Chapter 10: Files

Last Update: May 2, 2023

You may have to prepare a few text files for the programs to open.

Program 1:

10-open.py

```
def next():
    input("Next> ")

print('\tOpening and printing a text file:')
f = open("file-1.txt","r")
for line in f:
    print(f"{line}")
print("\tWhy double space?")
f.close()
next()

print('\tReusing the varaible f for another file')
f = open("file-2.txt","r")
for line in f:
    line = line.strip()
    print(f"[{line}]")
print('---')
next()

print('\n\tBefore closing:', f.closed)
f.close()
print('\tAfter closing:', f.closed)
```

Program 2:

11-iter with.py

```
count = 0
print("\t*** Case 1: using with, add line number")
with open("file-1.txt") as f:
    for line in f:
        count += 1
        print(f"{count}: {line}")
print(f.closed)

print("\t*** Case 2: remove double space, checking closed")
with open("file-1.txt") as f:
    for line in f:
        print(line.rstrip())
    print(f.closed)
print("---")
print(f.closed)
```

Program 3:

10 13-iter except.py

```
 1 # This is what one should do when opening a file to make sure the
 2 # file is actually there.
 3 # File name can be stored in a variable or read from input()
 4 # Create file-1.txt with some text. Change to the first line
 5 # to file-9.txt to trigger the exception
 6 #
 7 fname = 'file-1.txt'
 8 try:
 9     f = open(fname, 'r')
10     print(f'*1* {fname} opened successfully.')
11     for line in f:
12         print(line, end='')
13 except FileNotFoundError as ex:
14     print(f'*2* FileNotFoundError in opening file: {fname} at
15 location XYZ.')
16     print(f'*3* An exception of type "{ex.__class__.__name__}"
17 occurred. Arguments: {ex.args}.')
18     print(ex)
```

Program 4:

10 20-readline.py

```
 1 # I prefer the first method of using readline().
 2 # Note that the second method produces an extra (blank) line.
 3 # This is a good example of "look ahead".
 4 # We can't decide whether we must print "line" until we
 5 # read the line.  That's the look-ahead.  It's kind of like
 6 # you don't know where is the end of a word until you see a blank
 7 # (or end-of-line) after the word.
 8 def next():
 9     input("Next> ")
10
11 print('---')
12 f = open('file-1.txt')
13 line = f.readline()
14 while line:
15     print(f"[{line}]")
16     line = f.readline()
17 print('---')
18 f.close()
19 next()
20
21 print('---')
22 f = open('file-1.txt')
23 line = " xxx "
24 while line:
25     line=f.readline()
26     print(f"[{line}]")
27 print('---')
28 f.close()
```

Program 5:

10	22-readline.py

```
 1 # Using strip() to clean up the two ends of a string
 2
 3 def next():
 4     input("Next> ")
 5
 6 f = open('file-1.txt')
 7 done = False
 8 print('---')
 9 while not done:
10     line = f.readline()
11     print(line.strip())
12     done = not line
13 print('---')
14 f.close()
15 next()
16
17 f = open('file-1.txt')
18 print('---')
19 while True:
20     line = f.readline().strip()
21     if line!='':
22         print(line)
23     else:
24         break
25 print('---')
26 f.close()
27 next()
28
29 f = open('file-1.txt')
30 print('---')
31 line = f.readline()
32 while line:
33     print(line.strip())
34     line = f.readline()
35 print('---')
36 f.close()
```

Program 6:

```
10 | 23-iter vs readline.py
```

```
 1 # Comparing the iterator method with readline()
 2 # Any difference?
 3
 4 def next():
 5     input("> ")
 6
 7 f = open('file-1.txt')
 8 for line in f:
 9     print(f"[{line}]")
10 f.close()
11 next()
12
13 f = open('file-1.txt')
14 line = f.readline()
15 while line:
16     print(f"[{line}]")
17     line=f.readline()
18 f.close()
```

Program 7:

```
10 | dd-xxxx.py
```

```
 1 f1 = open('test.txt','r')
 2 f2 = open('test2.txt', 'w')
 3 for line in f1:
 4     line = line.strip()
 5     print(f"[{line}]")
 6     print(line, file=f2)
 7 f1.close()
```

Program 8:

```
10 | 33-read.py
```

```
 1 # This example shows how to use read(). It returns one long string of
 2 # all characters in the file.
 3
 4 f = open('file-1.txt')
 5 lines = f.read()
 6 # eoln characters embedded in the string
 7
 8 print('---')
 9 print(f"[{lines}]")  # list
10 print('---')
11 print(f"{len(lines)} characters total")
12 print(f"{lines[35:46]}")
13 f.close()
```

Program 9:

10	34-read split eoln.py

```
 1 # Some commonly ways to use the string.
 2 def next():
 3     input('Next> ')
 4
 5 # using splitline() to separate the long string into lines
 6 # Note eoln removed
 7 import pprint
 8 # If you want to process a line at a time, do this ...
 9 f = open('file-1.txt')
10 lines = f.read().splitlines()
11 print('---')
12 for line in lines:
13     print(f"[{line}]")
14 print('---')
15 print(f'Number of lines: {len(lines)}')
16 f.close()
17 next()
18
19 # If you want to process a line at a time w/o eoln, do this ...
20 # strip() will remove extra spaces on either end
21 # do strip() after the split(), otherwise ....
22 f = open('file-1.txt')
23 lines = f.read().splitlines()
24 print('---')
25 for line in lines:
26     print(f"[{line.strip()}]")
27 print('---')
28 print(f'Number of lines: {len(lines)}')
29 f.close()
30 next()
31
32 # If you want to process a word at a time, do this
33 f = open('file-1.txt')
34 lines = f.read().split()
35 pprint.pprint(lines)
36 print(f'Length of the list: {len(lines)}')
37 f.close()
```

Program 10:

10	35-read except.py

```
# read, print, eoln embedded
fname = 'Fruits.txt'
try:
    file = open(fname, 'r')
    print(f'*** {fname} opened successfully ***')
    text = file.read()
    print(f'---\n{text}\n---')
except FileNotFoundError:
    print(FileNotFoundError, fname)
```

Program 11:

10	dd-xxxx.py

```
# Common ways to use readlines()

# list of lines with eoln
def next():
    input("> ")

f = open('file-1.txt','r')
lines = f.readlines()
print(lines)
next()

# printing each line
print('---')
for line in lines:
    print(f"[{line}]")
next()

# removing the eoln
print('---')
for line in lines:
    print(f"[{line.strip()}]")
print('---')
f.close()
```

Program 12:

```
10  51-read write.py
 1  fin = open("file-1.txt", 'r')
 2  fout = open("out2.txt", 'w')
 3  count = 0
 4  for line in fin:
 5      count += 1
 6      fout.write(f'{count:03d}: {line}')
 7  fin.close()
 8  fout.close()
 9  print('\nDone.  See the output file.')
```

Program 13:

```
10  52-writelines.py
 1  f1 = open("out1.txt",'w')
 2  f2 = open("out2.txt",'w')
 3  list1 =["One","Two","Three"]
 4  list2 =["One\n","Two\n","Three\n"]
 5  f1.writelines(list1)
 6  f2.writelines(list2)
```

Program 14:

```
10  54-write vs print.py
 1  f1 = open("out1.txt",'w')
 2  f2 = open("out2.txt",'w')
 3  list1 = "Hello"
 4  f1.write(list1)  # write to f1 using write()
 5  print(list1, file=f2) # write to f2 using print()
 6  print(list1) # write to default
```

Program 15:

```
10  56-append.py
 1  # Adding one more line to a file, check the file after execution
 2  f = open("fruits.txt",'r')
 3  for line in f:
 4      print(line.strip())
 5  f.close()
 6  f = open("fruits.txt",'a')
 7  print("Peach       99",  file=f)
 8  f.close()
```