파이썬 라이브러리를 활용한 데이터 분석

3장 내장 자료구조, 함수, 파일

2020.06.18 3h

파일 ch03-study.ipynb

- **내용**
  - 튜플, 리스트, 사전, 집합
  - 함수
  - file

튜플

- **튜플**
  - 1차원의 고정된 크기를 가지는 변경 불가능한 순차 자료형
    - **변경 불가능**
      - **Hashable**
      - **Immutable**
  - tuple(a)
    - **a: 모든 순차 자료형이나 이터레이터를 튜플로 변환 가능**
      - **tuple([1, 2, 3]), tuple('python')**
  - 주의
    - **튜플 내에 저장된 객체는 그 위치에서 변경이 가능**
      - **Tup = tuple(['foo', [1, 2], True])**
      - **tup[1].append(3)**

리스트

- **리스트**
  - 크기와 내용이 수정되는 순차 자료형
  - *rest
    - **길이를 모르는 인자를 담는 방법**
      - **values = 1, 2, 3, 4**
      - **a, b, *rest = values**
  - list(iterator_or_generator)
    - **Iterator나 generator의 실제 값을 담기 위한 방법**

모듈 bisect

- **표준 라이브러리**
  - 이진 검색 알고리즘을 이용해서 시퀀스를 검색하고, 시퀀스에 항목을 삽입할 수 있는 함수를 제공
  - bisect()
    - **bisect.bisect(a, x, lo=0, hi=len(a))**
    - **a라는 오름차순으로 정렬된 시퀀스에 x값이 들어갈 위치를 리턴**
  - insort()
    - **bisect.insort(a, x, lo=0, hi=len(a))**
    - **a라는 오름차순으로 정렬된 시퀀스에 x값을 삽입**

사전

- **해쉬맵, 연관 배열**
  - update()
    - **기본 키는 값 수정, 새로운 키는 키-값 삽입**
  - get(키, 기본_값)
    - **키가 없으면 None이나 기본_값을 반환**
  - pop(키)
    - **키가 없으면 예외 발생**
  - setdefault(key[, default])
    - **키가 있으면, 값을 반환, 키가 없으면 새로운 키의 값으로 삽입될 default를 반환**
      - **If key is in the dictionary, return its value. If not, insert key with a value of default and return default. default defaults to None.**

```
In [80]: words = ['apple', 'bat', 'bar', 'atom', 'book']
         by_letter = {}

         for word in words:
             letter = word[0]
             by_letter.setdefault(letter, []).append(word)

         by_letter
Out[80]: {'a': ['apple', 'atom'], 'b': ['bat', 'bar', 'book']}
```

내장 모듈 collections

- **collections.defaultdict(default_factory, key=value,...)**
  - 딕셔너리(dictionary)와 거의 비슷하지만 key값이 없을 경우 미리 지정해 놓은 초기(default)값을 반환하는 dictionary
    - **default_factory는 defaultdict의 초기값을 지정하는 인자**
    - **key1=value1,key2=value2,...,keyn=valuen를 지정**
- **collections.defaultdict(list)**
  - 초기 값이 list

```
In [79]: from collections import defaultdict

         words = ['apple', 'bat', 'bar', 'atom', 'book']

         by_letter = defaultdict(list)
         for word in words:
             by_letter[word[0]].append(word)

         dict(by_letter)

Out[79]: {'a': ['apple', 'atom'], 'b': ['bat', 'bar', 'book']}
```

p105

유효한 사전 키

- **키**
  - 바뀌지 않는 객체만 가능
    - **Scalar**
      - **정수, 실수, 문자열**
    - **튜플**
    - **다른 사전도 키로 사용 불가능**
  - 해쉬 가능한 객체
    - **함수 hash()로 검사 가능**

집합

- **생성**
  - {1, 2, 4}
  - set([1, 2, 3])
  - 틀린 예
    - **{}**
    - **set(1, 2, 3)**
- **집합 원소**
  - 중복 불가능
  - 수정 불가능한 객체

일부 인자만 취하기

- **커링**
  - 함수의 전달 인자 몇 개를 미리 채움으로써 더 간단한 함수를 만드는 방법
  - 하나 이상의 인수가 이미 채워진 함수의 새 버전을 만들기 위해 사용
    - **커링은 수학자 하스켈 커리로부터 유래된 이름이고 함수를 변형하는 과정**
- **functools.partial**

```
In [142]: def power(base, exponent):
              return base ** exponent

          from functools import partial

          square = partial(power, exponent=2)
          cube = partial(power, exponent=3)

          cube(5)

Out[142]: 125
```

Generator

- **순회 가능한 객체(이터레이터)를 생성하는 방법**
  - 함수 안에서 yield를 사용하면 함수는 제너레이터가 되며 yield에는 값(변수)을 지정
    - **yield 값**
  - 작동 방식
    - **요청 시 순차적인 값을 yield에 의해 하나씩 반환**
- **호출**
  - 호출하더라고 실행되지 않음
  - 반복에 사용하던지, 리스트 등에 사용해야 함
- **제네레이터 표현식**
  - 괄호 사용
    - **gen = (x ** 2 for x in range(100))**

```
In [162]:
def number_generator():
    yield 0
    yield 1
    yield 2

number_generator()

Out[162]: <generator object number_generator at 0x000001ADF0B64B48>

In [150]:
for i in number_generator():
    print(i)

0
1
2

In [157]: g = number_generator()

In [155]: g.__next__()

Out[155]: 2

In [158]: next(g)

Out[158]: 0
```

모듈 itertools

- 일반 데이터 알고리즘을 위한 다양한 제너레이터 제공
  - groupby(리스트, 키)
    - 리스트를 키로 구분한 그룹의 제너레이터를 반환
      - 키와 제너레이터를 반환

itertools module

In [171]:
```
import itertools
first_letter = lambda x: x[0]
names = ['Alan', 'Adam', 'Wes', 'Will', 'Albert', 'Steven']
for letter, names in itertools.groupby(names, first_letter):
    print(letter, list(names)) # names is a generator
```

```
A ['Alan', 'Adam']
W ['Wes', 'Will']
A ['Albert']
S ['Steven']
```

주피터 셀 결과 한글 깨짐 처리

- **명령어 chcp(change code page)**
  - Code page:
    - **기본 949**
  - 한글 처리 utf8
    - **65001로 수정**
    - **!chcp 65001**

```
In [15]: !chcp 65001

Active code page: 65001

In [16]: !dir

 Volume in drive D is DATADRIVE0
 Volume Serial Number is 0AEE-91E2

 Directory of D:₩(0 2020 이공계 연수 저장소)₩Data Analysis & Visualization₩data code

2020-06-16  오후 05:37    <DIR>          .
2020-06-16  오후 05:37    <DIR>          ..
2020-06-16  오후 01:55    <DIR>          .ipynb_checkpoints
2020-06-14  오후 04:56            26,594 appa-study.ipynb
2020-05-30  오후 12:06            33,571 appa.ipynb
2020-06-16  오후 01:27            83,011 ch02-study.ipynb
2020-06-03  오후 03:54            31,341 ch02.ipynb
2020-06-16  오후 05:37            45,763 ch03-study.ipynb
2020-05-30  오후 12:06            46,733 ch03.ipynb
2020-06-05  오후 03:44           186,305 ch04-study.ipynb
2020-05-30  오후 12:06            35,253 ch04.ipynb
2020-06-06  오후 05:23            67,393 ch05-study.ipynb
2020-05-30  오후 12:06            40,706 ch05.ipynb
2020-05-30  오후 12:06            26,542 ch06.ipynb
2020-06-15  오후 06:26           175,557 ch07-study.ipynb
2020-05-30  오후 12:06            29,145 ch07.ipynb
2020-06-14  오후 04:05           367,841 ch08-study.ipynb
2020-05-30  오후 12:06            27,217 ch08.ipynb
2020-05-30  오후 12:06            25,195 ch09.ipynb
2020-06-11  오후 06:50            35,150 ch10-study.ipynb
2020-05-30  오후 12:06            28,332 ch10.ipynb
2020-05-30  오후 12:06            40,852 ch11.ipynb
2020-06-14  오후 04:04            68,684 ch12-study.ipynb
2020-05-30  오후 12:06            19,489 ch12.ipynb
2020-06-14  오후 04:40            47,342 ch13-study.ipynb
2020-05-30  오후 12:06            17,020 ch13.ipynb
2020-05-30  오후 12:06            46,194 ch14.ipynb
2020-06-15  오전 09:34    <DIR>          datasets
2020-06-12  오전 10:53    <DIR>          examples
2020-05-30  오전 11:34                22 hello.py
2020-06-05  오전 11:09             8,667 mynumpy1.ipynb
2020-06-06  오후 01:44           350,681 my_10_minutes_to_pandas.ipynb
2020-06-16  오전 09:05                95 some_module.py
2020-06-16  오전 09:06    <DIR>          __pycache__
              28 File(s)      1,910,695 bytes
               6 Dir(s)  1,902,721,912,832 bytes free
```

바이트와 유니코드

- **텍스트 모드와 바이너리 모드**
  - 텍스트 모드 파일 열기
    - **인코딩 방식 utf8이라면 chars = f.read(10)에서**
      - **10개의 문자를 읽어 옴**
      - **10 바이트~40바이트**
  - 바이너리 모드 파일 열기
    - **chars = f.read(10)에서**
      - **무조건 10바이트를 읽어 옴**

```
In [85]: with open(path) as f:
             lines = [x.rstrip() for x in f]

In [86]: f = open(path)
         f.read(10)
         f.tell()

Out[86]: 11

In [87]: f2 = open(path, 'rb')  # Binary mode
         f2.read(10)
         f2.tell()

Out[87]: 10
```