Files and utilities

In this lecture, we will go over ways to open/close/read/write/delete files, as well as some useful system utilities.

Created by John C.S. Lui, June 1, 2018.

Important note: *If you want to use and modify this notebook file, please acknowledge the author.*

File methods

f = open("filename")	# open a file, return file value
f = open("filename", "w")	# open a file for writing
f.read()	# return a single character value
f.read(n)	# return no more than n character values
f.readline()	# return the next line of input
f.readlines()	# return all the file content as a list
f.write(s)	# write string s to file
f.writelines(lst)	# write list lst to file
f.close()	# close file

In [ ]:
```python
# process a line at a time
f = open('message.txt')    # open a specific file
for line in f:             # process a line at a time
    print('Read in:', line)
```

In [ ]:
```python
# process a character at a time
f = open('message.txt')    # open a specific file
for char in f.read():        # process a character at a time
    print('Read in:', char)
```

In [ ]:
```
# process 2 character only
f = open('message.txt')    # open a specific file
for char in f.read(2):      # process a character at a time
    print('Read in:', char)
```

In [ ]:
```
# process a line only
f = open('message.txt')    # open a specific file
for line in f.readline():      # process a character in this line
    print('Read in:', line)
```

In [ ]:
```
# process a line at a time
f = open('message.txt')    # open a specific file
for line in f.readlines():      # process a character in this line
    print('Read in:', line)
```

In [ ]:
```
# write something to the file
f = open('message1.txt', "w")    # open a specific file with write permission
f.write("write 1st string\n")
f.write("write 2nd string\n")
f.write("write 3rd string\n")
f.write("write 4th string\n")
f.close()                       # remember to close the file
```

In [ ]:
```
# write something to the file
f = open('message1.txt', "w")    # open a specific file with write permission
f.writelines(["1st item\n", "2nd item\n", "3rd item\n", "4th item\n"])
f.close()                       # remember to close the file
```

Operating system support

At times, it will be useful to ask the server (or OS) to help us to process files. Let's take a look.

In [ ]:
```
import os
os.rename ("message1.txt", "message2.txt")  # rename a file

os.curdir                   # show current working director
os.system('ls -al')         # perform a long listing of files in current directory

os.remove("message2.txt") # want to remove the file we just created
```

Recovering from exception

Sometimes, when you want to open a file, there can be error (can you give example on this error?). We want to have a way to handle this problem. Let's illustrate.

In [ ]:
```
try:
    f = open('input.txt')    # try to open a file 'input.txt'

except OSError:
    print ('unable to open the file')
else:
    print ('continue with processing')
    f.close()
print ('continue')
```

Standard I/O

- print writes characters to a file normally attached to display window
- Input functions read from a file attached to keyboard
- These files can be accessed through **sys** module
- Input file : *sys.stdin*, output file: *sys.stdout*, error messages: *sys.stderr*
- *stderr* normally also goes to *stdout*

Various input and output options

- *str()* function is to return representation of values which are **human-readable**
- *repr()* function is to generate representations which can be read by the **interpreter**
- Many values, such as numbers or structures like lists and dictionaries, have the same representation using either function. Strings and floating point numbers have two distinct representations.

In [ ]:
```
s = 'Hello world'
print (str(s))
print (repr(s))
print (str(1.0/7))
print (repr(1.0/7))
x = 10*3.25
y = 200* 200
s = 'The value of x is ' + repr(x) + ', and y is ', repr(y) + '...'
print ('str(s):', str(s), "; repr(s):", repr(s))
print(s)
hello = 'hello, world'
hellos = repr(hello)
print (hellos)
repr((x,y,('spam','eggs')))
```

In [ ]:
```
for x in range (1,11):
    print(repr(x).rjust(2), repr(x*x).rjust(4), end='')
    print (repr(x*x*x).rjust(6))
```

str.format

It becomes a place holder, and we can use various *index* !!!

In [ ]:
```
# use as place holder
print('I am the {} who say "{}!"'.format('bat', 'man'))
```

In [ ]:
```
# use index to manipulate the ordering. This is what we call the positional argument
print('I am the {0} who say "{1}!"'.format('bat', 'man'))
print('I am the {1} who say "{0}!"'.format('bat', 'man'))
```

In [ ]:
```
# use of keyword argument
print ('This {food} is {adjective}.'.format
             (food='spam', adjective='absolutely horrible'))
```

In [ ]:
```
# Positional and keyword arguments can be arbitrarily combined
print ('The story of {0}, {1}, and {other}'.format
        ('Hill', 'Manfred', other='George'))
print ('The story of {0}, {other}, and {1}'.format
        ('Hill', 'Manfred', other='George'))
```

In [ ]:
```
# Format output {x:y}, where x is the positional argument
#                  y is the format
print ('The value of PI is approximately {0:.6f}'.format
       (3.1415926))
```

In [ ]:
```
# Passing an integer after ":" to make things neat

table ={'John': 1000, 'Peter':500, 'David': 10} # define dictionary
for name, amount in table.items():  # extract item from dictionary
    print ('{0:10} ==> {1:10d}'.format(name, amount))
```

In [ ]:
```

```

Mathematics library

The *math* module gives access to the underlying C library functions for floating point math

In [ ]:
```
# using pi and cosine
import math
print (math.cos(math.pi / 4.0))

print(math.log(1024,2))
```

The *random* module provides tools for making random selection

In [ ]:
```
import random
random.choice(['apple', 'pear', 'banana'])
```

In [ ]:
```
random.sample(range(100), 10)   # sampling without replacement of 10 items
```

In [ ]:
```
import random
print('a random float number: ', random.random())
print('a random integer from range(10): ', random.randrange(10))
```

Library for Internet Access

Here, we have two modules: *urllib* and *smtplib*, for retrieving data and sending email respectively. Please read them via the Python documentation.

In [ ]:
```
# read one html file

import urllib.request

response = urllib.request.urlopen('http://python.org/')
html = response.read()
print(html)
```

Library for Performance Measurement

The *timeit* module measures the performance of the program

```
In [ ]: from timeit import Timer
        print('time 1 = ', Timer('t=a; a=b; b=t', 'a=1; b=2').timeit())
        print('time 2 = ', Timer('a,b = b,a', 'a=1; b=2').timeit())
```

```
In [ ]:
```