Files in Python

- PoPI

Files

- File is a sequence of bytes (each byte has a value from 0 to 255)
  - E.g.: 25 46 0 250 32 0 45 250 32 0 42 145 32 0 45 250 32 0 45 145 …
    - If the file size is 1 Kbyte how many bytes are there?
- A special case of file is a text file
  - E.g.: I[SPACE]love[SPACE]computer[SPACE]science[NEWLINE]And[SPACE]you?...
  - Which is displayed on your screen as
    - I love computer science
      And you?...
- Still a normal file: I = 73, [SPACE]=32,… (it is slightly different in Unicode)

Access to files

- How Python accesses information from files is very intuitive:
  - When a file is opened, a cursor is set to position 0 of the file
  - Then, typical actions are:
    - Read next four bytes in the file, store them in a list of integers, and move the cursor by 4
    - Write four bytes into the file…
    - Move the cursor by 10 positions/bytes
    - Write a string to the text file and move the cursor by its length
    - Read a sequence of symbols from a text file which ends by [NEWLINE] and store in a string
      - Effectively reads one line
      - [NEWLINE] is \n

Reading from files

- Opening a file

  infile = open(“file1.txt”, "r")   or “rb” to inform Python the file is binary

- Read a number of symbols and return them as a string (and shift cursor)

  str = infile.read(10)

- Read a number of bytes and return them as a list

  ten_bytes = infile.read(10)

- Read a whole line and return as string:

  line = infile.readline()

  - reading includes \n symbol too, so line will end with a character \n
  - use line.rstrip() to remove that character
  - use

    words = line.split(‘ ‘)

    to get a list of words of line
  - infile.readline() returns empty string if cursor is at the end of file
    - useful to read a file in a while loop and check when to terminate the loop

Writing to files

- Opening a file

  outfile = open(“file2.txt”, “w") or “wb” to inform Python the file is binary

- Convenient output formatting tool is string substitution % operator

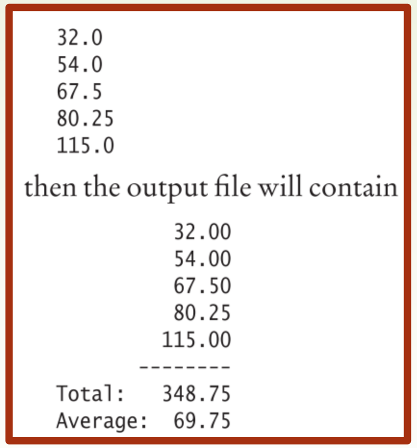
  formatted_string = string_where_substitutions_to_be_done % values

```
N = 42
formatted_string = “I have spotted %d camels” % N     # %d says that N should be
                                                      # formatted as decimal integer
print(formatted_string)
>>I have spotted 42 camels
value = 3.14159265359
print("%15.2f\n is pi" % value)        # "%15.2f says that value should be printed as float
>>          3.14                       # occupying 15 chars and only 2 chars after .
>> is pi
print(“I have spotted %d camels and pi is %15.2f” % (N, value))     #combined together
```

- Write a string to a file

  outfile.write(str)

Example



```
32.0
54.0
67.5
80.25
115.0
```

then the output file will contain

```
           32.00
           54.00
           67.50
           80.25
          115.00
        --------
Total:    348.75
Average:   69.75
```

```
# This program reads a file
#containing numbers and writes
#numbers to another file, lined up
# in a column and followed by their total
# and average.
#Prompt the user for the name of the input
# and output files.

inputFileName = input("Input file name: ")
outputFileName = input("Output file name: ")

# Open the input and output files.
infile = open(inputFileName, "r")
outfile = open(outputFileName, "w")
```

```
# Read the input and write the output.
total = 0.0
count = 0
line = infile.readline()
while line != "" :
    value = float(line)
    outfile.write("%15.2f\n" % value)
    total = total + value
    count = count + 1
    line = infile.readline()

# Output the total and average.
outfile.write("%15s\n" % "--------")
outfile.write("Total: %8.2f\n" % total)
avg = total / count
outfile.write("Average: %6.2f\n" % avg)

# Close the files.
infile.close()
outfile.close()
```

Reading The Whole File at Once

- Before we were reading the file line-by-line

```
line = infile.readline()
while line != "" :
        DoSomethingWith line
        line = infile.readline()
```

- Alternatively, you can read the whole text file at once

```
lines = infile.readlines()                    #returns a list of strings

for line in lines:

         DoSomethingWith line
```

- Do it with caution:
  - OK if you know that file contains a configuration of a 8x8 chessboard
  - Not OK if file contains a database of employees of potentially large company (500 MB)