NSSA 220
Task Automation with Interpreted Languages

Subprocesses in Python

Instructor: Dr. Fahed Jubair

RIT DUBAI

Review
Process

- A process is an instance of a running program
- When you execute a Linux command, you are creating a process
- When you execute an application (e.g., Zoom, Chrome, Python script), you are creating a process
- Each process has its own memory, i.e., a process cannot access the memory of other processes (e.g., Zoom cannot access the memory allocated for Chrome)

Subprocess Module

- In python, the *subprocess* module allows programmers to write code to create and run processes, and retrieve their output
- Common usages of the *subprocess* module in Python:
  - To run Linux (or Windows) commands within Python and retrieve their results
  - To run external executables (Perl, Bash, C++, Java, etc) within Python and retrieve their results
  - To run multiple tasks in parallel

subprocess.run Method

- A method for running commands or external executables as a subprocess
- The below example shows how to execute the *pwd* command using the run method

```
import subprocess

result = subprocess.run(["pwd"], capture_output=True, text=True)

print('output:', result.stdout)
print('error:', result.stderr)
```

To indicate whether to capture the standard output and standard error.

To return stdout and stderr as a String (otherwise, returned as Bytes)

Note: if you are using Windows, then add “shell=True” option

Running Bash Scripts

- Let us assume we have the following Bash script

```
#!/bash/bin
n=$1
for ((i=1; i<=n; i++))
do
echo -n "$i "
done
```

- The below example shows how to execute the bash script using the run method

The command to run and its arguments are passed as a list of strings.

```
import subprocess

result = subprocess.run(["bash", "hello.sh", "5"],
capture_output=True, text=True)

print(result.stdout)
print(result.stderr)
print(result.returncode)
```

Running Python Scripts

```python
import subprocess

result = subprocess.run(["python3", "script.py"], capture_output=True,
text=True)

print(result.stdout)

result = subprocess.run(["python3", "-c", "print([i for i in range(50)])"],
capture_output=True, text=True)

print(result.stdout)
```

subprocess.call Method

- Used to run a command in a separate process, and wait for its completion
- Returns the status code

```
import subprocess

return_code = subprocess.call(["python3", "--version"])

print(return_code)
```

subprocess.check_output Method

- Similar to subprocess.run method, except that it only returns the standard output (stdout)

```
import subprocess

output = subprocess.check_output(["python3", "--version"], text=True)

print(output)
```

Exercise 1

- Assume you have the below C++ code, called Hello.cpp
- Write a Python script that uses the subprocess module to compile and execute Hello.cpp program, and print the output of the program execution on the screen

```
#include <iostream>

using namespace std;

int main(){
    cout << "C++ is awesome!\n";
    return 0;
}
```

Exercise 2

- Repeat exercise 1, however, this time, write a bash script to compile and execute Hello.cpp, and print its output on the screen