File IO

Introduction to Computer Programming

Dr. Paul Vrbik

October 24, 2018

Memory Hierarchy

Type	Order	This Computer
CPU Cache L2	KB	256KB
CPU Cache L3	MB	8MB
Random Access Memory	GB	16GB
Disk	GB/TB	256GB
Cloud	PB	Functionally Infinite.

Why Files?

When Python launches it gets allocated space in RAM and it is possible to fill it up. This is a problem if we want to solve memory intensive problems (e.g. analyzing tweets, payroll, or scientific computing).

It will be necessary to instruct Python to use the disk memory one level up.

Example

The following is the contents of the file `hello.txt`.

```
What a\n
wonderful\n
hello world.\n
```

Here `\n` denotes a new-line character which normally is not displayed.

Open a file

To open a file assumed to be in the same directory as where your running Python (see: `>>> os.getcwd()`) do

```
file = open("file.dat", "<mode>")
```

Mode	Description
`r`	read
`w`	write
`a`	append

Note: append means write at end of file.

Read a file

```
>>> file = open("hello.txt", "r")
>>> file.readline()
'What a\n'
>>> file.readline()
'wonderful\n'
>>> file.readline()
'hello world.\n'
>>> file.readline()
''
>>> file.readline()
''
```

open for read.

Read a file

```
>>> file = open("hello.txt", "r")                    open for read.
>>> for line in file:
...     print( line )
What a\n
\n
wonderful\n
\n
hello world.\n
\n
```

The \n are inserted by Python's print.

Observe what happens if we repeat the loop from the last slide.

```
>>> for line in file:
...     print(line)
>>>
```

This is because the file-pointer `file` has already reached the end of the file the last loop.

We need to reset the file pointer to the beginning.

```
>>> file = open("hello.txt", "r")
>>> for line in file:
...     line
‘What a\n’
‘wonderful\n’
‘hello world.\n’
```

Closing files

If you do not close your files they remain open and vulnerable to side-effects.

That is, you may find some of your file is missing or extra bits in your file when you neglect to close after use.

```
>>> file = open("hello.txt", "r")
>>> for line in file:
...     line
'What a'\n
'wonderful'\n
'hello world.'\n
>>> file.close()
```

Alternative to close

Using the `with` construct has the advantage of closing your file for you — even if the program crashes while it is executing its code block.

```
with open("hello.txt", "r") as file:
    for line in file:
        print(line)
```

Question

Write a function that counts the number of empty lines in a file.

Answer

```
from typing import TextIO

def num_empty_lines(file:TextIO) -> int:
```

When reading from a file we are always reading strings.

Do not forget to cast your strings to a more appropriate types when necessary.

```
>>> with open("numbers.dat", "<mode>") as file:
>>>     ans = []
>>>     for line in file:
...         ans.append( int(line) )
>>> ans
[1, 2, 3]
```

Question

Write a function that finds the most popular band in a file that contains lines that look like:

```
band:str, rating:int, plays:int
```

Answer

```
from typing import TextIO
def highest_rated_band(file:TextIO) -> int:
```

Question

Extend the previous answer to take an arbitrary file with a header of attributes like

```
name,grade,age
```

and write a function that finds the maximum of that attribute.

Answer

```
from typing import TextIO
def most(file:TextIO, attribute:str):
```

Writing to Files

```
>>> with open("numbers.dat", "w") as file:
...     file.write("Hello World.\n")
```

Write single string.

```
...     file.writelines(["Hello\n", "World\n"])
```

Write a list of strings.

Careful!

Opening a file for write will create the `numbers.dat` file or overwrite the old one if it exists.

Appending to Files

Appending will open a file without over-writing, instead appending to the end of the file.

A file is created if one does not exist.

```
>>> with open("numbers.dat", "a") as file:
...     file.append("Hello World.\n")
```

What you should be doing.

Prepare for your exam!