Python Training.....

-- Jeetendra Bhattad

Agenda

- Program v/s Process
- Running commands / other scripts through a script
- Child process ?
  - subprocess
  - fork
  - system
- subprocess module in python
- psutil : to be studied https://code.google.com/archive/p/psutil/

Subprocess

- Allows :-
  - Spawn new process
  - Connect to their input/output/error pipes
  - Obtain their return codes
- Replaces :-
  - os.system()
  - os.spawn()
  - os.popen()
  - commands()
- subprocess modules methods :-
  - call
  - check_call
  - check_output
  - Popen

subprocess.call

* subprocess.call(args, *, stdin=None, stdout=None, stderr=None, shell=False)

* Run the command described by args. Wait for command to complete, then return the returncode.

```
>>> import subprocess

>>> subprocess.call([“ls”, “-l”])
```

* Command line arguments are passed as a list of strings

* Setting shell to true causes subprocess to spawn an intermediate shell process and command is run on that shell

```
>>> import subprocess

>>> subprocess.call(“echo $PATH”)

>>> subprocess.call(“echo $PATH”, shell=True)
```

subprocess.check_call subprocess.check_output

* subprocess.check_call(args, *, stdin=None,stdout=None, stderr=None, shell=False)

* Run command with arguments. Wait for command to complete. If the exit code was zero then return, otherwise raise CalledProcessError. The CalledProcessError object will have the return code in the returncode attribute.

```
>>> subprocess.check_call(['cp', '-l'], shell=True)
```

* subprocess.check_output : run command with arguments and return its output as a byte string.

subprocess.Popen

```
*  class Popen(args, bufsize=0, executable=None,
               stdin=None, stdout=None, stderr=None,
               preexec_fn=None, close_fds=False, shell=False,
               cwd=None, env=None, universal_newlines=False,
               startupinfo=None, creationflags=0)
```

- args : string or seqeunce of program arguments
- shell : False - on Unix Popen uses os.execvp to execute child program, True – first item specifies command string & any additional items will be treated as additional shell arguments.
- IO-redirection : stdin, stdout, stderr
- preexec_fn : set to callable object, will be called in the child process just before the child is executed.
- close_fds : if set to True all file descriptors except 0,1 and 2 will be closed before the child process is executed.
- cwd : if not None, current working directory will be changed to cwd before child is executed
- env : if not None, it defines the environment variables for the new process

Popen examples.....

```
#To run a process and read all of its output, set the stdout value to PIPE and call
communicate().
print '\nread:'
proc = subprocess.Popen(['echo', '"to stdout"'], stdout=subprocess.PIPE )
stdout_value = proc.communicate()[0]
print '\tstdout:', repr(stdout_value)

#set up a pipe to allow the calling program to write data to it, set stdin to PIPE.
print '\nwrite:'
proc = subprocess.Popen(['cat', '-'], stdin=subprocess.PIPE)
proc.communicate('\tJay Jay Ram-Krishna Hari to stdin\n')

#setup to do read & write as part of one subprocess
print '\npopen2:'
proc = subprocess.Popen(['cat', '-'], stdin=subprocess.PIPE, stdout=subprocess.PIPE)
stdout_value = proc.communicate('through stdin to stdout')[0]
print '\tpass through:', repr(stdout_value)
```

Assignment

Write a Python script to accept file-name from user and display word count, line count, character count of that file.

Write a Python script which runs periodically and displays status of the currently running processes