Inmanta - Asyncio - Task Scheduling

The goal of this assignment is to build a small stand alone program that can execute a set of tasks with a **maximal level of concurrency** while respecting the dependencies between the tasks.

The program should be written in python3.6 or higher and use **python/asyncio** as a primary concurrency mechanism.

You are expected to produce code at the quality level of a proof-of-concept:

- you are expected to produce sufficient test cases and documentation to ensure basic correctness and to assist review of the code
- you are not required to produce production level tests, documentation or logging/metering integration
- security is not a concern, all input is considered trusted
- user friendly error reporting when exceptions occur is not required
- the program is allowed to bail out with an exception if the input is not valid

You are allowed to use any python library available on pypi.
You should provide a readme file with instructions on how to run the program.

Detailed assignment

The program expects as input a json file, conforming to the schema included. This file contains a list of tasks, each task specifies

- name
- type (eval or exec)
- arguments
- dependencies (optional)

A task can start executing after all its dependencies have been successfully executed.
If a dependency fails or is skipped, the task should be skipped.

A task of type eval has a code snippet as argument, that is to be executed within the main process. If the code raises an exception, it is considered failed.

A task of type exec has a shell command as argument, that is to be executed in a shell. If the return code is none zero, it is considered failed.

The output of the program should provide

- while executing tasks:
    - a line for the start of a task of the form “`Started: %(name)s`”
    - a line for the end of a task of the form “`Ended   : %(name)s`”
    - all output produced by the tasks

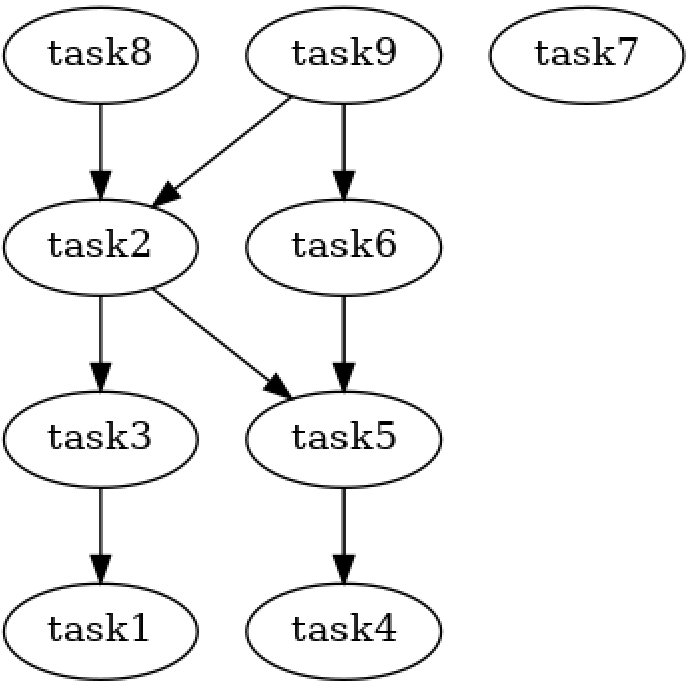
    - any exception produced by the tasks
- when done
    - a report with status for each task where the status is either ok, failed or skipped

Useful methods

Asyncio overview: https://docs.python.org/3/library/asyncio.html

Executing python code in process: https://docs.python.org/3/library/functions.html#exec
Spawning a process:
https://docs.python.org/3/library/asyncio-subprocess.html#asyncio.create_subprocess_shell
Running code of the main thread:
https://docs.python.org/3/library/asyncio-eventloop.html#asyncio.loop.run_in_executor
Get an eventloop:
https://docs.python.org/3/library/asyncio-eventloop.html#asyncio.get_event_loop
Starting the eventloop:
https://docs.python.org/3/library/asyncio-eventloop.html#id1
Starting a co-routine (fire-and-forget):
https://docs.python.org/3/library/asyncio-task.html#asyncio.create_task
Await multiple futures/co-routines:
https://docs.python.org/3/library/asyncio-task.html#asyncio.gather

Included example scenario


task8
task9
task7
task2
task6
task3
task5
task1
task4