**BCED321 Advanced Programming**

**Assessment Two**

**Practical Assessment 2**

Students:

Matthew Gordon

John Quiamco

Harry Lo

**Table of Contents**

[1 Class Diagram 2](#_Toc35816887)

[2 Matthew’s help file details 3](#_Toc35816888)

[3 John’s help file details 3](#_Toc35816889)

[4 Harry’s help file details 3](#_Toc35816890)

[5 Lists of Matthew’s own work, self-reflection on robustness, and self-reflection on the completeness and implement 4](#_Toc35816891)

[6 List of John’s own work, self-reflection on robustness, and self-reflection on the completeness and implement 6](#_Toc35816892)

[7 List of Harry’s own work, self-reflection on robustness, and self-reflection on the completeness and implement 6](#_Toc35816893)

[8 Location of GitHub repository 24](#_Toc35816894)

# Class Diagram

class\_dia\_cli

+run(args)

+main()

MyCli

+my\_name

+prompt

+file\_to\_data

-\_\_init\_\_(self, my\_name=**">"**)

+do\_exit(self)

+help\_exit(self)

+do\_diagram(self, inp)

+help\_diagram(self)

+do\_deletediagram(self, inp)

+help\_deletediagram(self)

+do\_pickle(self, inp)

+help\_pickle(self)

+do\_unpickle(self, inp)

+help\_unpickle(self)

+do\_deletepickle(self, inp)

+help\_deletepickle(self)

+do\_createtable(self, inp)

+help\_createtable(self)

+do\_addtotable(self, inp)

+help\_addtotable(self)

+do\_showtable(self, inp)

+help\_showtable(self)

+do\_deletetable(self, inp)

+help\_deletetable(self)

+do\_closedb(self)

+help\_closedb(self)

+do\_deletedb(self)

+help\_deletedb(self)

+default(self, inp)

-\_\_canExit(self)

-\_\_setCanExit(self, value)

+preloop(self)

+postloop(self)

+emptyline(self)

+do\_exit(self, args)

+do\_EOF(self, args)

+do\_shell(self, args)

+do\_help(self, args)

+do\_create(self, file\_name)

+run\_pyreverse(self)

+do\_pyr\_class\_diagram(self, file\_names)

+help\_pyr\_class\_diagram(self)

+do\_read\_source\_file(self, file\_name)

+help\_read\_source\_file(self)

+do\_validate\_class\_contents(self, file\_name)

+help\_validate\_class\_contents(self)

+do\_dot\_2\_png(self, input\_dot\_file\_name)

+help\_dot\_2\_png(self)

+do\_quit(self, line)

+help\_quit(self)

FileToData

+generic\_visit(self, node)

+read\_file(self, file\_name)

+show\_ast\_nodes(self)

+show\_all\_classes(self)

MyCreator

+image\_name

+file\_name

-\_\_init\_\_(self, image\_name, file\_name)

+create\_diagram(self)

+validate\_file\_name(self)

+validate\_image\_name(self)

+delete\_diagram(self)

MyPickle

+file\_to\_pickle

+pickle\_name

-\_\_init\_\_(self, file\_to\_pickle, pickle\_name)

+make\_pickle(self)

+unmake\_pickle(self)

+delete\_pickle(self)

MyDatabase

+Connection

+cursor

-\_\_init\_\_(self)

+create\_table(self, table\_name)

+add\_data(self, table\_name, file\_number, file\_name, file\_content)

+show\_data(self, table\_name)

+check\_table(self, table\_name)

+close\_database(self)

+delete\_database(self)

TestUM

+setUp(self)

+test\_validate\_file(self)

+test\_check\_table\_false(self)

+test\_check\_table\_true(self)

MyTestCase

+test\_show\_ast\_nodes(self)

+test\_read\_file\_wrong\_file\_name(self)

# Matthew’s help file details

|  |  |
| --- | --- |
| **Command** | **Help** |
| Diagram [inp] | Create a class diagram. Enter file location of py/dot file, then enter name/type of image. |
| Deletediagram [inp] | Deletes a diagram |
| Pickle [inp] | pickle [filename], enter file to pickle then the name of the pickle file |
| Unpickle [inp] | unpickle [picklefilename], enter the name of a text file that has been pickled |
| Deletepickle [inp] | Deletes a pickle file |
| Createtable [inp] | createtable [TABLE\_NAME], creates a table with: file\_number INTEGER PRIMARY KEY, file\_name VARCHAR(30),'  'file\_content VARCHAR(999) |
| Addtotable [inp] | addtotable [TABLE\_NAME], adds data to specified table |
| Showtable [inp] | showtable [TABLE\_NAME], shows data held within specified table |
| Deletetable [inp] | deletetable [TABLE\_NAME], deletes specified table |
| closedb | Closes current open database |
| deletedb | Deletes current database |
| default | Appears when an incorrect command is input |

# John’s help file details

# Harry’s help file details

1. python class\_dia\_cli.py --help (in the system command line)

usage: class\_dia\_cli.py [-h] [-l LETTER]

This is a program going to a CLI to generate UML class diagram from Source

Codes

optional arguments:

-h, --help show this help message and exit

-l LETTER optional: give a letter displaced at the beginning of each

command line. If user enter a string, only first character will

be shown.

1. >>>>> help pyr\_class\_diagram (in the line-oriented command interpreter)

Generate and display a class diagram in png format from a given python file

Syntax: pyr\_class\_diagram [output png file name suffix] [input source code file name.py])

1. >>>>> help read\_source\_file (in the line-oriented command interpreter)

Extract data from the given python file to be an ast node

Syntax: read\_source\_file [input source code file name.py]

1. >>>>> help validate\_class\_contents (in the line-oriented command interpreter)

Validate, list and display class names, function names and the total numbers of them in the given python file.

Class and function names are displayed in command line.

Total numbers of classes and functions are displayed in a bar graph.

Syntax: validate\_class\_contents [input source code file name.py].

1. >>>>> help dot\_2\_png (in the line-oriented command interpreter)

Generate and display png file from the given dot file.

Syntax: dot\_2\_png [input dot file name.dot].

1. >>>>> help quit (in the line-oriented command interpreter)

Quit from this CLI

:return: True

# Lists of Matthew’s own work, self-reflection on robustness, and self-reflection on the completeness and implement

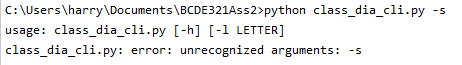
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Component | Location | Used by your peers (2 mark) | Robustness (2 mark) | Complete and well  implemented, i.e.,  “What is clever  about this?” (2  mark) | Marks |
| 1 | Support command-line arguments | Lines 27-134 in my\_cli.py | 2 marks | 2 marks. Encounters no unhandled exceptions | 2 marks. Code follows pep8 guidelines and is Pythonic | 6 |
| 2 | Has a line-oriented command interpreter based on cmd or similar package | Lines 27-134 in my\_cli.py | 2 marks | 2 marks. Encounters no unhandled exceptions | 2 marks. Code follows pep8 guidelines and is Pythonic | 6 |
| 3 | Display command line help of available commands | Lines 27-134 in my\_cli.py | 2 marks | 2 marks. Encounters no unhandled exceptions | 2 marks. All commands have a help command associated with them. Code follows pep8 guidelines and is Pythonic. | 6 |
| 4 | Change commands and options | Lines 27-134 in my\_cli.py | 2 marks | 2 marks. Encounters no unhandled exceptions | 2 marks. All commands have a help command associated with them that can be accessed by typing  ? [COMMAND]. Code follows pep8 guidelines and is Pythonic | 6 |
| 5 | Extract data | Lines 11-15 in PickleMaker.py and Line 14 in DiagramCreator.py | 2 marks | 2 marks. Encounters no unhandled exceptions | 2 marks. Data is extracted using Pickle. Code follows pep8 guidelines and is Pythonic | 6 |
| 6 | Validate data | Lines 88-94 in my\_cli.py and 17-24 in DiagramCreator.py | 2 marks | 2 marks. Encounters no unhandled exceptions | 2 marks. Code follows pep8 guidelines and is Pythonic | 6 |
| 7 | Provides object persistence / object serialization using either pickle or shelve | PickleMaker.py | 2 marks | 2 marks. Encounters no unhandled exceptions | 2 marks. Code follows pep8 guidelines and is Pythonic | 6 |
| 8 | Can load data from a file | Lines 11-15 in PickleMaker.py and Lines 11 to 15 in DiagramCreator.py | 2 marks | 2 marks. Encounters no unhandled exceptions | 2 marks. Code follows pep8 guidelines and is Pythonic | 6 |
| 9 | Can deal with file directory | PickleMaker.py and DiagramCreator.py | 2 marks | 1 mark. Some errors when creating a diagram in DiagramCreator.py due to it forcing a “classes.” Prefix. | 2 marks. Code follows pep8 guidelines and is Pythonic | 5 |
| 10 | Can raise exceptions and provide exception handling | Lines 88-94 in my\_cli.py and 17-24 in DiagramCreator.py | 2 marks | 2 marks. Encounters no unhandled exceptions | 2 marks. Code follows pep8 guidelines and is Pythonic | 6 |
| 11 | Amount of checking for pre- and post- conditions of methods | Lines 17-24 in DiagramCreator.py and 63-69 in SQLDatabase.py | 2 marks | 2 marks. Encounters no unhandled exceptions | 2 marks. Function in SQLDatabase.py checks that the table exists before allowing the code to process. Code in DiagramCreator.py checks that files exist before processing. Code follows pep8 guidelines and is Pythonic | 6 |
| 12 | Provide doctests | MattDoctests.py | 2 marks | 2 marks. All tests pass | 2 marks. 21 tests total. | 6 |
| 13 | Provide unittests | MattUnittests.py | 2 marks | 1 mark. All tests pass, but only 3 tests | 1 mark. Only 3 tests | 4 |
| 14 | Pretty print, i.e., displaying data in chart/ diagram, e.g., bar chart, pie chart, UML diagram, etc | DiagramCreator.py | 2 marks | 2 marks. Encounters no unhandled exceptions | 2 marks. Code creates a class diagram. Code follows pep8 guidelines and is Pythonic | 6 |
| 15 | Can save and read data from a database, e.g., SQLite, MySQL and MongoDB | SQLDatabase.py | 2 marks | 2 marks. Encounters no unhandled exceptions | 2 marks. Code follows pep8 guidelines and is Pythonic | 6 |

# List of John’s own work, self-reflection on robustness, and self-reflection on the completeness and implement

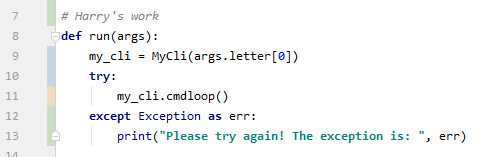
# List of Harry’s own work, self-reflection on robustness, and self-reflection on the completeness and implement

1. Support command-line arguments
   1. Used by peers

* File: class\_dia\_cli.py. I did the three functions below:
  + def run(args):
  + def main():
  + if \_\_name\_\_ == ‘\_\_main\_\_’: (note that: this is an entry point of the whole program)
  1. Robustness
* If user inputs wrong flag, my program will tell the user that the input was wrong as show below



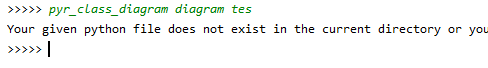
* There is exception handling as shown below. If there are any errors, the program will ask the user try again.



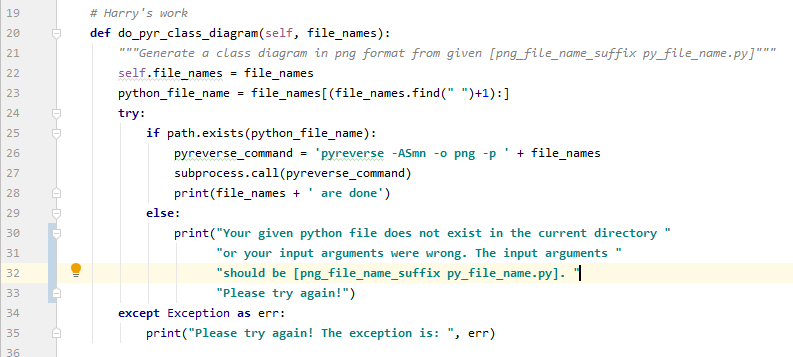
* 1. Complete and well implemented
* My code is pythonic. It complies with PEP 8 and is beautiful better than ugly. For example:
  + The code meets the naming convention of PEP 8.
  + There are two blank lines between functions.

1. Has a line-oriented command interpreter based on cmd or similar package
   1. Used by peers

* File: my\_cli.py. I did the functions below:
  + def \_\_init\_\_(self, my\_name=">"):
  + def do\_pyr\_class\_diagram(self, file\_names):
  + def help\_pyr\_class\_diagram(self):
  + def do\_read\_source\_file(self, file\_name):
  + def help\_read\_source\_file(self):
  + def do\_validate\_class\_contents(self, file\_name):
  + def help\_validate\_class\_contents(self):
  + def do\_dot\_2\_png(self, input\_dot\_file\_name):
  + def help\_dot\_2\_png(self):
  + def do\_quit(self, line):
  + def help\_quit(self):
  + if \_\_name\_\_ == '\_\_main\_\_': (for manual testing only)
* File: file\_to\_data.py. It is used by my\_cli.py. I did the functions below:
  + def generic\_visit(self, node):
  + def read\_file(self, file\_name):
  + def show\_ast\_nodes(self):
  + def show\_all\_classes(self):
  + if \_\_name\_\_ == "\_\_main\_\_": (for doctects and manual testing)
  1. Robustness
* If the file, which user inputs into “def do\_pyr\_class\_diagram(self, file\_names):” function, does not exist, my program will tell the user that Your given python file does not exist in the current directory or your input arguments were wrong. The input arguments should be [png\_file\_name\_suffix py\_file\_name.py]. Please try again! The screenshot is shown below:



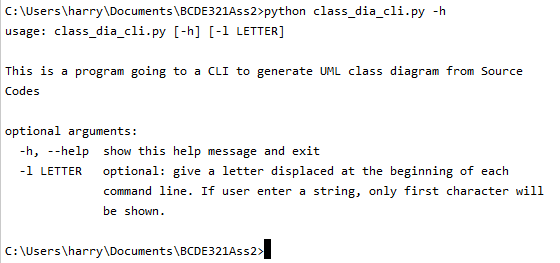
* There is exception handling as shown below. If there are any errors, the program will ask the user try again.



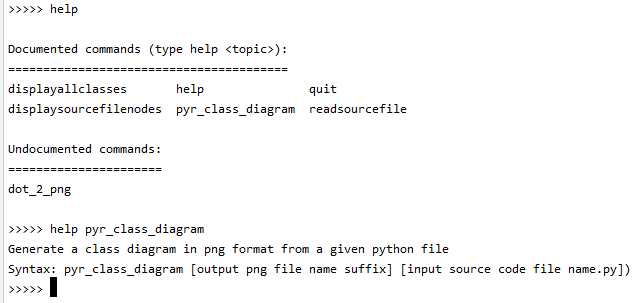
* 1. Complete and well implemented
* My code is pythonic. It complies with PEP 8 and is beautiful better than ugly. For example:
  + The code meets the naming convention of PEP 8.
  + There are either one or two blank lines between code blocks according to PEP8.

1. Display command line help of available commands
   1. Used by peers

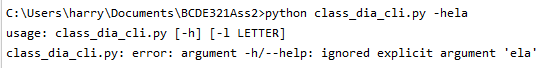
* File: class\_dia\_cli.py and my\_cli.py. Both files have the command line help as shown below:
  + For class\_dia\_cli.py, an example of the help function is below:



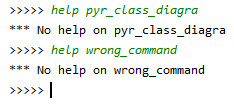
* + For my\_cli.py, two examples of the help functions are below:



* 1. Robustness
* File: class\_dia\_cli.py and my\_cli.py. They both have exception handling as shown below.
  + For class\_dia\_cli.py, my program will tell the user to use -h or - -help for help if the user used a wrong flag for help as shown below:



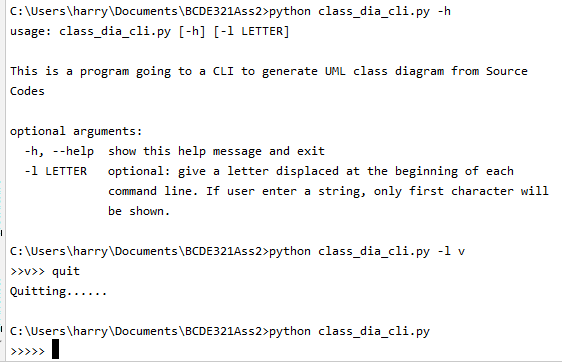
* + For my\_cli.py, my program will tell the user that no help on the command which does not exist or was wrongly spelled as shown below:



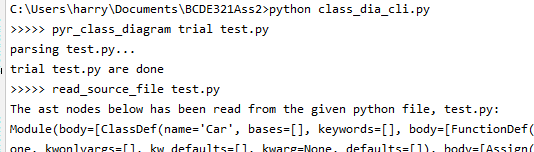
* 1. Complete and well implemented
* My code is pythonic. It complies with PEP 8 and is beautiful better than ugly. For example:
  + The code meets the naming convention of PEP 8.
  + There are either one or two blank lines between code blocks according to PEP8.

1. Change commands and options
   1. Used by peers

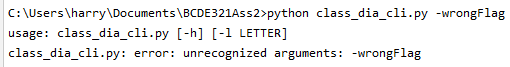
* File: my\_cli.py. It can change options as shown below:
  + There are three options: (i) -h or - -help flap for help; (ii) -l flap for adding a letter at the prompt as shown below (e.g. giving -l v flag will get the prompt of >>v>>); (iii) no flap for having a >>>>> prompt)



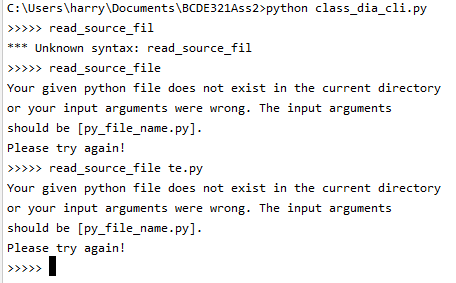
* + For my\_cli.py, there are more than one commands. An example of change commands (pyr\_class\_diagram and read\_source\_file commons) is below:



* 1. Robustness
* File: class\_dia\_cli.py and my\_cli.py. They both have exception handling as shown below.
  + For class\_dia\_cli.py, my program will tell the user what options (i.e. flags) are available if the user used a wrong option (i.e. wrong flag) which is not available as shown below:

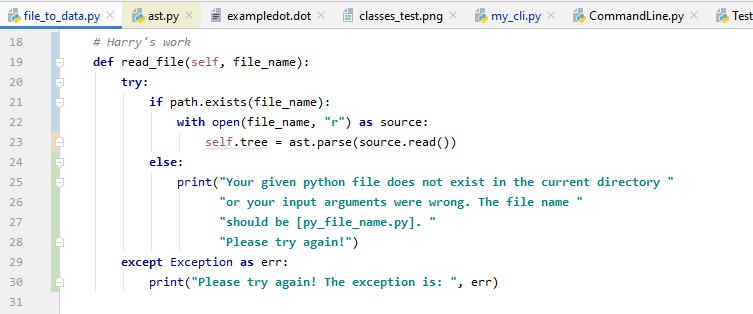


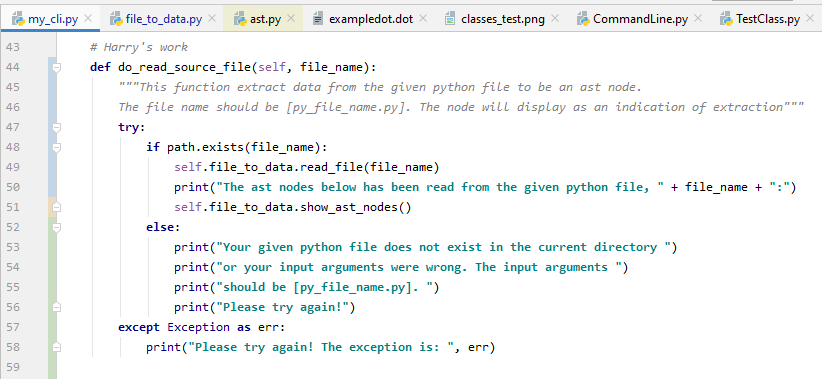
* + For my\_cli.py, my program will tell the user if the user used a wrong function or a wrong argument as shown below:



* 1. Complete and well implemented
* My code is pythonic. It complies with PEP 8 and is beautiful better than ugly. For example:
  + The code meets the naming convention of PEP 8.
  + There are either one or two blank lines between code blocks according to PEP8.

1. Extract data
   1. Data can be extracted from a python file through the def read\_file(self, file\_name): function in class FileToData(ast.NodeVisitor): in file\_to\_data.py. This read\_file function is used by few functions in the my\_cli.py, for example def do\_read\_source\_file(self, file\_name):. The codes of the two functions are shown below:

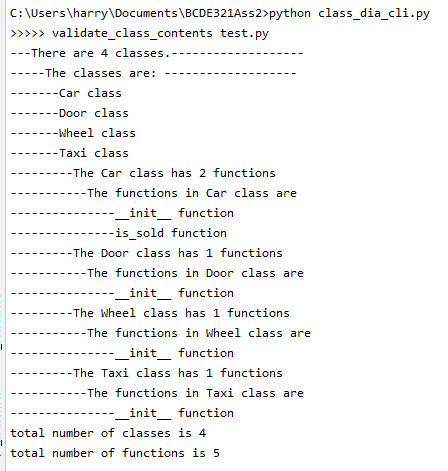


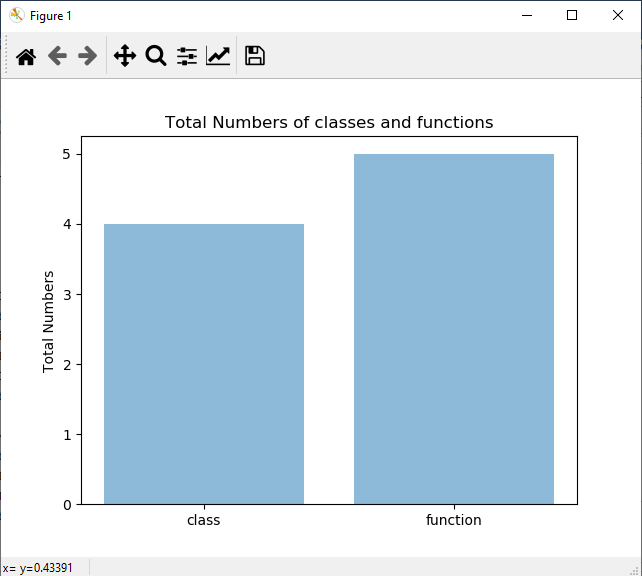


* 1. Robustness
* Both aforementioned def read\_file(self, file\_name): and def do\_read\_source\_file(self, file\_name): functions have exception handling which checks if the file exists or not and if there is error or not. My program will tell the users if file does not exist in current directory or there are errors as shown in the codes at item 5.1 above.
  1. Complete and well implemented
* My code is pythonic. It complies with PEP 8 and is beautiful better than ugly. For example:
  + The code meets the naming convention of PEP 8.
  + There are either one or two blank lines between code blocks according to PEP8.

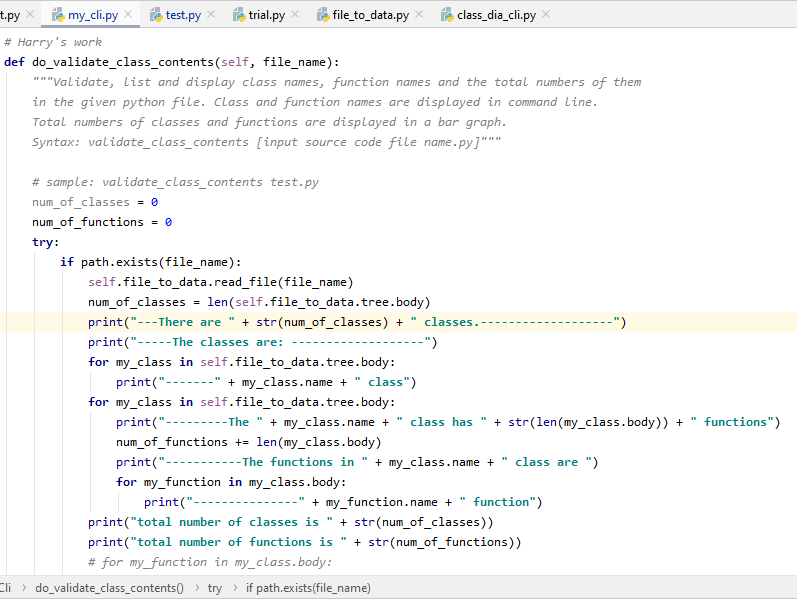
1. Validate data
   1. Used by peers

* File: my\_cli.py. The def do\_validate\_class\_contents(self, file\_name): function validates class names, function names and the total numbers of them in the given python file, and display them in command lines and a graph as below:





* 1. Robustness
* The def do\_validate\_class\_contents(self, file\_name): function has exception handling which checks if the file exists or not and if there is error or not. My program will tell the users if file does not exist in current directory or there are errors as shown in the codes below. There are “try” and “if path.exists(file\_name)”.



* 1. Complete and well implemented
* My code is pythonic. It complies with PEP 8 and is beautiful better than ugly. For example:
  + The code meets the naming convention of PEP 8.
  + There are either one or two blank lines between code blocks according to PEP8.

1. Provides object-persistence / object serialization using either pickle or shelve

* None

1. Can load data from a file
   1. Used by peers

* File: file\_to\_data.py. Data can be loaded from a python file through the def read\_file(self, file\_name): function as shown below in class FileToData(ast.NodeVisitor): in file\_to\_data.py file. This read file function is used by do\_read\_source\_file(self, file\_name): function, def do\_validate\_class\_contents(self, file\_name): function, and def do\_validate\_class\_contents(self, file\_name): function in my\_cli.py file.



* 1. Robustness
* The aforementioned def read\_file(self, file\_name): have exception handling which checks if the file exists or not and if there is error or not. My program will tell the users if file does not exist in current directory or there are errors as shown in the codes at item 8.1 above.
  1. Complete and well implemented
* My code is pythonic. It complies with PEP 8 and is beautiful better than ugly. For example:
  + The code meets the naming convention of PEP 8.
  + There are either one or two blank lines between code blocks according to PEP8.

1. Can deal with file directory
   1. Used by peers

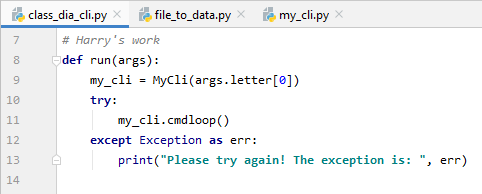
* File: file\_to\_data.py. The path.exists() function is used in def read\_file(self, file\_name): function to check if the given file is in the current file directory or not. The program will tell the user if the file is not in the current file directory. The corresponding code is below:



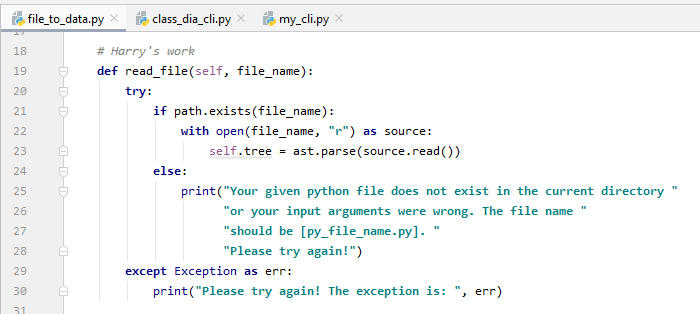
* 1. Robustness
* The aforementioned path.exists() function is for exception handling which checks if the file is in current file directory or not. My program will tell the users if the current directory does not have the file.
  1. Complete and well implemented
* My code is pythonic. It complies with PEP 8 and is beautiful better than ugly. For example:
  + The code meets the naming convention of PEP 8.
  + There are either one or two blank lines between code blocks according to PEP8.

1. Can raise exceptions and provide exception handling
   1. Used by peers

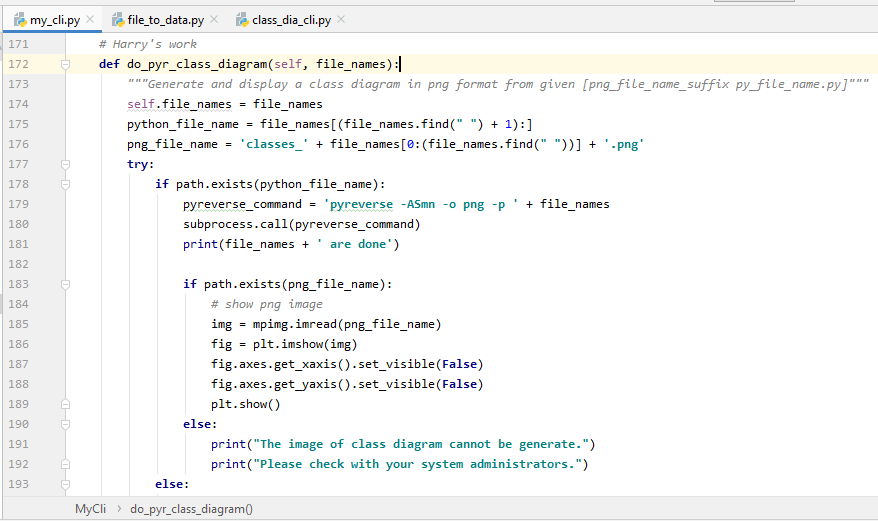
* I have provided exception handling in different parts of the codes. Few examples only (not all) are given below:
  + def run(args): function in class\_dia\_cli.py as shown below:



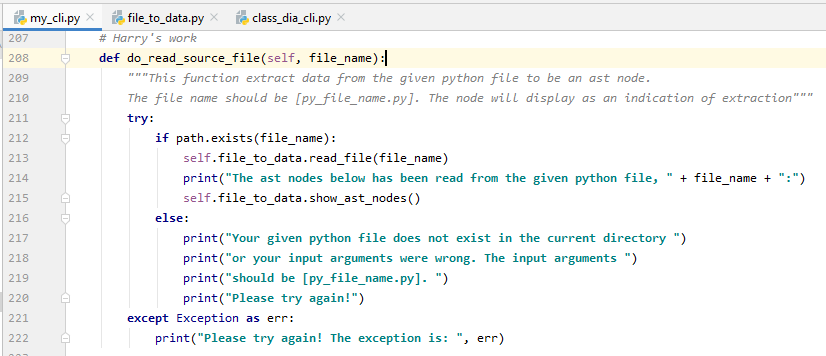
* + def read\_file(self, file\_name): function in file\_to\_data.py as shown below:



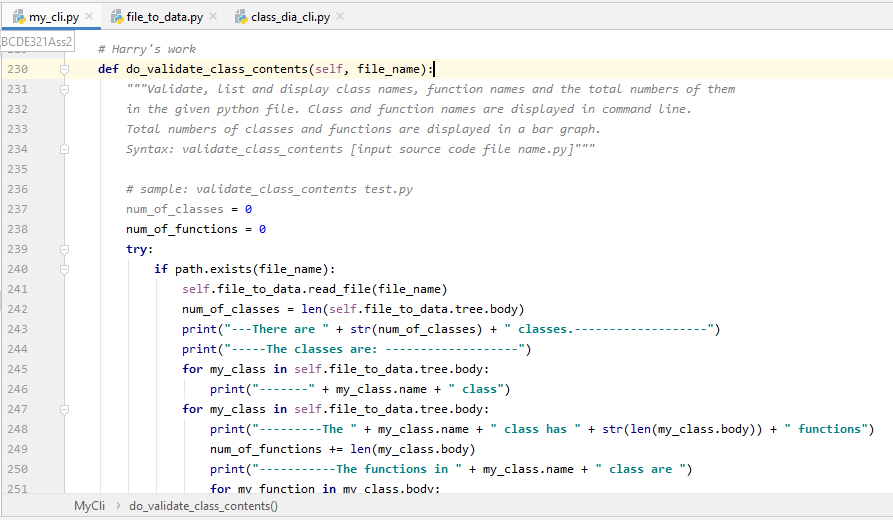
* + def do\_pyr\_class\_diagram(self, file\_names):: function in my\_cli.py as shown below:



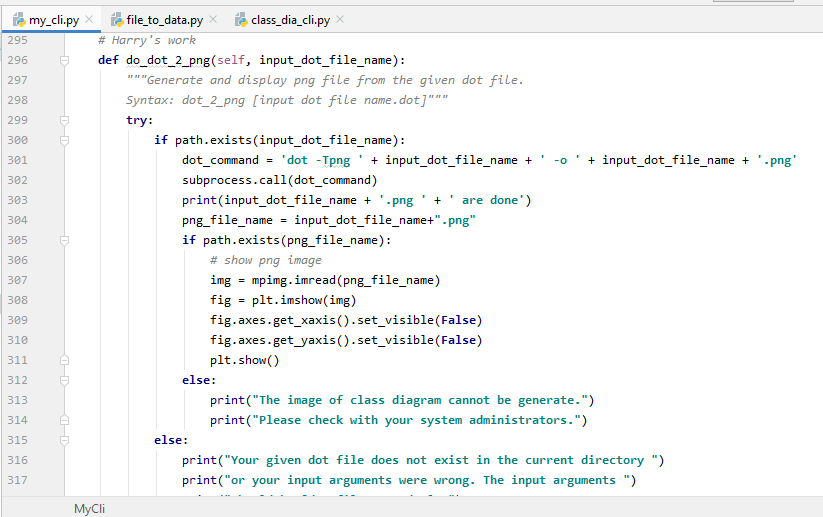
* + def do\_read\_source\_file(self, file\_names): function in my\_cli.py as shown below:



* + def do\_validate\_class\_contents(self, file\_name): function in my\_cli.py as shown below:



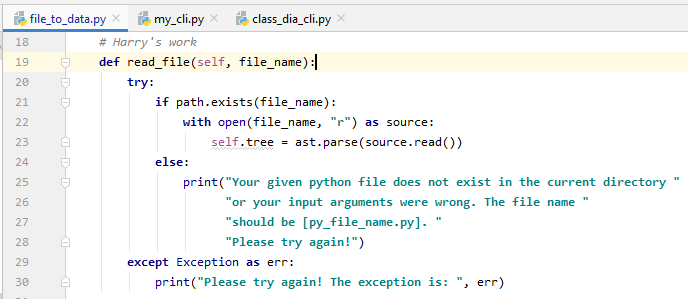
* + def do\_dot\_2\_png(self, input\_dot\_file\_name): function in my\_cli.py as shown below:



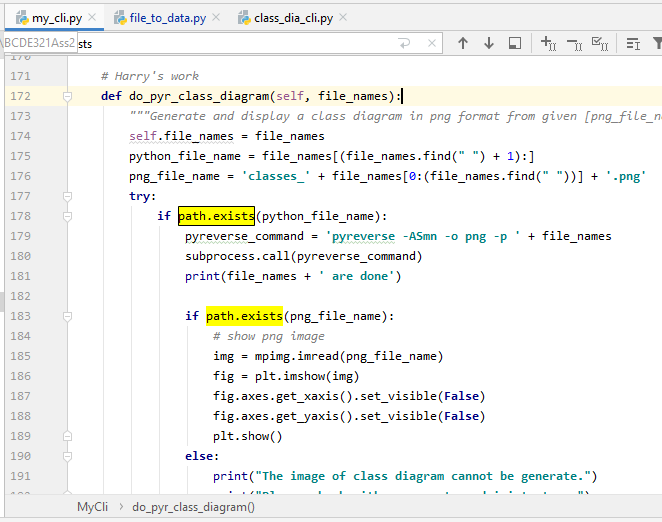
* 1. Robustness
* The aforementioned functions have exception handling which checks if there is error or not. My program will tell the users if error.
  1. Complete and well implemented
* My code is pythonic. It complies with PEP 8 and is beautiful better than ugly. For example:
  + The code meets the naming convention of PEP 8.
  + There are either one or two blank lines between code blocks according to PEP8.

1. Amount of checking for pre- and post- conditions of methods
   1. Used by peers

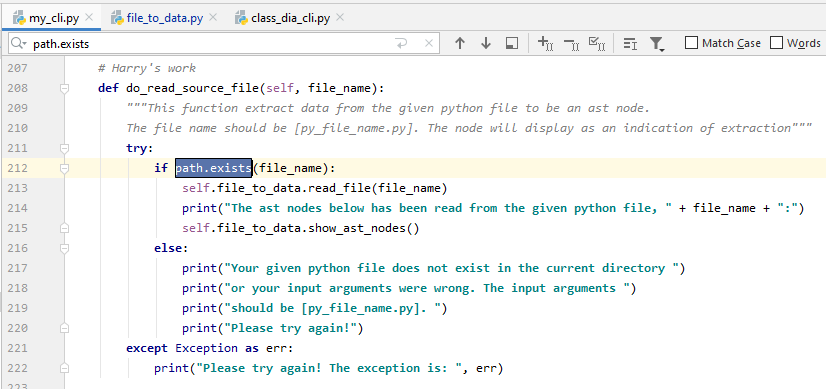
* Flies: file\_to\_data.py and my\_cli.py.
  + The def read\_file(self, file\_name): function as shown below in file\_to\_data.py has check pre-condition which checks if the required file exists in the current directory before the file is opened.



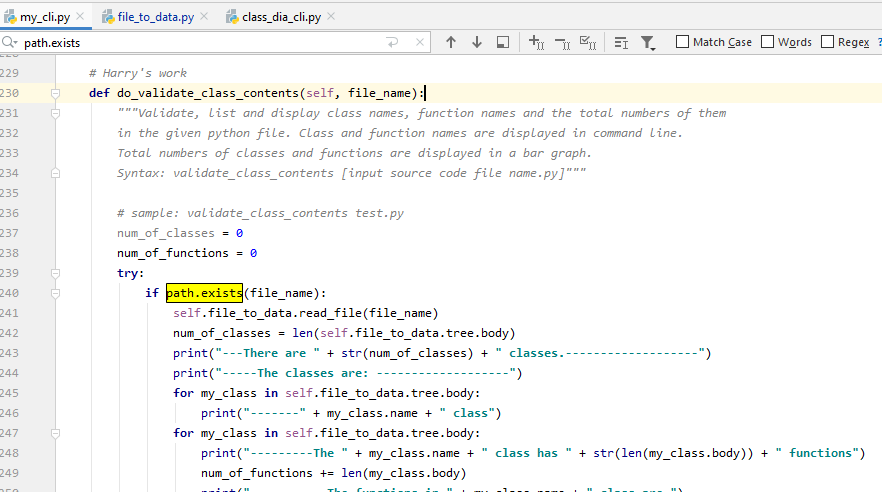
* + The def do\_pyr\_class\_diagram(self, file\_names): function as shown below in my\_cli.py has check both pre- and post- conditions which checks if both the required input file and the output file respectively as show below exist in the current directory before the files are opened.



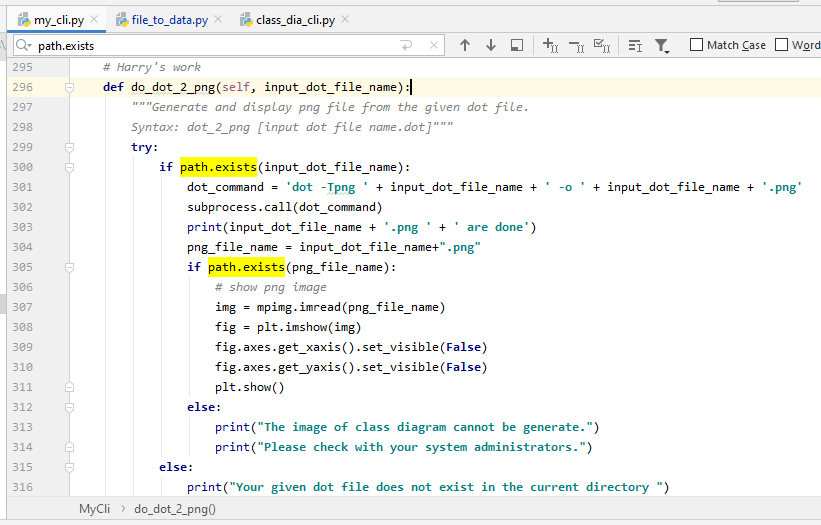
* + The def do\_read\_source\_file(self, file\_name): function as shown below in my\_cli.py has check pre-condition which checks if the required file exists in the current directory before the file is opened.



* + The def do\_validate\_class\_contents(self, file\_name): function as shown below in my\_cli.py has check pre-condition which checks if the required file exists in the current directory before the file is opened.



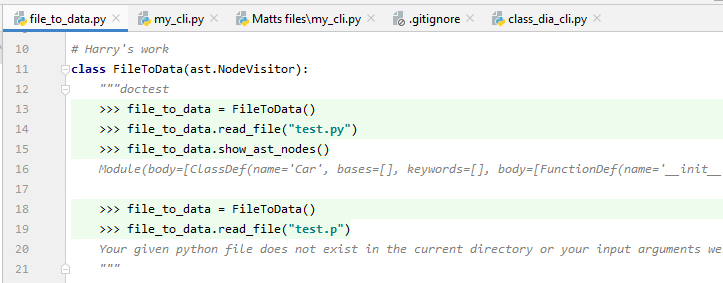
* + The def do\_dot\_2\_png(self, input\_dot\_file\_name): function as shown below in my\_cli.py has check both pre- and post- conditions which checks if both the required input file and the output file respectively as show below exist in the current directory before the files are opened.

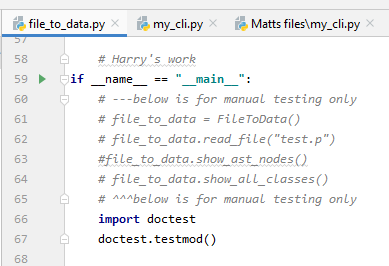


* 1. Robustness
* The path.exists() function in the aforementioned codes is for both pre- and post- conditions (for input file and output file respectively) which checks if the file is in current file directory or not. My program will tell the users if the current directory does not have the file.
  1. Complete and well implemented
* My code is pythonic. It complies with PEP 8 and is beautiful better than ugly. For example:
  + The code meets the naming convention of PEP 8.
  + There are either one or two blank lines between code blocks according to PEP8.

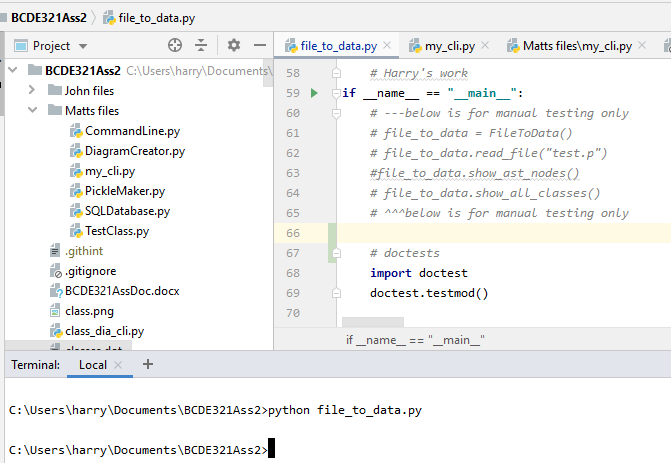
1. Provide doctests
   1. Used by peers

* File: file\_to\_data.py
  + Two doctests are shown below:





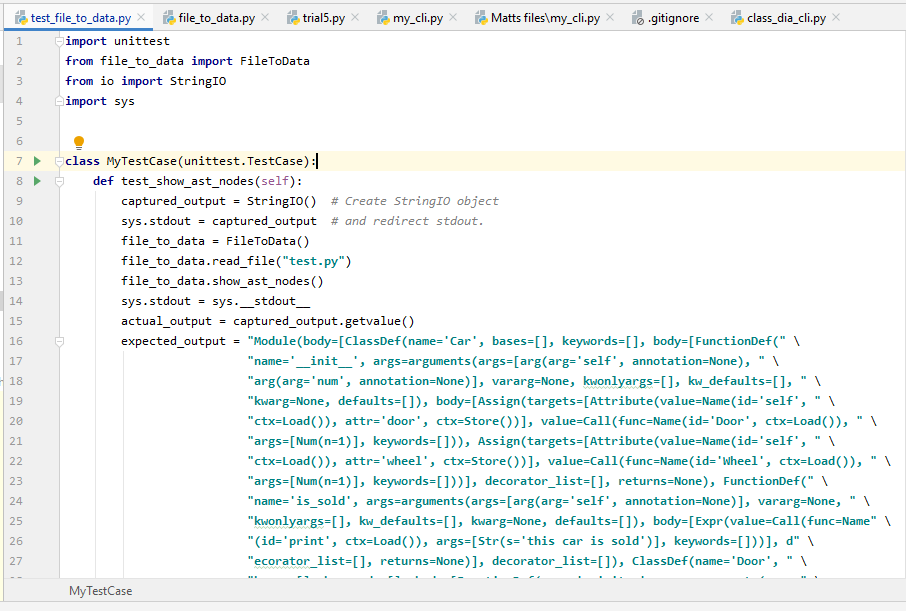
* 1. Robustness
* All doctests were passed as shown below

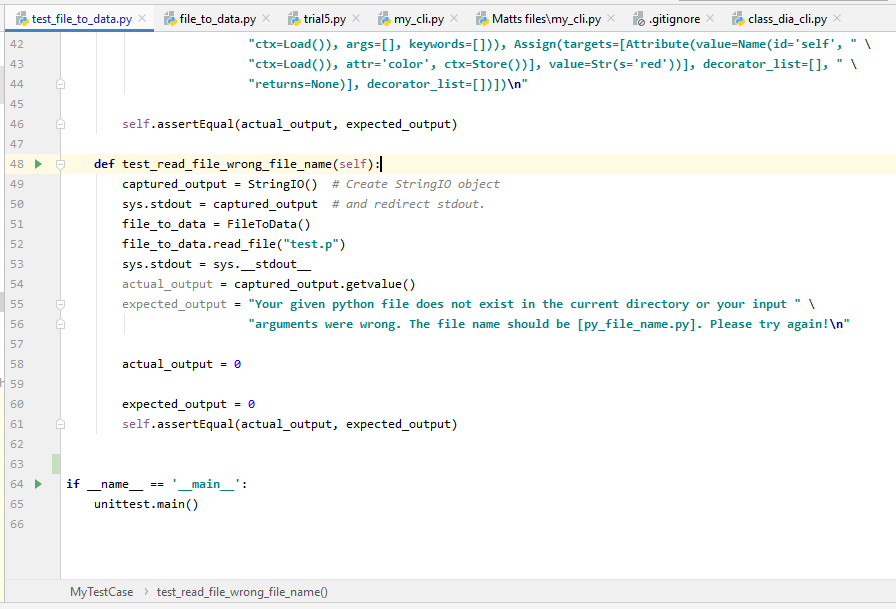


* 1. Complete and well implemented
* Less than 10 different doctests

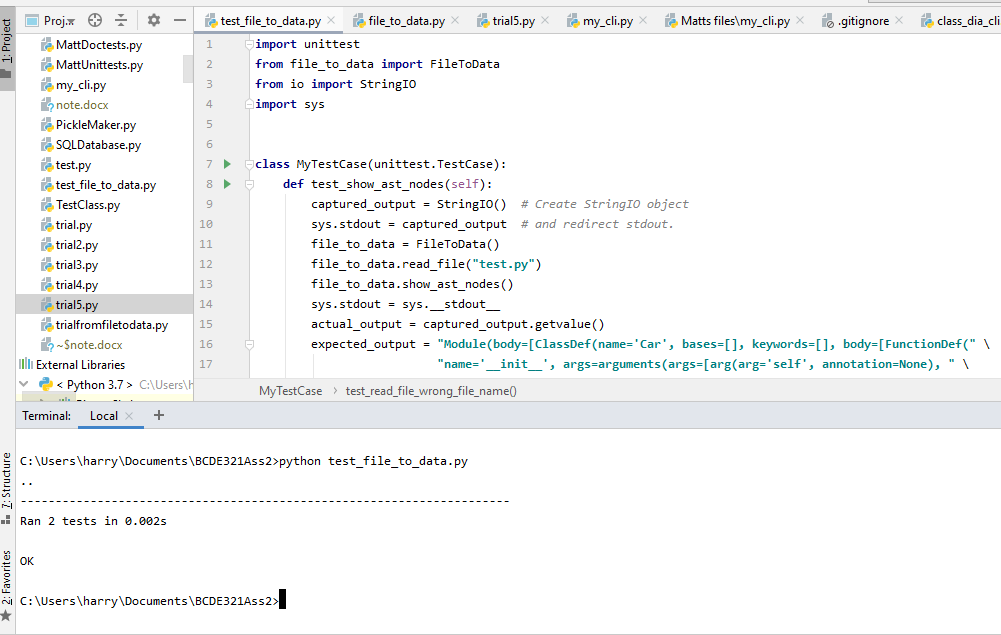
1. Provide unittests
   1. Used by peers

* File: test\_file\_to\_data.py
  + Two unit tests are shown below:





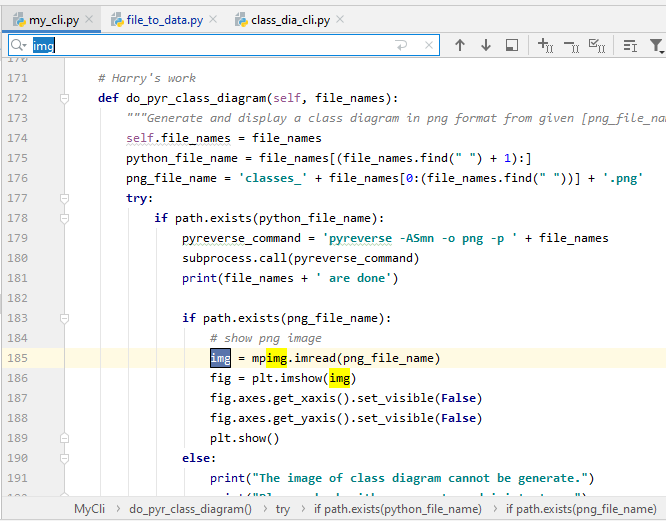
* 1. Robustness
* All unit tests were passed as shown below

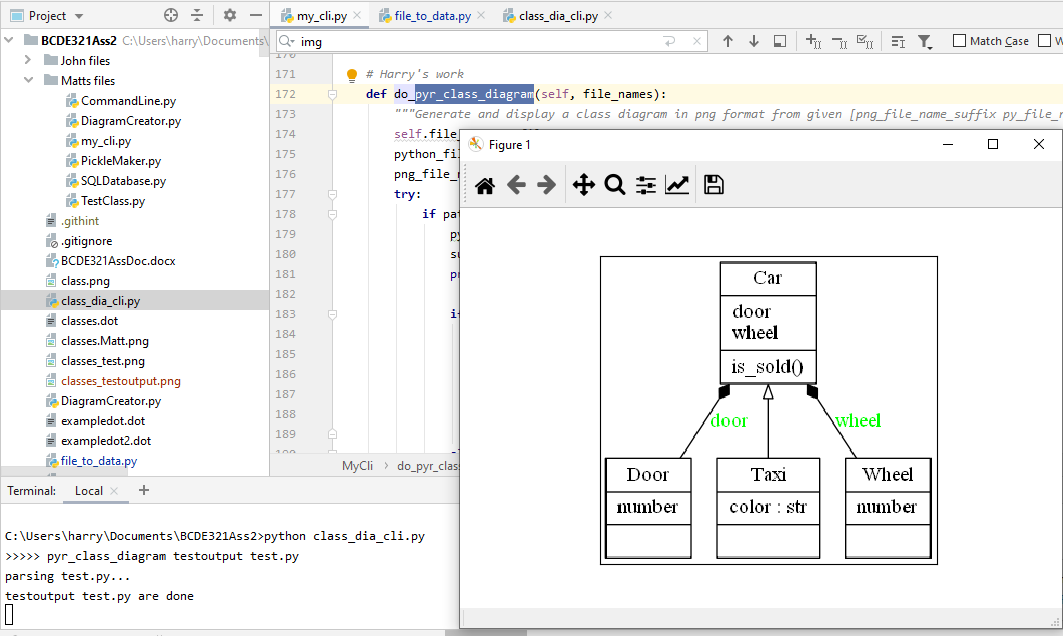


* 1. Complete and well implemented
* Less than 10 different unit tests

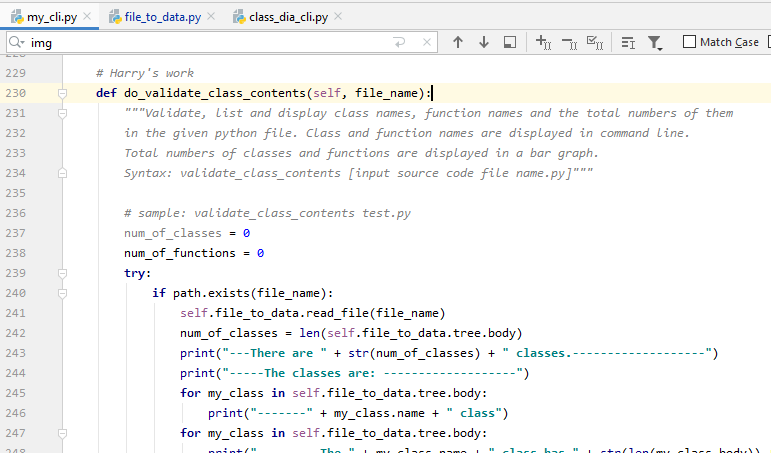
1. Pretty print, i.e., displaying data in chart / diagram, e.g., bar chart, pie chart, UML diagram, etc.
   1. Used by peers

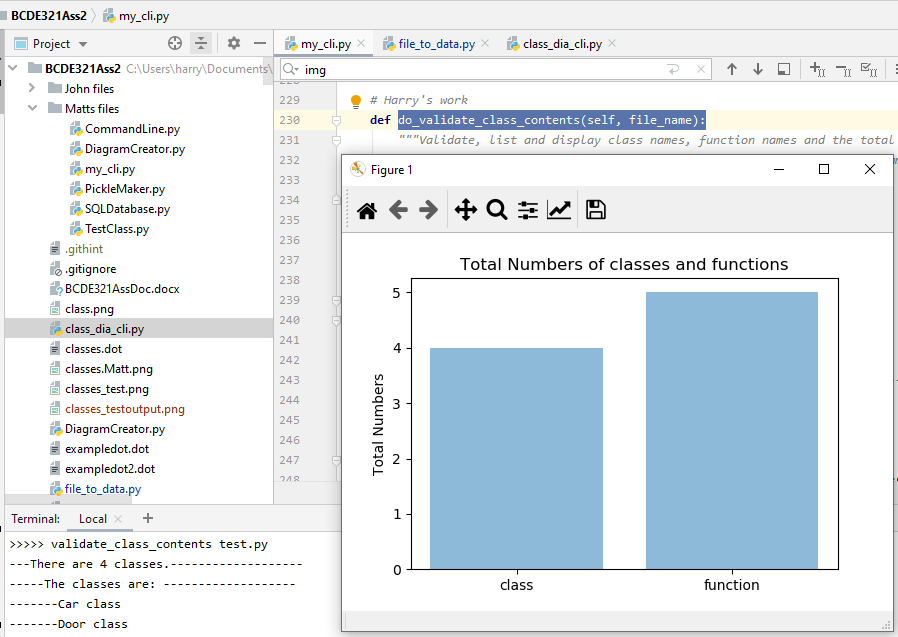
* File: my\_cli.py
  + The def do\_pyr\_class\_diagram(self, file\_names): function can display a class diagram based on a python file as shown below:



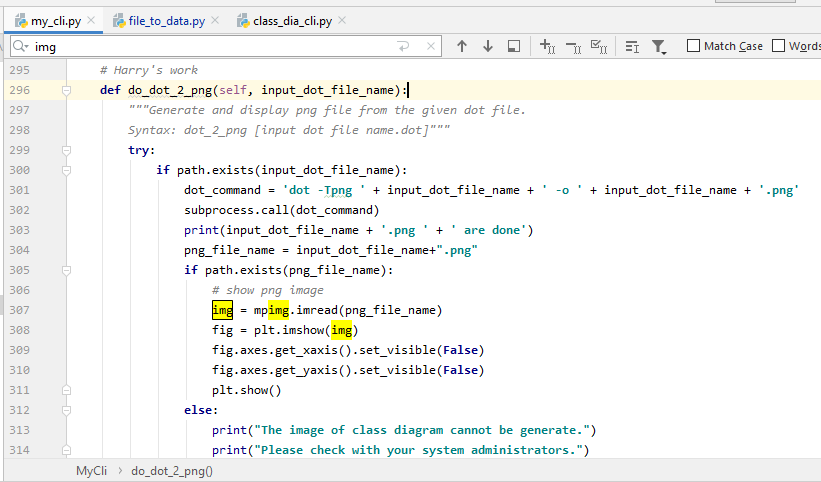


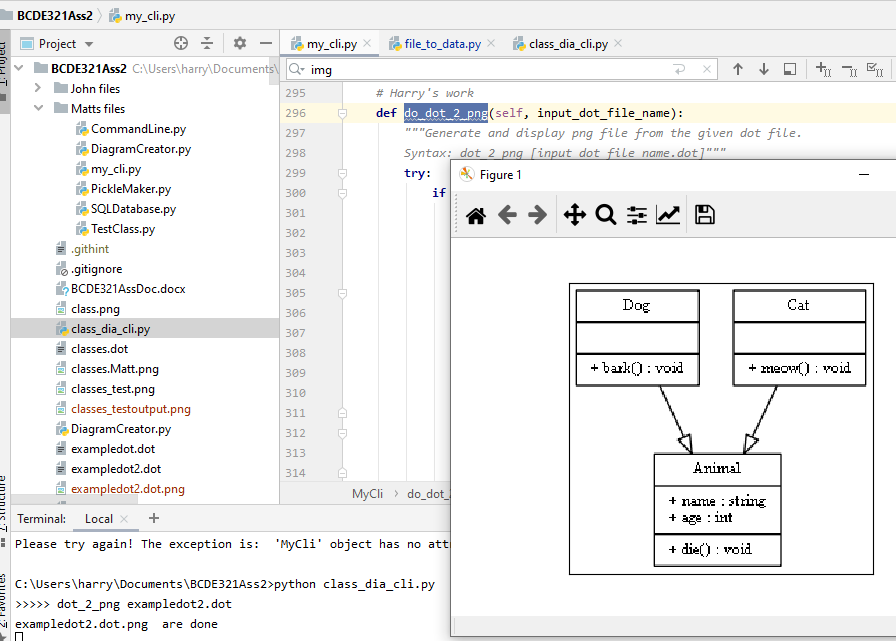
* The def do\_validate\_class\_contents(self, file\_name): function can display a bar diagram to show the total numbers of classes and functions in the input python file as shown below:





* + The def do\_dot\_2\_png(self, input\_dot\_file\_name): function can display a class diagram based on a dot file as shown below:





* 1. Robustness
* The aforementioned functions have exception handling which checks if the file exists or not and if there is error or not. My program will tell the users if file does not exist in current directory or there are errors as shown in the codes at item 14.1 above.
  1. Complete and well implemented
* My code is pythonic. It complies with PEP 8 and is beautiful better than ugly. For example:
  + The code meets the naming convention of PEP 8.
  + There are either one or two blank lines between code blocks according to PEP8.

1. Can save and read data from a database ,e.g., from a database, e.g., SQLite, MySQL and MongoDB.

* None

# Location of GitHub repository

<https://github.com/harrykhlo/BCDE321Ass2>