Python3 – Subprocess Overview

One-liner (Same Task)
- print(subprocess.check_output(['ls', '-1']).decode('utf-8'))
- print(''.join(map(chr, subprocess.check_output(['ls', '-1']))))

Waits (No Threading)
subprocess.call(), subprocess.check_output(),
subprocess.getoutput(),
subprocess.Popen().stdout.readlines()
Threading
subprocess.Popen()
#Windows only (threading)
DETACHED_PROCESS = 0x00000008
subprocess.Popen([sys.executable, 'ls'],
creationflags=DETACHED_PROCESS).pid
#BSD only (threading)
pid = subprocess.Popen([sys.executable, 'ls'],
stdout=subprocess.PIPE, stderr=subprocess.PIPE,
stdin=subprocess.PIPE)

Open Program or Execute Code
```
subprocess.call(['EXCUTABLE'])  # no threading
```

Get PID
```
pid = subprocess.Popen(['ls', '-1'], shell=True,
stdout=subprocess.PIPE,
stderr=subprocess.STDOUT).stdout.read()

print(pid) # <subprocess.Popen object at 0x7f32405e80b8>

type(pid) # subprocess.Popen

# .readline() reads one stdout line at a time
# .readlines() reads all stdout at once
```

Usage
```
x = subprocess.check_output(['cmd', 'arg1'])
# x = b'Desktop\nDocuments\nDownloads\n'
z = x.decode('utf-8'))  # plain text (str)

subprocess.getoutput('ls -l')  # one str param
# Output = 'String\nString\nString'
```

Tricks
```
shlex.split('ls -l')  # output: ['ls', '-l']
subprocess.check_output(shlex.split('ls -l'))
```

Shell2Python
Shell: output=`dmesg | grep hda`
Python: from subprocess import *
```
p1 = Popen(['dmesg'], stdout=PIPE)
p2 = Popen(['grep', 'hda'], stdin=p1.stdout, stdout=PIPE)
p1.stdout.close()
# allow p1 to receive a SIGPIPE if p2 exits
output = p2.communicate()[0]
```
Python:
```
output = check_output('dmesg | grep hda', shell=True)
```

Migrate os to subprocess
- os: output = os.spawnlp(os.P_WAIT, 'cmd', “arg')
- subprocess: output = call(['cmd', 'arg'])

- os: pid = os.spawnlp(os.P_NOWAIT, 'cmd', 'arg')
- subprocess: pid = Popen(['cmd', 'arg']).pid

Environment Variables
```
proc = subprocess.Popen(['echo',
os.environ['MY_ENV_VAR']])

proc = subprocess.Popen('echo '$MY_ENV_VAR'',
env=environ, shell=True)

proc = subprocess.Popen(['echo',
os.path.expandvars('$MY_ENV_VAR')])
```

- newenv = os.environ.copy()
- newenv['MY_ENV_VAR'] = 'value'

Example
```
import subprocess
proc = subprocess.Popen('mousepad')
print(proc)  # <subprocess.Popen object at
0x7f3240609d30>
pid = proc.pid  # 13731
print(proc.poll())  # None
proc.kill()
print(proc.poll())  # 0
proc = subprocess.Popen('mousepad')
pid = proc.pid
try:
    outs, errs = proc.communicate(timeout=15)
except TimeoutExpired:  # when timer expires
    proc.kill()  # close mousepad
    outs, errs = proc.communicate()
```

 # Example Lines
- proc.communicate(input='', timeout=int)
- Return Code: p_status = proc.wait()
- Send a Kill Signal: proc.send_signal(SIG)
- Stop the Process: proc.terminate()
- Wait for the Process: proc.wait()

Scripted Example
```
#!/usr/bin/env python3
import subprocess, sys
cmd = 'netstat -p --tcp'
p = subprocess.Popen(cmd, shell=True,
stderr=subprocess.PIPE)
while True:  # displaying output immediately
    out = p.stderr.read(1)
    if out == '' and p.poll() != None:
        break  # exit after netstat closes
    if out != '':  # release output
        sys.stdout.write(out)
        sys.stdout.flush()
```