3e-File.Write.Read

November 2, 2024

1 Writing to a File

1.1 Using the Shell

Using echo and the shell's redirect operator.

```
[6]: !echo hello world > hello.world.v01.txt
```

```
[7]: ls -la hello.world.v01.txt
```

```
-rw-r--r-- 1 root root 12 Jun 26 20:37 hello.world.v01.txt
```

```
[8]: !cat -n hello.world.v01.txt
```

```
     1  hello world
```

Using a "here document" structure.

```
[9]: %%bash
cat <<'asdfasdfasdf' > hello.world.v02.txt
  hello
  world
asdfasdfasdf
```

```
[10]: !cat -n hello.world.v02.txt
```

```
     1    hello
     2    world
```

1.2 Using writefile Magic

The `%%writefile` magic takes all the text after the magic and writes it to the specified file.

```
[11]: %%writefile hello.world.v03.txt
hello
world
```

```
Writing hello.world.v03.txt
```

```
[12]: !cat -n hello.world.v03.txt
```

```
     1  hello
     2  world
```

1.3 Using Python

The most “Pythonic” way to write to a file is to use a “context manager”. The `with` command creates a code block that automatically handles, i.e. provides a context for, closing a file.

[13]:
```
with open("hello.world.v04.txt", "w") as file:
  file.write("hello world")
```

[14]:
```
!cat -n hello.world.v04.txt
```

```
     1  hello world
```

[15]:
```
ls -la hello*
```

```
-rw-r--r-- 1 root root 12 Jun 26 20:37 hello.world.v01.txt
-rw-r--r-- 1 root root 16 Jun 26 20:37 hello.world.v02.txt
-rw-r--r-- 1 root root 12 Jun 26 20:38 hello.world.v03.txt
-rw-r--r-- 1 root root 11 Jun 26 20:39 hello.world.v04.txt
```

Another way to to use the pair `open` and `close`. This is generally avoided as it is easy to forget to include the `close`.

[16]:
```
file = open("hello.world.v05.txt", "w")
file.write("hello world\n")
file.close()
```

[17]:
```
!cat -n hello.world.v05.txt
```

```
     1  hello world
```

[18]:
```
ls -la hello*
```

```
-rw-r--r-- 1 root root 12 Jun 26 20:37 hello.world.v01.txt
-rw-r--r-- 1 root root 16 Jun 26 20:37 hello.world.v02.txt
-rw-r--r-- 1 root root 12 Jun 26 20:38 hello.world.v03.txt
-rw-r--r-- 1 root root 11 Jun 26 20:39 hello.world.v04.txt
-rw-r--r-- 1 root root 12 Jun 26 20:42 hello.world.v05.txt
```

2 Reading from a File

2.1 Using the Shell

The most common way to read from a file is to use `cat` and assign it to a Python variable. Each line becomes an entry in a list.

[19]:
```
!cat -n hello.world.v01.txt
```

```
     1  hello world
```

```
[20]: output = !cat -n hello.world.v01.txt
      print(output)
```

```
['     1\thello world']
```

```
[21]: output[0].upper()
```

```
[21]: '     1\tHELLO WORLD'
```

2.2 Using capture Magic

The `%%capture` magic reads all the output of a cell. If a variable is given, the output is assigned to the variable. Both the standard output and standard error are captured and can be accessed using the `#.stdout` and `#.stderr` attributes. They are both strings.

```
[28]: %%capture output
      !cat -n hello.world.v02.txt
```

```
[29]: print(output.stdout)
```

```
     1	hello
     2	world
```

```
[30]: print(output.stderr)
```

2.3 Using Python

Just the like writing, the most "Pythonic" way to read from a file is to use a "context manager".

```
[31]: output = ''
      with open("hello.world.v04.txt", "r") as file:
        output = file.read()

      print(output)
```

```
hello world
```

```
[36]: type(output)
```

```
[36]: str
```

```
[37]: type(output.split('\n'))
```

```
[37]: list
```

```
[38]: output.split('\n')
```

```
[38]: ['hello world']
```

You can also read one line at a time using a loop.

[39]:
```
!seq 1 10 > hello.world.v04.txt
```

[40]:
```
!cat -n hello.world.v04.txt
```

```
     1	1
     2	2
     3	3
     4	4
     5	5
     6	6
     7	7
     8	8
     9	9
    10	10
```

[41]:
```
output = ''
with open("hello.world.v04.txt", "r") as file:
  for line in file:
    output += line

print(output)
```

```
1
2
3
4
5
6
7
8
9
10

```

[42]:
```
output
```

[42]:
```
'1\n2\n3\n4\n5\n6\n7\n8\n9\n10\n'
```

[43]:
```
output = []
with open("hello.world.v04.txt", "r") as file:
  for line in file:
    output += [line.rstrip()]

print(output)
```

```
['1', '2', '3', '4', '5', '6', '7', '8', '9', '10']
```

3 Your turn

3.1 Part 1

Write some content to files using each of the three methods shown above - shell - %%writefile - Python open()

```
[ ]: # Solution
```

3.2 Part 2

Then use each of the three methods shown above to read the content into a Python variable. - shell - %%capture - Python open()

```
[ ]: # Solution
```

```
[44]: !cat --help
```

```
Usage: cat [OPTION]… [FILE]…
Concatenate FILE(s) to standard output.

With no FILE, or when FILE is -, read standard input.

  -A, --show-all           equivalent to -vET
  -b, --number-nonblank    number nonempty output lines, overrides -n
  -e                       equivalent to -vE
  -E, --show-ends          display $ at end of each line
  -n, --number             number all output lines
  -s, --squeeze-blank      suppress repeated empty output lines
  -t                       equivalent to -vT
  -T, --show-tabs          display TAB characters as ^I
  -u                       (ignored)
  -v, --show-nonprinting   use ^ and M- notation, except for LFD and TAB
      --help     display this help and exit
      --version  output version information and exit

Examples:
  cat f - g  Output f's contents, then standard input, then g's contents.
  cat        Copy standard input to standard output.

GNU coreutils online help: <https://www.gnu.org/software/coreutils/>
Full documentation <https://www.gnu.org/software/coreutils/cat>
or available locally via: info '(coreutils) cat invocation'
```

```
[45]: !cat -b hello.world.v01.txt
```

```
     1  hello world
```

[60]:
```
%%writefile foobar.txt
hello





world
```

```
Overwriting foobar.txt
```

[61]:
```
!cat -n foobar.txt
```

```
     1	hello\\t
     2
     3
     4
     5
     6
     7	world
```

[54]:
```
!cat -b foobar.txt
```

```
     1	hello





     2	world
```

[56]:
```
!cat -ns foobar.txt
```

```
     1	hello
     2
     3	world
```

[57]:
```
!cat -vet foobar.txt
```

```
hello$
$
$
$
$
$
world$
```

[ ]:
```

```