

Python language: reading/writing files

The FOSSEE Group

Department of Aerospace Engineering
IIT Bombay

Mumbai, India

Outline

- 1 Reading files
- 2 Writing files
- 3 Exercise: parsing data from file

Opening files

Recall:

```
% pwd # present working directory  
% cd /home/fossee # go to directory
```

The file is in our present working directory

```
In []: f = open('pendulum.txt')  
In []: f
```

- `f` is a file object
- Shows the mode in which the file is open (read mode)

Reading the whole file

```
In []: pend = f.read()
```

```
In []: print(pend)
```

- We have read the whole file into the variable `pend`

```
In []: type(pend)
```

```
In []: pend_list = pend.splitlines()
```

```
In []: pend_list
```

- `pend` is a string variable
- We can split it at the newline characters into a list of strings

Closing the file

- Close the file, when done; Also, if you want to read again

```
In []: f.close()
```

```
In []: f
```

Reading line-by-line

```
In []: for line in open('pendulum.txt'):  
.....:     print(line)
```

- The file object is an “iterable”
- We iterate over it and print each line
- Instead of printing, collect lines in a list

```
In []: line_list = [ ]  
In []: for line in open('pendulum.txt'):  
.....:     line_list.append(line)
```

Outline

- 1 Reading files
- 2 Writing files
- 3 Exercise: parsing data from file

Writing files

```
In []: f = open('new_file.txt', 'w')  
In []: f.write('Hello world!\n')  
In []: f.close()
```

- Note the mode **'w'**
- Will clobber existing file!
- **write** will not add new lines
- Always remember to call **close**

Using print to write files

On Python 2.x

```
In []: from __future__ import print_function
```

```
In []: f = open('new_file.txt', 'w')
```

```
In []: print('Hello world!', file=f)
```

```
In []: f.close()
```

- Just pass the **file** keyword arg
- **print** works normally, so adds new lines

Outline

- 1 Reading files
- 2 Writing files
- 3 Exercise: parsing data from file

File parsing – Problem

A; 010002; AMY A; 058; 037; 42; 35; 40; 212; P; ;

- File with records like the one above is given
- Each record has fields separated by ;
- region code; roll number; name;
- marks — 1st L; 2nd L; math; science; social; total
- pass/fail indicated by P/F; W if withheld and else empty
- We wish to calculate mean of math marks in region B

Tokenization

```
In []: line = "parse this          string"
In []: line.split()
```

- Original string is split on white-space
- Returns a list of strings
- It can be given an argument to split on that argument

```
r = "A;01;JOSE R;083;042;47;AA;72;244;;; "
r.split(';')
```

Tokenization ...

- Since we split on commas, fields may have extra spaces at ends
- We can strip out the spaces at the ends

```
In []: word = "      B      "  
In []: word.strip()
```

- `strip` is returning a new string

str to float

- After tokenizing, the marks we have are strings
- We need numbers to perform math operations

```
In []: mark_str = "1.25"
```

```
In []: mark = float(mark_str)
```

```
In []: type(mark_str)
```

```
In []: type(mark)
```

File parsing – Solution

```
math_B = [] # empty list to store marks
for line in open("sslc1.txt"):
    fields = line.split(";")
    reg_code = fields[0]
    reg_code = reg_code.strip()

    math_mark_str = fields[5]
    math_mark = float(math_mark_str)

    if reg_code == "B":
        math_B.append(math_mark)

math_B_mean = sum(math_B) / len(math_B)
print(math_B_mean)
```

File parsing – Solution

An Error!

```
ValueError: could not convert string to float: AA
```


File parsing – debugging

```
math_B = [] # empty list to store marks
for line in open("sslc1.txt"):
    fields = line.split(";")
    reg_code = fields[0]
    reg_code = reg_code.strip()
    print(fields) # <-- Added
    math_mark_str = fields[5]
    math_mark = float(math_mark_str)
    # ...
```

File parsing – debugging

```
math_B = [] # empty list to store marks
for i, line in enumerate(open("sslcl.txt")):
    fields = line.split(";")
    reg_code = fields[0]
    reg_code = reg_code.strip()
    print(i, fields) # <-- Added
    math_mark_str = fields[5]
    math_mark = float(math_mark_str)
    # ...
```

File parsing – Solution

```
math_B = [] # empty list to store marks
for line in open("sslcl.txt"):
    fields = line.split(";")
    reg_code = fields[0].strip()
    m = fields[5]
    mark = float(m) if m != 'AA' else 0
    if reg_code == "B":
        math_B.append(mark)

math_B_mean = sum(math_B) / len(math_B)
print(math_B_mean)
```

Summary

- Reading files
- Writing files
- Simple file parsing