

# Reading and Writing Files

- Every things in life related to data
- Decision depends on data
- Experiments depends on data...

Many way to store data:





- Plain Text
- Each piece of data in one line
- Each line have different data type separated by delimiter
- Data could span many line with pieces of data separated by delimiter

In this lecture Python deal with data (read / write/ process)  
Show different techniques for reading files

## What Kinds of Files Are There?

There are many kinds of files. Text files, music files, videos, and various word processor and presentation documents are common.

Normally formatted files takes more space than plain text file, here is example of some of the file types – size for file:

	empty.docx	21 KB	Microsoft Word document
	empty.odt	8 KB	OpenDocument Text (.odt) Document
	empty.pages	29 KB	Pages document
	empty.txt	Zero bytes	Plain Text File

# Opening a File

Important Notes:

Python assumes that the file is in the same directory as the program that is doing the reading.

Create your first file:

1. Make a directory, perhaps called file\_examples.
2. In IDLE, select File→New Window and type the following:

This is my first file

It is important to save data

Data is crucial for decision

3. Save under the name file\_example.txt in file\_examples.
4. In IDLE, select File→New, create read file code:

```
file = open('file_example.txt', 'r')  
contents = file.read()  
print(contents)  
file.close()
```

5. Save this as file\_reader.py in your file\_examples directory.

# Opening a File

- Built-in function `open` opens a file
- a *file cursor* acts much like a bookmark.
- The file cursor is initially at the beginning of the file,
- As we read or write data it moves to the end of what we just read or wrote.
- The first argument in the call function `open` is the name of the file to open
- The second argument, is the mode

file *mode*:

- 'r' : read mode
- 'w' for writing and
- 'a' for appending,
- omitting the mode, then the default is 'r'.

## The with Statement

Every opened file should be closed after process

Python provides a with statement that automatically closes a file when the end of the block is reached:

With statement :

Ex:

```
with open('file_example.txt', 'r') as file:  
    contents = file.read()  
    print(contents)
```

The general form of a with statement is as follows:

```
with open(«filename», «mode») as «variable»:  
    «block»
```

## How Files Are Organized on Your Computer

- A *file path* specifies a location in your computer's file system.
- A file path contains the sequence of directories to a file, starting at the *root directory*. The end of sequence is the file you want to process.
- By referring to the file with this sequence, you don't need to put the code and the file on the same directory

Ex:

```
with open('c:/temp/file_example.txt', 'r') as file:  
    contents = file.read()  
    print(contents)
```

# Techniques for Reading Files

- Python provides several techniques for reading files.
- All techniques work starting at the current file cursor.
- Techniques could be combine as needed.

The techniques:

- The Read Technique
- The Readlines Technique
- The “For Line in File” Technique
- The Readline Technique

We will go through these techniques with examples

## The Read Technique

Use this technique when you want to read the contents of a file into a single string

```
with open('file_example.txt', 'r') as file:  
    contents = file.read()  
    print(contents)
```

OR: when you want to specify exactly how many characters to read.

```
with open('file_example.txt', 'r') as example_file:  
    first_ten_chars = example_file.read(10)  
    the_rest = example_file.read()  
    print("The first 10 characters:", first_ten_chars)  
    print("The rest of the file:", the_rest)
```

Important note: When the you finish reading the file, you'll need to close and reopen the file.

This bring the cursor to null value to start from the beginning again



# The Readlines Technique

Use this technique when you want to get a Python list of strings containing the individual lines from a file. Function `readlines` works much like function `read`, except that it splits up the lines into a list of strings.

**EX**

```
with open('file_example.txt', 'r') as example_file:  
    lines = example_file.readlines()  
    print(lines)
```

Here is the output:

```
['First line of text.\n', 'Second line of text.\n', 'Third line of text.\n']
```

# Methods on read line

```
with open('planets.txt', 'r') as planets_file:  
    planets = planets_file.readlines()  
>>> planets
```

?

Then we can impose methods on the lines – it comes as string lists

```
>>> for planet in reversed(planets):  
... print(planet.strip())
```

?

```
>>> for planet in sorted(planets):  
... print(planet.strip())
```

?

```
>>> print(planets[1])
```

?

```
>>> planets[0]
```

?

## The "For Line in File" Technique

This technique read line by line in for loop

On each iteration, the file cursor is moved to the beginning of the next line.

You can perform methods on the line separately

```
with open('planets.txt', 'r') as data_file:  
    for line in data_file:  
        print(line, len(line))           ?
```

Q: Why is the length of Mars is 5?

Try this print:

```
print(line, len(line.strip()))
```

## The Readline Technique

This technique reads one line at a time, unlike the Readlines technique. Use this technique when you want to read only part of a file.

Advantage: you might want to treat lines differently depending on context

Let us consider understand the type of the data and the following code:

# Files over the Internet

- We are living in the age of the internet
- Data is accessible by all
- Data could be used for analysis, risk analysis, planning, research ...

Example: you can find the Hopedale data on line on the following URL:

<http://robjhyndman.com/tsdldata/ecology1/hopedale>

Library Module `urllib.urlrequest` contains a function called `urlopen` that opens a web page for reading.

- Files could be images, music, videos, text, and more),
- readline methods return the file into format bytes.

An Python interpreter (UTF-8 encoder) decode the bytes to integers, strings, functions, and documents.

# Files over the Internet

```
import urllib.request
```

```
"""Library Module urllib.urlrequest contains a function called  
urlopen that opens a web page for reading."""
```

```
url = 'http://robjhyndman.com/tsdldata/ecology1/hopedale.dat'
```

```
with urllib.request.urlopen(url) as webpage:
```

```
for line in webpage:
```

```
    line = line.strip()
```

```
    line = line.decode('utf-8')
```

```
    print(line)
```

# Writing Files

- To create a new file or to replace the contents of an existing file, we use write mode ('w').
- If the filename doesn't exist already, then a new file is created;
- If the file already there, the file contents are erased and replaced.
- Once opened for writing, a clean file is opened
- you can use method write to write a string to the file.

# Append file mode

- Write mode refresh the already existing file each time it is opened
- Append mode : keep the existing data on the file and append the new information after the existing one
- Ex: we will append more items, sales and prices to the existing files

**with** open('topics.txt', 'a') **as** output\_file:  
output\_file.write('Software Engineering')

Ex: append new items, sale/y, price to the items.sales.process files



## Writing Algorithms That Use the File-Reading Techniques

### Skipping the Header

Some files begin with a one-line description followed by comments in lines beginning with a #, Rest is the real data  
Readline technique can be used to skip that header.  
The technique ends when we read the first real piece of data, which will be the first line after the description that doesn't start with a #.

























