<파이썬을 이용한 실전 웹크롤링>

1. 웹 크롤링이란?

- 컴퓨터 소프트웨어 기술을 활용하여 웹 상의 정보를 자동적으로 수집하고 정형 데이터 형식으로 추출하는 행위

# 정형데이터란?

- 스프레드시트와 같은 고정된 틀에 저장된 데이터. 행, 열이 지정된 형태

# 비정형데이터

- 페이스북, 트위터, 유튜브 영상, 음원, 이미지 등 고정된 형태가 아닌 데이터 타입

2. 파이썬 코드 오류 대처

- 오류발생 코드의 빨간 부분을 구글검색.

- 많은 오류들은 stackoverflow 사이트에서 유사상황 검색 가능

- 패키지명을 검색/GITHUB사이트 접속/오류내용검색 및 문의

3. 반복문/조건문/제어문

1) for/while : 조건을 만족할 때까지 코드내용을 반복 수행. 조건만족시 바로 탈출

|  |  |
| --- | --- |
| for number in range(5): # 0~4를 number 입력  #for문 실행  print(number) #number 반복수행 결과 출력  >>>  0  1  2  3  4 | n=0 # 변수 n초기값 = 0  while n<6: # n이 6보다 작을 때까지 수행  print(n) #n값 출력  n=n+1 #n값에 1을 더한 값을 반환  >>>  0  1  2  3  4  5 |

2) if/else, elif(여러 개의 조건문 코드를 입력할 경우)

‘~라면, 그 밖의 경우에는’

|  |
| --- |
| n=7 # n 초기값은 7  if n>5: # 조건문 n값이 5보다 크다면  print('n is bigger than 5') #조건을 만족할 경우 반환되는 문구  else: #if 조건문에 해당되지 않을 경우  print('n is smaller than 5') #if 조건 만족하지 않을 경우 반환되는 문구  >>> n is bigger than 5 |

3) continue, pass

Continue : 반복문 내에서 continue 이하의 코드를 읽지 않고 반복문 시작코드로 이동.

Pass : 반복문 내에서 진행대로 코드 읽게 함. 설명 상 없어도 되는 명령어로 이해됨.

4. 외장함수

1) os모듈 : diretory확인 및 변경

|  |
| --- |
| import os #os모듈 가져오기  current\_directory=os.getcwd() #current 변수에 경로 저장  print(current\_directory) #변수 출력  >>> C:\Users\gusdu |

2) time모듈: 현재 시간을 출력

|  |
| --- |
| import time #time모듈 가져오기  time.time() #time모듈의 time함수실행  >>> 1532274578.1568818 #현재시간이지만 읽을 수 없음  time.localtime(time.time()) #볼 수 있는 현재 시간으로 반환  >>> time.struct\_time(tm\_year=2018, tm\_mon=7, tm\_mday=23, tm\_hour=0, tm\_min=49, tm\_sec=41, tm\_wday=0, tm\_yday=204, tm\_isdst=0)  time.strftime('%Y%m%d', time.localtime(time.time())) #연도/월/일 형태로 시간 반환  >>>'20180723' |

3) collection모듈 : 반복 문자, 숫자 등의 횟수

|  |
| --- |
| from collections import Counter #collection 모듈의 counter함수 실행  list\_ct=['a','b','c','d','d','d','c','a','f'] #list변수  apb\_count=Counter(list\_ct) # apb변수에 count한 결과 저장  apb\_count #apb변수 내용 출력  >>> Counter({'a': 2, 'b': 1, 'c': 2, 'd': 3, 'f': 1}) |

4) RE 모듈(정규표현식) - 데이터 중 반복되는 패턴의 구조를 표현하는 형식 언어

입력형식 : '[정규표현식]정규표현식'

* 숫자, 알파벳, 특수문자를 이용하여 원하는 문자열을 추출하거나 탐색.
* re.compile('[가나다]') / '가or나or다' 글자를 포함하는 문자열 패턴
* re.compile('[가-힣]') / 가~힣 까지 글자를 포함하는 문자열 패턴
* re.compile('[0-9]') / 숫자를 포함하는 문자열 패턴
* re.compile('[a-z]') / a-z를 포함하는 문자열 패턴
* re.compile('[A-Z]') / A-Z를 포함하는 문자열 패턴

[정규표현식].

* '.' : 메타문자. 엔터(/n)를 제외한 모든 문자열을 의미함.
* re.compile('문자열\*') : 문자열 0회이상 반복되는 문자열 패턴.

ex)'라'\* → ' ', '라', '라라', '라라라'

* re.compile('문자열+') : 문자열 1회이상 반복되는 문자열 패턴.

ex)'라'+ → '라', '라라', '라라라'

* re.compile('문자열{start, end}'

ex)[가-힣]{2,3} → 가~힣 사이의 문자열 둘or셋을 포함하는 패턴

* re.compile('문자열|문자열') (bar\_shift + \) : or
* re.compile('^문자열') : 문자열과 맨 처음이 같은 문자열 탐색
* re.compile('문자열$') : 문자열과 맨 끝이 같은 문자열 탐색
* re.compile('/b문자열/b)': 문장내에서 문자열과 일치하는 내용만 탐색.

ex) '/b지승/b' -- 지승 -- 만 탐색.

* 변수.search() 결과내 같은 부분이 있다면 추출
* 변수.match() 결과내 일치하는 부분이 있을 때만 추출

|  |
| --- |
| test1 = ['가지','나비','라디오'] #test 변수 선언  ex1 = re.compile('[가나다]') #ex1변수에 정규표현식 저장  print('='\*10,'예제1','='\*10)  for test in test1: #test1에 해당하는 내용을 test에 입력  print(ex1.match(test)) #ex1정규표현식 조건과 test내용의 일치여부 결과  >>> <\_sre.SRE\_Match object; span=(0, 1), match='가'>  <\_sre.SRE\_Match object; span=(0, 1), match='나'>  None |

* findall() : 탐색 결과를 [('문자열')]형태로 반환

>>> [('이기성', '1'), ('장병국', '2'), ('이희정', '1')]

* finditer() : 탐색결과를 열(series)구조로 반출

>>>이기성

장병국

이희정

* group() : 탐색된 패턴의 문자열 추출. findall의 결과에서 이름은 group(1), 숫자는 group(2)로 표현.

5. 내장함수 특징

- 문자열간의 덧셈 ex)'안+지승' = 안지승

- 문자열과 숫자의 곱셈 ex)'해'\*3 = 해해해

- 문자열에 index포함되어 문자열 자르기 가능

ex)'스위트케이'[0:3] = '스위트'

- 문자열.strip() : 양쪽 공백을 제거하는 함수

|  |
| --- |
| a = " hi "  a.strip()  >>>'hi' |

- 문자열.replace(a,b) : 문자열 안의 a를 b로 치환하는 함수

|  |
| --- |
| a = "Life is too short"  a.replace("Life", "Your leg")  >>>'Your leg is too short' |

|  |
| --- |
| a = "a:b:c:d"  a.split(':')  >>>['a', 'b', 'c', 'd'] |

- 문자열.split(a) : 'a'를 기준으로 문자열 분할하여 리스트형으로 반환하는 함수

#파이썬을 활용한 데이터분석

1. 설치 및 jupyter notebook실행

첨부파일 참고

한줄씩 결과를 확인할 수 있는 편리함.

1. 유용한 명령어

- Shift + Enter : 코드실행

- 변수명 입력 후 Shift+Tab : 변수정보

- 변수명 ? : 변수에 대한 디테일한 정보 확인가능

- %ls : 현재 파일이 위치한 폴더내 다른 파일 목록 나타남

- %pwd : 현재 파일이 위치한 경로

- %cd : 폴더 이동하는 명령어

ex)%cd.. : 한단계 상위 폴더로 경로 이동

- %cd 폴더명 : 현재경로에서 해당폴더로 경로 이동

- %whos : 입력한 변수 목록, type, 내용 확인하는 명령어

- %history : 지금까지 입력했던 내용 보여주는 명령어

- %timeit : 작업실행시간 표현 명령어

- %matplotlib inline : 그래프 결과 작업 창 내에 표현하도록 하는 명령어

1. 마크다운문서

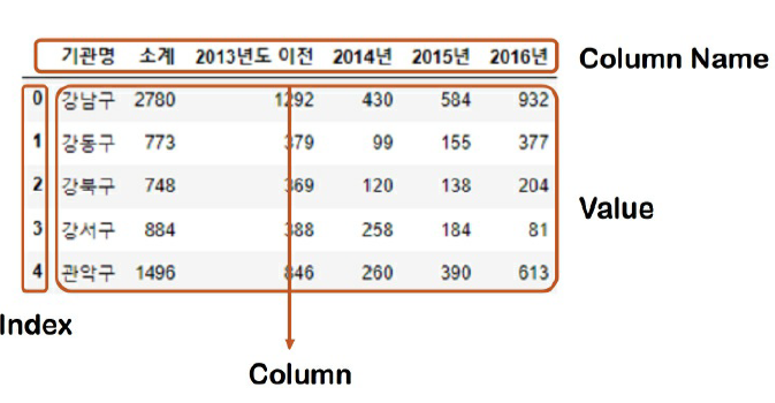
특수문자를 이용하여 텍스트의 서식(크기, 굵기, 기울임), 정렬(들여쓰기, 가운데 정렬 등),을 변경 이미지 삽입 등을 실행하는 환경.

# (#의 개수는 제목의 크기), \* (내용의 목록, 기울기), <(인용부호), ---(가로 구분선)

1. Pandas 모듈

* Pandas는 series(한 줄의 열), dataframe(행렬 2차원 구조) 두 형태를 갖는 데이터 구조.

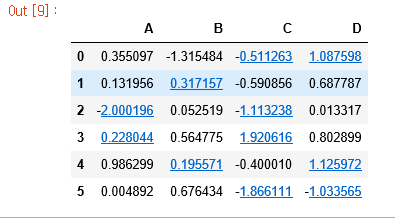
|  |
| --- |
| Import pandas as pd #pandas모듈을 불러오고 모듈을 pd라고 부른다.  col\_names=['구','인구수','남성','여성','내국인','남성내국인','여성내국인','외국인','남성외국인','여성외국인','세대당인구','고령자']  raw\_data=pd.read\_excel('../data/01\_OctagonExcel\_seoul.xls', header=2, usecols="B,D:N", names=col\_names, skipfooter=1) |

Dataframe 구조

* pd.read\_excel('../data/01\_OctagonExcel\_seoul.xls' : Excel파일 읽어오기
* ../ 는 자료가 위치한 상위 폴더를 의미함.
* header=2 : 데이터의 index값 2에 해당하는 행부터 자료 시작.(index 0,1값 제거)
* usecols="B,D:N" : A열(가장 첫번째 열), C열 제외한 나머지 데이터 사용
* skipfooter=1 : 데이터의 가장 마지막 행을 삭제.

1. 기초 명령어

|  |
| --- |
| import pandas as pd #pandas모듈 불러오기, pandas는 pd로 호출  import numpy as np #numpy모듈 불러오기, numpy는 np로 호출  df=pd.DataFrame(np.random.randn(6,4), columns=['A','B','C','D'])  # pandas dataframe을 구성. numpy모듈의 random함수에서 randn을 이용하여 0-1사이 임의의 값으로 6x4행렬 구성. 열의 이름은 ABCD로 함.  df |

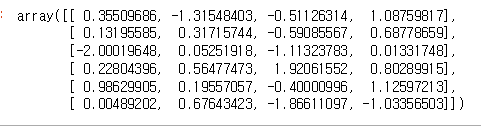


- df.head() : 상위 몇 개의 데이터 출력. 기본값은 5.

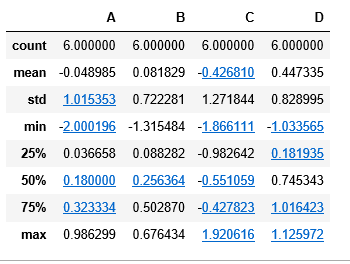
- df.tail() : 하위 몇 개의 데이터 출력. 기본값은 5.

- df.index/columns : 인덱스값, 열 제목값 호출(0-5)

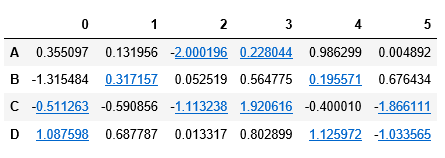
- df.values : ([[ ]])의 배열로 값 출력



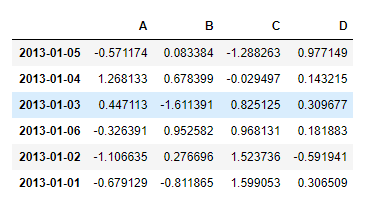
- df.describe() : 빈도, 평균, 분산, 표준편차, 최대값, 4분위 값 출력



-df.T : 행렬 변환. 6x4 4x6으로 변환



* df.sort\_values(by=’C’) : C열을 오름차순으로 정렬



* df.sort\_values(by=’A’, ascending=False) : 내림차순으로 정렬

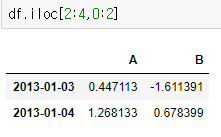
1. Pandas 응용(slice)

: 데이터의 선택적 추출

* df[‘A’] : 앞의 예제에서 A열만 출력. 특정 columns 출력
* df[0:3] : index 0-2 값에 해당하는 데이터 출력. 특정 raw 출력
* df.loc[index[0]] : index[0]에 해당하는 행 출력.
* df.loc[ : , [‘A’, ‘B’]] : 행전체와 A, B열에 해당하는 데이터 출력 ( : 전체를 의미)
* df.iloc[2,3] : 0x0부터 시작되는 데이터구조에서 2행 3열의 값 출력. (iloc : 행이나 열을 순서로 범위지정가능)

|  |
| --- |
|  |

* df.iloc[2:4,0:2] : 0부터 시작되는 행렬 순서 상 행 2,3에 해당하고 열 0,1에 해당하는 값 출력.



* df[df['B']<0] : B열이 0보다 작은 조건에 해당하는 값 출력



* 변수에 값을 복사할 때는 오른쪽 결과처럼 코드를 구성하면 c의 최초값까지 달라진다. 따라서 값을 복사할 때는 copy()옵션을 이용.

|  |  |
| --- | --- |
| c=[1,2,3]  d=c.copy()  d[1]=0  d  >>>[1, 0, 3] | c=[1,2,3]  d=c  d[1]=0  c  >>>[1, 0, 3] |

* 값의 삽입(기본적으로 열로 삽입)

|  |
| --- |
| df['E']=[1,2,3,4,5,6]  df  >>> |

* loc함수를 이용하여 행 데이터 삽입 가능

|  |
| --- |
| df.loc['2018-09-17']=[5,6,7,8,9]  df  >>> |

* A열과 C열의 연산 가능

|  |  |
| --- | --- |
|  | df['F']=df['A']+df['C']  df['F']  >>> |