Python: Reading and Writing Text Files*

Arash Pourhabib†

1 Reading Files

- To access the contents of a file, first we need to "open" the file.
- The command **open** creates a file object

```
infile = open('data.txt', 'r')
#(1) To read line-by-line:
lines=[]
for line in infile:
    lines.append(line)

#(2) Or equivalently
lines = infile.readlines()

# (3) Read file as a single string
data_file = infile.read()
```

Make sure you close the file using `infile.close()` after each reading.

Note the difference between `infile.readlines()` (which reads from the current line to the end of file) and `infile.readline()` (which only reads the current line).

If you use `infile.read()`, the file contents will be stored as a string: Use `lines.split()` to split the string, and **float** to convert the strings to (float) numbers (if applicable).

*References: (1) Langtangen, Hans Petter. A primer on scientific programming with Python, Fourth Edition. Springer, 2014. (2) https://docs.python.org/2/reference/

†School of Industrial Engineering and Management, Oklahoma State University

2 Writing Files

```
outfile = open('result.txt', 'w')
outfile.write('A sample text\n')
temp = 65.0
S = 'Today the temperature was {:.1f}'.format(temp)
outfile.write(S)
outfile.close()
```

Use `outfile = open('result.txt', 'a')` to append contents to the exiting file.

To read and write (append) at the same time use 'r+' in **open** command.

3 Alternative Way to Read/Write Files

```
with open('data.txt','r') as infile:
    lines = infile.readlines()

with open('result.txt','w') as outfile:
    outfile.write('Some more text\n')
```

There is no need to close the file when using the **with** statement.