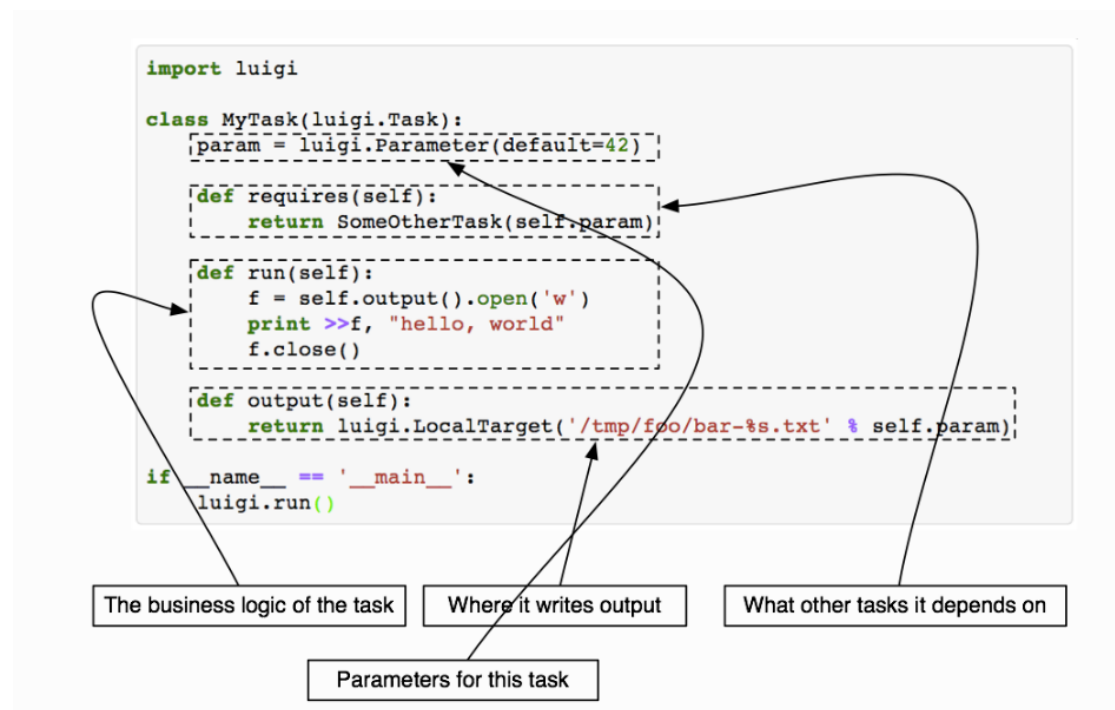


# Luigi

Luigi is a Python package that helps you build complex pipelines of batch jobs. It handles dependency resolution, workflow management, visualization, handling failures, command line integration, and much more.

There are two fundamental building blocks of Luigi - the Task class and the Target class. Both are abstract classes and expect a few methods to be implemented. In addition to those two concepts, the parameter class is an important concept that governs how a Task is run.

The Target class corresponds to a file on a disk, a file on HDFS or some kind of a checkpoint, like an entry in a database. Actually, the only method that Targets have to implement is the `exists` method which returns True if and only if the Target exists. Tasks are where the execution takes place. Tasks depend on each other and output targets. Here is an example of a Task class:



Here is a demo of how to use it:

1. Install luigi package

Input `conda install luigi` in anaconda prompt

```
Select Anaconda Prompt - luigid

(base) C:\Users\wenqi>conda install luigi
Solving environment: done

# All requested packages already installed.
```

2. Add work directory into environment variable "PYTHONPATH"

### 3. Check the visualizer page

Input *luigid* in anaconda prompt, then check visualizer page in <http://localhost:8082>

The image shows two screenshots. The top screenshot is an Anaconda Prompt window with the command `luigid` entered. The output shows the Luigi scheduler starting up and the web server listening on port 8082. The bottom screenshot is a web browser showing the Luigi Task Visualiser interface at `localhost:8082/static/visualiser/index.html`. The interface has a green header with tabs for Task List, Dependency Graph, Workers, and Resources. A 'Running' button is on the right. The main area displays task status cards for various task families: PEN... (0), RUN... (0), BAT... (0), DO... (0), FAIL... (0), UPS... (0), DIS... (0), and UPS... (0). At the bottom, there are filters for 'Show 10 entries', 'Filter table:', and 'Filter on Server'.

### 4. Create a python file named “example.py”

```
import time
import luigi
from luigi.local_target import LocalTarget
class RunAllTasks(luigi.Task):
    def requires(self):
        for i in range(10):
            yield RunExampleTask(i)
    def run(self):
        with self.output().open('w') as f:
            f.write('All done!')
    def output(self):
        return LocalTarget('tmp/RunAllTasks.txt')
class RunExampleTask(luigi.Task):
    number = luigi.IntParameter()
    def run(self):
```

```

time.sleep(self.number)

with self.output().open('w') as f:
    f.write("This is task # {}".format(self.number))
def output(self):
    return LocalTarget('tmp/runExampleTask_{}.txt'.format(self.number))
if __name__ == "__main__":
    luigi.run()

```

5. Change directory to your work directory, then run the task

Anaconda Prompt

```

(base) C:\Users\wenqi>cd C:\Users\wenqi\Desktop\7390\luigi
(base) C:\Users\wenqi\Desktop\7390\luigi>luigi --module example RunAllTasks

```

6. We can check the task status in visualizer page

The top screenshot shows the Luigi Task Visualiser interface. The dashboard displays the following task counts:

- PENDING TASKS: 0
- RUNNING TASKS: 0
- BATCH RUNNING TASKS: 0
- DONE TASKS: 11
- FAILED TASKS: 0
- UPSTREAM FAILURE: 0
- DISABLED TASKS: 0
- UPSTREAM DISABLED: 0

Below the summary is a table of task entries, all marked as 'DONE'.

Name	Details	Priority	Time	Actions
RunExampleTask	number=9	0	10/15/2018, 5:28:08 PM	[Icon]
RunExampleTask	number=8	0	10/15/2018, 5:28:16 PM	[Icon]
RunExampleTask	number=7	0	10/15/2018, 5:28:23 PM	[Icon]
RunExampleTask	number=6	0	10/15/2018, 5:28:29 PM	[Icon]
RunExampleTask	number=5	0	10/15/2018, 5:28:34 PM	[Icon]
RunExampleTask	number=4	0	10/15/2018, 5:28:38 PM	[Icon]
RunExampleTask	number=3	0	10/15/2018, 5:28:41 PM	[Icon]

The bottom screenshot shows the 'Dependency Graph' for the task 'RunAllTasks\_\_99914b932b'. The graph shows a root node 'RunAllTasks' connected to multiple 'RunExampleTask' nodes, illustrating the task dependencies.

Reference:

Luigi documentation : <https://luigi.readthedocs.io/en/stable/>

Luigi github: <https://github.com/spotify/luigi>