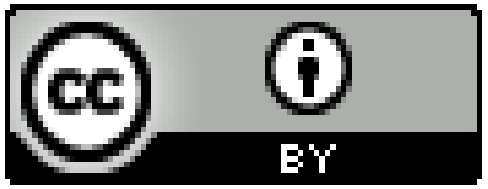
Python

Input and Output - Working with files



Copyright © Software Carpentry 2010
This work is licensed under the Creative Commons Attribution License
See http://software-carpentry.org/license.html for more information.

Been using `print` to see what programs are doing

Been using `print` to see what programs are doing

How to save data to files?

Been using `print` to see what programs are doing

How to save data to files?

And read data from them?

Been using `print` to see what programs are doing

How to save data to files?

And read data from them?

Python's solution looks very much like C's

Been using `print` to see what programs are doing

How to save data to files?

And read data from them?

Python's solution looks very much like C's

– A file is a sequence of bytes

Been using `print` to see what programs are doing

How to save data to files?

And read data from them?

Python's solution looks very much like C's

- A file is a sequence of bytes
- But it's often more useful to treat it as a sequence of lines

Sample data file: "haiku.txt"

```
Three things are certain:
Death, taxes, and lost data.
Guess which has occurred.

Errors have occurred.
We won't tell you where or why.
Lazy programmers.

With searching comes loss
and the presence of absence:
"My Thesis" not found.

A crash reduces
your expensive computer
to a simple stone.
```

How many characters in a file?

How many ~~characters~~ in a file?

bytes

How many ~~characters~~ in a file?

bytes ← Assume 1-to-1 for now

How many ~~characters~~ in a file?

bytes ← Assume 1-to-1 for now

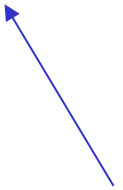
Revisit later

How many characters in a file?

```
with open('haiku.txt', 'r') as reader:
    data = reader.read()

print len(data)
```

How many characters in a file?

```
with open('haiku.txt', 'r') as reader:
    data = reader.read()

print len(data)
```



Create a file object

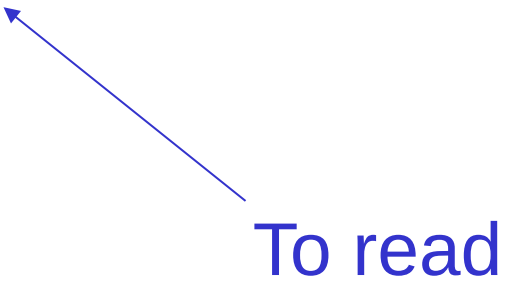
How many characters in a file?

```
with open('haiku.txt', 'r') as reader:
    data = reader.read()

print len(data)
```

File to connect to

How many characters in a file?

```
with open('haiku.txt', 'r') as reader:
    data = reader.read()

print len(data)
```



How many characters in a file?

```
with open('haiku.txt', 'r') as reader:
    data = reader.read()

print len(data)
```

Now holds file object

How many characters in a file?

```
with open('haiku.txt', 'r') as reader:
    data = reader.read()

print len(data)
```

Read entire content

of file into a string

How many characters in a file?

```
with open('haiku.txt', 'r') as reader:
    data = reader.read()

print len(data)
```

Now has a copy of all the bytes that were in the file

How many characters in a file?

```
with open('haiku.txt', 'r') as reader:
    data = reader.read()

print len(data)
```

Why don't we need to *close* the file?

Since Python now uses the "with" statement we can trust the file will be automatically closed when we leave the context of the "with" (indented) block.

How many characters in a file?

```
with open('haiku.txt', 'r') as reader:
    data = reader.read()

print len(data)
```

Report how many characters were read

How many characters in a file?

```
with open('haiku.txt', 'r') as reader:
    data = reader.read()

print len(data)
```

Report how many ~~characters~~ were read bytes

How many characters in a file?

```
with open('haiku.txt', 'r') as reader:
    data = reader.read()

print len(data)
```

293

If the file might be large, better to read in chunks

If the file might be large, better to read in chunks

```
with open('haiku.txt', 'r') as reader:
    data = reader.read(64)
    while data != '':
        print len(data)
        data = reader.read(64)
    print len(data)
```

If the file might be large, better to read in chunks

```
with open('haiku.txt', 'r') as reader:
    data = reader.read(64)
    while data != '':
        print len(data)
        data = reader.read(64)
    print len(data)
```

Read (at most) 64 bytes

If the file might be large, better to read in chunks

```
with open('haiku.txt', 'r') as reader:
    data = reader.read(64)
    while data != '':
        print len(data)
        data = reader.read(64)
    print len(data)
```

Read (at most) 64 bytes

Or the empty string

if there is no more data

If the file might be large, better to read in chunks

```
with open('haiku.txt', 'r') as reader:
    data = reader.read(64)
    while data != '':
        print len(data)
        data = reader.read(64)
    print len(data)
```

Keep looping as long as the last read returned some data

If the file might be large, better to read in chunks

```
with open('haiku.txt', 'r') as reader:
    data = reader.read(64)
    while data != '':
        print len(data)
        data = reader.read(64)
    print len(data)
```

Do something with the data

If the file might be large, better to read in chunks

```
with open('haiku.txt', 'r') as reader:
    data = reader.read(64)
    while data != '':
        print len(data)
        data = reader.read(64)
    print len(data)
```

(Try to) reload

If the file might be large, better to read in chunks

```
with open('haiku.txt', 'r') as reader:
    data = reader.read(64)
    while data != '':
        print len(data)
        data = reader.read(64)
    print len(data)
```

Should be 0 (or the loop would still be running)

If the file might be large, better to read in chunks

```
with open('haiku.txt', 'r') as reader:
    data = reader.read(64)
    while data != '':
        print len(data)
        data = reader.read(64)
    print len(data)
```

```
64
64
64
64
37
0
```

If the file might be large, better to read in chunks

```
with open('haiku.txt', 'r') as reader:
    data = reader.read(64)
    while data != '':
        print len(data)
        data = reader.read(64)
    print len(data)
```

```
64
64
64
64
37
0
```

Don't do this unless

If the file might be large, better to read in chunks

```
with open('haiku.txt', 'r') as reader:
    data = reader.read(64)
    while data != '':
        print len(data)
        data = reader.read(64)
    print len(data)
```

```
64
64
64
64
37
0
```

Don't do this unless the file really might be very large (or infinite)

More common to read one line at a time

More common to read one line at a time

```
with open('haiku.txt', 'r') as reader:
    line = reader.readline()
    total = 0
    count = 0
    while line != '':
        count += 1
        total += len(line)
        line = reader.readline()

print 'average', float(total) / float(count)
```

More common to read one line at a time

```
with open('haiku.txt', 'r') as reader:
    line = reader.readline()
    total = 0
    count = 0
    while line != '':
        count += 1
        total += len(line)
        line = reader.readline()

print 'average', float(total) / float(count)
```

Read a single line

More common to read one line at a time

```
with open('haiku.txt', 'r') as reader:
    line = reader.readline()
    total = 0
    count = 0
    while line != '':
        count += 1
        total += len(line)
        line = reader.readline()

print 'average', float(total) / float(count)
```

Keep looping until

no more lines in file

More common to read one line at a time

```
with open('haiku.txt', 'r') as reader:
    line = reader.readline()
    total = 0
    count = 0
    while line != '':
        count += 1
        total += len(line)
        line = reader.readline()

print 'average', float(total) / float(count)
```

(Try to) reload

More common to read one line at a time

```
with open('haiku.txt', 'r') as reader:
    line = reader.readline()
    total = 0
    count = 0
    while line != '':
        count += 1
        total += len(line)
        line = reader.readline()

print 'average', float(total) / float(count)
```

Average 19.53333333

Often more convenient to read all lines at once

Often more convenient to read all lines at once

```
with open('haiku.txt', 'r') as reader:
    contents = reader.readlines()
    total = 0
    count = 0
    for line in contents:
        count += 1
        total += len(line)

print 'average', float(total) / float(count)
```

Often more convenient to read all lines at once

```
with open('haiku.txt', 'r') as reader:
    contents = reader.readlines()
    total = 0
    count = 0
    for line in contents:
        count += 1
        total += len(line)

print 'average', float(total) / float(count)
```

All lines in file

as list of strings

Often more convenient to read all lines at once

```
with open('haiku.txt', 'r') as reader:
    contents = reader.readlines()
    total = 0
    count = 0
    for line in contents:
        count += 1
        total += len(line)

print 'average', float(total) / float(count)
```

Loop over lines with `for`

Often more convenient to read all lines at once

```
with open('haiku.txt', 'r') as reader:
    contents = reader.readlines()
    total = 0
    count = 0
    for line in contents:
        count += 1
        total += len(line)

print 'average', float(total) / float(count)
```

Average 19.53333333

"Read lines as list" + "loop over list" is common idiom

"Read lines as list" + "loop over list" is common idiom

So Python provides "loop over lines in file"

"Read lines as list" + "loop over list" is common idiom

So Python provides "loop over lines in file"

```
with open('haiku.txt', 'r') as reader:
    total = 0
    count = 0
    for line in reader:
        count += 1
        total += len(line)

    print 'average', float(total) / float(count)
```

"Read lines as list" + "loop over list" is common idiom

So Python provides "loop over lines in file"

```
with open('haiku.txt', 'r') as reader:
    total = 0
    count = 0
    for line in reader:
        count += 1
        total += len(line)

    print 'average', float(total) / float(count)
```

Assign lines of text in file

to loop variable one by one

"Read lines as list" + "loop over list" is common idiom

So Python provides "loop over lines in file"

```
with open('haiku.txt', 'r') as reader:
    total = 0
    count = 0
    for line in reader:
        count += 1
        total += len(line)

    print 'average', float(total) / float(count)
```

```
19.53333333
```

Put data in a file using `write` or `writelines`

Put data in a file using `write` or `writelines`

```
with open('temp.txt', 'w') as writer:
    writer.write('elements')
    writer.writelines(['He', 'Ne', 'Ar', 'Kr'])
```

Put data in a file using `write` or `writelines`

```
with open('temp.txt', 'w') as writer:
    writer.write('elements')
    writer.writelines(['He', 'Ne', 'Ar', 'Kr'])
```

Same function

Put data in a file using `write` or `writelines`

```
with open('temp.txt', 'w') as writer:
    writer.write('elements')
    writer.writelines(['He', 'Ne', 'Ar', 'Kr'])
```

File to write to

(is created if it doesn't exist)

Put data in a file using `write` or `writelines`

```
with open('temp.txt', 'w') as writer:
    writer.write('elements')
    writer.writelines(['He', 'Ne', 'Ar', 'Kr'])
```

For writing instead of reading

Put data in a file using `write` or `writelines`

```
with open('temp.txt', 'w') as writer:
    writer.write('elements')
    writer.writelines(['He', 'Ne', 'Ar', 'Kr'])
```

Write a single string

Put data in a file using `write` or `writelines`

```
with open('temp.txt', 'w') as writer:
    writer.write('elements')
    writer.writelines(['He', 'Ne', 'Ar', 'Kr'])
```

Write each string

in a list as a line

Put data in a file using `write` or `writelines`

```
with open('temp.txt', 'w') as writer:
    writer.write('elements')
    writer.writelines(['He', 'Ne', 'Ar', 'Kr'])
```

```
elementsHeNeArKr
```

Put data in a file using `write` or `writelines`

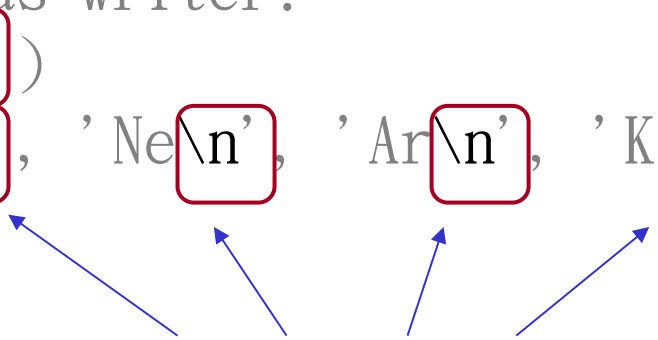
```
with open('temp.txt', 'w') as writer:
    writer.write('elements')
    writer.writelines(['He', 'Ne', 'Ar', 'Kr'])
```

```
elementsHeNeArKr
```

Python only writes what you tell it to

Put data in a file using `write` or `writelines`

```
with open('temp.txt', 'w') as writer:
    writer.write('elements\n')
    writer.writelines(['He\n', 'Ne\n', 'Ar\n', 'Kr\n'])
```



Have to provide end-of-line characters yourself

Put data in a file using `write` or `writelines`

```
with open('temp.txt', 'w') as writer:
    writer.write('elements\n')
    writer.writelines(['He\n', 'Ne\n', 'Ar\n', 'Kr\n'])
```

```
elements
He
Ne
Ar
Kr
```