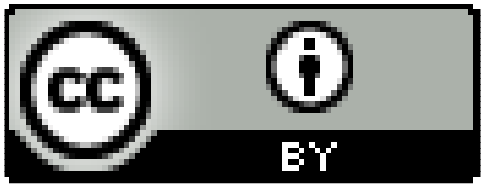
Python

Input and Output



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

```
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

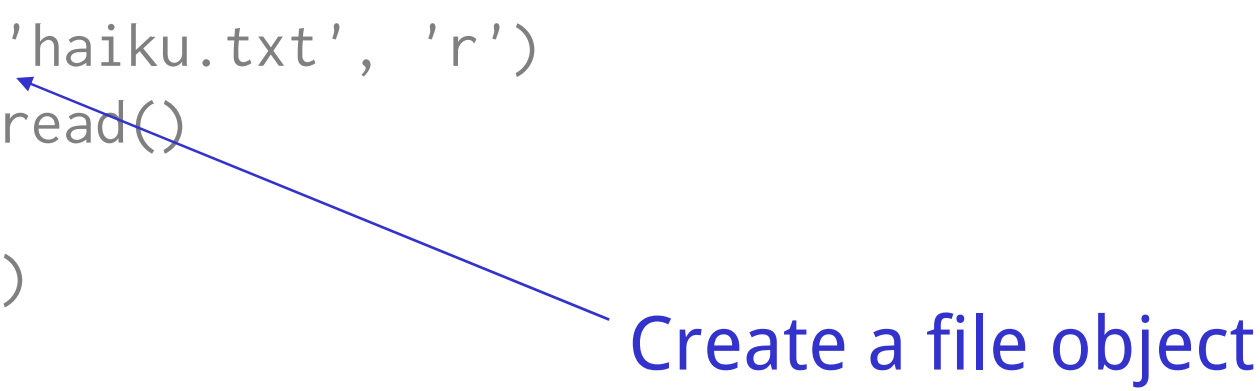
Revisit later

How many characters in a file?

```
reader = open('haiku.txt', 'r')
data = reader.read()
reader.close()
print len(data)
```

How many characters in a file?

```
reader = open('haiku.txt', 'r')
data = reader.read()
reader.close()
print len(data)
```

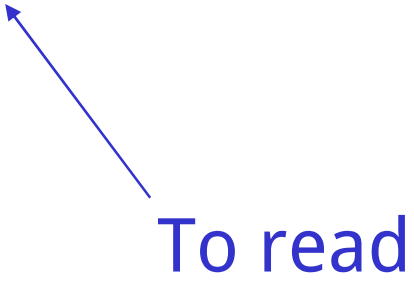


How many characters in a file?

```
reader = open('haiku.txt', 'r')
data = reader.read()
reader.close()
print len(data)
```

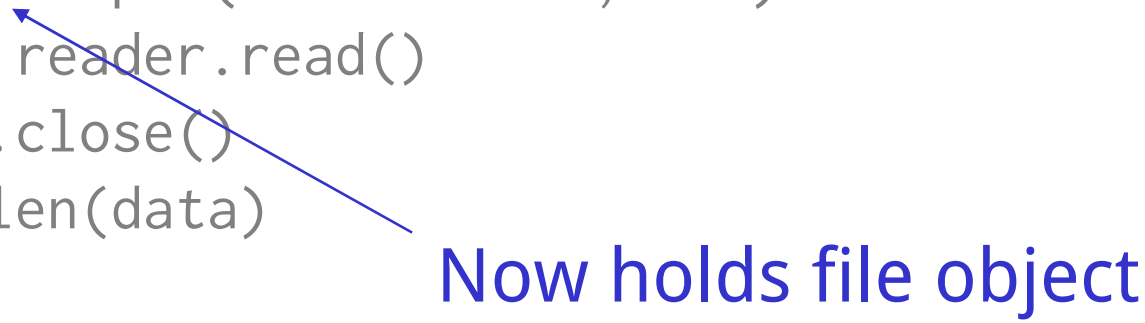
How many characters in a file?

```
reader = open('haiku.txt', 'r')
data = reader.read()
reader.close()
print len(data)
```



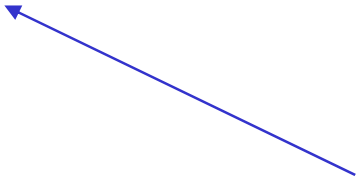
How many characters in a file?

```
reader = open('haiku.txt', 'r')
data = reader.read()
reader.close()
print len(data)
```



How many characters in a file?

```
reader = open('haiku.txt', 'r')
data = reader.read()
reader.close()
print len(data)
```



Read entire content of file into a string

How many characters in a file?

```
reader = open('haiku.txt', 'r')
data = reader.read()
reader.close()
print len(data)
```

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

How many characters in a file?

```
reader = open('haiku.txt', 'r')
data = reader.read()
reader.close()
print len(data)
```

Disconnect from the file

How many characters in a file?

```
reader = open('haiku.txt', 'r')
data = reader.read()
reader.close()
print len(data)
```

Disconnect from the file

Not strictly necessary

in small programs, but

good practice

How many characters in a file?

```
reader = open('haiku.txt', 'r')
data = reader.read()
reader.close()
print len(data)
```

Report how many characters were read

How many characters in a file?

```
reader = open('haiku.txt', 'r')
data = reader.read()
reader.close()
print len(data)
```

Report how many ~~characters~~ were read bytes

How many characters in a file?

```
reader = open('haiku.txt', 'r')
data = reader.read()
reader.close()
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

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

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

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

Read (at most) 64 bytes

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

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

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

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

Keep looping as long as the last read returned some data

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

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

Do something with the data

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

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

(Try to) reload

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

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

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

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

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

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

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

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

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

Don't do this unless

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

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

```
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

```
reader = open('haiku.txt', 'r')
line = reader.readline()
total = 0
count = 0
while line != '':
    count += 1
    total += len(line)
    line = reader.readline()
reader.close()
print 'average', float(total) / float(count)
```

More common to read one line at a time

```
reader = open('haiku.txt', 'r')
line = reader.readline()
total = 0
count = 0
while line != '':
    count += 1
    total += len(line)
    line = reader.readline()
reader.close()
print 'average', float(total) / float(count)
```

Read a single line

More common to read one line at a time

```
reader = open('haiku.txt', 'r')
line = reader.readline()
total = 0
count = 0
while line != '':
    count += 1
    total += len(line)
    line = reader.readline()
reader.close()
print 'average', float(total) / float(count)
```

Keep looping until

no more lines in file

More common to read one line at a time

```
reader = open('haiku.txt', 'r')
line = reader.readline()
total = 0
count = 0
while line != '':
    count += 1
    total += len(line)
    line = reader.readline()
reader.close()
print 'average', float(total) / float(count)
```

(Try to) reload

More common to read one line at a time

```
reader = open('haiku.txt', 'r')
line = reader.readline()
total = 0
count = 0
while line != '':
    count += 1
    total += len(line)
    line = reader.readline()
reader.close()
print 'average', float(total) / float(count)
```

```
19.53333333
```

Often more convenient to read all lines at once

Often more convenient to read all lines at once

```
reader = open('haiku.txt', 'r')
contents = reader.readlines()
reader.close()
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

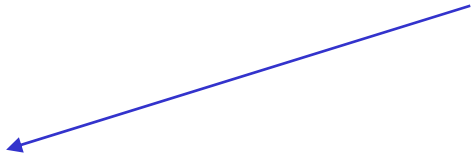
```
reader = open('haiku.txt', 'r')
contents = reader.readlines()
reader.close()
total = 0
count = 0
for line in contents:
    count += 1
    total += len(line)
print 'average', float(total) / float(count)
```

All lines in file

as list of strings

Often more convenient to read all lines at once

```
reader = open('haiku.txt', 'r')
contents = reader.readlines()
reader.close()
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

```
reader = open('haiku.txt', 'r')
contents = reader.readlines()
reader.close()
total = 0
count = 0
for line in contents:
    count += 1
    total += len(line)
print 'average', float(total) / float(count)
```

```
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"

```
reader = open('haiku.txt', 'r')
total = 0
count = 0
for line in reader:
    count += 1
    total += len(line)
reader.close()
print 'average', float(total) / float(count)
```

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

So Python provides "loop over lines in file"

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

```
reader = open('haiku.txt', 'r')
total = 0
count = 0
for line in reader:
    count += 1
    total += len(line)
reader.close()
print 'average', float(total) / float(count)
```

```
19.53333333
```

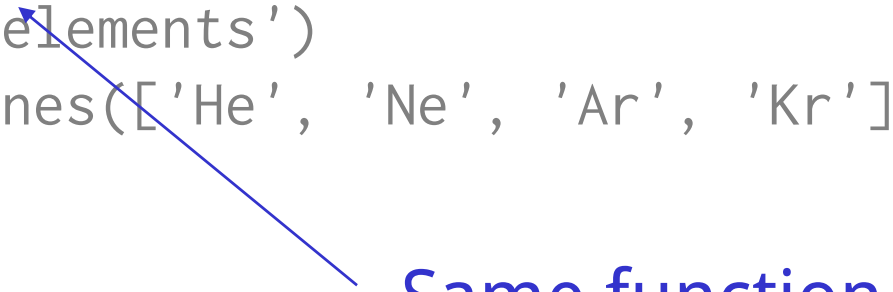
Put data in a file using `write` or `writelines`

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

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

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

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



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

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

File to write to

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

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

File to write to

Created if it doesn't exist

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

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

For writing instead of reading

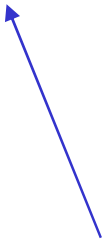
Put data in a file using `write` or `writelines`

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

Write a single string

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

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



Write each string in a list

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

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

```
elementsHeNeArKr
```

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

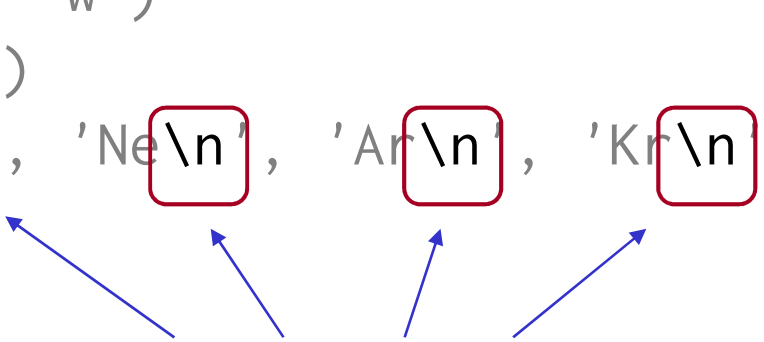
```
writer = open('temp.txt', 'w')
writer.write('elements')
writer.writelines(['He', 'Ne', 'Ar', 'Kr'])
writer.close()
```

```
elementsHeNeArKr
```

Python only writes what you tell it to

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

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



Have to provide end-of-line characters yourself

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

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

```
elements
He
Ne
Ar
Kr
```

Often simpler to use `print >>`

Often simpler to use `print >>`

```
writer = open('temp.txt', 'w')
print >> writer, 'elements'
for gas in ['He', 'Ne', 'Ar', 'Kr']:
    print >> writer, gas
writer.close()
```

Often simpler to use `print >>`

```
writer = open('temp.txt', 'w')
print >> writer, 'elements'
for gas in ['He', 'Ne', 'Ar', 'Kr']:
    print >> writer, gas
writer.close()
```

Specify open file after >>

Often simpler to use `print >>`

```
writer = open('temp.txt', 'w')
print >> writer, 'elements'
for gas in ['He', 'Ne', 'Ar', 'Kr']:
    print >> writer, gas
writer.close()
```

`print` automatically adds the newline

Copy a file

Copy a file

```
reader = open('haiku.txt', 'r')
data = reader.read()
reader.close()
writer = open('temp.txt', 'w')
write.write(data)
writer.close()
```

Copy a file

```
reader = open('haiku.txt', 'r')
data = reader.read()
reader.close()
writer = open('temp.txt', 'w')
write.write(data)
writer.close()
```

Copy a file

```
reader = open('haiku.txt', 'r')
data = reader.read()
reader.close()
writer = open('temp.txt', 'w')
write.write(data)
writer.close()
```

Copy a file

```
reader = open('haiku.txt', 'r')
data = reader.read()
reader.close()
writer = open('temp.txt', 'w')
write.write(data)
writer.close()
```

Probably won't work with a terabyte...

Copy a file

```
reader = open('haiku.txt', 'r')
data = reader.read()
reader.close()
writer = open('temp.txt', 'w')
write.write(data)
writer.close()
```

Probably won't work with a terabyte...

...but we probably don't care

Copy a file (version 2)

Copy a file (version 2)

```
reader = open('haiku.txt', 'r')
writer = open('temp.txt', 'w')
for line in reader:
    writer.write(line)
reader.close()
writer.close()
```

Copy a file (version 2)

```
reader = open('haiku.txt', 'r')
writer = open('temp.txt', 'w')
for line in reader:
    writer.write(line)
reader.close()
writer.close()
```

Assumes the file is text

Copy a file (version 2)

```
reader = open('haiku.txt', 'r')
writer = open('temp.txt', 'w')
for line in reader:
    writer.write(line)
reader.close()
writer.close()
```

Assumes the file is text

Or at least that the end-of-line character appears

frequently

This version doesn't make an exact copy

This version doesn't make an exact copy

```
reader = open('haiku.txt', 'r')
writer = open('temp.txt', 'w')
for line in reader:
    print >> writer, line
reader.close()
writer.close()
```

This version doesn't make an exact copy

```
reader = open('haiku.txt', 'r')
writer = open('temp.txt', 'w')
for line in reader:
    print >> writer, line
reader.close()
writer.close()
```

Python keeps the newline when reading

This version doesn't make an exact copy

```
reader = open('haiku.txt', 'r')
writer = open('temp.txt', 'w')
for line in reader:
    print >> writer, line
reader.close()
writer.close()
```

Python keeps the newline when reading

`print` automatically adds a newline

This version doesn't make an exact copy

```
reader = open('haiku.txt', 'r')
writer = open('temp.txt', 'w')
for line in reader:
    print >> writer, line
reader.close()
writer.close()
```

Python keeps the newline when reading

`print` automatically adds a newline

Result is double-spaced output

Copy a file (version 3)

Copy a file (version 3)

```
BLOCKSIZE = 1024
reader = open('haiku.txt', 'r')
writer = open('temp.txt', 'w')
data = reader.read(BLOCKSIZE)
while len(data) > 0:
    writer.write(data)
    data = reader.read(BLOCKSIZE)
reader.close()
writer.close()
```

Copy a file (version 3)

```
BLOCKSIZE = 1024
reader = open('haiku.txt', 'r')
writer = open('temp.txt', 'w')
data = reader.read(BLOCKSIZE)
while len(data) > 0:
    writer.write(data)
    data = reader.read(BLOCKSIZE)
reader.close()
writer.close()
```

(Needlessly?) harder to understand

created by

Greg Wilson

October 2010