
print(n1 + n2)  
print(n1 - n2)  
print(n1 * n2)  
print(n1 / n2)  
print(n1 % n2)
```

30  
10  
200  
2.0  
0

```
[10]: carts = ["계란", "라면", "콩나물"]  
       print(carts)
```

```
for cart in carts:  
    print(cart)
```

['계란', '라면', '콩나물']  
계란  
라면  
콩나물



# 세팅(Settings)

- Settings Editor 설정

- 자동 완성 기능

Home > Settings > Settings Editor > Enable autocompletion

- 줄번호, 글꼴 등

Home > Settings > Notebook

- line numbers
- Font Size : 20

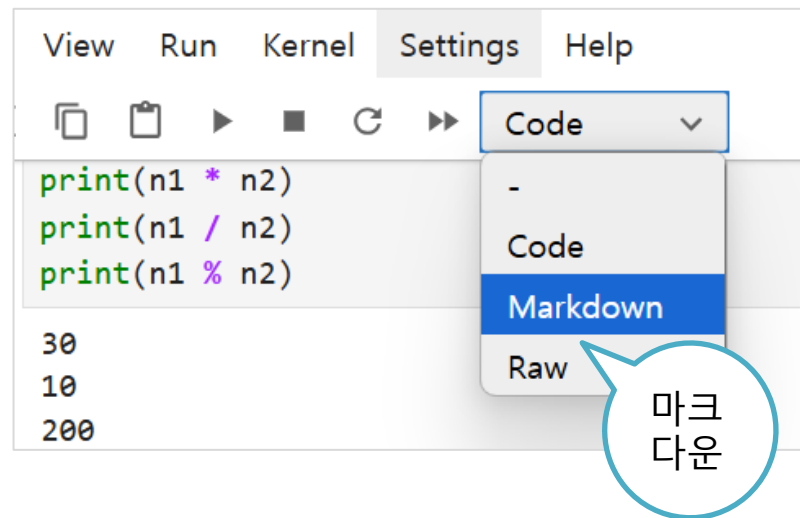
# Markdown 문서 이해하기

- 마크다운(Markdown)

마크다운(Markdown)은 일반 텍스트 기반의 마크업 언어이다.

- 기호 종류

- 제목: #의 개수가 클수록 작은 제목( ## 텍스트)
- 목록 - '\*' 사용



# Markdown 문서 이해하기

- 마크다운(Markdown)

## ### 함수 만들기

- \* 기능 - 절대값 계산
- \* 설명 - 음수는 양수로 양수는 양수로 반환

```
def my_abs(x):  
    if x < 0:  
        return -x  
    else:  
        return x  
  
print(my_abs(-3))
```

3

## 함수 만들기

- 기능 - 절대값 계산
- 설명 - 음수는 양수로 양수는 양수로 반환

```
def my_abs(x):  
    if x < 0:  
        return -x  
    else:  
        return x  
  
print(my_abs(-3))
```

3

# 웹 스크래핑 = 웹 크롤링

## ➤ 웹 크롤링이란?

+ 파이썬의 웹 크롤링에 대해 알려줘



웹 페이지에서 **자동으로 데이터를 수집**하는 기술입니다.  
업무 자동화나 데이터 분석에 정말 많이 쓰여요.

- 뉴스 제목 수집
- 쇼핑몰 상품 가격 모니터링
- 공지사항 자동 수집
- 환율, 날씨 데이터 수집

# 웹 스크래핑 = 웹 크롤링

## ➤ 웹 크롤링의 기본 구조

요청(Request) → 웹 서버 → 응답(Response, HTML) → 파싱(분석)  
→ 데이터 추출

## ➤ 웹 크롤링에 주로 사용되는 라이브러리

- requests
- BeautifulSoup(bs4)

# 웹 스크래핑 = 웹 크롤링

## ▷ requests 모듈

Python 프로그래밍 언어용 HTTP 라이브러리이다.

HTTP 프로토콜을 이용하여 웹 사이트로부터 데이터를 송수신할 수 있다.

**url 요청 - requests.get(url)**

```
import requests

url = "https://www.python.org"
response = requests.get(url)
print(response) #<Response [200]>
print(response.status_code) #200

html = response.text
print(html)
```

<Response [200]>

정상

<Response [404]>

페이지 없음

# 웹 스크래핑 = 웹 크롤링

## ▷ requests 모듈(라이브러리)

```
url2 = "https://www.python.org/3"  
response = requests.get(url2)  
print(response)
```

<Response [200]>

정상

<Response [404]>

페이지 없음

# 로봇 배제 표준

## ▷ 로봇 배제 표준

로봇 배제 표준이란?

웹사이트에 로봇이 접근하는 것을 방지하기 위한 규약 robots.txt에 기술하고 있다.

- 로봇에 의한 접근이 허용되는 경우라도 웹 서버에 무리가 갈 만큼 반복적으로 웹 페이지를 요청하는 것과 같이 서비스 안정성을 해칠 수 있는 행위를 하지 않아야 함
- 크롤링(또는 스크래핑)으로 취득한 자료를 임의로 배포하거나 변경하는 등의 행위는 저작권을 침해할 수 있으므로 저작권 규정을 준수해야 함



# 로봇 배제 표준

## ▷ 로봇 배제 표준

템플릿 태그	설 명
User-agent: * Disallow: /	모든(*) 로봇(검색엔진 봇)에게 루트 디렉터리(/) 이하 모든 문서에 대한 접근을 차단한다.
User-agent: * Allow: /	모든(*) 로봇에게 루트 디렉터리(/) 이하 모든 문서에 대한 접근을 허락한다.
User-agent: * Disallow: /temp/	모든(*) 로봇에게 특정 디렉터리(/temp/)에 대한 접근을 차단한다.

✓ **User-agent:** 지침을 적용할 크롤러 이름

# 로봇 배제 표준

## ▷ 로봇 배제 표준 – python.org

```
← ↻ 🔒 https://www.python.org/robots.txt

# Directions for robots.  See this URL:
# http://www.robotstxt.org/robotstxt.html
# for a description of the file format.

User-agent: HTTrack
User-agent: puf
User-agent: MSIECrawler
Disallow: /

# The Krugle web crawler (though based on Nutch) is OK.
User-agent: Krugle
Allow: /
Disallow: /~guido/orlijn/
Disallow: /webstats/

# No one should be crawling us with Nutch.
User-agent: Nutch
Disallow: /

# Hide old versions of the documentation and various large
User-agent: *
Disallow: /~guido/orlijn/
Disallow: /webstats/
```

HTTrack, puf, MSIECrawler 같은  
사이트 복제용 툴이 웹사이트  
전체를 긁어가지 못하게 차단하  
는 역할

# 로봇 배제 표준

## ▷ 로봇 배제 표준

```
import requests

urls = ["https://www.naver.com/", "https://www.python.org/"]
filename = "robots.txt"

v for url in urls:
    url_path = url + filename
    print(url_path)
    response = requests.get(url_path)
    print(response)

https://www.naver.com/robots.txt
<Response [200]>
https://www.python.org/robots.txt
<Response [200]>
```

# HTML이란?

- HTML(HyperText Markup Language)

- 하이퍼텍스트를 마크업 하는 언어, HTML5(현재 버전)
- **하이퍼텍스트** : 웹 사이트에서 링크를 클릭해 다른 문서나 사이트로 이동하는 기능
- **마크업** : **tag**(태그)를 사용해 문서에서 어느 부분이 제목이고 본문인지, 어느 부분이 사진이고 링크인지 표시하는 명령어(코드)

- HTML의 역사

인물 : **팀버너스리** – 웹의 아버지, 영국의 컴퓨터 과학자

WorldWideWeb(월드와이드웹) 하이퍼텍스트 시스템 고안 – cern(유럽 입자 물리연구소)에서 개발됨.

URL, HTML, HTTP 최초 설계, W3C 창립

# 웹 스크레이핑 = 웹 크롤링

## ❖ BeautifulSoup 라이브러리(모듈)

HTML과 XML 문서를 파싱하기 위한 파이썬 라이브러리이다.

웹 서버로 부터 HTML 소스코드를 가져온 다음에는 HTML 태그 구조를 해석하기 위한 과정이 필요하다.

HTML 소스 코드를 해석하는 것을 **파싱(parsing)**이라고 부른다.

### ▶ BeautifulSoup 설치

```
pip install BeautifulSoup4
```

### ▶ BeautifulSoup 사용

```
from bs4 import BeautifulSoup
```

# 웹 스크레이핑 = 웹 크롤링

## ❖ BeautifulSoup 라이브러리(모듈)

- ✓ `soup.find(태그)`  
처음 나오는 태그로 찾기
- ✓ `soup.find_all(태그)`  
태그에 해당하는 모든 요소 찾아서 리스트로 반환함
- ✓ `soup.find(태그, attrs={'class': css_selector})`  
태그에 해당하는 선택자로 찾기

# 선택자(Selector)

## 선택자(Selector)란?

스타일의 속성을 적용하는 요소를 '선택자(selector)'라고 부른다. 이 선택자는 태그 하나가 될 수도 있지만 여러 개의 요소를 묶어 별도의 선택자로 지정할 수도 있다.

- 태그 선택자

문서에서 특정 태그를 사용한 모든 요소에 스타일 적용

- id 선택자, class 선택자

- ✓ id 선택자 : 문서 안에서 **한 번만 사용**한다면 id 선택자로 정의

- #(샷)** 다음에 id 이름 지정

- ✓ class 선택자 : 문서 안에서 **여러 번 반복할 스타일**이라면 클래스 선택자로 정의.

- 마침표(.)** 다음에 클래스 이름 지정

# 서울시청 웹 크롤링하기

## ✓ 메뉴 글자 수집하기



## 웹 브라우저 > 우클릭 > 검사(단축키 F12)





# 서울시청 웹 크롤링하기

✓ 메뉴 글자 수집하기

```
import requests
from bs4 import BeautifulSoup

url = "https://www.seoul.go.kr/main/index.jsp"
response = requests.get(url)
html = BeautifulSoup(response.text, 'html.parser')
# print(html)

# find()로 찾기
first_li = html.find('li', attrs={'class': 'public'})
print(first_li)
print(first_li.text)
```

```
<li class="public">
<a href="//yeyak.seoul.go.kr" onclick="action_logging({tr_code:'service01'});" target="_blank"
</a>
</li>
```

공공서비스예약

# 서울시청 웹 크롤링하기

## ✓ 메뉴 글자 수집하기

```
# find_all()로 찾기
div = html.find('div', attrs={'class': 'm_service'})
all_li = div.find_all('li')
# print(all_li)

v for li in all_li:
    print(li.text)

print(all_li[1].text)
```

공공서비스예약

응답소(민원신고)

서울일자리

# 웹 스크레이핑 = 웹 크롤링

## ❖ 주요 검색 함수

- ✓ `select_one(태그이름.선택자이름)`  
첫번째 요소로 찾기
- ✓ `select(태그이름.선택자이름 > 하위 태그)`  
태그에 해당하는 모든 요소 찾기

# 서울시청 웹 크롤링하기

✓ 메뉴 글자 수집하기

```
# select_one()으로 찾기
first_li = html.select_one('li.public')
print(first_li)
print(first_li.text)

# select()로 찾기
all_li = html.select('div.m_service ul li')

for li in all_li:
    print(li.text)

print(all_li[1].text)  #응답소(민원신고)
```

```
<li class="public">
<a href="//yeyak.seoul.go.kr" onclick="action_logging({tr_code:'service
</a>
</li>
```

공공서비스예약

# 실습1. 국립중앙박물관 관람 정보

- 국립 중앙 박물관 관람 정보

홈페이지 : <https://www.museum.go.kr/>

관람 정보 > 관람 안내 > 검사(F12)

## 관람시간

월, 화, 목, 금, 일요일: 10:00 ~ 18:00 (입장 마감: 17:30)

수, 토요일: 10:00 ~ 21:00 (입장 마감: 20:30)

· 옥외 전시장(정원)은 오전 7시부터 오후 10시까지 관람하실 수 있습니다.

## 관람료

무료

상설전시관, 어린이박물관, 무료 특별전시 해당

유료

유료 특별전시 해당

관람권 구입하는 곳: 특별전시실 1 앞 매표소

관람권 판매시간: 관람 종료 30분 전까지

# 실습1. 국립중앙박물관 관람 정보

- 국립 중앙 박물관 관람 정보

```
url = "https://www.museum.go.kr/MUSEUM/contents/M0101000000.do?menuId=tour-guidance"
response = requests.get(url)
html = BeautifulSoup(response.text, 'html.parser')

# 관람안내
# select_one()
first_ul = html.select_one('ul.display-content')
print(first_ul)
print(first_ul.text)
```

# 실습1. 국립중앙박물관 관람 정보

- 국립 중앙 박물관 관람 정보

```
# select()로 찾기
contents = html.select('ul.display-content-area > li > ul')
print(contents)

for content in contents:
    | | print(content.text)

# 관람시간
print(contents[0].text)

# 휴관일 및 휴실일
print(contents[1].text)

# 관람료
print(contents[2].text)
```

# KBS 뉴스 기사

## ➤ 뉴스 기사 크롤링하기

홈페이지 : <https://news.kbs.co.kr/>

KBS > 뉴스 > 메인 기사

### 이 대통령 “다주택자보다 집값 폭등 고통받는 국민 더 배려받아야”

입력 2026.02.04 (07:19) | 수정 2026.02.04 (08:27)



AI 요약





# KBS 뉴스 기사

## ➤ 뉴스 기사 크롤링하기

```
# 메인 기사 스크랩
url = "https://news.kbs.co.kr/news/pc/view/view.do?ncd=8295309"
response = requests.get(url)
html = BeautifulSoup(response.text, 'html.parser')

# 제목 스크랩
title = html.select_one("h4.headline-title")
print(title)
print(title.text)

# 내용 스크랩
content = html.select_one('div.detail-body')
print(content.text.strip())
```

# KBS 뉴스 기사

## ➤ 뉴스 기사 크롤링하기

### ■ 제목 스크랩

```
<h4 class="headline-title">트럼프 “4일부터 각국에 관세 서한 보낼 것”</h4>  
트럼프 “4일부터 각국에 관세 서한 보낼 것”
```

### ■ 내용 스크랩

[앵커] 다음 주 상호 관세 유예 종료를 앞두고 트럼프 대통령이 4일부터 각국에 관세 서한을 보낼 거라고 말했습니다. 복잡하다며 간단한 거래를 하는 편이 낫다는 겁니다. 워싱턴 김지숙 특파원입니다. [리포트] 상호 관세 유예 종료를 앞둔 트럼프 대통령은 내일부터 각국에 관세 서한을 보낼 거라고 했습니다. 170여 개국과 상대하고 있는데 얼마나 많은 합의를 할 수 있겠습니까. [도널드 트럼프/미국 대통령/화면출처:폭스TV 유튜브 : "아마 내일부터 여러 나라에 편지를 보내기 시작할 것인데, 그 편지에는 미국에서 사업을 하려면 얼마를 내야 하는지가 적혀 있을 것입니다."] 스콧 베센트 미 재무장관도 합의를 이끌어낼 수 있다고 생각해 마지막 순간까지 기다리는 국가들은 기존에 책정된 상호 관세율이 적용됩니다. /미 재무장관/CNBC 인터뷰 : "제가 다른 언론 인터뷰에서도 경고했듯, 이 국가들은 조심해야 합니다. 왜냐하면 그들의 이익이 있기 때문입니다."] 상호 관세 유예 연장 가능성에 대해선 결승선을 통과해야 할 시점에 공개적으로 연장하겠다고 말하며 모호성을 견지하면서, 각국에 미국과의 합의를 서두를 것을 압박한 걸로 풀이됩니다. 그러면서 유예 기간이 끝나기

# KBS 뉴스 기사

## ➤ 대표 이미지 저장하기

+ 헤드라인 대표 이미지 가져오는 방법을 알려줘



```
# 이미지 경로
img_src = thumbnail.get("src")
print(img_src)

# 전체 url
base_url = "https://news.kbs.co.kr"
img_url = base_url + img_src
print(img_url)

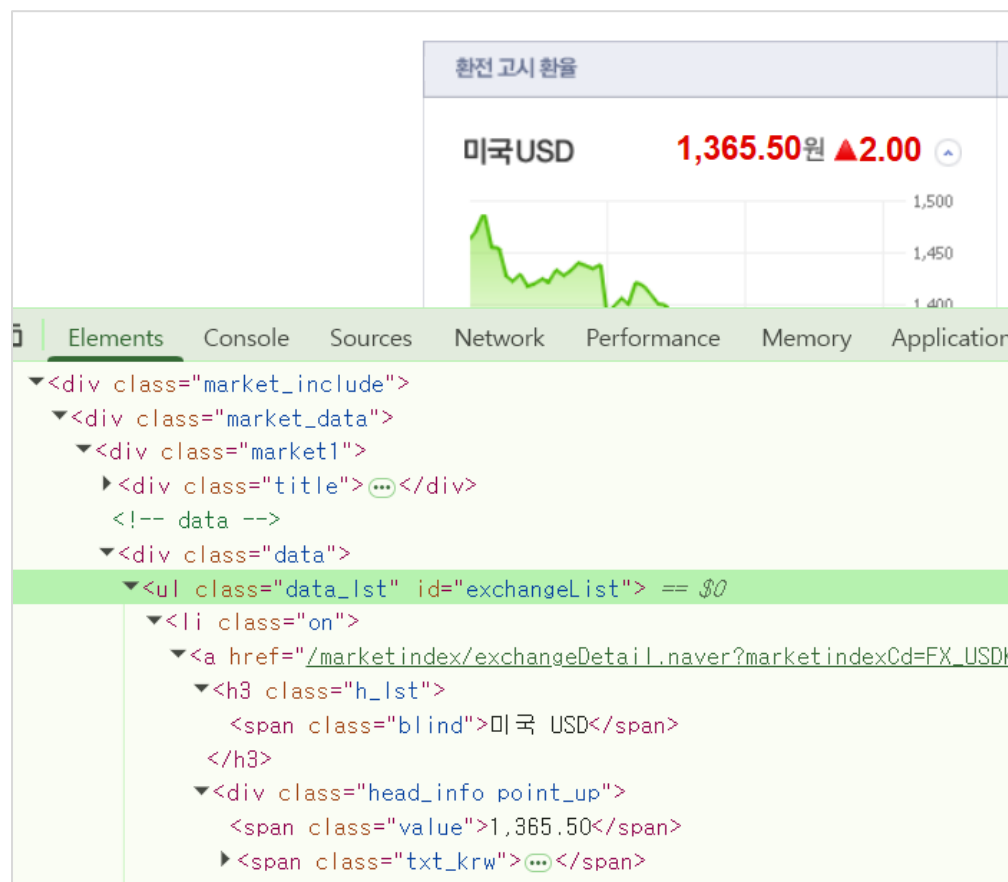
# 사진 저장하기
img_data = requests.get(img_url).content
# print(img_data)

with open("kbs_thumbnail.jpg", "wb") as f:
    f.write(img_data)
print("대표 이미지 저장 완료!")
```

# 네이버 금융 크롤링하기

## ● 환율정보 수집하기

네이버 > 증권 > 시장지표 > 환전 고시 환율



The screenshot shows the Naver Finance '환전 고시 환율' (Exchange Rate) page. The main content displays the USD exchange rate as 1,365.50 KRW, up by 2.00. Below this is a line chart showing the historical trend. The browser's developer tools are open to the 'Elements' tab, showing the DOM tree. The selected element is a list of exchange rates, with the USD entry highlighted. The DOM structure is as follows:

```
<div class="market_include">
  <div class="market_data">
    <div class="market1">
      <div class="title">...</div>
      <!-- data -->
      <div class="data">
        <ul class="data_lst" id="exchangeList"> == $0
          <li class="on">
            <a href="/marketindex/exchangeDetail.naver?marketindexCd=FX_USD">
              <h3 class="h_lst">
                <span class="blind">미국 USD</span>
              </h3>
              <div class="head_info point_up">
                <span class="value">1,365.50</span>
                <span class="txt_krw">...</span>
              </div>
            </a>
          </li>
        </ul>
      </div>
    </div>
  </div>
</div>
```

USD 1,366.30  
JPY(100엔) 944.39  
EUR 1,607.52  
CNY 190.57

# 네이버 금융 크롤링하기

- 환율정보 수집하기 – find() 사용하여 첫번째 환율 찾기

```
resp = requests.get("https://finance.naver.com/marketindex/")
soup = BeautifulSoup(resp.text, "html.parser")

# find()
first_ul = soup.find('ul', attrs={'class': 'data_lst'})
# print(first_ul)
first_li = first_ul.find('li')
print(first_li)

# 환율 종류
exchange = first_ul.find('span', attrs={'class': 'blind'})
print(exchange.text)
print(exchange.text.split(" ")[1]) #미국 USD
```

# 네이버 금융 크롤링하기

- 환율정보 수집하기 – find\_all() 사용

```
# 환율 지수
value = first_ul.find('span', attrs={'class': 'value'})
print(value.text) #1,366.60
print(exchange.text, value.text)

# find_all() - 전체 환율 찾기
all_li = first_ul.find_all('li')
# print(all_li)

for li in all_li:
    exchange = li.find('span', attrs={'class': 'blind'})
    value = li.find('span', attrs={'class': 'value'})
    print(exchange.text.split(" ")[-1], value.text)
```

# 네이버 금융 크롤링하기

- 환율정보 수집하기 – select() 사용

```
all_li = soup.select("div.market1 ul li")
# all_li = soup.select("ul.data_lst li") #차이 비교
# print(all_li)

# 환율 종류 - select_one() : 1개 선택
exchange = soup.select_one("span.blind")
# print(exchange.text) #미국 USD

# 환율 지수
value = soup.select_one("span.value")
# print(value.text) #1,388.80

# 환율 전체 출력
for li in all_li:
    exchange = li.select_one("span.blind")
    value = li.select_one("span.value")
    # 공백문자로 텍스트 분리 - 리스트 반환
    # text 대신 string 가능
    print(exchange.string.split(' ')[-1], value.string)
```