

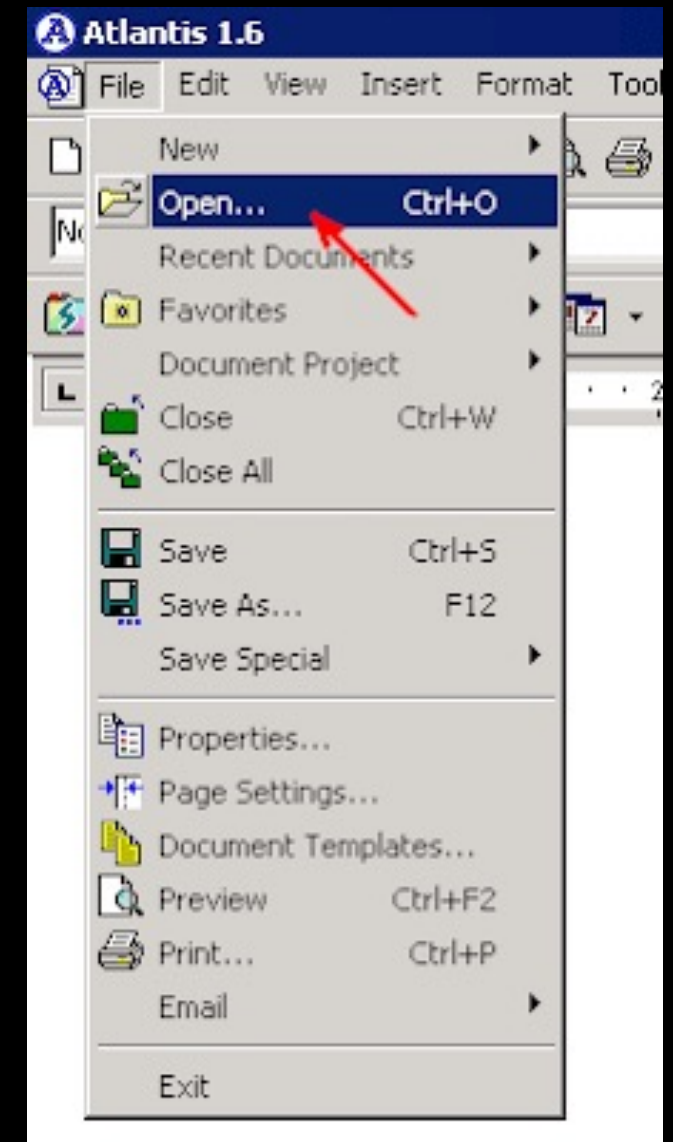
Working with Files

Lecture 6



Opening a File

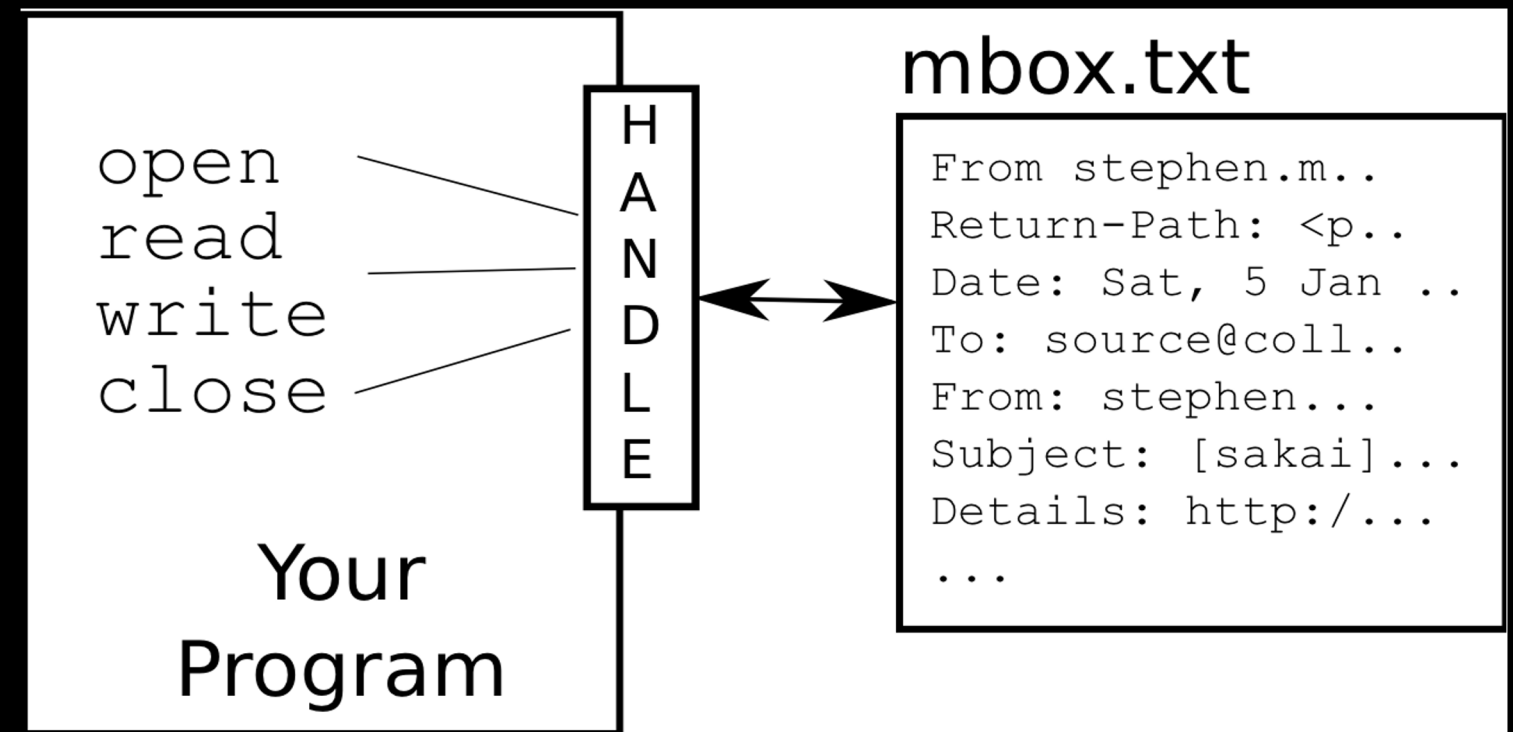
- This is done with the `open()` function
- Similar to “File -> Open” in a Word Processor



- `open()` returns a “file handle” - a variable used to perform operations on the file

What is a Handle?

```
>>> fhand = open('mbox.txt')
>>> print(fhand)
<_io.TextIOWrapper name='mbox.txt' mode='r' encoding='UTF-8'>
```



Using open()

```
fhand = open('mbox.txt', 'r')
```

- `handle = open(filename, mode)`
- `filename` is a string
- `mode` is optional
 - 'r' read
 - 'w' write

Basic syntax

```
fhand = open( 'mbox.txt' , 'r')
content = fhand.read()
fhand.close()
print(content)
```

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008\nReturn-Path:  
<postmaster@collab.sakaiproject.org>\nDate: Sat, 5 Jan 2008 09:12:18 -  
0500\nTo:source@collab.sakaiproject.org\nFrom:stephen.marquard@uct.ac.za  
\nSubject: [sakai] svn commit: r39772\n
```

<http://www.py4e.com/code/mbox-short.txt>

Reading the *Whole* File

We can **read** the whole file (newlines and all) into a **single string**

```
>>> fhand = open('mbox.txt', 'r')
>>> content = fhand.read()
>>> print(len(content))
90000
>>> print(content[:20])
From stephen.marquar
```

Iterating line by line

- **file handle** → a **sequence** of strings
- **for**: iterate through a **sequence**
- Remember - a **sequence** is an ordered set

```
xfile = open( 'mbox.txt' )  
for line in xfile:  
    print(line)
```

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008\nReturn-Path: <postmaster@collab.sakaiproject.org>\nDate: Sat, 5 Jan 2008 09:12:18 -0500\nTo:source@collab.sakaiproject.org\nFrom:stephen.marquard@uct.ac.za\n
```

Counting Lines in a File

- Open a **file** read-only
- Use a **for** loop to read each line
- **Count** the lines and print out the number of lines

```
fhand = open('mbox.txt', 'r')  
count = 0  
for line in fhand:  
    count = count + 1
```

```
fhand.close()  
print('Line Count:', count)
```

```
Line Count: 132045
```


Writing Files in Python

Basic syntax

```
fhand = open('test.txt', 'w')  
fhand.write('some text!\n')  
fhand.write('... and some more \n')  
fhand.close()
```

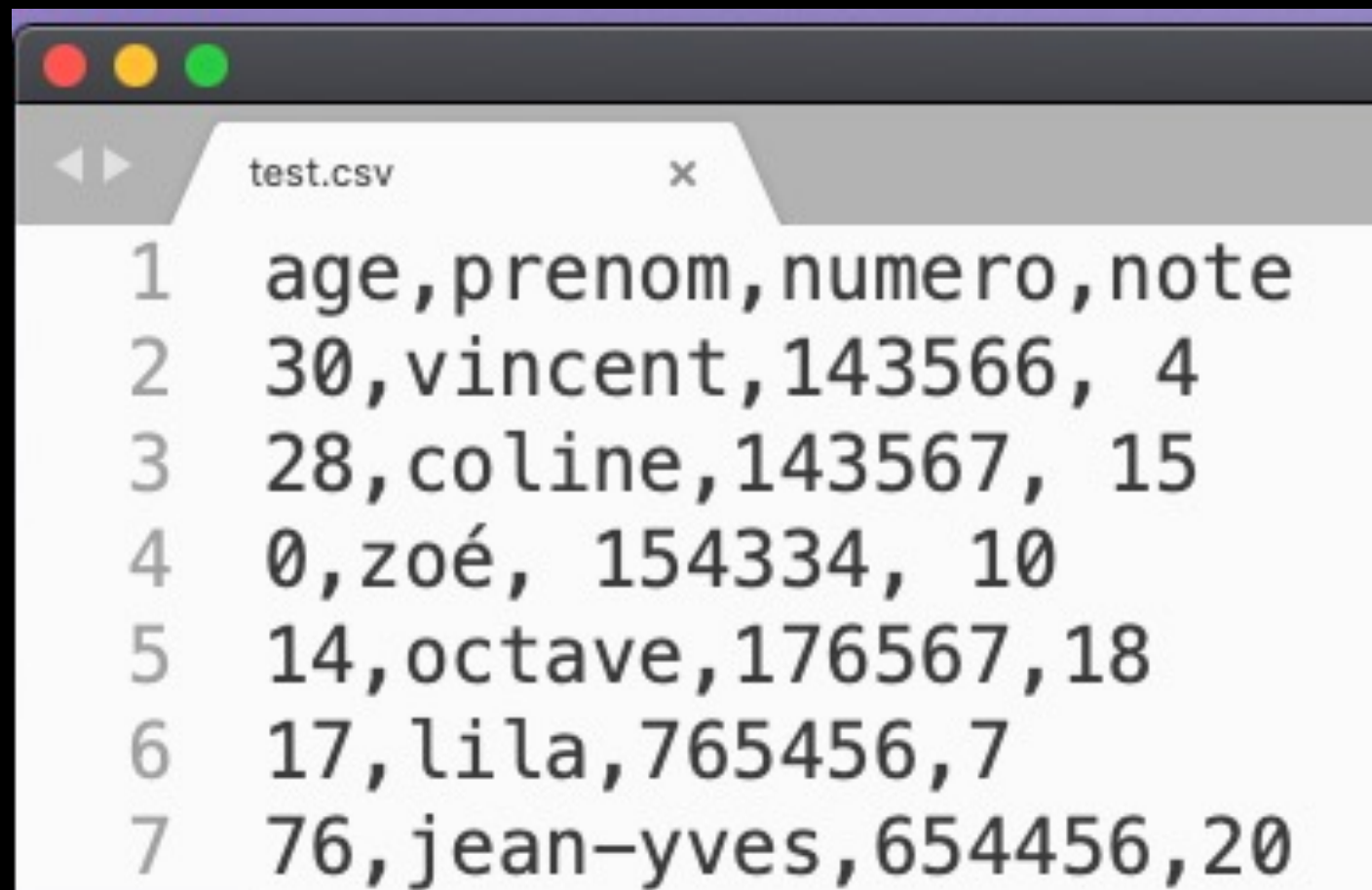
test.txt

some text!
... and some more

Other data formats

.csv format files

Comma Separated Values

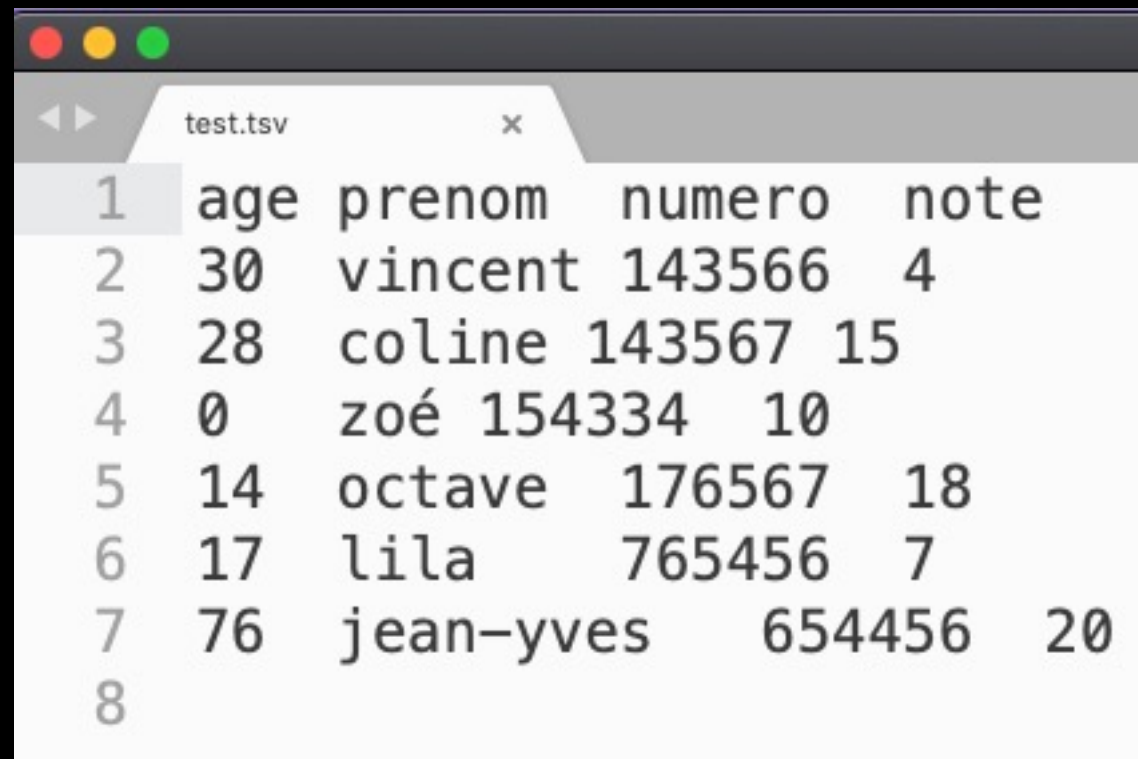


	age,prenom,numero,note
1	30,vincent,143566, 4
2	28,coline,143567, 15
3	0,zoé, 154334, 10
4	14,octave,176567,18
5	17,lila,765456,7
6	76,jean-yves,654456,20
7	

```
tfile = open('test.tsv')  
for line in tfile:  
    #do sth
```

.tsv format files

- Tab-Separated Values



A screenshot of a text editor window with a single tab titled 'test.tsv'. The window displays a table of data with 5 columns: 'age', 'prenom', 'numero', and 'note'. The first row is the header, and the following seven rows contain data. Line numbers 1 through 8 are visible on the left side of the editor.

1	age	prenom	numero	note
2	30	vincent	143566	4
3	28	coline	143567	15
4	0	zoé	154334	10
5	14	octave	176567	18
6	17	lila	765456	7
7	76	jean-yves	654456	20
8				

```
tfile = open('test.tsv')  
for line in tfile:  
    #do sth
```

.conll format text files

```
# sent_id = 1
# text = They buy and sell books.
1  They      they      PRON      PRP      Case=Nom|Number=Plur
2  buy       buy       VERB      VBP      Number=Plur|Person=3|Tense=Pres
3  and       and       CONJ      CC       _
4  sell      sell      VERB      VBP      Number=Plur|Person=3|Tense=Pres
5  books     book      NOUN      NNS      Number=Plur
6  .         .         PUNCT     .         _
```

```
c_file = open('test.conll')
for line in c_file:
    if not line.startswith("#"):
        row = line.rstrip().split("\t")
        print(row)
```

Pandas library

Pandas library

- Before you can use pandas module in your program, you must import the library using “import pandas”
- recall the functions of this module by pandas.function_name()

```
import pandas
df_s = pandas.read_csv( 'test.csv' )
df_t = pandas.read_csv( 'test.tsv', sep="\t" )
```


Data frames methods

In [1]:

```
df_s = pandas.read_csv('test.csv')  
# List first 5 rows  
df_s.head()
```

	age	prenom	numero	note
0	30	vincent	143566	4
1	28	coline	143567	15
2	0	zoé	154334	10
3	14	octave	176567	18
4	17	lila	765456	7

df.method()	description
head([n]), tail([n])	first/last n rows
max(), min()	return max/min values for all numeric columns
mean(), median()	return mean/median values for all numeric columns
std()	standard deviation

Data frames attributes

```
In [2]: # Get column names  
cols = list(df_s.columns)  
print(cols)
```

```
['age', 'prenom', 'numero', 'note']
```

df.attribute	description
columns	list the column names
size	number of elements
dtypes	list the types of the columns

Selecting a column

In [3]:

```
# selecting a column  
print(df_s['prenom'])
```

```
0      vincent  
1      coline  
2        zoé  
3      octave  
4        lila  
5    jean-yves  
Name: prenom, dtype: object
```

	age	prenom	numero	note
0	30	vincent	143566	4
1	28	coline	143567	15
2	0	zoé	154334	10
3	14	octave	176567	18
4	17	lila	765456	7

Selecting lines

In [4]:

```
# selecting one or several lines  
sliced_df = df_s.loc[1:2]  
print(sliced_df)
```

	age	prenom	numero	note
1	28	coline	143567	15
2	0	zoé	154334	10

	age	prenom	numero	note
0	30	vincent	143566	4
1	28	coline	143567	15
2	0	zoé	154334	10
3	14	octave	176567	18
4	17	lila	765456	7
5	76	jean-yves	654456	20

**Note that `.loc[1:2]` is interpreted as a label of the index,
And never as an integer position along the index*

Iterate through lines

In [5]:

```
# iterate through every row and print the  
# item at position 1 in that row  
for idx,row in df_s.iterrows():  
    print(row[1])
```

```
vincent  
coline  
zoé  
octave  
lila  
jean-yves
```

	age	prenom	numero	note
0	30	vincent	143566	4
1	28	coline	143567	15
2	0	zoé	154334	10
3	14	octave	176567	18
4	17	lila	765456	7
5	76	jean-yves	654456	20

Basic operations

In [6]:

```
n_stu= df_s['note']  
print(n_stu.min())  
print(n_stu.max())  
print(n_stu.mean())  
print(n_stu.median())  
print(n_stu.std())
```

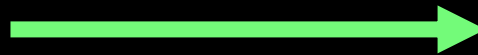
```
↳ 4  
   20  
   12.333333333333334  
   12.5  
   6.3456021516217564
```

	age	prenom	numero	note
0	30	vincent	143566	4
1	28	coline	143567	15
2	0	zoé	154334	10
3	14	octave	176567	18
4	17	lila	765456	7
5	76	jean-yves	654456	20

Sorting data

```
# Create a new data frame sorted by column note  
d_s = df_s.sort_values('note', ascending=False)  
print(d_s)
```

	age	prenom	numero	note
0	30	vincent	143566	4
1	28	coline	143567	15
2	0	zoé	154334	10
3	14	octave	176567	18
4	17	lila	765456	7
5	76	jean-yves	654456	20



age	prenom	numero	note
76	jean-yves	654456	20
14	octave	176567	18
28	coline	143567	15
0	zoé	154334	10
17	lila	765456	7
30	vincent	143566	4

Writing a csv file with Pandas

1. Create a data frame

```
data = {"prenom": ["vincent", "coline", "zoé"],  
        "note": [5, 15, 18]}  
df3 = pandas.DataFrame(data)  
print(df3)
```

```
[>]      prenom  note  
0  vincent     5  
1  coline    15  
2    zoé     18
```


Writing a csv file with Pandas

2. Export the **Data Frame** to a CSV file

```
df3.to_csv( 'mycsv.csv' )
```

```
[>]      prenom  note
0  vincent     5
1   coline    15
2     zoé     18
```