Working with files (write)

- Creating a new file:
  - `new_file = open(‘new.txt’, mode=‘w’)`
- Writing some text in the file:
  - `new_file.write(‘this is a text file’)`
- Text will be shown when you close the file:
  - `new_file.close()`
- NOTE: if you open the file again, all of the previous text will be deleted.

Scope

- Variables can only reach the area in which they are defined, which is called *scope*. Think of it as the area of code where variables can be used. Python supports global variables and local variables.
- By default, all variables declared in a function are local variables. To access a global variable inside a function, it's required to explicitly define 'global variable'.
- Defining a global variable is usually a bad approach, instead we use the following code to change the value of a global variable:
  - `global_variable = function(local_variable)`

Scope

case 1

```
>> x = 1
>> def test():
...     x = 2
>> test()
>> print(x) ☞ returns 1
```

case 2

```
>> x = 1
>> def test():
...     global x
...     x = 2
>>test()
>>print(x) ☞ returns 2
```

Scope

- A few examples:
  - Run the following codes and see the results:

```
def f():
    print(s)
s = "I love Paris in the summer!"
f()
```

```
def f():
    s = "I love London!"
    print(s)

s = "I love Paris!"
f()
print(s)
```

```
>>> def f():
...   print(s)
...   s = "I love London!"
...   print(s)
...
>>> s = "I love Paris!"
>>> f()
```

Scope
(Celsius to Fahrenheit conversion)

- Correct way of changing a variable through a function

```
>> temp = 80
>> def celsius_to_fahrenheit(temp):
...     temp = temp*1.8+32
...     return temp
>> temp = celsius_to_fahrenheit(temp)
>> temp
176.0
```

Locating modules

- When you import a module, the Python interpreter searches for the module in the following sequences:
  1. The current directory.
  2. If the module isn't found, Python then searches each directory in the shell variable PYTHONPATH.
  3. If all else fails, Python checks the default path.
- The module search path (all of the three locations) is stored in the system module sys as the `sys.path` variable.
  - `>> import sys`
  - `>> print(sys.path)` 👉 returns a list of directories.

os, os.path and shutil modules

- OS module: provides functions for interacting with the operating system.
- The *os* and *os.path* modules include many functions to interact with the file system.

command	meaning
`os.getcwd()`	returns the Current Working Directory(CWD) of the file used to execute the code
`os.listdir(dir)`	list of filenames in that directory path (not including . and ..). The filenames are just the names in the directory, not their absolute paths.
`os.path.join(dir, filename)`	given a filename from the above list, use this to put the dir and filename together to make a path
`os.path.abspath(path)`	given a path, return an absolute form, e.g. C:\Users\Farbod\Desktop\test.py

os, os.path and shutil modules

command	meaning
`os.path.dirname(path)`, `os.path.basename(path)`	given dir/foo/bar.html, return the dirname "dir/foo" and basename "bar.html"
`os.path.exists(path)`	true if it exists
`os.mkdir(dir_path)`	makes one dir

- shutil module: its main use is for file copying
  - `>> import shutil`
  - `>> shutil.copy(src, dst)`
  - src is source file and dst is the destination directory, both should be strings.

zipfile module

- Your Python programs can both create and open (or extract) ZIP files using functions in the `zipfile` module

command	meaning
`exampleZip = zipfile.ZipFile('example.zip‘, mode = ‘r’)`	Opens the example.zip file as ZipFile object in read mode. (can be changed as in `open()` command)
`exampleZip.namelist()`	returns a list of strings for all the files and folders contained in the ZIP file.
`exampleZip.extractall()` OR `exampleZip.extractall('C:\\my_folder')`	extracts all the files and folders from a ZIP file into the current working directory (or the one you specified)
`exampleZip.extract(‘test.txt’)`	will extract a single file from the ZIP file.

Creating and adding to zip file

- This code will create a new ZIP file named new.zip that has the compressed contents of spam.txt.
  - `>> import zipfile`
  - `>> newZip = zipfile.ZipFile('new.zip', 'w')`
  - `>> newZip.write('spam.txt', compress_type=zipfile.ZIP_DEFLATED)`
  - `>> newZip.close()`
- The second argument is the compression type parameter, which tells the computer what algorithm it should use to compress the files; you can always just set this value to `zipfile.ZIP_DEFLATED`. (This specifies the *deflate* compression algorithm, which works well on all types of data.)