Simple Cralwer

개요

패키지 준비

`bs4` 패키지의 설치가 필요함

```
$ pip install --upgrade beautifulsoup4
```

수집 대상 URL

https://data.hossam.kr/sample.html

코드 구현

```
# 필요한 모듈 참조
import requests
from bs4 import BeautifulSoup
```

```
# 브라우저 버전정보
userAgent = "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
```

```
# 접속객체 생성
session = requests.Session()

# 접속객체에 부가정보(header) 삽입하기
session.headers.update({
    "Referer": "",
    "User-Agent": userAgent
})
```

```
# 수집할 컨텐츠가 있는 웹 페이지의 주소
url = "https://data.hossam.kr/sample.html"
```

```
# 생성한 접속객체를 활용하여 API에 접속
r = session.get(url)

# 접속에 실패한 경우
if r.status_code != 200:
    # 에러코드와 에러메시지 출력
    msg = "[%d Error] %s 에러가 발생함" % (r.status_code, r.reason)
```

```
    # 에러를 강제로 생성시킴
    raise Exception(msg)
```

```
# 인코딩 형식 지정하여 beautifulsoup 객체를 생성
r.encoding = "utf-8"
#print(r.text)
soup = BeautifulSoup(r.text)
soup
```

```
<!DOCTYPE html>

<html lang="en">
<head>
<meta charset="utf-8"/>
<meta content="width=device-width, initial-scale=1.0" name="viewport"/>
<title>Document</title>
<style>
        /* HTML 태그 방식 */
        h1 { color: #f0f; }
        h2 { color: #06f; }

        /** Class 방식 */
        .myclass { color: #f00; }

        /** id 방식 */
        #myid { color: #f60; }

        /** 자식 선택자  */
        .syllabus > li > ol > li {
            text-decoration: underline;
        }

        /** 자손 선택자 */
        .syllabus ol {
            font-weight: bold;
        }

        .part1 {
            background-color: #eeeeee;
        }

        .part2 {
            background-color: #d5d5d5;
        }

        /** 특정 대상을 구체적으로 명시  */
        div.sub.part1 {
            border: 1px dotted #000;
        }

        div.sub.part2#hello {
            border: 1px solid #555;
        }

        /** 특정 속성을 갖고 있는 요소  */
```

```
        a[href] {
            font-size: 20px;
        }

        /** 특정 속성 값에 대한 적용  */
        a[href='#'] {
            color: green;
        }
    </style>
</head>
<body>
<h1>Hello World</h1>
<a>link0</a>
<a href="#">link1</a>
<a href="https://www.naver.com">link2</a>
<h2 id="myid">Python</h2>
<div class="sub part1">
<ul class="syllabus">
<li>변수와 데이터 타입</li>
<li class="myclass">연산자</li>
<li>
                연속성 자료형
                <ol>
<li>리스트(list)</li>
<li>딕셔너리(dict)</li>
<li>집합(set)</li>
</ol>
</li>
<li>프로그램 흐름제어</li>
<li>함수</li>
</ul>
</div>
<h2>Data Analysis</h2>
<div class="sub part2" id="hello">
<ul>
<li>데이터 수집</li>
<li class="myclass">데이터 전처리</li>
<li>
                탐색적 데이터 분석
                <ol class="myclass">
<li>기초통계</li>
<li>데이터 시각화</li>
</ol>
</li>
<li>확증적 데이터 분석</li>
<li>데이터 마이닝</li>
</ul>
</div>
</body>
</html>
```

soup 객체로 부터 원하는 부분 추출하기

1) HTML 태그에 의한 추출

soup 객체의 `select()` 메서드의 리턴값은 항상 리스트 타입

```
test1 = soup.select("h1")
test1
```

```
[<h1>Hello World</h1>]
```

리스트 유형 이므로 인덱스 번호를 통한 접근이 가능

```
h1 = test1[0]
h1
```

```
<h1>Hello World</h1>
```

추출한 태그에서 내용만 추출

외부에서 추출한 내용은 앞뒤 여백 제거 필수

```
result1 = h1.text.strip()
result1
```

```
'Hello World'
```

2) class에 의한 추출

```
test2 = soup.select(".myclass")
test2
```

```
[<li class="myclass">연산자</li>,
 <li class="myclass">데이터 전처리</li>,
 <ol class="myclass">
 <li>기초통계</li>
 <li>데이터 시각화</li>
 </ol>]
```

```
for i, v in enumerate(test2):
    # 추출한 요소가 하위 태그를 포함하는 경우 그 안의 텍스트만 일괄 추출
    print("%d번째 요소 : %s" % (i, v.text.strip()))
```

```
0번째 요소 : 연산자
1번째 요소 : 데이터 전처리
2번째 요소 : 기초통계
데이터 시각화
```

하위 요소 추출하기

`select()` 메서드로 추출한 요소를 활용하여 그 하위요소를 추가적으로 추출할 수 있다.

```
li = test2[2].select("li")
li
```

```
[<li>기초통계</li>, <li>데이터 시각화</li>]
```

```
for i in li:
    print(i.text.strip())
```

```
기초통계
데이터 시각화
```

3) id에 의한 추출

`id` 값은 해당 웹페이지 안에 단 하나만 존재하기 때문에 반복문을 적용할 필요는 없다.

```
test3 = soup.select("#myid")
test3
```

```
[<h2 id="myid">Python</h2>]
```

```
print(test3[0].text.strip())
```

```
Python
```