Reading and Writing CSV files

In [137]:

```
import csv
```

In [55]:

```
%precision 2
#실수형이 나왔을때 소수점 2자리 까지만 보여주도록 하는 것
```

Out[55]:

```
'%.2f'
```

In [56]:

```
with open('car.csv') as csvfile:
    cars = list(csv.DictReader(csvfile))
```

In [57]:

```
cars[:3]   #처음부터 3개까지 읽어들임
```

Out[57]:

```
[{'': '1',
  'manufacturer': 'audi',
  'model': 'a4',
  'displ': '1.8',
  'year': '1999',
  'cyl': '4',
  'trans': 'auto(l5)',
  'drv': 'f',
  'cty': '18',
  'hwy': '29',
  'fl': 'p',
  'class': 'compact'},
 {'': '2',
  'manufacturer': 'audi',
  'model': 'a4',
  'displ': '1.8',
  'year': '1999',
  'cyl': '4',
  'trans': 'manual(m5)',
  'drv': 'f',
  'cty': '21',
  'hwy': '29',
  'fl': 'p',
  'class': 'compact'},
 {'': '3',
  'manufacturer': 'audi',
  'model': 'a4',
  'displ': '2',
  'year': '2008',
  'cyl': '4',
  'trans': 'manual(m6)',
  'drv': 'f',
  'cty': '20',
  'hwy': '31',
  'fl': 'p',
  'class': 'compact'}]
```

In [58]:

```
len(cars)    #데이터의 개수
```

Out[58]:

```
234
```

In [59]:

```
cars[0].keys()
```

Out[59]:

```
dict_keys(['', 'manufacturer', 'model', 'displ', 'year', 'cyl', 'trans', 'drv', 'ct
y', 'hwy', 'fl', 'class'])
```

In [60]:

```
sum(float(d['cty']) for d in cars) / len(cars)
```

Out[60]:

```
16.86
```

In [61]:

```
sum(float(d['hwy']) for d in cars) / len(cars)    #하이웨이(hwy)연비 평균 계산
```

Out[61]:

```
23.44
```

In [62]:

```
cylinder = set(d['cyl'] for d in cars)    #set -> 집합의 형태
cylinder    #cyl = 실린더 (몇기통인지)
```

Out[62]:

```
{'4', '5', '6', '8'}
```

In [63]:

```
CtyMpgByCyl = []
```

In [64]:

```
for c in cylinder :
    summpg = 0
    cyltypecount = 0

    for d in cars :
        if d['cyl'] == c :
            summpg += float(d['cty'])
            cyltypecount += 1
    CtyMpgByCyl.append((c, summpg/cyltypecount))
```

In [65]:

```
CtyMpgByCyl
```

Out[65]:

```
[('5', 20.50), ('6', 16.22), ('8', 12.57), ('4', 21.01)]
```

In [66]:

```
CtyMpgByCyl.sort(key = lambda x : x[0])
CtyMpgByCyl
```

Out[66]:

```
[('4', 21.01), ('5', 20.50), ('6', 16.22), ('8', 12.57)]
```

In [67]:

```
vehicleclass = set(d['class'] for d in cars)
vehicleclass
```

Out[67]:

```
{'2seater', 'compact', 'midsize', 'minivan', 'pickup', 'subcompact', 'suv'}
```

In [68]:

```
HwyMpgByClass = []
```

In [69]:

```
for c in vehicleclass :
    summpg = 0
    classtypecount = 0

    for d in cars :
        if d['class'] == c :
            summpg += float(d['hwy'])
            classtypecount += 1
    HwyMpgByClass.append((c, summpg/classtypecount))
```

In [70]:

```
HwyMpgByClass
```

Out[70]:

```
[('midsize', 27.29),
 ('compact', 28.30),
 ('2seater', 24.80),
 ('minivan', 22.36),
 ('pickup', 16.88),
 ('subcompact', 28.14),
 ('suv', 18.13)]
```

In [71]:

```
HwyMpgByClass.sort(key = lambda x : x[1])
HwyMpgByClass
```

Out[71]:

```
[('pickup', 16.88),
 ('suv', 18.13),
 ('minivan', 22.36),
 ('2seater', 24.80),
 ('midsize', 27.29),
 ('subcompact', 28.14),
 ('compact', 28.30)]
```

The Python Programming : Dates and Time

In [72]:

```
import datetime as dt    #datetime이라는 모듈을 dt라는 변수로 사용하겠다는 것
import time as tm
```

In [73]:

```
tm.time()
```

Out[73]:

```
1616393785.31
```

In [74]:

```
dtnow = dt.datetime.fromtimestamp(tm.time())
dtnow
```

Out[74]:

```
datetime.datetime(2021, 3, 22, 15, 59, 5, 825947)
```

In [75]:

```
dtnow.year
```

Out[75]:

```
2021
```

In [76]:

```
dtnow.month
```

Out[76]:

```
3
```

In [77]:

```
delta = dt.timedelta(days = 100)    #days가 100인 timedelta를 의미
```

In [78]:

```
today = dt.date.today()
today
```

Out[78]:

```
datetime.date(2021, 3, 22)
```

In [79]:

```
today - delta    #오늘로부터 100일 전
```

Out[79]:

```
datetime.date(2020, 12, 12)
```

In [80]:

```
today > today - delta    #today가 더 미래냐
```

Out[80]:

```
True
```

The Python Programming : Objects and map() function

객체지향

In [81]:

```
class Person : #Person을 선언하면 만들어지는 기본적인 변수를 department로 정의
    department = 'Department of Computer Science'

    def set_name(self, new_name) :
        self.name = new_name
```

In [82]:

```
person = Person()
```

In [86]:

```
person.set_name('eunchae')
```

In [87]:

```
print('{} in department {}'.format(person.name, person.department))
```

```
eunchae in department Department of Computer Science
```

In [88]:

```
store1 = [10.00, 11.00, 12.34, 2.34]
store2 = [9.00, 11.10, 12.34, 2.87]
```

In [89]:

```
cheapest = map(min, store1, store2)    #store1이랑 store2에서 가장 싼 것 찾으려고 mapping한 것
```

In [90]:

```
cheapest     #map이라고 하는 fuction의 주소
```

Out[90]:

```
<map at 0x2669d711e50>
```

In [91]:

```
for item in cheapest :
    print(item)
```

```
9.0
11.0
12.34
2.34
```

The Python Programming : Numerical Python (NumPy)

In [92]:

```
import numpy as np
```

In [93]:

```
mylist = [1, 2, 3]
mylist
```

Out[93]:

```
[1, 2, 3]
```

In [94]:

```
x = np.array(mylist)
x
```

Out[94]:

```
array([1, 2, 3])
```

In [95]:

```
y = np.array([4, 5, 6])
y
```

Out[95]:

```
array([4, 5, 6])
```

In [97]:

```
m = np.array([[7, 8, 9], [10, 11, 12]])
m
```

Out[97]:

```
array([[ 7,  8,  9],
       [10, 11, 12]])
```

In [98]:

```
m.shape    #2행(row) 3열(column)
```

Out[98]:

```
(2, 3)
```

In [100]:

```
n = np.arange(0, 30, 2)    #0에서 30까지 2씩 건너뛰어서
n
```

Out[100]:

```
array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28])
```

In [102]:

```
n = n.reshape(3, 5)    #3행 5열로 만들어줌
n
```

Out[102]:

```
array([[ 0,  2,  4,  6,  8],
       [10, 12, 14, 16, 18],
       [20, 22, 24, 26, 28]])
```

In [104]:

```
o = np.linspace(0, 4, 9)    #0부터 4까지의 데이터를 일정하게 9개로 나눠서 배열로 만듦
o
```

Out[104]:

```
array([0. , 0.5, 1. , 1.5, 2. , 2.5, 3. , 3.5, 4. ])
```

In [105]:

```
o.resize(3, 3)
o
```

Out[105]:

```
array([[0. , 0.5, 1. ],
       [1.5, 2. , 2.5],
       [3. , 3.5, 4. ]])
```

In [106]:

```
np.ones((4, 3))    #1로만 (4,3)행렬을 만듦
```

Out[106]:

```
array([[1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.],
       [1., 1., 1.]])
```

In [107]:

```
np.zeros((2, 3))    #0으로만 (2, 3)행렬을 만듦
```

Out[107]:

```
array([[0., 0., 0.],
       [0., 0., 0.]])
```

In [108]:

```
np.eye(4)    #대각선만 1로 해서 행렬 만듦
```

Out[108]:

```
array([[1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.]])
```

In [109]:

```
y
```

Out[109]:

```
array([4, 5, 6])
```

In [110]:

```
np.diag(y)    #대각선에 y 원소 넣어서 행렬 만듦
```

Out[110]:

```
array([[4, 0, 0],
       [0, 5, 0],
       [0, 0, 6]])
```

In [113]:

```
np.array([1, 2, 3] * 3)
```

Out[113]:

```
array([1, 2, 3, 1, 2, 3, 1, 2, 3])
```

In [114]:

```
np.repeat([1, 2, 3], 3)
```

Out[114]:

```
array([1, 1, 1, 2, 2, 2, 3, 3, 3])
```

In [116]:

```
p = np.ones([2, 3], int)
p
```

Out[116]:

```
array([[1, 1, 1],
       [1, 1, 1]])
```

In [119]:

```
p1 = np.vstack([p, 2 * p])    #아래에 쌓음
p1
```

Out[119]:

```
array([[1, 1, 1],
       [1, 1, 1],
       [2, 2, 2],
       [2, 2, 2]])
```

In [121]:

```
p2 = np.hstack([p, 2 * p])    #오른쪽에 쌓음
p2
```

Out[121]:

```
array([[1, 1, 1, 2, 2, 2],
       [1, 1, 1, 2, 2, 2]])
```

In [122]:

```
x, y
```

Out[122]:

```
(array([1, 2, 3]), array([4, 5, 6]))
```

In [123]:

```
x + y
```

Out[123]:

```
array([5, 7, 9])
```

In [124]:

```
x - y
```

Out[124]:

```
array([-3, -3, -3])
```

In [125]:

```
x * y
```

Out[125]:

```
array([ 4, 10, 18])
```

In [126]:

```
x.dot(y)    #1() * 4 )+ (2 * 5 )+ (3 * 6)
```

Out[126]:

```
32
```

In [127]:

```
x / y    #x를 y로 나눔
```

Out[127]:

```
array([0.25, 0.4 , 0.5 ])
```

In [128]:

```
x ** 2    #제곱
```

Out[128]:

```
array([1, 4, 9], dtype=int32)
```

In [129]:

```
z = np.vstack([x, y])
z
```

Out[129]:

```
array([[1, 2, 3],
       [4, 5, 6]])
```

In [130]:

```
z.T    #Transpose 행렬 전치
```

Out[130]:

```
array([[1, 4],
       [2, 5],
       [3, 6]])
```

In [131]:

```
z.shape
```

Out[131]:

```
(2, 3)
```

In [132]:

```
z.T.shape
```

Out[132]:

```
(3, 2)
```

In [133]:

```
z.dtype
```

Out[133]:

```
dtype('int32')
```

In [134]:

```
z = z.astype('f')    #float로 데이터 타입 바꾸기
z.dtype
```

Out[134]:

```
dtype('float32')
```

In [135]:

```
z.min()    #z에서 제일 작은 것
```

Out[135]:

```
1.0
```

In [136]:

```
z.argmin()
```

Out[136]:

```
0
```