# Πανεπιστήμιο Πατρών

# Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής

Αρχές Γλωσσών Προγραμματισμού και Μεταφραστών

Προεραιτική εργασία Python

Ακαδ. Έτος 2020-2021

Οδυσσέας Παπαδάκης

papadako@ceid.upatras.gr

ΑΜ 1041152 ( παλιός ΑΜ 5394 )

Έτος: 10ο

Έτος εισαγωγής 2011

**Σύντομη περιγραφή του κώδικα:**

Η εκτέλεση του προγράμματος χωρίζεται σε 5 Φάσεις, και είναι διαιρεμένη σε 5 συναρτήσεις :

1. Επιλογή του Directory στο οποίο θα κατέβουν τα αρχεία απο το site της Eurostat, και στο οποίο θα αποθηκευτούν τα τελικά .csv αρχεία. ( Συνάρτηση **directory\_change** )
2. Κατέβασμα των συμπιεσμένων αρχείων, αποσυμπίεση αυτών και εγγραφή τους στον δίσκο, ως αρχεία .tsv  
   ( Συνάρτηση **downloader** )  
   Στο τέλος της συνάρτησης αυτής, ο χρήστης ρωτάται αν θέλει να συνεχίσει με την επξεργασία των αρχείων ή αν θέλει να σταματήσει έχοντας μόνο κατεβάσει τα αρχεία .tsv
3. Επεξεργασία των κατεβασμένων αρχείων ώστε να κρατηθούν μόνο τα δεδομένα που θέλουμε. Αποθήκευση των δεδομένων αυτών στον δίσκο ως αρχεία .csv ( Συνάρτηση **data\_processor** )
4. Δημιουργία της βάσης δεδομένων (MySql) και INSERTS των tables. ( Συνάρτηση **db\_store**)
5. Δημιουργία των γραφημάτων. ( Συνάρτηση **make\_charts** )

Η συνάρτηση main ( main.py) είναι η κύρια συνάρτηση του προγράμματος, η οποία καλεί τις παραπάνω συναρτήσεις.

**Παραδοχές:**

Αντλήθηκαν δεδομένα για τα έτη 2016, 2017, 2018, 2019 για τις χώρες Ελλάδα και Ισπανία.

To όνομα της βάσης δεδομένων που δημιουργείται είναι arxes\_db.

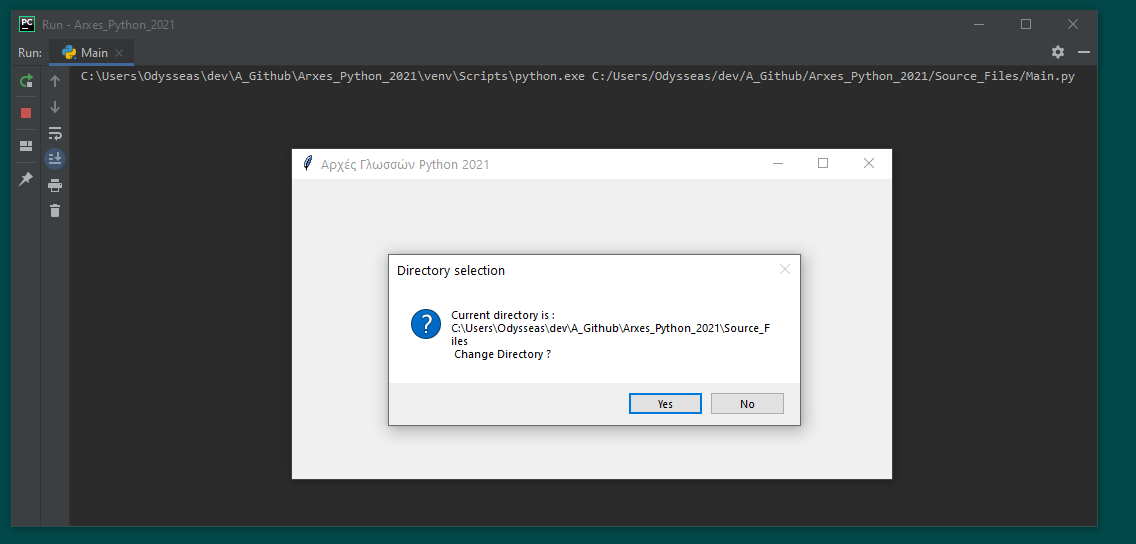
Τα γραφήματα εμφανίζονται σαν ένα μεγάλο plot με 4 subplots.

**Ακολουθεί ο κώδικας:**

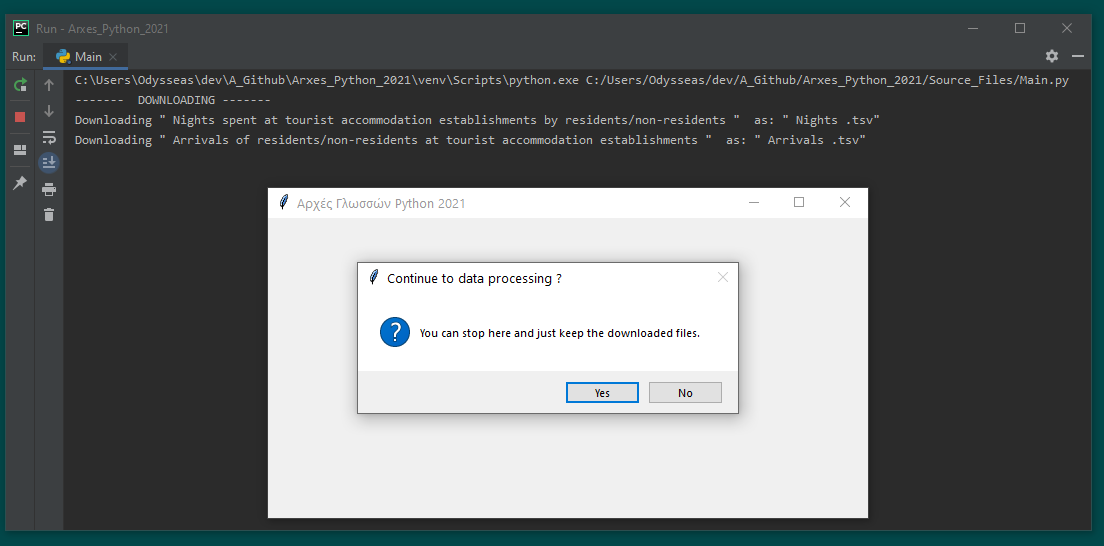
1. # This is a program that downloads some data from eurostat.
2. # It cleans it up
3. # Saves it as csv files
4. # Creates some charts
5. # And stores it into a mysql database
6. # Created by Odysseas papadakis
7. # papadako@ceid.upatras.gr
8. # AM: 104115
9. # 2021
10. **import** matplotlib**.**pyplot **as** plt
11. **from** matplotlib**.**ticker **import** FuncFormatter
12. **import** tkinter **as** tk
13. **from** tkinter **import** messagebox **as** mb
14. **from** tkinter **import** filedialog
15. **import** pandas **as** pd
16. **import** numpy **as** np
17. **import** gzip
18. **import** requests
19. **from** os **import** path
20. **from** os **import** remove
21. **import** os
22. **import** re
23. ***# Use this !!!***
24. ***# pip install mysql-connector-python***
25. **import** mysql**.**connector
26. ***# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***
27. ***# | Main code is at line 437 |***
28. ***# |\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|***
29. ***# ------------------------------ DIRECTORY SELECTOR -----------------------------------------------------------***
30. ***# This Function asks the user to specify a directory to store the files.***
31. **def** directory\_change**():**
32. ***# Get the current directory***
33. current\_directory **=** os**.**getcwd**()**
34. ***# Do you want to change directory ?***
35. dir\_change **=** mb**.**askquestion**(**"Directory selection"**,** "Current directory is :\n" **+**
36. str**(**current\_directory**)** **+**
37. "\n Change Directory ? "**)**
38. **if** dir\_change **==** 'yes'**:**
39. requested\_directory **=** tk**.**filedialog**.**askdirectory**()**
40. ***# If the user selects a directory***
41. **if** requested\_directory**:**
42. ***# Change to that directory***
43. os**.**chdir**(**requested\_directory**)**
44. ***# If the user presses the escape button or closes the directory selection window***
45. **else:**
46. mb**.**showerror**(**"DIRECTORY ERROR"**,** "No directory selected. \n " **+** "Exiting..."**)**
47. exit**(**0**)**
48. ***# This Python function***
49. ***# Downloads a compressed file from eurostat,***
50. ***# extracts the file from the gzip***
51. ***# and saves it to disk.***
52. ***# ------------------------------ DOWNLOADER -----------------------------------------------------------***
53. ***# list\_in contains 3 items which are :***
54. ***# 1) url: The url that holds the file we want***
55. ***# 2) user\_title: A user decided string, that will help identify the downloaded data***
56. ***# 3) original\_name: The original file name ( The title from eurostat)***
57. **def** downloader**(**list\_in**):**
58. url **=** list\_in**[**0**]**
59. user\_title **=** list\_in**[**1**]**
60. original\_name **=** list\_in**[**2**]**
61. **print(**"Downloading"**,** '"'**,** original\_name**,** '"'**,** ' as: "'**,** user\_title**,** '.tsv"'**)**
62. **try:**
63. ***# Download the gz to memory***
64. gz\_file **=** requests**.**get**(**url**,** allow\_redirects**=True,** timeout**=**9.00**)** ***# Expect a response within 9 seconds***
65. **except** requests**.**exceptions**:**
66. **print(**"ERROR! \n Unable to reach Website"**,** url**)**
67. ***# print(a\_ex) # Return the error type***
68. ***# exit(0)***
69. mb**.**showerror**(**"Downloader Error"**,** "URL unreachable.\t\t\n " **+** "Exiting"**)**
70. **return** 1
71. ***# Extract the gz to memory***
72. extracted\_data **=** gzip**.**decompress**(**gz\_file**.**content**)**
73. ***# The Data in the gzip is stored as a .tsv file***
74. ***# Create a title for file to be extracted from the user title***
75. filename **=** "Data\_"**+**user\_title**+**".tsv"
76. ***# Check for existing files. If they exist, ask to overwrite.***
77. **if** path**.**isfile**(**filename**):**
78. overwrite **=** mb**.**askquestion**(**"File already exists"**,** "Overwrite -> " **+** filename **+** " <- ?? "**)**
79. **if** overwrite **==** "no"**:**
80. mb**.**showinfo**(**"No Changes made "**,** "File-> " **+** filename **+** "<- not saved \t\t\t"**)**
81. **return** 1
82. ***# If the files don't exist or we want to overwrite***
83. **try:**
84. ***# Write the files to disk***
85. f **=** open**(**filename**,** "wb"**)**
86. f**.**write**(**extracted\_data**)**
87. f**.**close**()**
88. **except** IOError **as** ex\_IO**:**
89. mb**.**showinfo**(**" Problem writing file:"**,** filename **+** "\n Error: " **+** str**(**ex\_IO**))**
90. **return** **None**
91. ***# Return***
92. ***# 1) the filename of the .tsv file on the disk***
93. ***# 2) The title set by the user***
94. ***# 3) The original name from eurostat***
95. **return** filename**,** user\_title**,** original\_name
96. ***# ------------------------------ DATA PROCESSOR -----------------------------------------------------------***
97. ***# This function "Cleans" the data to keep just the years / countries / type of visitor we want.***
98. ***# In this instance we will keep the data for:***
99. ***# Countries: Greece , Spain***
100. ***# Years : 2016,2017,2018,2019***
101. ***# Types of visitors: Foreigners and Total.***
102. ***# It also stores the cleaned data in a csv file.***
103. ***# This function takes as input a list that contains:***
104. ***# 1) The filename of the .tsv file on disk that has all the data.***
105. ***# 2) The user created title of the data***
106. ***# 3) The original filename***
107. **def** data\_processor**(**list\_in**):**
108. filename **=** list\_in**[**0**]**
109. user\_title **=** list\_in**[**1**]**
110. original\_name **=** list\_in**[**2**]**
111. **print(**"Processing "**,** filename**)**
112. ***# ---------- Selection of years, countries, visitor types --------------------------------***
113. ***# Select years to keep data for.***
114. start\_year **=** 2016
115. end\_year **=** 2019
116. ***# Select countries to keep data for***
117. selected\_countries **=** "EL|ES"
118. ***# Slect visitor type to keep data for. Options are (FOR|LOC|TOTAL)***
119. visitor\_type\_RE **=** "FOR|TOTAL"
120. ***# ---------- Regular expressions creation --------------------------------***
121. ***# Create the regular expression that holds the years to be kept in the dataframe***
122. selected\_years **=** str**(**start\_year**)**
123. **for** i **in** range**(**start\_year**+**1**,** end\_year **+** 1**):**
124. selected\_years **=** selected\_years **+** "|" **+** str**(**i**)**
125. ***# For example "2016|2017|2018|2019"***
126. ***# Add the 'ends with character' regex***
127. selected\_countries\_RE **=** selected\_countries **+** "\Z"
128. ***# load the as a pandas dataframe***
129. df **=** pd**.**read\_table**(**filename**)**
130. ***# Change the title of the first column, because it has weird characters that cause problems.***
131. df **=** df**.**rename**(**columns**={**df**.**columns**[**0**]:** 'COUNTRY'**})**
132. ***# Keep the COUNTRY column, in addition to the selected years column***
133. selected\_years\_RE **=** re**.**compile**(**selected\_years **+** "|COUNTRY"**)**
134. ***# ---------- Filtering of the Data Frame --------------------------------***
135. ***# Filter out the columns that do not match the selected years***
136. df **=** df**.**filter**(**regex**=**selected\_years\_RE**,** axis**=**1**)**
137. ***# Clear rows that do not match the countries Regex***
138. df **=** df**[(**df**[**'COUNTRY'**].**str**.**contains**(**selected\_countries\_RE**,** regex**=True))]**
139. ***# Clear rows that do not start with the visitor type Regex***
140. df **=** df**[(**df**[**'COUNTRY'**].**str**.**match**(**visitor\_type\_RE**))]**
141. ***# Create the csv files that we will store.***
142. size **=** len**(**filename**)**
143. ***# Strip the last 4 character ( remove ".tsv" of the original filename )***
144. filename\_out **=** filename**[:**size **-** 4**]** **+** ".csv"
145. ***# Check for file existence and ask to write / overwrite***
146. **if** path**.**isfile**(**filename\_out**):**
147. overwrite **=** mb**.**askquestion**(**"File already exists"**,** "Overwrite --> " **+** filename\_out **+** " <-- ?? "**)**
148. **if** overwrite **==** "no"**:**
149. mb**.**showinfo**(**"No Changes made"**,** " Exiting\t\t"**)**
150. **return** 1
151. ***# Check for write access***
152. **try:**
153. ***# Try to open file to check for write permission***
154. f **=** open**(**filename\_out**,** "wb"**)**
155. f**.**close**()**
156. **except** IOError **as** ex\_IO**:**
157. mb**.**showinfo**(**" Error writing file:"**,** "File: \n" **+** filename\_out **+** "\n Error: " **+** str**(**ex\_IO**))**
158. **return** 1
159. ***# Create a csv with the cleaned data frame***
160. df**.**to\_csv**(**filename\_out**,** encoding**=**'utf-8'**,** index**=False)**
161. ***# Ask user whether to keep the downloaded ".tsv" file***
162. keep\_original\_files **=** mb**.**askquestion**(**"Keep Downloaded File ?"**,**
163. "KEEP : " **+** filename **+** "\t\t"**)**
164. **if** keep\_original\_files **==** "no"**:**
165. **try:**
166. remove**(**filename**)**
167. **except** IOError **as** ex\_IO**:**
168. mb**.**showinfo**(**" Error Deleting file:"**,** "File: \n" **+** filename **+** "\n Error: " **+** str**(**ex\_IO**))**
169. ***# Function Returns***
170. ***# 1) The " cleaned " pandas dataframe***
171. ***# 2) The user appointed title***
172. ***# 3) The original file name from eurostat***
173. **return** df**,** user\_title**,** original\_name
174. ***# ----------------------------------- DATABASE STORAGE-----------------------------------------------------***
175. ***# This function creates mySQL database from the input data***
176. ***# This function requires a MySQL database to be up and running.***
177. ***# The inputis a list of lists, each consistsing of 3 items***
178. ***# 1 ) A pandas data frame list\_in ,***
179. ***# 2) The user appointed name***
180. ***# 3) The original name of the tsv file***
181. **def** db\_store**(**list\_in**):**
182. ***# Create the connection to the local mySQL database and test it***
183. **try:**
184. db\_connection **=** mysql**.**connector**.**connect**(**
185. host**=**"localhost"**,**
186. user**=**"root"**,**
187. password**=**"toor"
188. **)**
189. **except** mysql**.**connector**.**Error **as** err**:**
190. **print(**"DATABASE ERROR "**,** "ERROR Something went wrong:\n {}"**.**format**(**err**))**
191. mb**.**showerror**(**"DATABASE ERROR "**,** "ERROR Something went wrong:\n {}"**.**format**(**err**))**
192. ***# print("ERROR Something went wrong: {}".format(err))***
193. **return** 1 ***# Return 1 if unable to connect to a database***
194. ***# Create a cursor***
195. mycursor **=** db\_connection**.**cursor**(**buffered**=True)**
196. ***# Create the database***
197. mycursor**.**execute**(**"DROP DATABASE IF EXISTS arxes\_db;"**)**
198. mycursor**.**execute**(**"CREATE DATABASE arxes\_db;"**)**
199. mycursor**.**execute**(**"use arxes\_db;"**)**
200. **for** k **in** range**(**len**(**list\_in**)):**
201. ***# Get the dataframe from the list***
202. df **=** list\_in**[**k**][**0**]**
203. ***# Get the user title from the list ( will be the name of the table )***
204. user\_title **=** list\_in**[**k**][**1**]**
205. ***# First step is to create a table which will be named with the user provided name .***
206. table\_name **=** user\_title
207. sql **=** **(**"CREATE TABLE " **+**
208. table\_name **+**
209. "(id INT AUTO\_INCREMENT PRIMARY KEY," **+**
210. " country\_visitor\_type VARCHAR(255)," **+**
211. "`2016` INT," **+**
212. "`2017` INT," **+**
213. "`2018` INT," **+**
214. "`2019` INT)"**)**
215. ***# print(sql)***
216. mycursor**.**execute**(**sql**)**
217. ***# For each row in our table***
218. **for** j **in** range**(**len**(**list\_in**[**k**])** **+** 1**):**
219. ***# Base sql insertion query string , concatenate stuff to it , in order to make the insertions queries***
220. sql\_insert **=** "INSERT INTO " **+** table\_name **+** \
221. " (`country\_visitor\_type`,`2016`,`2017`,`2018`,`2019`) VALUES ("
222. ***# for each column item in a row***
223. **for** i **in** range**(**len**(**df**.**columns**)):**
224. temp **=** "'"
225. temp **+=** df**.**iloc**[**j**,** i**]**
226. ***# Delete the whitespace after the df item***
227. temp **=** temp**.**rstrip**(**temp**[-**1**])**
228. sql\_insert **+=** temp **+** "',"
229. ***# Delete the last comma***
230. sql\_insert **=** sql\_insert**.**rstrip**(**sql\_insert**[-**1**])**
231. sql\_insert **+=** ");"
232. ***# # Show the SQL query***
233. ***# print(sql\_insert)***
234. ***# Execute it***
235. mycursor**.**execute**(**sql\_insert**)**
236. ***# Save the changes to the database***
237. db\_connection**.**commit**()**
238. ***# -------------------------------- CHART CREATOR --------------------------------------------------------***
239. ***# This function takes 4 inputs:***
240. ***# 1) A list that contains:***
241. ***# 1.1) a pandas dataframe***
242. ***# 1.2) The user title***
243. ***# 1.3) The original file name to be used as the subplot title***
244. ***# 2) The number of the subplot***
245. ***# 3) The list of country codes***
246. ***# 4) The list of country names***
247. **def** make\_charts\_2**(**list\_in**,** subplot\_number**,** country\_code**,** country\_name**):**
248. ***# There will be 4 plots, in a 2 x 2 grid***
249. plot **=** plt**.**subplot**(**2**,** 2**,** subplot\_number**)**
250. ***# Set the plot title as the original file name***
251. plot\_title **=** list\_in**[**2**]**
252. ***# Set the title of the subplot***
253. plot**.**set\_title**(**plot\_title **+** "\n" **+** country\_name**,** fontsize**=**14**)**
254. ***# set the subplot background color for better readability***
255. plot**.**set\_facecolor**(**"gainsboro"**)**
256. ***# Set the label for the y axis***
257. plot**.**set\_ylabel**(**'People'**,** fontsize**=**14**)**
258. ***# Set the label for the x axis***
259. plot**.**set\_xlabel**(**'YEAR'**,** fontsize**=**14**)**
260. ***# Get the dataframe from the list***
261. df **=** list\_in**[**0**]**
262. ***# Get the names of all the columns into a list***
263. ***# ( will be used to title each bar for the bar plot )***
264. years **=** df**.**columns**.**tolist**()**
265. ***# drop the first column from the list***
266. years**.**pop**(**0**)**
267. ***# Return evenly spaced values based on the length of the list supplied***
268. ***# example: For 4 years, x will be [ 0 1 2 3]***
269. x **=** np**.**arange**(**len**(**years**))** ***# the label locations***
270. ***# Place ticks(labels) on the x axis, on the evenly spaced values***
271. plot**.**set\_xticks**(**x**)**
272. ***# Source for labels text to attach to each tick is the years***
273. plot**.**set\_xticklabels**(**years**)**
274. ***# ------------- Code to keep the correct country rows -------------***
275. ***# Keep only the rows that have the country column ends with the country code we want***
276. ***# example : keep only the rows in which the country column ends with 'EL'***
277. df1 **=** df**[(**df**[**'COUNTRY'**].**str**.**endswith**(**country\_code**))]**
278. ***# ------------- Code to keep the number of Foreign visitors -------------***
279. ***# Keep only the row that have the country column BEGIN with FOR***
280. ***# To keep the foreigners = non residents***
281. data\_foreign **=** df1**[(**df1**[**'COUNTRY'**].**str**.**startswith**(**'FOR'**))]**
282. ***# Convert the dataframe into a list of lists***
283. data\_foreign **=** data\_foreign**.**values**.**tolist**()**
284. ***# Keep the only item of the list***
285. data\_foreign **=** data\_foreign**[**0**]**
286. ***# Delete the first item of the list, which is the country code and data type ( FOR|TOTAL)***
287. data\_foreign**.**pop**(**0**)**
288. ***# Make the list of strings into a list of integers***
289. data\_foreign **=** **[int(**i**)** **for** i **in** data\_foreign**]**
290. ***# ------------- Code to keep the total number of visitors -------------***
291. ***# Keep in a dataframe only the row that has the country column that begins with TOTAL***
292. ***# To keep the total number of visitors***
293. data\_total **=** df1**[(**df1**[**'COUNTRY'**].**str**.**startswith**(**'TOTAL'**))]**
294. ***# Convert the dataframe into a list of lists***
295. data\_total **=** data\_total**.**values**.**tolist**()**
296. ***# Keep the only item of the list***
297. data\_total **=** data\_total**[**0**]**
298. ***# Delete the first item of the list, which is the country code and data type ( FOR|TOTAL)***
299. data\_total**.**pop**(**0**)**
300. ***# Make the list of strings into a list of integers***
301. data\_total **=** **[int(**i**)** **for** i **in** data\_total**]**
302. width **=** 0.3 ***# the width of the bars of the plot***
303. ***# The two bars are created***
304. rect1 **=** plot**.**bar**(**x **-** width **/** 2**,** data\_foreign**,** width**,** label**=**'Non Residents'**)**
305. rect2 **=** plot**.**bar**(**x **+** width **/** 2**,** data\_total**,** width**,** label**=**'Total'**)**
306. ***# The labels for the two bars are created***
307. plot**.**bar\_label**(**rect1**,** padding**=**5**,** fmt**=**"%d"**,** color**=**'#1f77b4'**,** backgroundcolor**=**'0.8'**,** rotation**=**10**,** size**=**9**)**
308. plot**.**bar\_label**(**rect2**,** padding**=**5**,** fmt**=**'%d'**,** color**=**'#ff7f0e'**,** backgroundcolor**=**'0.8'**,** rotation**=**10**,** size**=**9**)**
309. ***# Create the formatting for the vertical axis***
310. ***# This code was taken from stackoverflow***
311. ***# https://stackoverflow.com/questions/40511476/how-to-properly-use-funcformatterfunc***
312. **def** millions**(**x**,** pos**):**
313. **return** '%1.1fM' **%** **(**x **\*** 1e-6**)**
314. ***# Create the formatting for the vertical axis***
315. formatter **=** FuncFormatter**(**millions**)**
316. ***# Set the formatting for the vertical axis***
317. plot**.**get\_yaxis**().**set\_major\_formatter**(**formatter**)**
318. ***# Show a legend***
319. plot**.**legend**()**
320. ***# This function takes as input***
321. ***# 1) A list of lists, each list consists of 3 items***
322. ***# 1.1) A pandas data frame***
323. ***# 1.2) The user appointed name***
324. ***# 1.3) The original name of the tsv file***
325. ***# 2) The number of the subplot to be created***
326. ***# 3) A list of country codes ['EL', 'ES']***
327. ***# 4) A list of country names ['Greece', 'Spain']***
328. **def** make\_charts**(**in\_list**,** country\_codes**,** country\_names**):**
329. nigths\_list **=** in\_list**[**0**]**
330. arrivals\_list **=** in\_list**[**1**]**
331. make\_charts\_2**(**nigths\_list**,** 1**,** country\_codes**[**0**],** country\_names**[**0**])**
332. make\_charts\_2**(**nigths\_list**,** 3**,** country\_codes**[**1**],** country\_names**[**1**])**
333. make\_charts\_2**(**arrivals\_list**,** 2**,** country\_codes**[**0**],** country\_names**[**0**])**
334. make\_charts\_2**(**arrivals\_list**,** 4**,** country\_codes**[**1**],** country\_names**[**1**])**
335. plt**.**show**()**
336. ***# ------------------------------ MAIN CODE -----------------------------------------------------------***
337. ***# The list "URL\_list" contains lists that have 3 items each :***
338. ***# 1) The url for each file we want to download***
339. ***# 2) A name created by the user to easily distinguish the file***
340. ***# 3) The original file name from eurostat***
341. ***# This list will be passed to the downloader function, to download the data.***
342. URL\_list **=** **[**
343. ***# list 1***
344. **[**
345. "https://ec.europa.eu/eurostat/estat-navtree-portlet-prod/BulkDownloadListing?file=data/tin00175.tsv.gz"
346. **,**
347. "Nights"
348. **,**
349. "Nights spent at tourist accommodation establishments by residents/non-residents"
350. **]**
351. **,**
352. ***# list 2***
353. **[**
354. "https://ec.europa.eu/eurostat/estat-navtree-portlet-prod/BulkDownloadListing?file=data/tin00174.tsv.gz"
355. **,**
356. "Arrivals"
357. **,**
358. "Arrivals of residents/non-residents at tourist accommodation establishments"
359. **]**
360. **]**
361. ***# Initialize tkInter***
362. root **=** tk**.**Tk**()**
363. ***# root.withdraw()***
364. ***# Tkinter window miscellaneous options***
365. ***# root.iconbitmap("../Images/favicon.ico")***
366. root**.**title**(**"Αρχές Γλωσσών Python 2021"**)**
367. root**.**geometry**(**"600x300+650+400"**)** ***# Width x Height + Padding left + Padding top***
368. ***# Ask for a location to download the data into***
369. directory\_change**()**
370. ***# The list "downloaded\_files" contains information about the files that the downloader has downloaded***
371. ***# It Holds lists, that have 3 items each:***
372. ***# 1) The filename of the .tsv file that was downloaded***
373. ***# 3) The user created title***
374. ***# 2) The original filename from the website***
375. downloaded\_files **=** **[]**
376. **print(**"------- DOWNLOADING -------"**)**
377. **for** i **in** range**(**len**(**URL\_list**)):**
378. ***# Feed each item of the URL list to the downloader***
379. temp **=** downloader**(**URL\_list**[**i**])**
380. **if** temp **is** **not** **None:**
381. downloaded\_files**.**append**(**temp**)**
382. ***# IF no files have been downloaded, abort***
383. **if** **not** downloaded\_files**:**
384. **print(**"No Data Downloaded, Exiting...."**)**
385. mb**.**showerror**(**"No Data Downloaded"**,** "No Data. \n " **+** "Exiting..."**)**
386. exit**(**0**)**
387. ***# Ask user whether to continue with processing the data***
388. continue\_after\_download **=** mb**.**askquestion**(**"Continue to data processing ?"**,**
389. "You can stop here and just keep the downloaded files." "\t\t"**)**
390. **if** continue\_after\_download **==** "no"**:**
391. exit**(**0**)**
392. ***# The list "cleaned\_files" contains the cleaned pandas dataframes + additional info***
393. ***# Holds lists that have 3 items :***
394. ***# 1) pandas dataframe that has been " cleaned "***
395. ***# 2) The user created title***
396. ***# 3) The original filename from the website***
397. cleaned\_files **=** **[]**
398. **print(**"------- PROCESSING DATA -------"**)**
399. **for** i **in** range**(**len**(**downloaded\_files**)):**
400. cleaned\_files**.**append**(**data\_processor**(**downloaded\_files**[**i**]))**
401. **if** **not** cleaned\_files**:**
402. **print(**"No Data, Exiting...."**)**
403. mb**.**showerror**(**"No Data"**,** "No Data. \n " **+** "Exiting..."**)**
404. exit**(**0**)**
405. **print(**"------- STORING TO DATABASE -------"**)**
406. db\_store**(**cleaned\_files**)**
407. ***# The list of country codes the data will be plotted for***
408. ***# global country\_code***
409. country\_codes **=** **[**'EL'**,** 'ES'**]**
410. ***# The equivalent names for the above country codes***
411. ***# global country\_name***
412. country\_names **=** **[**'Greece'**,** 'Spain'**]**
413. **print(**"------- MAKING CHARTS -------"**)**
414. make\_charts**(**cleaned\_files**,** country\_codes**,** country\_names**)**
415. mb**.**showinfo**(**"Done!\t\t"**)**

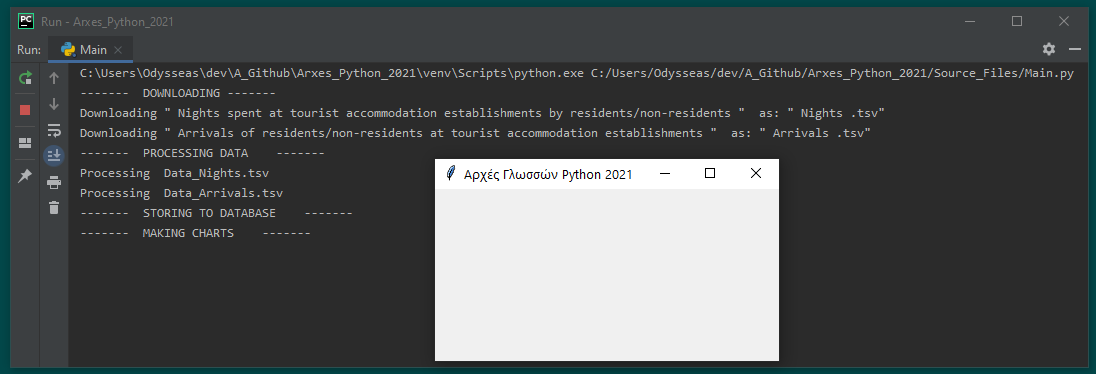
**Screenshots απο την εκτέλεση της εφαρμογής:**

Αρχικά, η ερώτηση για την επιλογή directory

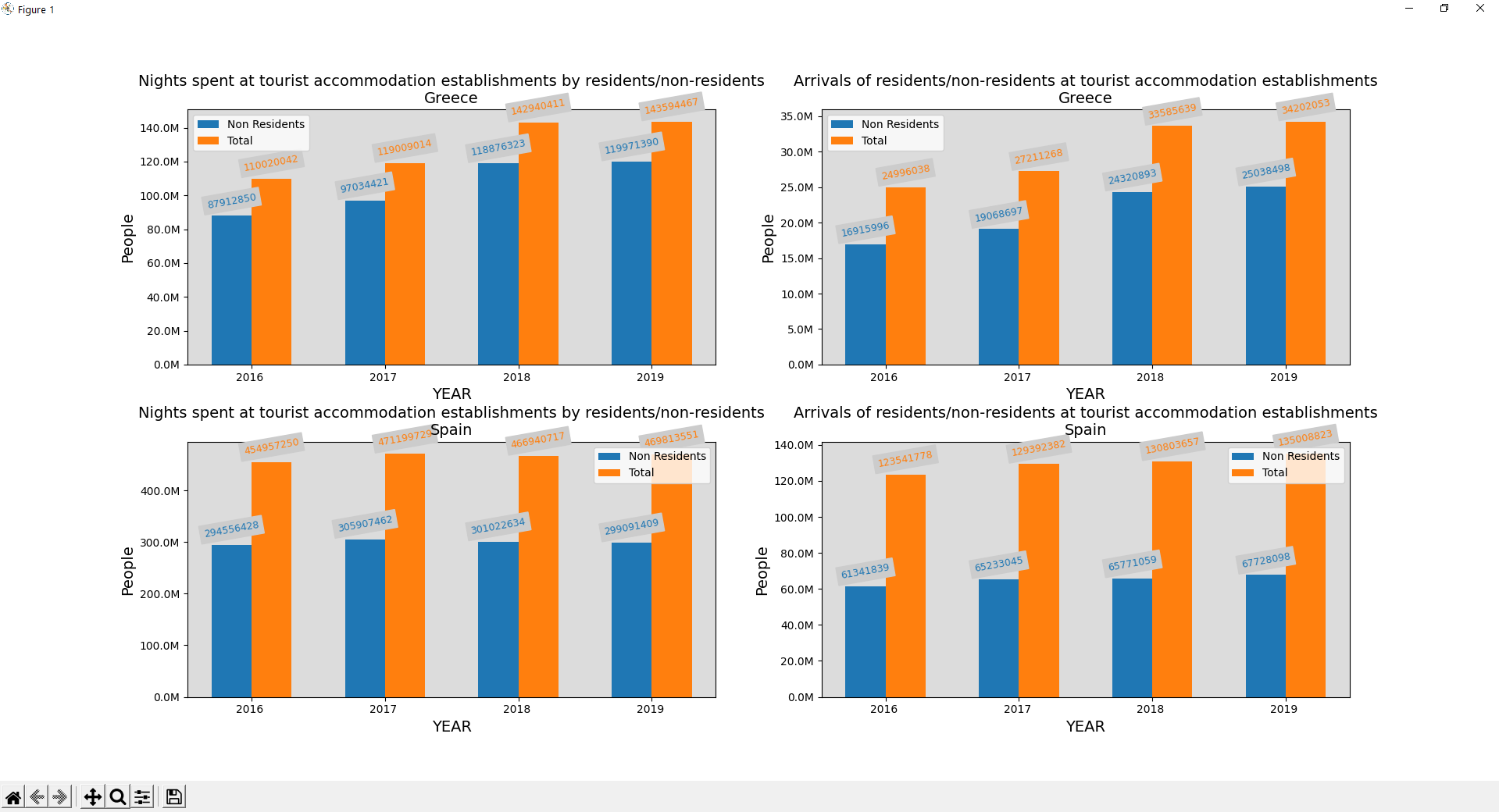


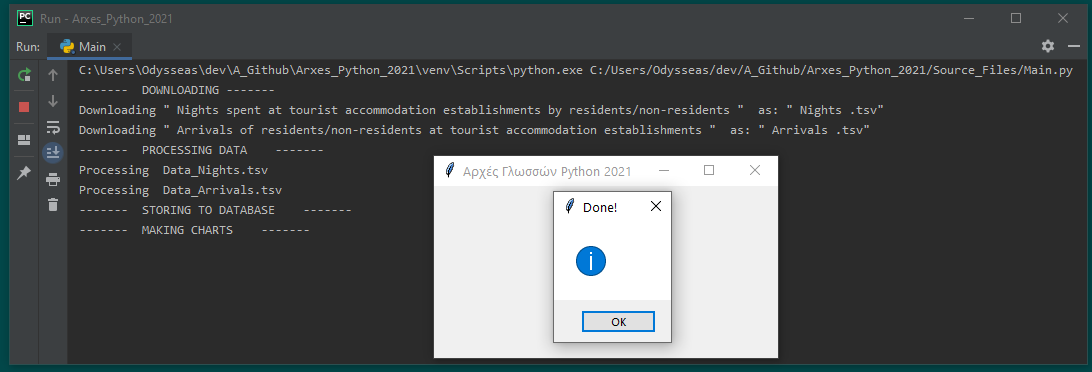
Ερώτηση αν ο χρήστης θέλει να συνεχίσει ή αν θέλει μόνο να κρατήσει τα κατεβασμένα αρχεία.



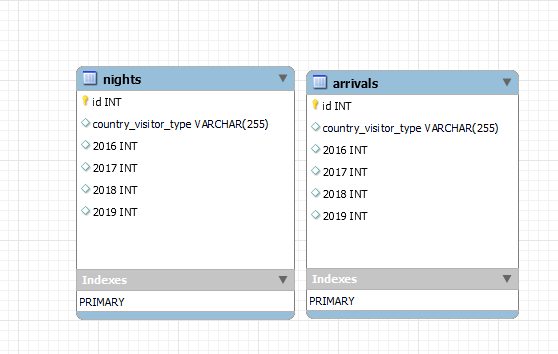
Screenshot απο το output της γραμμης εντολών κατα την επεξεργασία των δεδομένων και την αποθήκευση στη βάση δεδομένων.

Τα γραφήματα όπως παράγονται απο το matplotlib. ( Υπάρχουν σε υψηλότερη ανάλυση, σε μορφή png , μαζί με την παρούσα αναφορά ).



Τέλος της εκτέλεσης

Η βάση δεδομένων αποτελείται απο 2 tables με την εξής μορφή:



Και με το εξής περιεχόμενο:

