**This is a basic introductory document for someone who is getting into Python from Unix.**

**File and directory maintenance:**

**Import os** – is the main library to import

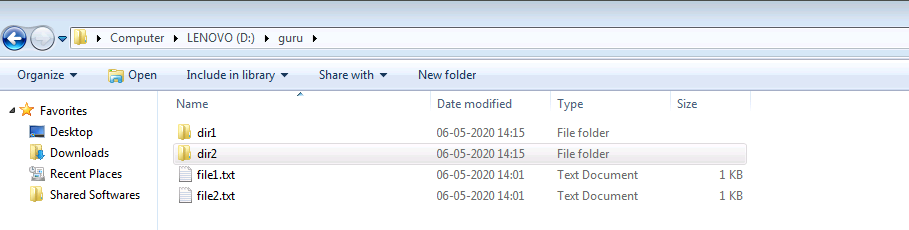
Ls = **os.listdir()** –this is for current path, this will show all directory and files.

Ls -l = **os.system('ls -l')**

Ls -ls = **os.system('ls -ls')**

Ls -ltr = **os.system('ls -ltr')**

If my directory is like this:



And you want to return only files then use:

**print(list(filter(os.path.isfile, os.listdir())))**

output is file1.txt.file.txt

for directory listing:

**print(list(filter(os.path.isdir, os.listdir())))**

Cat= os.system(“file1.txt”)

**Cd ~ = os.path.expanduser('~')**—to go to home directory

Cd = **Os.getcwd()**—change directory

# change the current directory

# to specified directory

**os.chdir("C:\Users\Gfg\Desktop\geeks")**

**os.chdir(".\Directory 1")**

The dot (.) at the beginning of the relative path .\Directory 1 represents the current working directory. A double dot ( ..) is used to move up the hierarchy, to the "parent" directory.

os.chdir(‘..’)—one level up (‘../..)—two levels up

**os.mkdir(specify path)**

**os.rmdir()**

sample script in python to remove directories. Note the usage on try, except(exception like catch) and else specified with colon(:)

import os

*# define the name of the directory to be deleted*

path = "/tmp/year"

try:

os.rmdir(path)

except OSError:

print ("Deletion of the directory %s failed" % path)

else:

print ("Successfully deleted the directory %s" % path)

**Os.makedirs()**—multiple directory

Remove file =os.remove(“file3.txt”)

Rename file=os.rename(“file3.txt”,”file4.txt”)

*In case you would like to remove an entire directory tree the rmtree() method from the****shutil****module will help you with that task.*

If you have dir as D://guru/gp2/gp3

Then

Import shutil

Os.rmtree(“D:\\guru”)—this will remove directories gp2 and gp3

Copy file= shutil.copyfile(“file1.txt”,”./file3.txt”)

Append to file:

f = open("./helloworld.txt", "a")

f.write("\nIt's good to have been born!")

f.close()

**how to use grep?**

<https://pythonhosted.org/grepfunc/>

**How to clear screen -** os.system(‘cls’)

**Sort:**

Sort works on a list of values

numbers = [1, 3, 4, 2]

words = ["Geeks", "For", "Geeks"]

numbers.sort()

works.sort()

The above 2 functions work in ascending order.

You can always print the output as print(numbers) or print(works)

If you want to sort descending then use numbers.sort(**reverse=True**) that’s it.

Now if you want to sort based on keys and you have a 2d matrix data like below:

list1 = [(1,2),(3,3),(1,1)]

**def** sortSecond(val):

    return val[1]

# list1 to demonstrate the use of sorting

# using using second key

list1 = [(1,2),(3,3),(1,1)]

# sorts the array in ascending according to

# second element

list1.sort(key=**sortSecond**)

print(list1)

Output is: [(1, 1), (1, 2), (3, 3)]

Note you are sorting on the 2nd element.

**Good to know:**

Sorted(). - method sorts the given sequence either in ascending order or in descending order and always return the sorted list. This method does not affect the original sequence whereas sort() does.

Now, **how do you sort a file**? – We need a script

Suppose you have a file with all fruits in mixed order

Here is the script for it:

#first assign a empty list

Fruits=list()

#read the fruits from file

Filename='fruits.txt'

with open (Filename) as fin:

for line in fin:

Fruits.append(line.strip())

#adding strip() removes \n

#sort the fruits

Fruits.sort()

print(Fruits)

#the above section was for print

#now use the same to write out to another file

filename='sorted\_fruits.txt'

with open(filename,'w') as fout:

for line in Fruits:

fout.write(line + '\n')

**Duplicate removal:**

lines\_seen = set() # holds lines already seen

with open("Output\_file.txt", "w") as output\_file:

for each\_line in open("Input\_file.txt", "r"):

if each\_line not in lines\_seen: # check if line is not duplicate

output\_file.write(each\_line)

lines\_seen.add(each\_line)

Ps: listing, search and killing

>>> import psutil

>>> psutil.pids()

[1, 2, 3, 5, 7, 8, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, ..., 32498]

>>> psutil.pid\_exists(32498)

True

>>> p = psutil.Process(32498)

>>> p.name()

'python'

>>> p.cmdline()

['python', 'script.py']

>>> p.terminate()

>>> p.wait()

Search and kill:

for p in psutil.process\_iter():

if 'nginx' in p.name() or 'nginx' in ' '.join(p.cmdline()):

p.terminate()

p.wait()

Cut: read slicing for this

<https://www.programiz.com/python-programming/methods/built-in/slice>

Wordcount (wc):

A=”iamabeginnerinpython”

Print (len(a))

20

Use filename with date:

from datetime import date

today = str(date.today())

b=[212,213]

file111 = 'file111\_'+today+'.txt'

with open(file111,'w') as fout:

for i in b:

fout.write(str(i) + '\n')

print("today",today)

**Head and tail is not direct. But below is the script:**

Head:

Import os

def head(filename, count=1):

This one is fairly trivial to implement but it is here for completeness.

with open(filename, 'r') as f:

lines = [f.readline() for line in xrange(1, count+1)]

return filter(len, lines)

Tail:

import os

def tail(filename, count=1, offset=1024):

"""

A more efficent way of getting the last few lines of a file.

Depending on the length of your lines, you will want to modify offset(size in bytes)

to get better performance.

"""

f\_size = os.stat(filename).st\_size

if f\_size == 0:

return []

with open(filename, 'r') as f:

if f\_size <= offset:

offset = int(f\_size / 2)

while True:

seek\_to = min(f\_size - offset, 0)

f.seek(seek\_to)

lines = f.readlines()

# Empty file

if seek\_to <= 0 and len(lines) == 0:

return []

# count is larger than lines in file

if seek\_to == 0 and len(lines) < count:

return lines

# Standard case

if len(lines) >= (count + 1):

return lines[count \* -1:]