Part VII

OS Functions

Module *os*

- Interface to OS services
- Variables

```
import os
os.environ    # environment variables, dict
os.linesep    # line separation; \r\n win, \n linux
os.name       # to find out what system
```

- Some general purpose methods

```
os.getcwd()
os.chdir(path)
os.getgroups()   # (UNIX)
os.uname()       # system information (UNIX)
os.times()       # (usr, sys, c_usr, c_sys, wct)
...
```

Module *os* (2)

- Some methods related to files

```
from os import *
chmod(path, mode)           # chmod
chmod("tmp.py", 0664)       # four digits required
listdir(path)               # ls
mkdir(path)                 # mkdir
makedirs(path)              # mkdir -p
rename(src, dst)            # mv
remove(path)                # rm
rmdir(path)                 # rmdir
removedirs(path)            # rm -rf
stat(path)                  # stats (atime, ...)

walk(path [, topdown])      # traverse dir rec.
# (dirpath, dirnames, fnames) for each dir
```

Module *os* (3)

Submodule `os.path`

- To manipulate pathnames and check for file properties

```
os.path.abspath(myPath)
os.path.basename(myPath)

os.path.exists(myPath)
os.path.isfile(myPath)
os.path.isdir(myPath)
os.path.islink(myPath)

os.path.getsize(myPath)
os.path.getatime(myPath)

os.path.join(path1, path2, ...)
```

Module *os* (4)

Example traversing a directory

```
#!/usr/bin/python3
import os

def print_tree(path):
    """Print the contents of the directory specified
       in path and all its subdirectories."""
    for (dirpath, dirnames, fnames) in os.walk(path):
        # get the directory level
        level = len(dirpath.split(os.sep))-1
        # print(directory)
        print("|"*level+"+␣"+os.path.basename(dirpath))
        # print all files
        for f in fnames:
            print("|"*level + "|␣␣" + f)

print_tree(".")
```

Module *glob*

Fast file / directory search using wildcards (* and ?)

- Find all files and directories including the letters `py` using `*`.

```
#!/usr/bin/python3
import glob

pyFiles = glob.glob('./*py*')
print(pyFiles)

#pyFiles = ['.\\glob.py', '.\\subp.py', '.\\printTree.py
```

- Use the wildcard `?` to match only one character.

```
for s in glob.glob('./myScripts/simulation?.py'):
    print(s)

# simulation1.py
# simulation2.py
# simulation3.py
```

Module *subprocess*

Easier and more robust access to command line.

- Call the terminal command `ls -l` from Python

```
#!/usr/bin/python
import subprocess
subprocess.call(["ls", "-l"])
```

- Store the output of a command in a string

```
d = subprocess.check_output(["ls","-l"])
print(d)
```

Module *ConfigParser*

Read and write configuration files from Python

- Create `*.ini` files with simulation parameters using *sections* with pairs of *keys* and *values*
- `myConfig.ini`:

```
[section1]
key1 = value1
key2 = value2
key3 = value3
...

[section2]
key4 = value4
key5 = value5
...
```

Module *ConfigParser* (2)

Read and write configuration files from Python

- Create `*.ini` files with simulation parameters using *sections* with pairs of *keys* and *values*
- `myConfig.ini`:

```
[parameters]
energy = 0.01
density = 2.456
k = 1e-4

[simulation]
stopEarly = True
timesteps = 10000
```

Module *ConfigParser* (3)

Read and write configuration files from Python

- Create parser objects to read in parameters

```
import configparser as cp
parser = cp.ConfigParser()
parser.read('myConfig.ini')
print(parser.sections())
# ['parameters', 'simulation']

timesteps = parser.get('simulation', 'timesteps')
timesteps = int(timesteps)
```

- Values are read in as strings! Should be casted if necessary.