Lecture 9: files and streams

Files

- `open(filename, mode)` returns a *file object*
  - `filename` is a path to a file
  - `mode` is a string where
    - 'r' - open for reading (default)
    - 'w' - open for writing, truncating the file first
    - 'x' - open for exclusive creation, failing if the file already exists
    - 'a' - open for writing, appending to the end of the file if it exists
    - 'b' - binary mode
    - 't' - text mode (default)
    - '+' - open a disk file for updating (reading and writing)
- file objects are *iterable* and support methods to *read*, *write*, and *flush* data as well as *close* the file

cat.py - streaming

```
1  import sys
2
3  fin = open(sys.argv[1],'r')
4  for line in fin:
5      print(line, end='')
6  fin.close()
```

cat.py - non-streaming

```
1  import sys
2
3  fin = open(sys.argv[1], 'r')
4  print(fin.read())
5  fin.close()
```

make_big_file.py

```
1  import sys
2
3  def make_big_file(filename, num):
4      with open(filename, 'w') as fout:
5          for i in range(num):
6              print(i, file=fout)
7
8
9  if __name__ == '__main__':
10     make_big_file(sys.argv[1], int(sys.argv[2]))
```

odd lines using zip, itertools, and count

```
1  import sys
2  from itertools import count
3
4  with open(sys.argv[1], 'r') as fin:
5      for line, lineno in zip(fin, count(1)):
6          if (lineno % 2 == 1):
7              print(line, end='')
```

interleaving files

Write a program called `merge.py` that takes two files as input and outputs to the terminal the contents of those files interleaved. For example, suppose `file1` and `file2` have the following contents:

```
file1:
1
3
5
7
```

```
file2:
2
4
6
8
```

Now consider running `interleave.py` on those files

```
$ python3 interleave.py file1 file2
1
2
3
4
5
6
7
8
```

interleaving files

```
1  import sys
2
3  with open(sys.argv[1],'r') as fin1, open(sys.argv[2],'r') as fin2:
4      for line1,line2 in zip(fin1,fin2):
5          print(line1,end='')
6          print(line2,end='')
```