Files

Reading Files

In [1]:
```
f = open('10_01_file.txt', 'r')
print(f)
```

```
<_io.TextIOWrapper name='10_01_file.txt' mode='r' encoding='UTF-8'>
```

In [4]:
```
f.readline()
```

Out[4]:
```
'Simple is better than complex.\n'
```

In [5]:
```
f.readlines()
```

Out[5]:
```
['Complex is better than complicated.\n',
 'Flat is better than nested.\n',
 'Sparse is better than dense.\n',
 'Readability counts.\n',
 "Special cases aren't special enough to break the rules.\n",
 'Although practicality beats purity.\n',
 'Errors should never pass silently.\n',
 'Unless explicitly silenced.\n',
 'In the face of ambiguity, refuse the temptation to guess.\n',
 'There should be one-- and preferably only one --obvious way to do it.\n',
 "Although that way may not be obvious at first unless you're Dutch.\n",
 'Now is better than never.\n',
 'Although never is often better than *right* now.\n',
 "If the implementation is hard to explain, it's a bad idea.\n",
 'If the implementation is easy to explain, it may be a good idea.\n',
 "Namespaces are one honking great idea -- let's do more of those!"]
```

In [7]:
```
f = open('10_01_file.txt', 'r')
for line in f.readlines():
    print(line.strip())
```

```
Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
```

Writing Files

In [11]:
```
f = open('10_01_output.txt', 'w')
print(f)
```

```
<_io.TextIOWrapper name='10_01_output.txt' mode='w' encoding='UTF-8'>
```

In [12]:
```
f.write('Line 1\n')
f.write('Line 2\n')
```

Out[12]:
```
7
```

In [13]:
```
f.close()
```

Appending Files

In [14]:
```
f = open('10_01_output.txt', 'a')
f.write('Line 3\n')
f.write('Line 4\n')
f.close()
```

In [15]:
```
with open('10_01_output.txt', 'a') as f:
    f.write('some stuff\n')
    f.write('some other stuff\n')

print(f)
```

```
<_io.TextIOWrapper name='10_01_output.txt' mode='a' encoding='UTF-8'>
```

In [16]:
```
f.write('PS. I forgot some stuff')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [16], in <module>
----> 1 f.write('PS. I forgot some stuff')

ValueError: I/O operation on closed file.
```

In [ ]:
```

```