Bill McG…

March 20, 2022

Foundations of Programming: Python

Assignment 08

GitHub Link: https://github.com/wwm787/Assignment\_08

# Working with Classes and Objects, Constructors, Fields, Attributes and Methods

# Introduction

In this assignment, I learned about objects that are made from a class, the purpose of a constructor, when to use keywords ‘self’ and ‘@staticmethod’, methods and properties. Details are contained in the summary below.

# Assignment 08 – CD Inventory Program Using Classes and Objects, Constructors, Fields, Attributes and Methods

In assignment 08, the script allows the user to inventory a CD collection. The program asks the user to either load data from a text file, add item to memory, view inventory, delete an item from memory, save to text file or exit the program. It is like assignment 06, except it utilizes Object Oriented Programming.

* Line 1: Import sys library to gain access to exit() function
* Lines 8 - 53: DataProcesser class for processing deletions and additions
* Lines 56 – 115: CD class
  + Create constructor for the fields
  + Create properties cd\_id, cd\_title and cd\_artist
  + Create setters cd\_id, cd\_title and cd\_artist
* Lines 119 – 181: Group file read and write functions into a class
  + Functions
    - Read text file that is called at program startup and when inventory is selected in menu
      * Added structured error handling to check for missing files
    - Write to text file that is called when save is selected from the menu
      * Added structured error handling to check for missing files
* Lines 186 – 280: Group IO functions into a class
  + Functions
    - Display menu at startup
    - Get user menu choice
    - Show current inventory in memory (not saved to text file)
    - Add function when user selects ‘a’ in program menu that allows the user to enter a new CD ID, Title and Artist via input command
      * Added structured error handling to check for
        + The ID is an integer
        + Title and Artist are not blank
* Line 286: Load currently saved inventory from text file at startup
* Lines 288 – 348: Series of if, elif and else statements to process menu items and call the specified data processing, read/write or IO functions
  + Choices are the exit the program, load from text file, view inventory, add a record, delete a record or save to text file.
  + For choice ‘d’ delete, added structured error handling to ensure only integers are entered

1. import sys
2. # -- DATA -- #
3. strFileName = 'CDInventory.txt'
4. lstOfCDObjects = []
5. class DataProcessor:
6. def myDeleteDataProcFunc(intIDDelReceived):
7. """ Function to delete CD based on ID passed to function
8. Args:
9. intIDDelReceived (int): ID of CD to delete.
10. Returns:
11. None.
12. """
13. # search thru table and delete CD
14. intRowNr = -1
15. blnCDRemoved = False
16. for row in lstOfCDObjects:
17. intRowNr += 1
18. if lstOfCDObjects[intRowNr].cd\_id\_a == intIDDelReceived:
19. del lstOfCDObjects[intRowNr]
20. blnCDRemoved = True
21. break
22. if blnCDRemoved:
23. print("The CD was removed")
24. else:
25. print("Could not find this CD!")
26. IO.disp\_current\_data\_screen(lstOfCDObjects) # display inventory
27. return
28. @staticmethod
29. def myAddProcCode(myID, myTitle, myArtist):
30. """ Function to process ID, Title and Artist
32. Args:
33. myID (string): ID of CD.
34. myTitle (string): Title of CD.
35. myArtist (string): Artist name.
36. Returns:
37. None.
38. """
39. # Add item to the table
40. intID = int(myID)
41. lstTbl = [intID, myTitle, myArtist]
42. lstOfCDObjects.append(CD(lstTbl[0], lstTbl[1], lstTbl[2]))
43. IO.disp\_current\_data\_screen(lstOfCDObjects)
44. class CD(object):
45. """Stores data about a CD:
46. properties:
47. cd\_id: (int) with CD ID
48. cd\_title: (string) with the title of the CD
49. cd\_artist: (string) with the artist of the CD
50. methods:
51. """
52. #fields#
53. \_\_numCans = 0
54. #constructor#
55. def \_\_init\_\_(self, cd\_id, cd\_title, cd\_artist):
56. #attributes#
57. self.\_\_cd\_id = cd\_id
58. self.\_\_cd\_title = cd\_title
59. self.\_\_cd\_artist = cd\_artist
60. CD.\_\_incrementCount()
61. @property
62. def cd\_id\_a(self):
63. return self.\_\_cd\_id
64. @property
65. def cd\_title\_a(self):
66. return self.\_\_cd\_title.title()
67. @property
68. def cd\_artist\_a(self):
69. return self.\_\_cd\_artist.title()
70. @staticmethod
71. def \_\_incrementCount():
72. CD.\_\_numCans += 1
73. @cd\_id\_a.setter
74. def cd\_id\_a(self, value):
75. if str(value).isnumeric():
76. raise Exception("This message can\'t be cryptic")
77. else:
78. self.\_\_cd\_id = value
79. @cd\_title\_a.setter
80. def cd\_title\_a(self, value):
81. if str(value).isnumeric():
82. raise Exception("This message can\'t be cryptic")
83. else:
84. self.\_\_cd\_title = value
85. @cd\_artist\_a.setter
86. def cd\_artist\_a(self, value):
87. if str(value).isnumeric():
88. raise Exception("This message can\'t be cryptic")
89. else:
90. self.\_\_cd\_artist = value
91. # -- PROCESSING -- #
92. class FileIO:
93. """Processes data to and from file:
94. properties:
95. methods:
96. read\_file(file\_name, lst\_Inventory): -> None
97. write\_file(file\_name): -> (a list of CD objects)
98. """
99. @staticmethod
100. def read\_file(file\_name, lstTbl):
101. """Function to manage data ingestion from file to a list of dictionaries
102. Reads the data from file identified by file\_name into a 2D table
103. (list of dicts) table one line in the file represents one dictionary row in table.
104. Args:
105. file\_name (string): name of file used to read the data from
106. table (list of dictionary): 2D data structure (list of dicts) that holds the data during runtime
107. Returns:
108. None.
109. """
110. try:
111. lstTbl.clear() # this clears existing data and allows to load data from file
112. lstOfCDObjects.clear()
113. objFile = open(file\_name, "r")
114. for line in objFile:
115. data = line.strip().split(",")
116. lstTbl = [int(data[0]), data[1], data[2]]
117. lstOfCDObjects.append(CD(lstTbl[0], lstTbl[1], lstTbl[2]))
118. # startTbl = CD(line)
119. objFile.close()
120. except Exception as e:
121. print("\nYou need to create a CDInventory.txt file first. \n")
122. print(e)
123. print("Exiting Program\n")
124. sys.exit()
125. @staticmethod
126. def write\_file(file\_name, recTbl): # save data
127. """ Function to save table data to text file
129. Args:
130. file\_name (string): name of the file used to write data to.
131. recTbl (list of dictionary): 2D data structure (list of dicts) that holds the data during runtime.
132. Returns:
133. None.
134. """
135. # Save to text file
136. billIntA = 0
137. objFile = open(file\_name, "w")
138. for row in recTbl: # Parse each row
139. lstValues = [lstOfCDObjects[billIntA].cd\_id\_a,
140. lstOfCDObjects[billIntA].cd\_title\_a, lstOfCDObjects[billIntA].cd\_artist\_a]
141. lstValues[0] = str(lstValues[0])
142. objFile.write(",".join(lstValues) + "\n")
143. billIntA += 1
144. objFile.close()
145. # -- PRESENTATION (Input/Output) -- #
146. class IO(object):
147. """Handling Input / Output"""
148. @staticmethod
149. def show\_menu():
150. """Displays a menu of choices to the user
151. Args:
152. None.
153. Returns:
154. None.
155. """
156. print(
157. "Menu\n\n[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory")
158. print("[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit\n")
159. @staticmethod
160. def capture\_users\_choice():
161. """Gets user input for menu selection
162. Args:
163. None.
164. Returns:
165. choice (string): a lower case sting of the users input out of the choices l, a, i, d, s or x
166. """
167. choice = " "
168. while choice not in ["l", "a", "i", "d", "s", "x"]:
169. choice = input(
170. "Which operation would you like to perform? [l, a, i, d, s or x]: ").lower().strip()
171. print() # Add extra space for layout
172. return choice
173. @staticmethod
174. def disp\_current\_data\_screen(lstOfCDObjects):
175. """Displays current inventory of table invTbl
176. Args:
177. invTbl (list of dict): 2D data structure (list of dicts) that holds the data during runtime.
178. Returns:
179. None.
180. """
181. print("======= The Current Inventory: =======")
182. print("ID\tCD Title (by: Artist)\n")
183. billIntA = 0
184. for row in lstOfCDObjects:
185. print("{}\t{} (by:{})".format(
186. lstOfCDObjects[billIntA].cd\_id\_a, lstOfCDObjects[billIntA].cd\_title\_a, lstOfCDObjects[billIntA].cd\_artist\_a))
187. billIntA += 1
188. print("======================================")
189. @staticmethod
190. def get\_CD\_add\_data\_from\_user():
191. """ Function for input / ouput
192. Ask the user CD ID, Title and Artist
194. Returns:
195. strID1 (string): User inputted CD ID.
196. strTitle1 (string): User inputted CD Title.
197. strArtist1 (string): User inputted CD Artist.
199. """
200. while True:
201. strID1 = input("Enter ID: ").strip()
202. try:
203. intID = int(strID1)
204. except ValueError as e:
205. print("\n")
206. print("That is not an Integer!")
207. continue
208. while True:
209. strTitle1 = input("What is the CD\"s title? ").strip()
210. try: # if blank raise error and start over
211. if len(strTitle1) == 0:
212. raise ValueError("You must enter a Title!")
213. except ValueError as e:
214. print("\n")
215. print(e)
216. continue
217. while True:
218. strArtist1 = input("What is the Artist\"s name? ").strip()
219. try: # if blank raise error and start over
220. if len(strArtist1) == 0:
221. raise ValueError("You must enter an Artist!")
222. except ValueError as e:
223. print("\n")
224. print(e)
225. continue
226. return strID1, strTitle1, strArtist1
227. # -- Main Body of Script -- #
228. # Load data from file into a list of CD objects on script start
229. FileIO.read\_file(strFileName, lstOfCDObjects)
230. while True:
231. # Display Menu to user and get choice
232. IO.show\_menu()
233. strChoice = IO.capture\_users\_choice()
234. # Process menu selection
235. # Process exit first
236. if strChoice == "x":
237. break
238. # Process load inventory
239. if strChoice == "l":
240. print(
241. "WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.")
242. strYesNo = input(
243. "type \"yes\" to continue and reload from file. otherwise reload will be canceled: ")
244. if strYesNo.lower() == "yes":
245. print("reloading...")
246. FileIO.read\_file(strFileName, lstOfCDObjects)
247. IO.disp\_current\_data\_screen(lstOfCDObjects)
248. else:
249. input(
250. "canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.")
251. IO.disp\_current\_data\_screen(lstOfCDObjects)
252. continue # start loop back at top.
253. # Process add a CD
254. elif strChoice == "a":
255. # Ask user for new ID, CD Title and Artist
256. strID, strTitle, strArtist = IO.get\_CD\_add\_data\_from\_user()
257. #IO.get\_CD\_add\_data\_from\_user()
258. DataProcessor.myAddProcCode(strID, strTitle, strArtist)
259. continue # start loop back at top.
260. # Process display current inventory
261. elif strChoice == "i":
262. IO.disp\_current\_data\_screen(lstOfCDObjects)
263. continue # start loop back at top.
264. elif strChoice == "d": # process delete a CD
265. IO.disp\_current\_data\_screen(lstOfCDObjects) # display inventory
266. # Get Userinput for which CD to delete
267. try:
268. intIDDelInput = int(
269. input("Which ID would you like to delete? ").strip())
270. except ValueError as e:
271. print("\n")
272. print("That is not an Integer!")
273. continue
274. DataProcessor.myDeleteDataProcFunc(intIDDelInput)
275. continue # start loop back at top.
276. elif strChoice == "s": # process save inventory to file
277. # Display current inventory and ask user for confirmation to save
278. IO.disp\_current\_data\_screen(lstOfCDObjects)
279. strYesNo = input(
280. "Save this inventory to file? [y/n] ").strip().lower()
281. # Process choice
282. if strYesNo == "y":
283. FileIO.write\_file(strFileName, lstOfCDObjects)
284. else:
285. input(
286. "The inventory was NOT saved to file. Press [ENTER] to return to the menu.")
287. continue # start loop back at top.
288. # Catch-all should not be possible, as user choice gets vetted in IO, but to be save:
289. else:
290. print("General Error")

Listing - CDInventory.py code

|  |  |
| --- | --- |
| Figure - CDInventory.py Spyder Demo | Figure - CDInventory.py Spyder Demo |

|  |
| --- |
| Figure - CDInventory.py Terminal Window Demo |

# LAB 08-E: Working with Files and Dictionaries:

This lab works by creating a class called TrackInfo with fields position, title and length. A constructor is created with the position, title and length attributes. Getter and Setter methods are then created to allow accessing track position, title and length as well as writing track position, title and length.

# Summary

The difference between a class and the objects made from a class is that a class is not an object, instead a class is the design / blueprint for an object. One benefit is that multiple objects that are instantiated from a single class will have a similar structure. Classes are typically composed of Fields, Constructors, Attributes and Methods. The ‘self’ keyword is used to represent the current instance and allows access to attributes and methods. Whereas the ‘@staticmethod’ is used when you do not want to reference individual objects, instead you want to reference at the class level. The connected method from the CanOnAString class is an example of a @staticmethod. Fields are data stores of a class, attributes are the variables and properties are special methods that access the attribute and write to the attribute. These methods are also known as Getters and Setters.

# Appendix

## Listing CDInventory.py

1. #-----------------------------------------------------------------------#
2. # Title: CD\_Inventory.py
3. # Desc: Assignnment 08 - Working with Classes and Objects, Constructors, Fields, Attributes and Methods
4. # Change Log: (Who, When, What)
5. # DBiesinger, 2030-Jan-01, created file
6. # DBiesinger, 2030-Jan-01, added pseudocode to complete assignment 08
7. # Bill McG…., 2022-Mar-20, Modified File to add functionality
8. #-----------------------------------------------------------------------#
9. import sys
10. # -- DATA -- #
11. strFileName = 'CDInventory.txt'
12. lstOfCDObjects = []
13. class DataProcessor:
14. def myDeleteDataProcFunc(intIDDelReceived):
15. """ Function to delete CD based on ID passed to function
16. Args:
17. intIDDelReceived (int): ID of CD to delete.
18. Returns:
19. None.
20. """
21. # search thru table and delete CD
22. intRowNr = -1
23. blnCDRemoved = False
24. for row in lstOfCDObjects:
25. intRowNr += 1
26. if lstOfCDObjects[intRowNr].cd\_id\_a == intIDDelReceived:
27. del lstOfCDObjects[intRowNr]
28. blnCDRemoved = True
29. break
30. if blnCDRemoved:
31. print("The CD was removed")
32. else:
33. print("Could not find this CD!")
34. IO.disp\_current\_data\_screen(lstOfCDObjects) # display inventory
35. return
36. @staticmethod
37. def myAddProcCode(myID, myTitle, myArtist):
38. """ Function to process ID, Title and Artist
40. Args:
41. myID (string): ID of CD.
42. myTitle (string): Title of CD.
43. myArtist (string): Artist name.
44. Returns:
45. None.
46. """
47. # Add item to the table
48. intID = int(myID)
49. lstTbl = [intID, myTitle, myArtist]
50. lstOfCDObjects.append(CD(lstTbl[0], lstTbl[1], lstTbl[2]))
51. IO.disp\_current\_data\_screen(lstOfCDObjects)
52. class CD(object):
53. """Stores data about a CD:
54. properties:
55. cd\_id: (int) with CD ID
56. cd\_title: (string) with the title of the CD
57. cd\_artist: (string) with the artist of the CD
58. methods:
59. """
60. #fields#
61. \_\_numCans = 0
62. #constructor#
63. def \_\_init\_\_(self, cd\_id, cd\_title, cd\_artist):
64. #attributes#
65. self.\_\_cd\_id = cd\_id
66. self.\_\_cd\_title = cd\_title
67. self.\_\_cd\_artist = cd\_artist
68. CD.\_\_incrementCount()
69. @property
70. def cd\_id\_a(self):
71. return self.\_\_cd\_id
72. @property
73. def cd\_title\_a(self):
74. return self.\_\_cd\_title.title()
75. @property
76. def cd\_artist\_a(self):
77. return self.\_\_cd\_artist.title()
78. @staticmethod
79. def \_\_incrementCount():
80. CD.\_\_numCans += 1
81. @cd\_id\_a.setter
82. def cd\_id\_a(self, value):
83. if str(value).isnumeric():
84. raise Exception("This message can\'t be cryptic")
85. else:
86. self.\_\_cd\_id = value
87. @cd\_title\_a.setter
88. def cd\_title\_a(self, value):
89. if str(value).isnumeric():
90. raise Exception("This message can\'t be cryptic")
91. else:
92. self.\_\_cd\_title = value
93. @cd\_artist\_a.setter
94. def cd\_artist\_a(self, value):
95. if str(value).isnumeric():
96. raise Exception("This message can\'t be cryptic")
97. else:
98. self.\_\_cd\_artist = value
99. # -- PROCESSING -- #
100. class FileIO:
101. """Processes data to and from file:
102. properties:
103. methods:
104. read\_file(file\_name, lst\_Inventory): -> None
105. write\_file(file\_name): -> (a list of CD objects)
106. """
107. @staticmethod
108. def read\_file(file\_name, lstTbl):
109. """Function to manage data ingestion from file to a list of dictionaries
110. Reads the data from file identified by file\_name into a 2D table
111. (list of dicts) table one line in the file represents one dictionary row in table.
112. Args:
113. file\_name (string): name of file used to read the data from
114. table (list of dictionary): 2D data structure (list of dicts) that holds the data during runtime
115. Returns:
116. None.
117. """
118. try:
119. lstTbl.clear() # this clears existing data and allows to load data from file
120. lstOfCDObjects.clear()
121. objFile = open(file\_name, "r")
122. for line in objFile:
123. data = line.strip().split(",")
124. lstTbl = [int(data[0]), data[1], data[2]]
125. lstOfCDObjects.append(CD(lstTbl[0], lstTbl[1], lstTbl[2]))
126. # startTbl = CD(line)
127. objFile.close()
128. except Exception as e:
129. print("\nYou need to create a CDInventory.txt file first. \n")
130. print(e)
131. print("Exiting Program\n")
132. sys.exit()
133. @staticmethod
134. def write\_file(file\_name, recTbl): # save data
135. """ Function to save table data to text file
137. Args:
138. file\_name (string): name of the file used to write data to.
139. recTbl (list of dictionary): 2D data structure (list of dicts) that holds the data during runtime.
140. Returns:
141. None.
142. """
143. # Save to text file
144. billIntA = 0
145. objFile = open(file\_name, "w")
146. for row in recTbl: # Parse each row
147. lstValues = [lstOfCDObjects[billIntA].cd\_id\_a,
148. lstOfCDObjects[billIntA].cd\_title\_a, lstOfCDObjects[billIntA].cd\_artist\_a]
149. lstValues[0] = str(lstValues[0])
150. objFile.write(",".join(lstValues) + "\n")
151. billIntA += 1
152. objFile.close()
153. # -- PRESENTATION (Input/Output) -- #
154. class IO(object):
155. """Handling Input / Output"""
156. @staticmethod
157. def show\_menu():
158. """Displays a menu of choices to the user
159. Args:
160. None.
161. Returns:
162. None.
163. """
164. print(
165. "Menu\n\n[l] load Inventory from file\n[a] Add CD\n[i] Display Current Inventory")
166. print("[d] delete CD from Inventory\n[s] Save Inventory to file\n[x] exit\n")
167. @staticmethod
168. def capture\_users\_choice():
169. """Gets user input for menu selection
170. Args:
171. None.
172. Returns:
173. choice (string): a lower case sting of the users input out of the choices l, a, i, d, s or x
174. """
175. choice = " "
176. while choice not in ["l", "a", "i", "d", "s", "x"]:
177. choice = input(
178. "Which operation would you like to perform? [l, a, i, d, s or x]: ").lower().strip()
179. print() # Add extra space for layout
180. return choice
181. @staticmethod
182. def disp\_current\_data\_screen(lstOfCDObjects):
183. """Displays current inventory of table invTbl
184. Args:
185. invTbl (list of dict): 2D data structure (list of dicts) that holds the data during runtime.
186. Returns:
187. None.
188. """
189. print("======= The Current Inventory: =======")
190. print("ID\tCD Title (by: Artist)\n")
191. billIntA = 0
192. for row in lstOfCDObjects:
193. print("{}\t{} (by:{})".format(
194. lstOfCDObjects[billIntA].cd\_id\_a, lstOfCDObjects[billIntA].cd\_title\_a, lstOfCDObjects[billIntA].cd\_artist\_a))
195. billIntA += 1
196. print("======================================")
197. @staticmethod
198. def get\_CD\_add\_data\_from\_user():
199. """ Function for input / ouput
200. Ask the user CD ID, Title and Artist
202. Returns:
203. strID1 (string): User inputted CD ID.
204. strTitle1 (string): User inputted CD Title.
205. strArtist1 (string): User inputted CD Artist.
207. """
208. while True:
209. strID1 = input("Enter ID: ").strip()
210. try:
211. intID = int(strID1)
212. except ValueError as e:
213. print("\n")
214. print("That is not an Integer!")
215. continue
216. while True:
217. strTitle1 = input("What is the CD\"s title? ").strip()
218. try: # if blank raise error and start over
219. if len(strTitle1) == 0:
220. raise ValueError("You must enter a Title!")
221. except ValueError as e:
222. print("\n")
223. print(e)
224. continue
225. while True:
226. strArtist1 = input("What is the Artist\"s name? ").strip()
227. try: # if blank raise error and start over
228. if len(strArtist1) == 0:
229. raise ValueError("You must enter an Artist!")
230. except ValueError as e:
231. print("\n")
232. print(e)
233. continue
234. return strID1, strTitle1, strArtist1
235. # -- Main Body of Script -- #
236. # Load data from file into a list of CD objects on script start
237. FileIO.read\_file(strFileName, lstOfCDObjects)
238. while True:
239. # Display Menu to user and get choice
240. IO.show\_menu()
241. strChoice = IO.capture\_users\_choice()
242. # Process menu selection
243. # Process exit first
244. if strChoice == "x":
245. break
246. # Process load inventory
247. if strChoice == "l":
248. print(
249. "WARNING: If you continue, all unsaved data will be lost and the Inventory re-loaded from file.")
250. strYesNo = input(
251. "type \"yes\" to continue and reload from file. otherwise reload will be canceled: ")
252. if strYesNo.lower() == "yes":
253. print("reloading...")
254. FileIO.read\_file(strFileName, lstOfCDObjects)
255. IO.disp\_current\_data\_screen(lstOfCDObjects)
256. else:
257. input(
258. "canceling... Inventory data NOT reloaded. Press [ENTER] to continue to the menu.")
259. IO.disp\_current\_data\_screen(lstOfCDObjects)
260. continue # start loop back at top.
261. # Process add a CD
262. elif strChoice == "a":
263. # Ask user for new ID, CD Title and Artist
264. strID, strTitle, strArtist = IO.get\_CD\_add\_data\_from\_user()
265. #IO.get\_CD\_add\_data\_from\_user()
266. DataProcessor.myAddProcCode(strID, strTitle, strArtist)
267. continue # start loop back at top.
268. # Process display current inventory
269. elif strChoice == "i":
270. IO.disp\_current\_data\_screen(lstOfCDObjects)
271. continue # start loop back at top.
272. elif strChoice == "d": # process delete a CD
273. IO.disp\_current\_data\_screen(lstOfCDObjects) # display inventory
274. # Get Userinput for which CD to delete
275. try:
276. intIDDelInput = int(
277. input("Which ID would you like to delete? ").strip())
278. except ValueError as e:
279. print("\n")
280. print("That is not an Integer!")
281. continue
282. DataProcessor.myDeleteDataProcFunc(intIDDelInput)
283. continue # start loop back at top.
284. elif strChoice == "s": # process save inventory to file
285. # Display current inventory and ask user for confirmation to save
286. IO.disp\_current\_data\_screen(lstOfCDObjects)
287. strYesNo = input(
288. "Save this inventory to file? [y/n] ").strip().lower()
289. # Process choice
290. if strYesNo == "y":
291. FileIO.write\_file(strFileName, lstOfCDObjects)
292. else:
293. input(
294. "The inventory was NOT saved to file. Press [ENTER] to return to the menu.")
295. continue # start loop back at top.
296. # Catch-all should not be possible, as user choice gets vetted in IO, but to be save:
297. else:
298. print("General Error")