Chap_7 Cheat Sheet

by Theo666 via cheatography.com/193738/cs/40335/

File mode

r	Open a file for reading only. The file pointer is placed at the beginning of the file.
w	Open a file for writing only. Overwrites the file if the file exists. If the file does not exist, creates a new file.
a	Open a file for appending. The file pointer is placed at the end of the file. If the file does not exist, creates a new file for writing.
r+	Open a file for both reading and writing. The file pointer is placed at the beginning. Overwrites the exsiting content but the new content is shorter than the existing content, the remaining old content will still be present in the file.
w+	Open a file for both reading and writing. Overwrites the file if it exists. If it does not exist, creates a new file for reading and writing.
a+	Open a file for both reading and writing. The file pointer is placed at the end of the file. If it does not exist, creates a new file for reading and writing.

File mode (cont)

and so on

open(file, mdoe)
The main difference between the r+ and w+ modes is that w+ overwrites all the content in the file, while r+ keeps the remaining old content intact.
To add ew content to the beginning of a file without overwriting the old content, we can save the old content as an object then overwriting and rewrite the old content with 'w+' mode in Python

Common Escape Sequences

\n	newline
\t	tab
\\	backslash
\'	single quote
\"	double quote
\r	carriage return

- \" \' \\ to help us differentiate between characters that have a special meaning in the programming language and characters that are part of the string itself
- \r to move the cursor to the beginning of the current line without moving to the next line.
One common use case for \r is to create progress bars or other types of dynamic text output. To overwrite the previous output and create the illusion of a progress bar that updates in real-time.

Common Functions

`readli nes (fil e)`	read a file and returns them as a list of strings. Each element reprensents a line

Common Functions (cont)

readli-ne(file)	read a single line from a file and returns it as a string. The line includes the newline character \n
read(file)	read a file and returns it as a single string. The resulting string includes all the characters, including \n
seek(o-ffset)	change the position of the file handle, which is like a cursor that defines from where the data has to be read or written
next()	retrieve the next item from an iterator
f.write(c-ontent)	write content to a file.
f.writeli-nes()	write a list of strings to a file

- readline() returns "" while there's no the next line
- `f.seek (of fset, from_what)` offset specifies the number of positions to move, from_what = c(0, 1, 2) 0 (by default): the beginning of a file; 2 : the end of a file
- `f.write() f.writ elines` they not automatically add \n at the end of the string or the string element of a list
- Read methods systematically return content as string. Similarly, when you use f.write() and f.writelines(), you must provide a string and a list of strings

File opening

```
# Method 1
with open(file, mode = ' ') as
obejct :
       ...
#Method 2
object = open(file, mode)
object.cl ose()
# open multiple files
with open(f ile1, " r") as f1,
open(f ile2, " w") as f2:
       ...
```

Examples

```
# Iteration of a file using
readline() & while loop
with open(file, " r") as filin:
       line = filin.r ea dline()
          while line != " ":
                  pri nt( line)
                 line =
filin.r ea dline()
# keep a line using readline() &
next()
with open(' fil e.txt', 'r') as
f:
        line1 = f.read line()
        next(f)
        line3 = f.read line() #
return the line after next
# Iteration of a list resturned
by readli nes()
with open(file, " r") as filin:
        lines = filin.r ea dli -
nes() # [s:f] to keep line
       for line in lines:
                pri nt( line)
# Iteration directly for a file
# could return error, specially
too many strings
# but much more efficient than
readli nes()
with open(file, " r") as filin:
       for line in filin:
                pri nt( line)
```

Examples (cont)

```
> # change the position
with open(file, 'r') as f:
   f.seek(20)
   print(f.tell()) # Prints the new position
with open(file, 'r') as f:
   f.seek(-10, 2)
# writing
with open(file, "w") as filout:
    for element in my_list:
       filout.write(f"{element}\n")
# \r
import time
total = 10
for i in range(total + 1):
   progress = i / total * 100
   bar = '#' * i + '-' * (total - i)
   print(f'Progress: [{bar}] {progress:.1f}%',
end='\r')
   time.sleep(0.5)
print('\nDone!')
```