26.Python Scrapy I

1. Python 고급 Crawling 기술 Scrapy

1. **Scrapy 개념**

1) 함수와 코드를 미리 작성해놓은 Crawling Framework .

- 특정 함수를 특정 위치에 어떻게 사용해야 하는지, 작성해야 하는지 정해놓은 프로그램

2) 처음에는 난해하지만, 사용법을 알면 BeautifulSoup 라이브러리보다 편함

3) 크롤링을 좀더 안정적으로 할 수 있음(Scrapy 내부에서 다양한 안정장치)

4) 크롤링을 좀더 빠르게 할 수 있음(크롤링 프로그램을 동시에 여러개 실행시켜서 많은 량의 데이터 크롤링시, 시간 단축)

5) 다양한 크롤링 관련 기능(크롤링한 데이터를 다양한 포멧으로 저장도 가능)

2 **Scrapy** 사용 Process

1) 실제 크롤링할 스파이더(spider, scrapy 기반 크롤링 프로그램) 생성

2) 크롤링할 사이트(시작점)와 크롤링할 아이템(item)에 대한 selector 설정

3) 크롤러 실행

3. Windows 기반 **Scrapy** 설치

1) 기본 설치

- pip install scrapy

2) 윈도우에서 정상 설치가 안될 시 Upgrade

pip install --upgrade setuptools

pip install pypiwin32

pip install twisted[tls]

3) 2번 으로도 안되면

https://visualstudio.microsoft.com/ko/downloads/

2. Python 고급 Crawling 기술 Scrapy Project 생성1

1. Terminal Open
   - Key Board로 명령어를 내릴수 있는 Program

2. **scrapy Project 생성**

1) C:₩Users₩ktg₩python-env>virtualenv virtScrapy # 가상 환경 만들기
2) virtScrapy₩Scripts₩activate.bat → (virtScrapy) C:₩Users₩ktg₩python-env> # 가상 환경 전환됨
3) (virtScrapy) C:₩Users₩ktg₩python-env>pip install scrapy # 가상 환경 에서 scrapy Framework 설치하기
4) (virtScrapy) C:₩Users₩ktg₩python-env>cd C:₩PyCharmProject₩Sources # 가상 환경 에서 만들 Project로 이동
5) (virtScrapy) C:₩PyCharmProject₩Sources>scrapy startproject ktgScrapy # 가상 환경 에서 Project 만들기
6) spider (크롤러이름) 생성 방법
   - 크롤링 프로젝트 내에, 여러 가지 크롤러(scrapy에서는 spider라고 함) 있을 수 있으므로, 각 크롤러의 이름을 지정

[1] scrapy genspider 이용

① (virtScrapy) C:₩PyCharmProject₩Sources>cd C:₩PyCharmProject₩Sources₩ktgScrapy₩ktgScrapy

② (virtScrapy) C:₩PyCharmProject₩Sources₩ktgScrapy₩ktgScrapy>scrapy genspider gmarket1 www.gmarket.co.kr
(virtScrapy) C:₩PyCharmProject₩Sources₩ktgScrapy₩ktgScrapy>scrapy genspider gmarket2 www.gmarket.co.kr
(virtScrapy) C:₩PyCharmProject₩Sources₩ktgScrapy₩ktgScrapy>scrapy genspider gmarket3 www.gmarket.co.kr
(virtScrapy) C:₩PyCharmProject₩Sources₩ktgScrapy₩ktgScrapy>
scrapy genspider gmarket5 corners.gmarket.co.kr/Bestsellers

[2] pycharm 으로 작성

① ktgScrapy₩ktgScrapy/spiders 디 렉토리에 gmarket1,2,3.5,6 py 파일(템플릿)이 생김

② 직접 scrapy genspider 명령을 사용하지 않고, 만들어도 됨

2. Python 고급 Crawling 기술 Scrapy Project 생성2

3. spider (크롤러) 실행

1) (virtScrapy) C:₩PyCharmProject₩Sources₩ktgScrapy₩ktgScrapy>scrapy crawl gmarket

(virtScrapy) C:₩PyCharmProject₩Sources₩ktgScrapy₩ktgScrapy>scrapy crawl gmarket1

(virtScrapy) C:₩PyCharmProject₩Sources₩ktgScrapy₩ktgScrapy>scrapy crawl gmarket2

(virtScrapy) C:₩PyCharmProject₩Sources₩ktgScrapy₩ktgScrapy>scrapy crawl gmarket3

(virtScrapy) C:₩PyCharmProject₩Sources₩ktgScrapy₩ktgScrapy>scrapy crawl gmarket5

2) 다양한 데이터 포멧으로 아이템 저장하기(gmarket5 예시)

① csv, xml, json 포멧

② 터미널 환경에서, ecommerce 폴더에서 다음 명령

scrapy crawl 크롤러명 -o 저장할파일명 -t 저장포멧

(virtScrapy) C:₩PyCharmProject₩Sources₩ktgScrapy₩ktgScrapy>scrapy crawl gmarket5 -o gmarket5.csv -t csv

(virtScrapy) C:₩PyCharmProject₩Sources₩ktgScrapy₩ktgScrapy>scrapy crawl gmarket5 -o gmarket5.xml -t xml

json 파일을 확인하면, 한글문자가 깨져나옴

(virtScrapy) C:₩PyCharmProject₩Sources₩ktgScrapy₩ktgScrapy>scrapy crawl gmarket5 -o gmarket5.json -t json

3. Python 고급 Crawling 기술 Scrapy 기본 구조1

1. Scrapy 기본 Template 구조

1) 클래스 이름은 마음대로 정하면 됨, 단 scrapy.Spider 를 상속

2) name이 크롤러(spider)의 이름

3) allowed_domains는 옵션 (삭제해도 무방함)

- 별도 상세 설정으로 허용된 주소 이외의 주소는 크롤링 못하게끔 하는 기능을 위한 변수

4) start_urls 가 중요함. 크롤링할 페이지 주소를 나타냄.

- parse 함수는 클래스의 메서드로 response를 반드시 인자로 받아야 함
- response에 start_urls 에 기록된 주소의 크롤링 결과가 담아 오기 때문

5) response 확인하기

- response.text 에 크롤링된 데이터가 담겨져 있음

2. gmarket.py 예시 문장

```
# -*- coding: utf-8 -*-
import  scrapy


class GmarketSpider(scrapy.Spider):
    name = 'gmarket'
    allowed_domains = ['www.gmarket.co.kr']
    start_urls = ['http://www.gmarket.co.kr/']

    def parse(self, response):
        pass
```

3. Python 고급 Crawling 기술 Scrapy 기본 구조2

1. Scrapy 기본 Template 구조 , start_urls 상세

1) start_urls는 리스트로 크롤링할 주소를 여러개 쓰는 것 가능

2) start_urls 동작 방식

① start_urls에서 주소를 하나씩 가져와서 크롤링

② response 에 넣고, parse 함수를 호출

③ parse 함수에 response에 담아져있는 크롤링 결과를 원하는 대로 처리

2. gmarket.py 예시 문장

```
# -*- coding: utf-8 -*-
import  scrapy

class GmarketSpider(scrapy.Spider):
    name = 'gmarket'
     start_urls = ['https://corners.gmarket.co.kr/Bestsellers/', 'https://sports.v.daum"]


    def parse(self, response):
        pass
```

3. Python 고급 Crawling 기술 Scrapy 기본 구조3

1. Scrapy 기본 Template 구조 , response 상세

1) response.css(): css selector 로 데이터 가져오기

① response.css('head > title').get()

② response.css('head > title').getall()

③ response.css('head > title::text').get()

④ gmarket 베스트 상품의 타이틀 가져오기

- response.css('div.best-list li > a::text').getall()

⑤ gmarket 베스트 상품의 특정 순번째의 타이틀 가져오기

- response.css('div.best-list li > a::text')[1].get()

2) response.xpath(): xpath 로 데이터 가져오기

① response.xpath('//div[@class="best-list"]/ul/li/a').getall()

② response.xpath('//div[@class="best-list"]/ul/li/a/text()').getall()

3) re(): 정규표현식 쓰기

① response.css('div.best-list li > a::text')[1].re('(₩w+)')

② response.xpath('//div[@class="best-list"]/ul/li/a/text()')[1].re('(₩w+)')

4) 예시

- 터미널 환경에서, ecommerce 디렉토리에서 scrapy crawl gmarket 명령
- scrapy genspider 로 spider 작성하기

scrapy genspider gmarket http://corners.gmarket.co.kr/Bestsellers

2. gmarket.py 예시 문장

```
-*- coding: utf-8 -*-
import scrapy
class GmarketSpider(scrapy.Spider):
    name = 'gmarket'
    allowed_domains =
            ['http://corners.gmarket.co.kr/Bestsellers']
     start_urls =
            ['http://corners.gmarket.co.kr/Bestsellers/']

     def parse(self, response):
         titles = response.css('div.best-list li > a::text').getall()
         for title in titles:
              print (title)
```

3. Python 고급 Crawling 기술 Scrapy 기본 구조4

1. 크롤링 데이터 다루기: **저장**하기

1) items.py 를 작성

① 크롤링하고자 하는 데이터를 아이템(item)으로 선언

② 클래스를 만들고, scrapy.Item을 상속받고, 아이템의 이름을 만들고, scrapy.Field()를 넣어줘야 함

2) gmarket.py 수정

- yield 명령어로 아이템(item)에 저장할 수 있다!
- 아이템 클래스 생성 및 아이템 저장

① 선언: from 프로젝트이름.items import 아이템클래스명

② 클래스 생성: item = 아이템클래스명()

③ 아이템 저장 : item['아이템명'] = 아이템데이터

yield item

2. items.py 예시 문장

```
# -*- coding: utf-8 -*-
# Define here the models for your scraped items
#
# See documentation in:
# https://doc.scrapy.org/en/latest/topics/items.html
import scrapy
class KtgscrapyItem(scrapy.Item):
    # define the fields for your item here like:
    # name = scrapy.Field()
    title = scrapy.Field()
```

2. gmarket.py 예시 문장

```
import scrapy
from  ktgScrapy.items import KtgscrapyItem

class GmarketSpider(scrapy.Spider):
    name = 'gmarket'
    allowed_domains = ['http://corners.gmarket.co.kr/Bestsellers']
    start_urls = ['http://corners.gmarket.co.kr/Bestsellers/']

    def parse(self, response):
        titles = response.css('div.best-list li > a::text').getall()
        for title in titles:
            item = EcommerceItem()
            item['title'] = title
            yield item
```

3. Python 고급 Crawling 기술 Scrapy 기본 구조5

1. 크롤링 데이터 다루기: item Data **처리**하기

1) 다양한 데이터 포멧으로 아이템 저장하기

① csv, xml, json 포멧

② 터미널 환경에서, ecommerce 폴더에서 다음 명령

scrapy crawl 크롤러명 -o 저장할파일명 -t 저장포멧

(virtScrapy) C:₩PyCharmProject₩Sources₩ktgScrapy₩ktgScrapy>scrapy crawl gmarket3 -o gmarket3.csv -t csv

(virtScrapy) C:₩PyCharmProject₩Sources₩ktgScrapy₩ktgScrapy>scrapy crawl gmarket3 -o gmarket3.xml -t xml

json 파일을 확인하면, 한글문자가 깨져나옴

(virtScrapy) C:₩PyCharmProject₩Sources₩ktgScrapy₩ktgScrapy>scrapy crawl gmarket3 -o gmarket3.json -t json

2) settings.py 수정

- 문자의 유니코드 문제는 항상 존재!

FEED_EXPORT_ENCODING = 'utf-8'

- 기존 json 파일을 지우고! 다시 다음 명령을 실행하면, 한글문자가 정상적으로 저장되는 것을 확인

(virtScrapy) C:₩PyCharmProject₩Sources₩ktgScrapy₩ktgScrapy>scrapy crawl gmarket3 -o gmarket3.json -t json

3. Python 고급 Crawling 기술 Scrapy 기본 구조6

1. 크롤링 데이터 다루기: item Data **후처리** 하기

1) item Data **후처리** 필요

① 일부 아이템은 저장하지 않음

② 중복되는 아이템을 저장하지 않음

③ 데이터베이스등에 저장

④ 특별한 포멧으로 아이템을 저장

⑤ 프레임워크 스타일로 구조화시키기 위해 별도 파일에 작성할 수 있도록 하였음 (parse 함수에서 작성해도 됨)

2) pipelines.py

- 아이템이 저장되려 할 때마다, pipelines.py의 process_item 함수를 호출

4. Python 고급 Crawling 기술 Crawler(Spider)1

1. 우선 gmarket.py 에서 상품과 가격을 모두 가져와보도록 코드를 수정

```
import scrapy
from  ktgScrapy.items import KtgscrapyItem

class Gmarket5Spider(scrapy.Spider):
    name = 'gmarket5'
    allowed_domains = ['corners.gmarket.co.kr/Bestsellers']
    start_urls = ['http://corners.gmarket.co.kr/Bestsellers/']

    def parse(self, response):
        titles = response.css('div.best-list > ul > li[id] > a::text').getall()
        prices = response.css('div.best-list > ul > li[id] > div.item_price > div.s-price > strong > span > span::text').getall()
        for i in range(len(titles)):
            item = KtgscrapyItem()
            item['title'] = titles[i]
            item['price'] = int(prices[i].replace("원", "").replace(",", ""))
            yield item
```

2. pipelines.py 사용을 위해 별도 설정 필요

여러 Class 있을경우 숫자가 낮을수록 먼저 수행

우선순위 번호 : 0~1000 숫자중 임의의 숫자 부여

스파이더 실행후 Enable Item Pipilines를 터미날에서 확인 , 설정 적용 여부 확인 가능

2. settings.py 수정 예시 문장

```
# Obey robots.txt rules
    ROBOTSTXT_OBEY = False
# 한글 Setting
  FEED_EXPORT_ENCODING = 'utf-8'
  ITEM_PIPELINES = {
        'ktgScrapy.pipelines.KtgscrapyPipeline': 300,
  }
```

3. pipeline.py 작성

- 각 아이템 생성시, pipeline.py 에 있는 process_item 함수를 거쳐가게 되어 있음
- 필요한 아이템만 return 해주고, 필터링할 아이템은 DropItem 을 통해, 더이상의 아이템처리를 멈추게 해줘야 함

4. Python 고급 Crawling 기술 Crawler(Spider)2

3. pipeline.py 예시

```
from scrapy.exceptions import DropItem
# 각 아이템 생성시, pipeline.py 에 있는 process_item 함수를 거쳐가게 되어 있음
class KtgscrapyPipeline(object):
    def process_item(self, item, spider):
        print (item)
        if item['price'] > 10000:
            return item
        else:
            raise DropItem("drop item having lower price than 10000")
```