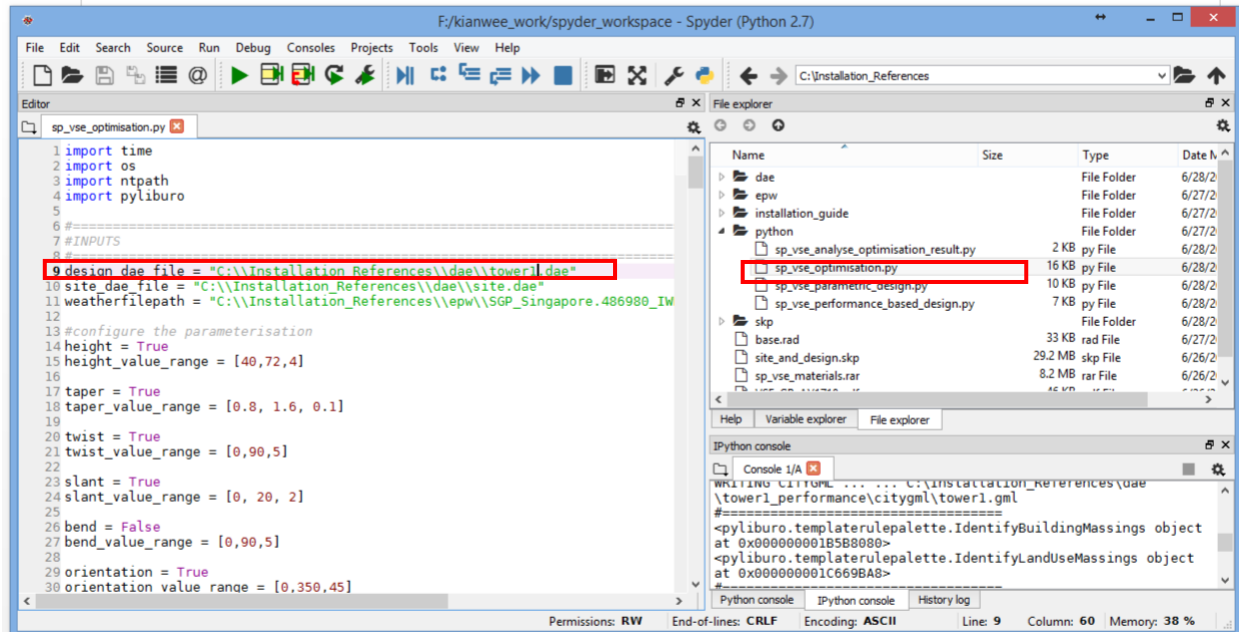


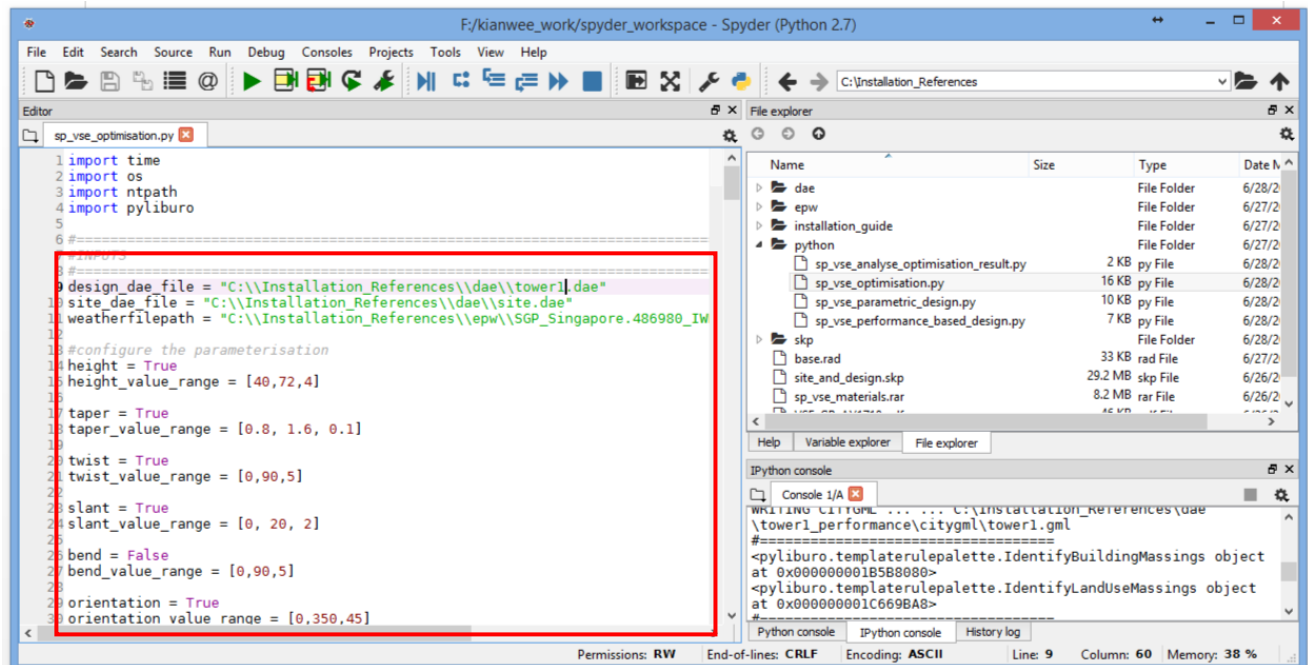
Pre-requisite:

- You must have read the installation guide for pyliburo and installed pyliburo before attempting this exercise.
- You must already finish exercise 1 and 2 to attempt exercise 3.

1.) Open spyder and go to the “sp_vse_optimisation.py” script by double clicking it. We will use the example from exercise 1, “c:\\Installation_References\\dae\\tower1.dae”.



2.) You can optimise your model according to 6 parameters. Height, taper, twist, slant, bend and orientation. Turn on each parameters by typing in True or False. Set the range of each parameter for the optimisation algorithm to explore. For example, if you set: height = True, height_value_range = [40, 72, 4]. It means the optimisation will generate design between the height of 40 to 72 and in intervals of 4 (40, 44, 48, 52, 56, 60, 64, 68, 72). There are 9 possible height variations. You can configure the rest of the parameters.



```
1 import time
2 import os
3 import ntpath
4 import pyliburo
5
6 #=====
7 design_dae_file = "C:\\Installation_References\\dae\\tower1.dae"
8 site_dae_file = "C:\\Installation_References\\dae\\site.dae"
9 weatherfilepath = "C:\\Installation_References\\epw\\SGP_Singapore.486980_IW
10
11 #configure the parameterisation
12 height = True
13 height_value_range = [40,72,4]
14
15 taper = True
16 taper_value_range = [0.8, 1.6, 0.1]
17
18 twist = True
19 twist_value_range = [0,90,5]
20
21 slant = True
22 slant_value_range = [0, 20, 2]
23
24 bend = False
25 bend_value_range = [0,90,5]
26
27 orientation = True
28 orientation_value_range = [0.350,45]
```

The screenshot shows the Spyder Python IDE interface. The main editor window displays a Python script named `sp_vse_optimisation.py`. The script imports `time`, `os`, `ntpath`, and `pyliburo`. It then defines several file paths for design, site, and weather data. The core of the script is a configuration section for parameterisation, where six parameters are set: `height`, `taper`, `twist`, `slant`, `bend`, and `orientation`. Each parameter is assigned a boolean value (True or False) and a list representing its value range and interval. For example, `height` is set to `True` with a range of `[40, 72, 4]`. The right-hand side of the IDE features a File Explorer showing the project structure, including folders like `dae`, `epw`, `installation_guide`, `python`, and `skp`. Below the File Explorer is the IPython console, which displays the output of the script, showing the execution of `pyliburo` objects for building and land use massings.

3.) Configure your parameters and value_ranges according to the parametric_log.csv shown in the previous exercise. Look out for minimum and maximum parameter values and use them to set the parameter range and intervals.

parametric_log.csv - Excel

Chen Kian Wee

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	
	design_file_name	plot_ratio	nshffai	height	height_value	orientation	orientation	twist	twist_val	taper	taper_value	slant	slant_val	bend	bend_val	nparms	number_of_surfaces	total
1	C:\installa	5.2	0	TRUE	80	TRUE	90	FALSE	45	FALSE	1.6	FALSE	5	FALSE	30	2	6	0.234
2	C:\installa	7.6	0.03	TRUE	60	TRUE	90	FALSE	45	TRUE	1.6	FALSE	5	FALSE	30	3	1384	0.653
3	C:\installa	4	0	TRUE	60	FALSE	90	FALSE	45	FALSE	1.6	FALSE	5	FALSE	30	1	6	0.227
4	C:\installa	5.9	0.04	FALSE	60	FALSE	90	FALSE	45	TRUE	1.6	FALSE	5	FALSE	30	1	904	0.475
5	C:\installa	4	0	TRUE	60	FALSE	90	FALSE	45	FALSE	1.6	FALSE	5	FALSE	30	1	6	0.251
6	C:\installa	3.2	0.02	FALSE	60	FALSE	90	TRUE	45	FALSE	1.6	FALSE	5	FALSE	30	1	904	0.482
7	C:\installa	3.2	0.02	FALSE	60	FALSE	90	FALSE	45	FALSE	1.6	TRUE	20	FALSE	30	1	904	0.426
8	C:\installa	3.7	0	FALSE	60	FALSE	90	FALSE	45	FALSE	1.6	FALSE	20	TRUE	30	1	904	0.44
9	C:\installa	3.2	0	FALSE	60	TRUE	45	FALSE	45	FALSE	1.6	FALSE	20	FALSE	30	1	6	0.231
10	C:\installa	9.6	0.02	TRUE	60	TRUE	45	TRUE	45	TRUE	1.6	TRUE	20	TRUE	30	6	6358	4.432
11																		
12																		
13																		

parametric_log

READY

4.) For example for the problem shown in the figure:

- a. Height = [40, 72, 4] = 9 possibilities
- b. Taper = [0.8, 1.6, 0.1] = 9 possibilities
- c. Twist = [0, 90, 5] = 10 possibilities
- d. Slant = [0, 20, 2] = 10 possibilities
- e. Orientation = [0, 350, 45] = 8 possibilities
- f. Total possibilities = $9 \times 9 \times 10 \times 10 \times 8 = 64800$

The screenshot shows the Spyder Python IDE interface. The main editor window displays a Python script named `sp_vse_optimisation.py`. The script imports `time`, `os`, `ntpath`, and `pyliburo`. It defines several input parameters and their value ranges for optimization:

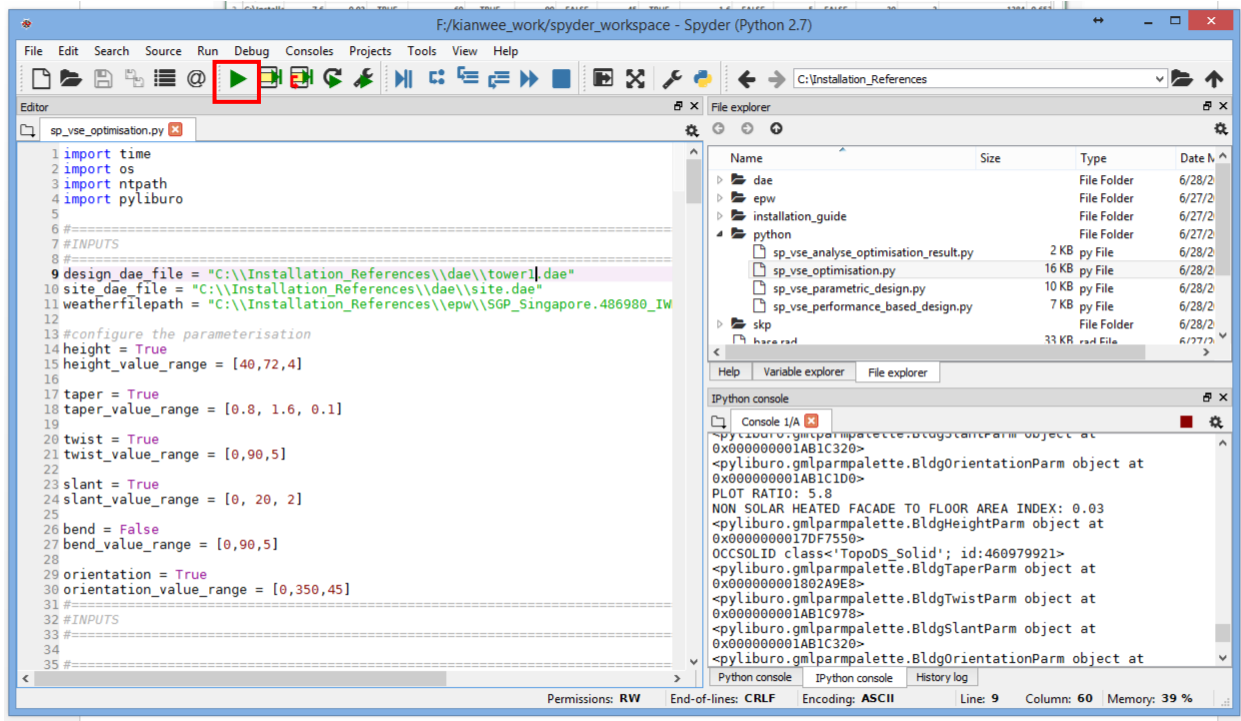
```
1 import time
2 import os
3 import ntpath
4 import pyliburo
5
6 #-----
7 #INPUTS
8 #-----
9 design_dae_file = "C:\\Installation_References\\dae\\tower1.dae"
10 site_dae_file = "C:\\Installation_References\\dae\\site.dae"
11 weatherfilepath = "C:\\Installation_References\\epw\\SGP_Singapore.486980_IW
12
13 #configure the parameterisation
14 height = True
15 height_value_range = [40,72,4]
16
17 taper = True
18 taper_value_range = [0.8, 1.6, 0.1]
19
20 twist = True
21 twist_value_range = [0,90,5]
22
23 slant = True
24 slant_value_range = [0, 20, 2]
25
26 bend = False
27 bend_value_range = [0,90,5]
28
29 orientation = True
30 orientation value range = [0,350,45]
```

The right-hand pane shows the 'File explorer' view, displaying a directory structure with folders like `dae`, `epw`, `installation_guide`, `python`, and `skp`. The 'python' folder contains several files, including `sp_vse_analyse_optimisation_result.py`, `sp_vse_optimisation.py`, `sp_vse_parametric_design.py`, and `sp_vse_performance_based_design.py`. The bottom pane shows the 'IPython console' with the following output:

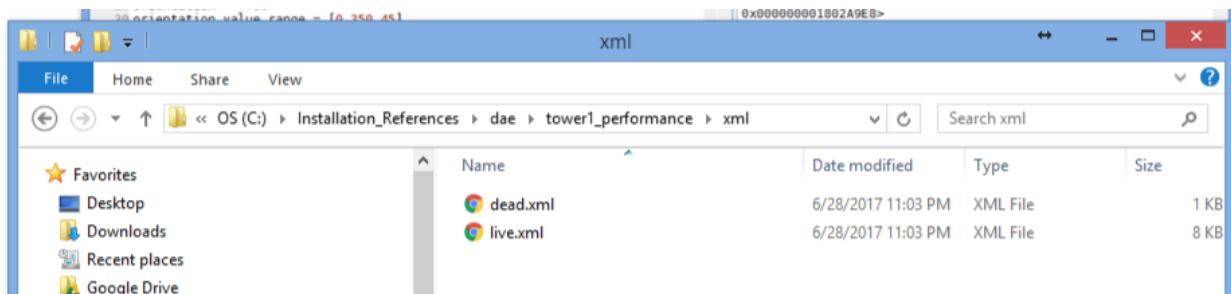
```
WRITING CITITOML ... C:\\Installation_References\\dae\\tower1_performance\\citygml\\tower1.gml
#-----
<pyliburo.templaterulepalette.IdentifyBuildingMassings object
at 0x000000001B5B8080>
<pyliburo.templaterulepalette.IdentifyLandUseMassings object
at 0x000000001C669BA8>
```

The status bar at the bottom indicates 'Permissions: RW', 'End-of-lines: CRLF', 'Encoding: ASCII', 'Line: 9', 'Column: 60', and 'Memory: 38 %'.

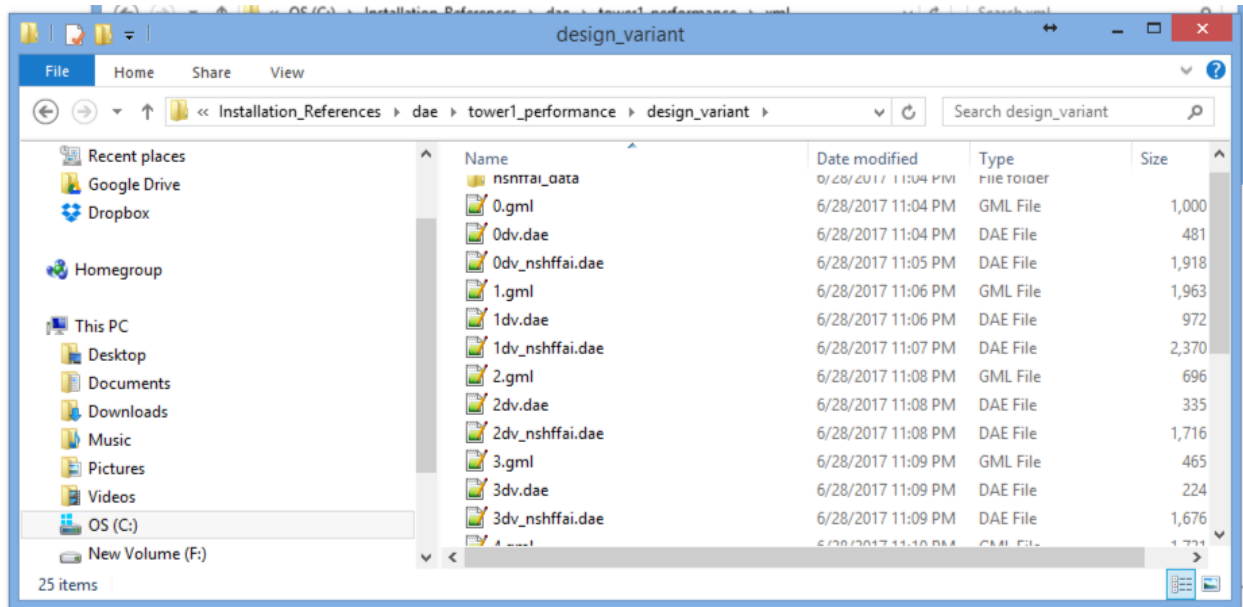
5.) Press run at spyder. The script will start running for 6-12 hours depending how complicated is your model.



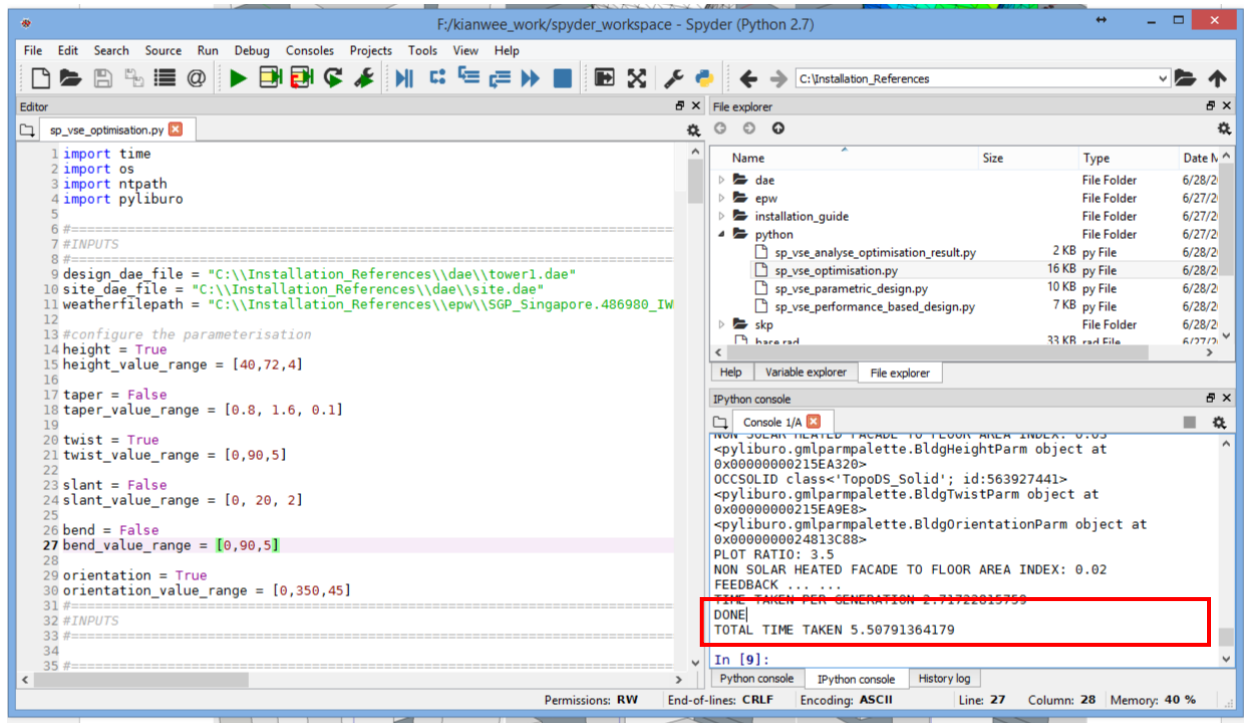
6.) Once the optimisation runs, it will produce 2 xml files named “dead.xml” and “live.xml”. These 2 files documents all the results of the optimisation. (PLEASE DO NOT OPEN THE FILE WHILE THE OPTIMISATION IS RUNNING)



7.) You can check the generated design variants in the design variant folder.



8.) If you have successfully run the optimisation. You will see the DONE message at the console, with the total time taken.



9.) Create an archive folder, copy and paste the “dead.xml” and “live.xml” file into the archive folder. This is to make sure you keep an extra copy of your result for the 6-12 hours wait, as the xml file documents all the results of the optimisation.

