July-18

head → first load whole data & then give required rows
nrows=10 → just read first 10 rows

Python-tutorial 11

(*)

```
pd.options.display.max_rows=6
import pandas as pd
result = pd.read_csv('ex6.csv')
result
```

```
       one   two   three   four
0      .     .     .       .
1      .     .     .       .
2      .     .     .       .
---    ---   ---   ---     ---
597    .     .     .       .
598    .     .     .       .
599    .     .     .       .
```

(600 x 4)

(*)

```
pd.read_csv('ex6.csv', nrows=5)
```

(just reads first 5 rows & doesn't loads whole data unlike in head()).

(*) Chunksize:-

```
chunk=pd.read_csv('ex6.csv',chunksize=1000)
```

(iterate using for loop).

(∵ reads data upto 1000 bytes ← you may loss data in last row).

by_default sep = ','

• Writing to file:

Reading
```
data = pd.read_csv('[illegible] ex5.csv')
data
```
Any-table

writing: `data.to_csv('xyz.csv')` → (in any format you want).
```
data.to_csv(
```
(*) replace null data with some data:
```
import sys
data.to_csv('out1.csv', sep='@')
data
```
```
      A  B  C   D
  0   a  b  c   NaN
  1   d  e  NaN g
  2   h  i  j   k
```
standard output
```
data.to_csv(sys.stdout, na_rep='d')
```
```
      A  B  C  D
  0   a  b  c  d
  1   d  e  d  g
  2   h  i  j  k
```
(*) remove header & index, & selecting colum (filter)

~~[illegible]~~

Imp
```
data.to_csv('xyz.csv', index=False,
                       header=False)

data.to_csv('xyz.csv', index=False,
                       columns=['A', 'C', [illegible]])
```

⊛ date_range : (inbuilt)

```
dates = pd.date_range('1/1/2000', periods=7)
```

```
type(dates)
→ pandas.core.indexes.datetimes.DatetimeIndex
```

```
dates
→ (['2000-01-01', '2000-01,02', - - -, '2000-01-07'])
```

• Convert this in series : (use Numpy library)

```
import numpy as np
~~ts = pd.Series~~
dates = pd.date_range('1/1/2000', periods=3)
ts = pd.Series(np.arrange.(3), index=dates)
```

(by default index is 0,1,2,....).

```
ts.to_csv('tseries.csv')
→ 2000-01-01 , 0
  2000-01-02 , 1
  2000-01-03 , 2
```

• Reading data without using Pandas (not recommended)

```
import csv
f = open('ex7.csv')
reader = csv.reader(f)
```

not in form of dataframe

```
for line in reader:
print(line)
```

```
['a', 'b', 'c']
['1', '2', '3']
['1', '2', '3']
```

```
list(reader) → [['a','b','c'], ['1','2','3'], ['1','2','3']]
```

{JSON → in form of dictionary :}

• another method: (not recommended)

```
with open('ex7.csv') as f:
    lines = list(csv.reader(f))
    lines
→[['a','b','c'], ['1','2','3'], ['1','2','3']]
    header, values = lines[0], lines[1:]

    header → ['a','b','c']
    values → [['1','2','3'], ['1','2','3']]
```

zip:

```
list(zip(header, zip(*values)))
 [('a',('1','1')), ('b',('2','2')), ('c',('3','3'))]
```

dictionary-comprehension:

```
l1 = [1,2,3,4]
l2 = [5,6,7,8]
{a:b for a,b in zip(l1,l2)}
→ {1:5, 2:6, 3:7, 4:8}
```

• JSON Data: (Java Script object Notation):

```
import json                          obj=
result = json.loads(obj)             '''{
result
→ {dic}                                }
                                     '''
type(result) → dict                  type(obj) → str
```

[str data → json file → data frame]
(or dict)

So, we changed from [illegible] str → dict

Now, [illegible] dict → str.

```
asjson = json.dumps(result)
type(asjson) → str
```

• converting dict of json file into dataframe: (first import & load json)

```
result ['siblings']
~~[{'name': 'A', 'B', 'C'~~
[{'A':1, 'B':2, 'C':[3,4,5]},
 {'A':6, 'B':7, 'C':[8,9,10]}]
siblings = pd.DataFrame(result['siblings'],
              columns = ['A', 'B', 'C'])
```

you may select any columns

siblings =

	A	B	C
0	1	2	[3,4,5]
1	6	7	[8,9,10]

• if you directly has json-data, then how to read:

```
data = pd.read_json('ex7.json')
data → dataframe
```

• store data in json:

```
data.to_json()
[illegible] print(data.to_json())
        {dict}. (with some changes)
print(data.to_json(orient = 'rec
```

(*) Serde: { [obj → byte code
serelzation/write
[byte code → obj ← (desrelization/read)

● Binary Data Formats:
(pickle = binary data file)

```
frame = pd.read_csv('ex1.csv')
```
→ human-readable

```
frame.to_pickle('frame-pickle')
```
not human readble

```
pd.read_pickle('frame-pickle')
```

	a	b	c	d
0	A	B	C	D
1	E	F	G	H
2	I	J	K	L

Note: There are many kinds of binary formats. (readbility, meta-structure etc. are diff.)

●

$$\begin{tikzcd}
\text{Google} \arrow[r, "\text{bridge}", "\text{API}"'] & \text{HDFC}
\end{tikzcd}$$

diff tech.   diff. tech

heterogeneous platforms

- Web APIs (Appli. Programming Interface)

(connectivity b/w two homo./hetro. platforms)

```
import requests
url = 'https: ...'
resp = requests.get(url)
resp        → <Response [200]>
type(resp)  → requests.models.Response

data = resp.json[]
data[0]['user']['id] → 13159005
```