python™

Unit 6

File processing

Special thanks to Roy McElmurry, John Kurkowski, Scott Shawcroft, Ryan Tucker, Paul Beck for their work.

Reading Files

name = open("**filename**")

- opens the given file for reading, and returns a file object

name.read() – file's entire contents as a string

name.readline() – next line from file as a string

name.readlines() – file's contents as a list of lines

- the lines from a file object can also be read using a `for` loop

```
>>> f = open("hours.txt")
>>> f.read()
'123 Susan 12.5 8.1 7.6 3.2\n
456 Brad 4.0 11.6 6.5 2.7 12\n
789 Jenn 8.0 8.0 8.0 8.0 7.5\n'
```

python™

File Input Template

- A template for reading files in Python:

```
name = open("filename")
for line in name:
    statements
```

```
>>> input = open("hours.txt")
>>> for line in input:
...     print(line.strip())   # strip() removes \n

123 Susan 12.5 8.1 7.6 3.2
456 Brad 4.0 11.6 6.5 2.7 12
789 Jenn 8.0 8.0 8.0 8.0 7.5
```

Exercise

- Write a function `input_stats` that accepts a file name as a parameter and that reports the longest line in the file.
  - example input file, `carroll.txt`:

```
Beware the Jabberwock, my son,
the jaws that bite, the claws that catch,
Beware the JubJub bird and shun
the frumious bandersnatch.
```

  - expected output:

```
>>> input_stats("carroll.txt")
longest line = 42 characters
the jaws that bite, the claws that catch,
```

Exercise Solution

```
def input_stats(filename):
    input = open(filename)
    longest = ""
    for line in input:
        if len(line) > len(longest):
            longest = line

    print("Longest line =", len(longest))
    print(longest)
```

Recall: String Methods

Java	Python
`length`	`len(str)`
`startsWith, endsWith`	`startswith, endswith`
`toLowerCase, toUpperCase`	`upper, lower, isupper, islower, capitalize, swapcase`
`indexOf`	`find`
`trim`	`strip`
	`ord, chr`

```
>>> name = "Martin Douglas Stepp"
>>> name.upper()
'MARTIN DOUGLAS STEPP'
>>> name.lower().startswith("martin")
True
>>> len(name)
20
```

String Splitting

- `split` breaks a string into tokens that you can loop over.

  name`.split()` `# break by whitespace`

  name`.split(`**delimiter**`)` `# break by delimiter`

- `join` performs the opposite of a `split`

  delimiter`.join(`**list of tokens**`)`

```
>>> name = "Brave Sir Robin"
>>> for word in name.split():
...     print(word)
Brave
Sir
Robin
>>> "LL".join(name.split("r"))
'BLLave SiLL Robin
```

Splitting into Variables

- If you know the number of tokens, you can `split` them directly into a sequence of variables.

  var1, **var2**, ..., **varN** = **string**.split()

- may want to convert type of some tokens: **type**(**value**)

```
>>> s = "Jessica 31 647.28"
>>> name, age, money = s.split()
>>> name
'Jessica'
>>> int(age)
31
>>> float(money)
647.28
```

Exercise

- Suppose we have this `hours.txt` data:

```
123 Suzy 9.5 8.1 7.6 3.1 3.2
456 Brad 7.0 9.6 6.5 4.9 8.8
789 Jenn 8.0 8.0 8.0 8.0 7.5
```

- Compute each worker's total hours and hours/day.
  - Assume each worker works exactly five days.

```
Suzy ID 123 worked 31.4 hours: 6.3 / day
Brad ID 456 worked 36.8 hours: 7.36 / day
Jenn ID 789 worked 39.5 hours: 7.9 / day
```

Exercise Answer

hours.py

```
 1  input = open("hours.txt")
 2  for line in input:
 3      id, name, mon, tue, wed, thu, fri = line.split()
 4
 5      # cumulative sum of this employee's hours
 6      hours = float(mon) + float(tue) + float(wed) + \
 7              float(thu) + float(fri)
 8
 9      print(name, "ID", id, "worked", \
10            hours, "hours: ", hours/5, "/ day"
```

Writing Files

name = open("**filename**", "w")
name = open("**filename**", "a")

- opens file for write (deletes previous contents), or
- opens file for append (new data goes after previous data)

name.write(**str**) – writes the given string to the file

name.close() – saves file once writing is done

```
>>> out = open("output.txt", "w")
>>> out.write("Hello, world!\n")
>>> out.write("How are you?")
>>> out.close()

>>> open("output.txt").read()
'Hello, world!\nHow are you?'
```

Exercise

- Write code to read a file of gas prices in USA and Belgium:

```
8.20    3.81    3/21/11
8.08    3.84    3/28/11
8.38    3.92    4/4/11
...
```

- Output the average gas price for each country to an output file named `gasout.txt`.