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

Περιεχόμενα

Σύντομη περιγραφή του κώδικα:

Η εκτέλεση του προγράμματος χωρίζεται σε 5 Φάσεις :

1) Συνάρτηση **directory_change**
   Επιλογή του Directory στο οποίο θα κατέβουν τα αρχεία απο το site της Eurostat, και στο οποίο θα αποθηκευτούν τα τελικά .csv αρχεία.

2) Συνάρτηση **downloader**
   Κατέβασμα των συμπιεσμένων αρχείων, αποσυμπίεση αυτών και εγγραφή τους στον δίσκο, ως αρχεία .tsv
   Στο τέλος της συνάρτησης αυτής, ο χρήστης επιλέγει αν θέλει να συνεχίσει με την επξεργασία των αρχείων ή αν θέλει να σταματήσει έχοντας μόνο κατεβάσει τα αρχεία .tsv .

3) Συνάρτηση **data_processor**
   Επεξεργασία των κατεβασμένων αρχείων ώστε να κρατηθούν μόνο τα δεδομένα που θέλουμε.
   Αποθήκευση των δεδομένων αυτών στον δίσκο ως αρχεία .csv
   Στο τέλος της συνάρτησης αυτής ο χρήστης επιλέγει αν θέλει να κρατήσει τα αρχικά, μη επεξεργασμένα αρχεία .tsv.
   Τα παραγόμενα csv αρχεία αποθηκέυονται ανεξαρτήτως της επιλογής του χρήση.

4) Συνάρτηση **db_store**
   Δημιουργία της βάσης δεδομένων (MySql) και INSERTS των tables.

5) Συνάρτηση **make_charts**
   (Μαζί με την υποσυνάρτηση make_charts_2)
   Δημιουργία των γραφημάτων.

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.
Το όνομα της βάσης δεδομένων που δημιουργείται είναι arxes_db.
Τα γραφήματα εμφανίζονται σαν ένα μεγάλο plot με 4 subplots.

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: 1041152
9 # 2021
10
11
12 import matplotlib.pyplot as plt
13 from matplotlib.ticker import FuncFormatter
14 import tkinter as tk
15 from tkinter import messagebox as mb
16 from tkinter import filedialog
17 import pandas as pd
18 import numpy as np
19 import gzip
20 import requests
21 from os import path
22 from os import remove
23 import os
24 import re
25
26 #  Use this !!!
27 #  pip install mysql-connector-python
28 import mysql.connector
29
30 #    ___________________________________
31 #   |   Main code is at line 441    |
32 #   |___________________________________|
33
34 # ------------------------------   DIRECTORY SELECTOR --------------------------------
------------------------------
35 # This Function asks the user to specify a directory to store the files.
36
37
38 def directory_change():
39
40     # Get the current directory
41     current_directory = os.getcwd()
42
43     # Do you want to change directory ?
44     dir_change = mb.askquestion("Directory selection", "Current directory is :\n" +
45                                 str(current_directory) +
46                                 "\n Change Directory ? ")
47
48     if dir_change == 'yes':
```

```
49          requested_directory = tk.filedialog.askdirectory()
50
51          # If the user selects a directory
52          if requested_directory:
53              # Change to that directory
54              os.chdir(requested_directory)
55
56          # If the user presses the escape button or closes the directory selection
window
57          else:
58              mb.showerror("DIRECTORY ERROR", "No directory selected. \n " +
"Exiting...")
59              exit(0)
60
61  # This Python function
62  # Downloads a compressed file from eurostat,
63  # extracts the file from the gzip
64  # and saves it to disk.
65
66  # ------------------------------   DOWNLOADER ------------------------------------
-------------------
67  #      list_in contains 3 items which are :
68  #       1) url: The url that holds the file we want
69  #       2) user_title: A user decided string, that will help identify the downloaded
data
70  #       3) original_name: The original file name ( The title from eurostat)
71
72
73  def downloader(list_in):
74
75      url = list_in[0]
76      user_title = list_in[1]
77      original_name = list_in[2]
78
79      print("Downloading", '"', original_name, '"', ' as: "', user_title, '.tsv"')
80
81      try:
82          # Download the gz to memory
83          gz_file = requests.get(url, allow_redirects=True, timeout=9.00)    # Expect a
response within 9 seconds
84      except requests.exceptions:
85          print("ERROR! \n Unable to reach Website", url)
86          # print(a_ex)                           # Return the error type
87          # exit(0)
88          mb.showerror("Downloader Error", "URL unreachable.\t\t\n " + "Exiting")
89          return 1
90
91      # Extract the gz to memory
92      extracted_data = gzip.decompress(gz_file.content)
93
94      # The Data in the gzip is stored as a .tsv file
```

```
95      #  Create a title for file to be extracted from the user title
96      filename = "Data_"+user_title+".tsv"
97
98      # Check for existing files. If they exist, ask to overwrite.
99      if path.isfile(filename):
100         overwrite = mb.askquestion("File already exists", "Overwrite -> " + filename
+ " <- ?? ")
101         if overwrite == "no":
102             mb.showinfo("No Changes made ", "File-> " + filename + "<- not saved
\t\t\t")
103             return 1
104     # If the files don't exist or we want to overwrite
105     try:
106         # Write the files to disk
107         f = open(filename, "wb")
108         f.write(extracted_data)
109         f.close()
110     except IOError as ex_IO:
111         mb.showinfo(" Problem writing file:", filename + "\n Error: " + str(ex_IO))
112         return None
113
114     # Return
115     # 1) the filename of the .tsv file on the disk
116     # 2) The title set by the user
117     # 3) The original name from eurostat
118     return filename, user_title, original_name
119
120 # ------------------------------   DATA PROCESSOR  ----------------------------------
------------------------
121
122 # This function "Cleans" the data to keep just the years / countries / type of
visitor we want.
123 # In this instance we will keep the data for:
124 # Countries: Greece , Spain
125 # Years : 2016,2017,2018,2019
126 # Types of visitors: Foreigners and Total.
127 # It also stores the cleaned data in a csv file.
128
129 #  This function takes as input a list that contains:
130 # 1) The filename of the .tsv file on disk that has all the data.
131 # 2) The user created title of the data
132 # 3) The original filename
133
134
135 def data_processor(list_in):
136
137     filename = list_in[0]
138     user_title = list_in[1]
139     original_name = list_in[2]
140
141     print("Processing ", filename)
```

```
142
143     #  ----------  Selection of years, countries, visitor types ----------------------
----------
144
145     # Select years to keep data for.
146     start_year = 2016
147     end_year = 2019
148
149     # Select countries to keep data for
150     selected_countries = "EL|ES"
151
152     # Slect visitor type to keep data for. Options are (FOR|LOC|TOTAL)
153     visitor_type_RE = "FOR|TOTAL"
154
155     #  ---------- Regular expressions creation  --------------------------------
156
157     # Create the regular expression that holds the years to be kept in the dataframe
158     selected_years = str(start_year)
159     for i in range(start_year+1, end_year + 1):
160         selected_years = selected_years + "|" + str(i)
161     # For example "2016|2017|2018|2019"
162
163     # Add the 'ends with character'  regex
164     selected_countries_RE = selected_countries + "\Z"
165
166     # load the as a pandas dataframe
167     df = pd.read_table(filename)
168
169     # Change the title of the first column, because it has weird characters that
cause problems.
170     df = df.rename(columns={df.columns[0]: 'COUNTRY'})
171
172     # Keep the COUNTRY column, in addition to the selected years column
173     selected_years_RE = re.compile(selected_years + "|COUNTRY")
174
175     #  ---------- Filtering of the Data Frame --------------------------------
176
177     # Filter out the columns that do not match the selected years
178     df = df.filter(regex=selected_years_RE, axis=1)
179
180     # Clear rows that do not match the countries Regex
181     df = df[(df['COUNTRY'].str.contains(selected_countries_RE, regex=True))]
182
183     # Clear rows that do not start with the visitor type Regex
184     df = df[(df['COUNTRY'].str.match(visitor_type_RE))]
185
186     # Create the csv files that we will store.
187     size = len(filename)
188     # Strip the last 4 character ( remove ".tsv" of the original filename )
189     filename_out = filename[:size - 4] + ".csv"
190
```

```
191     # Check for file existence and ask to write / overwrite
192     if path.isfile(filename_out):
193         overwrite = mb.askquestion("File already exists", "Overwrite --> " +
filename_out + " <-- ?? ")
194         if overwrite == "no":
195             mb.showinfo("No Changes made", " Exiting\t\t")
196             return 1
197 
198     # Check for write access
199     try:
200         # Try to open file to check for write permission
201         f = open(filename_out, "wb")
202         f.close()
203     except IOError as ex_IO:
204         mb.showinfo(" Error writing file:", "File: \n" + filename_out + "\n Error: "
+ str(ex_IO))
205         return 1
206 
207     # Create a csv with the cleaned data frame
208     df.to_csv(filename_out, encoding='utf-8', index=False)
209 
210     # Ask user whether to keep the downloaded ".tsv" file
211     keep_original_files = mb.askquestion("Keep Downloaded File ?",
212                                          "KEEP : " + filename + "\t\t")
213     if keep_original_files == "no":
214         try:
215             remove(filename)
216         except IOError as ex_IO:
217             mb.showinfo(" Error Deleting file:", "File: \n" + filename + "\n Error: "
+ str(ex_IO))
218 
219     #  Function Returns
220     # 1) The " cleaned " pandas dataframe
221     # 2) The user appointed title
222     # 3) The original file name from eurostat
223     return df, user_title, original_name
224 
225 # ---------------------------------  DATABASE STORAGE-----------------------------
----------------------
226 
227 # This function creates mySQL database from the input data
228 # This function requires a MySQL database to be up and running.
229 
230 # The inputis a list of  lists, each consistsing of 3 items
231 # 1 ) A pandas data frame list_in ,
232 # 2)  The user appointed name
233 # 3) The original name of the tsv file
234 
235 
236 def db_store(list_in):
237 
```

```
    # Create the connection to the local mySQL database and test it
    try:
        db_connection = mysql.connector.connect(
            host="localhost",
            user="root",
            password="toor"
        )
    except mysql.connector.Error as err:
        print("DATABASE ERROR ", "ERROR Something went wrong:\n {}".format(err))
        mb.showerror("DATABASE ERROR ", "ERROR Something went wrong:\n
{}".format(err))
        # print("ERROR Something went wrong: {}".format(err))
        return 1   # Return 1 if unable to connect to a database

    # Create a cursor
    mycursor = db_connection.cursor(buffered=True)

    # Create the database
    mycursor.execute("DROP DATABASE IF EXISTS arxes_db;")
    mycursor.execute("CREATE DATABASE arxes_db;")
    mycursor.execute("use arxes_db;")

    for k in range(len(list_in)):

        #  Get the dataframe from the list
        df = list_in[k][0]
        #  Get the user title from the list ( will be the name of the table )
        user_title = list_in[k][1]

        # First step is to create a table which will be named with the user provided
name .
        table_name = user_title
        sql = ("CREATE TABLE " +
               table_name +
               "(id INT AUTO_INCREMENT PRIMARY KEY," +
               " country_visitor_type VARCHAR(255)," +
               "`2016` INT," +
               "`2017` INT," +
               "`2018` INT," +
               "`2019` INT)")
        # print(sql)
        mycursor.execute(sql)

        # For each row in our table
        for j in range(len(list_in[k]) + 1):

            # Base sql insertion query string , concatenate stuff to it , in order to
make the insertions queries
            sql_insert = "INSERT INTO " + table_name + \
                         " (`country_visitor_type`,`2016`,`2017`,`2018`,`2019`)
VALUES ("
```

```
285
286             # for each column item in a row
287             for i in range(len(df.columns)):
288                 temp = "'"
289                 temp += df.iloc[j, i]
290
291                 # Delete the whitespace after the df item
292                 temp = temp.rstrip(temp[-1])
293
294                 sql_insert += temp + "',"
295
296             #  Delete the last comma
297             sql_insert = sql_insert.rstrip(sql_insert[-1])
298             sql_insert += ");"
299
300             # # Show the SQL query
301             # print(sql_insert)
302
303             # Execute it
304             mycursor.execute(sql_insert)
305
306             # Save the changes to the database
307             db_connection.commit()
308
309 # ------------------------------- CHART CREATOR ------------------------------------
--------------------
310
311 # This function takes 4 inputs:
312 # 1) A list that contains:
313 #   1.1) a pandas dataframe
314 #   1.2) The user title
315 #   1.3) The original file name to be used as the subplot title
316 # 2) The number of the subplot
317 # 3) The list of country codes
318 # 4) The list of country names
319
320
321 def make_charts_2(list_in, subplot_number, country_code, country_name):
322
323     # There will be 4 plots, in a 2 x 2  grid
324     plot = plt.subplot(2, 2, subplot_number)
325
326     # Set the plot title as the original file name
327     plot_title = list_in[2]
328
329     # Set the title of the subplot
330     plot.set_title(plot_title + "\n" + country_name, fontsize=14)
331
332     # set the subplot background color for better readability
333     plot.set_facecolor("gainsboro")
334
```

```
335     # Set the label for the y axis
336     plot.set_ylabel('People', fontsize=14)
337
338     # Set the label for the x axis
339     plot.set_xlabel('YEAR', fontsize=14)
340
341     # Get the dataframe from the list
342     df = list_in[0]
343
344     # Get the names of all the columns into a list
345     # ( will be used to title each bar for the bar plot )
346     years = df.columns.tolist()
347     # drop the first column  from the list
348     years.pop(0)
349
350     # Return evenly spaced values based on the length of the list supplied
351     # example: For 4 years, x will be [ 0 1 2 3]
352     x = np.arange(len(years))  # the label locations
353
354     # Place  ticks(labels) on the x axis, on the evenly spaced values
355     plot.set_xticks(x)
356
357     # Source for labels text  to attach to each tick is the years
358     plot.set_xticklabels(years)
359
360     # -------------   Code to keep the correct country rows  -------------
361
362     # Keep only the rows that have the country column ends with  the country code we
want
363     # example : keep only the rows in which the country column ends with 'EL'
364     df1 = df[(df['COUNTRY'].str.endswith(country_code))]
365
366     # -------------   Code to keep the  number of Foreign visitors -------------
367
368     # Keep only the row that have the country column BEGIN with FOR
369     # To keep the foreigners = non residents
370     data_foreign = df1[(df1['COUNTRY'].str.startswith('FOR'))]
371     # Convert the  dataframe into a list of lists
372     data_foreign = data_foreign.values.tolist()
373     #  Keep the only item of the list
374     data_foreign = data_foreign[0]
375     # Delete the first item of the list, which is the country code and data type (
FOR|TOTAL)
376     data_foreign.pop(0)
377     # Make the list of strings into a list of integers
378     data_foreign = [int(i) for i in data_foreign]
379
380     # -------------   Code to keep the total number of visitors -------------
381
382     # Keep in a dataframe only the row that has the country column that begins with
TOTAL
```

```
383     # To keep the total number of visitors
384     data_total = df1[(df1['COUNTRY'].str.startswith('TOTAL'))]
385     # Convert the  dataframe into a list of lists
386     data_total = data_total.values.tolist()
387     #  Keep the only item of the list
388     data_total = data_total[0]
389     # Delete the first item of the list, which is the country code and data type (
FOR|TOTAL)
390     data_total.pop(0)
391     # Make the list of strings into a list of integers
392     data_total = [int(i) for i in data_total]
393
394     width = 0.3  # the width of the bars of the plot
395
396     # The two bars are created
397     rect1 = plot.bar(x - width / 2, data_foreign, width, label='Non Residents')
398     rect2 = plot.bar(x + width / 2, data_total, width, label='Total')
399
400     #  The labels for the two bars are created
401     plot.bar_label(rect1, padding=5, fmt="%d", color='#1f77b4',
backgroundcolor='0.8', rotation=10, size=9)
402     plot.bar_label(rect2, padding=5, fmt='%d', color='#ff7f0e',
backgroundcolor='0.8', rotation=10, size=9)
403
404     # Create the formatting for the vertical axis
405     # This code was taken from stackoverflow
406     # https://stackoverflow.com/questions/40511476/how-to-properly-use-
funcformatterfunc
407     def millions(x, pos):
408         return '%1.1fM' % (x * 1e-6)
409     # Create the formatting for the vertical axis
410     formatter = FuncFormatter(millions)
411
412     # Set the formatting for the vertical axis
413     plot.get_yaxis().set_major_formatter(formatter)
414
415     # Show a legend
416     plot.legend()
417
418
419 # This function takes as input
420 # 1) A list of lists, each list consists of 3 items
421 #   1.1) A pandas data frame
422 #   1.2)  The user appointed name
423 #   1.3) The original name of the tsv file
424 # 2) The number of the subplot to be created
425 # 3) A list of country codes ['EL', 'ES']
426 # 4) A list of country names ['Greece', 'Spain']
427
428
429 def make_charts(in_list, country_codes, country_names):
```

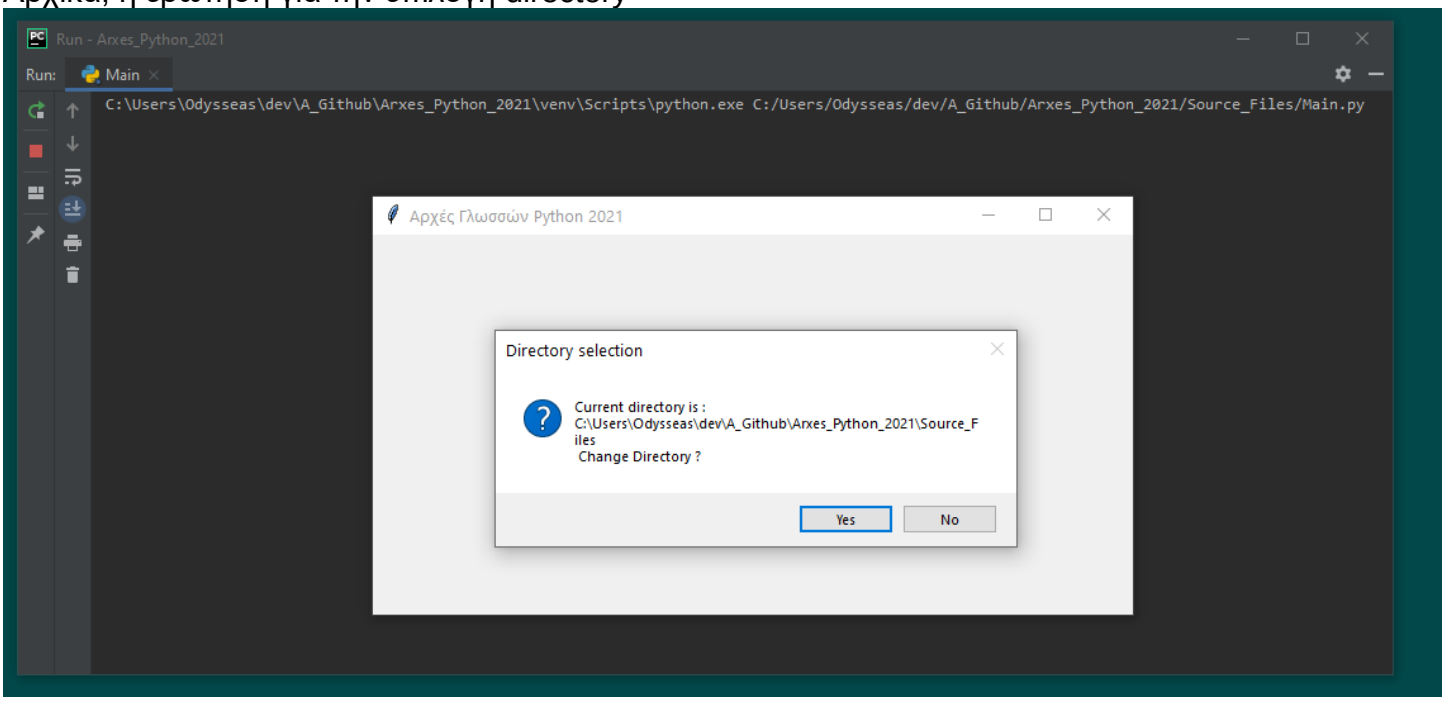
```
430
431     nigths_list = in_list[0]
432     arrivals_list = in_list[1]
433
434     make_charts_2(nigths_list,     1, country_codes[0], country_names[0])
435     make_charts_2(nigths_list,     3, country_codes[1], country_names[1])
436     make_charts_2(arrivals_list,   2, country_codes[0], country_names[0])
437     make_charts_2(arrivals_list,   4, country_codes[1], country_names[1])
438
439     plt.show()
440
441 # ------------------------------   MAIN CODE  ------------------------------------
--------------------
442
443 # The list "URL_list"  contains  lists that have 3  items each :
444 # 1) The url for each file we want to download
445 # 2) A name created by the user to easily distinguish the file
446 # 3) The original file name from eurostat
447 #  This list will be passed to the downloader function, to download the data.
448
449
450 URL_list = [
451             # list 1
452             [
453                 "https://ec.europa.eu/eurostat/estat-navtree-portlet-
prod/BulkDownloadListing?file=data/tin00175.tsv.gz"
454                 ,
455                 "Nights"
456                 ,
457                 "Nights spent at tourist accommodation establishments by
residents/non-residents"
458             ]
459             ,
460             # list 2
461             [
462                 "https://ec.europa.eu/eurostat/estat-navtree-portlet-
prod/BulkDownloadListing?file=data/tin00174.tsv.gz"
463                 ,
464                 "Arrivals"
465                 ,
466                 "Arrivals of residents/non-residents at tourist accommodation
establishments"
467             ]
468            ]
469
470 # Initialize tkInter
471 root = tk.Tk()
472 # root.withdraw()
473
474 # Tkinter window miscellaneous options
475 # root.iconbitmap("../Images/favicon.ico")
```

```
476 root.title("Αρχές Γλωσσών Python 2021")
477 root.geometry("600x300+650+400")  # Width x Height + Padding left + Padding top
478
479 # Ask for a location to download the data into
480 directory_change()
481
482 # The list "downloaded_files" contains information about the files that the
downloader has downloaded
483 # It Holds lists, that have 3 items each:
484 # 1) The filename of the .tsv file that was downloaded
485 # 3) The user created title
486 # 2) The original filename from the website
487 downloaded_files = []
488
489 print("-------  DOWNLOADING -------")
490 for i in range(len(URL_list)):
491     # Feed each item of the URL list to the downloader
492     temp = downloader(URL_list[i])
493     if temp is not None:
494         downloaded_files.append(temp)
495
496 # IF no files have been downloaded, abort
497 if not downloaded_files:
498     print("No Data Downloaded, Exiting....")
499     mb.showerror("No Data Downloaded", "No Data. \n " + "Exiting...")
500     exit(0)
501
502 # Ask user whether to continue with processing the data
503 continue_after_download = mb.askquestion("Continue to data processing ?",
504                                           "You can stop here and just keep the
downloaded files."  "\t\t")
505 if continue_after_download == "no":
506     exit(0)
507
508 # The list "cleaned_files" contains the cleaned pandas dataframes + additional info
509 # Holds lists that have 3 items :
510 # 1) pandas dataframe that has been " cleaned "
511 # 2) The user created title
512 # 3) The original filename from the website
513 cleaned_files = []
514
515 print("-------  PROCESSING DATA    -------")
516 for i in range(len(downloaded_files)):
517     cleaned_files.append(data_processor(downloaded_files[i]))
518
519 if not cleaned_files:
520     print("No Data, Exiting....")
521     mb.showerror("No Data", "No Data. \n " + "Exiting...")
522     exit(0)
523
524 print("-------  STORING TO DATABASE    -------")
```

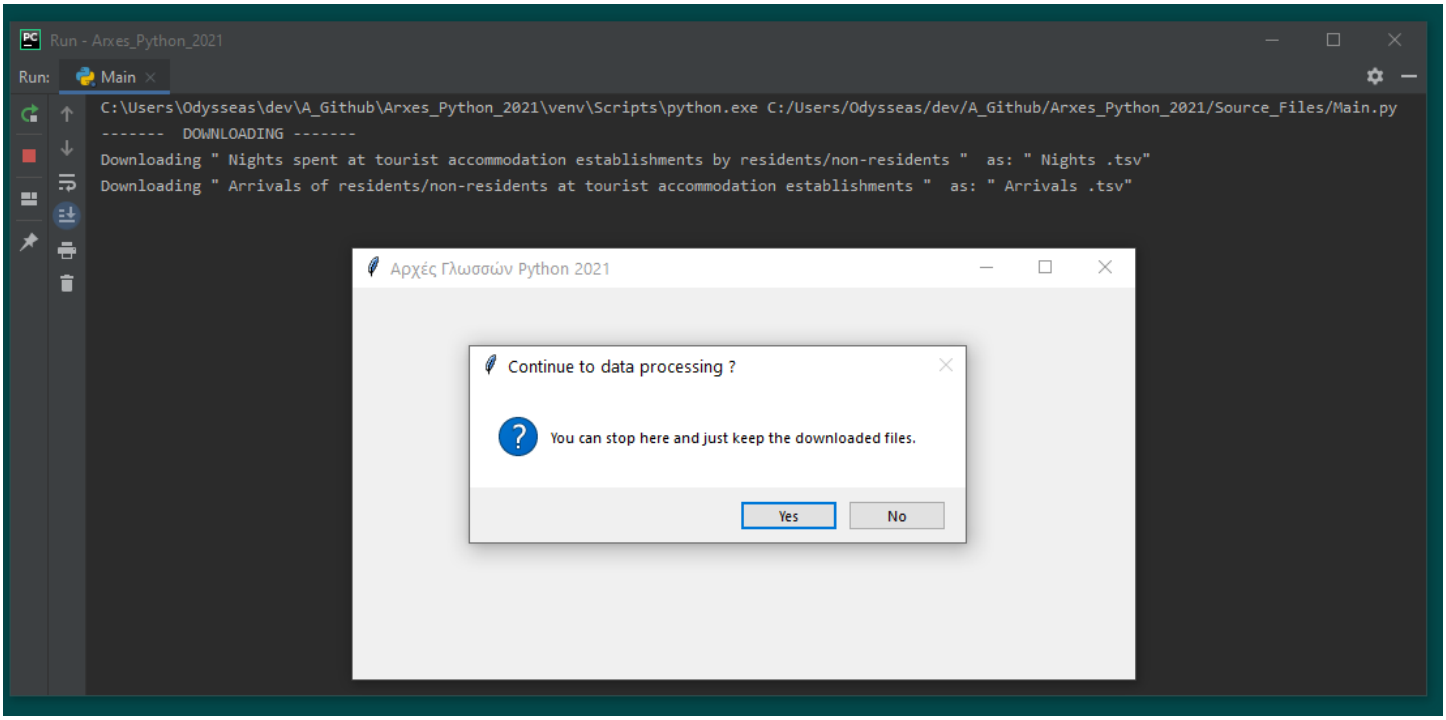
```
525 db_store(cleaned_files)
526
527
528 #  The list of country codes the data will be plotted for
529 # global country_code
530 country_codes = ['EL', 'ES']
531
532 # The equivalent names for the above country codes
533 # global country_name
534 country_names = ['Greece', 'Spain']
535
536 print("-------  MAKING CHARTS    -------")
537 make_charts(cleaned_files, country_codes, country_names)
538
539 mb.showinfo("Done!\t\t")
540
```

Screenshots απο την εκτέλεση της εφαρμογής:

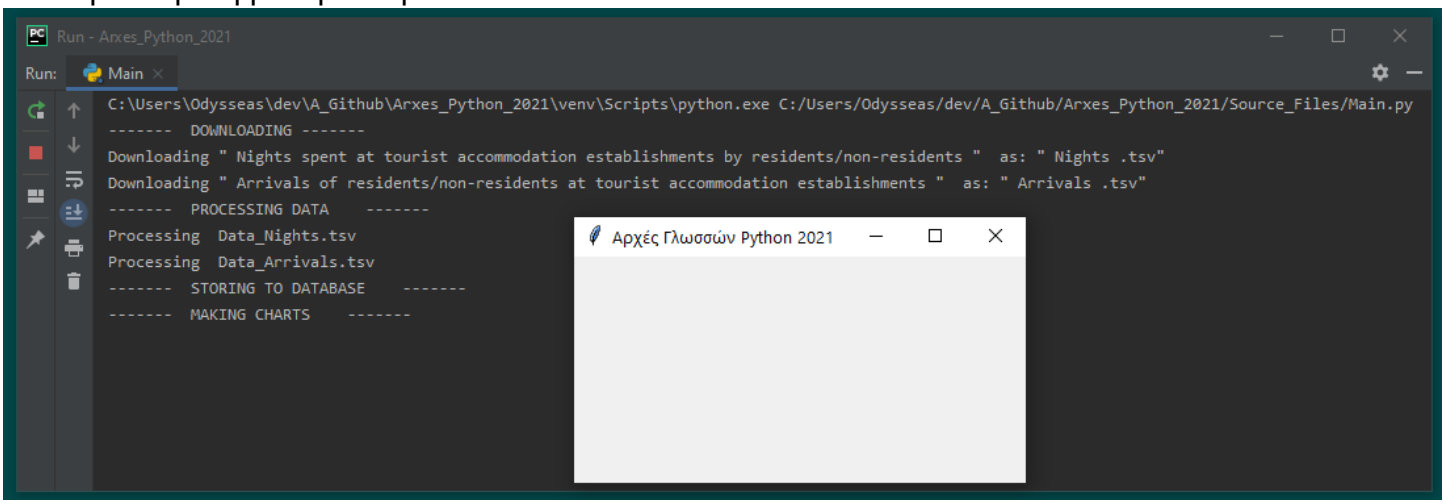
Αρχικά, η ερώτηση για την επιλογή directory



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

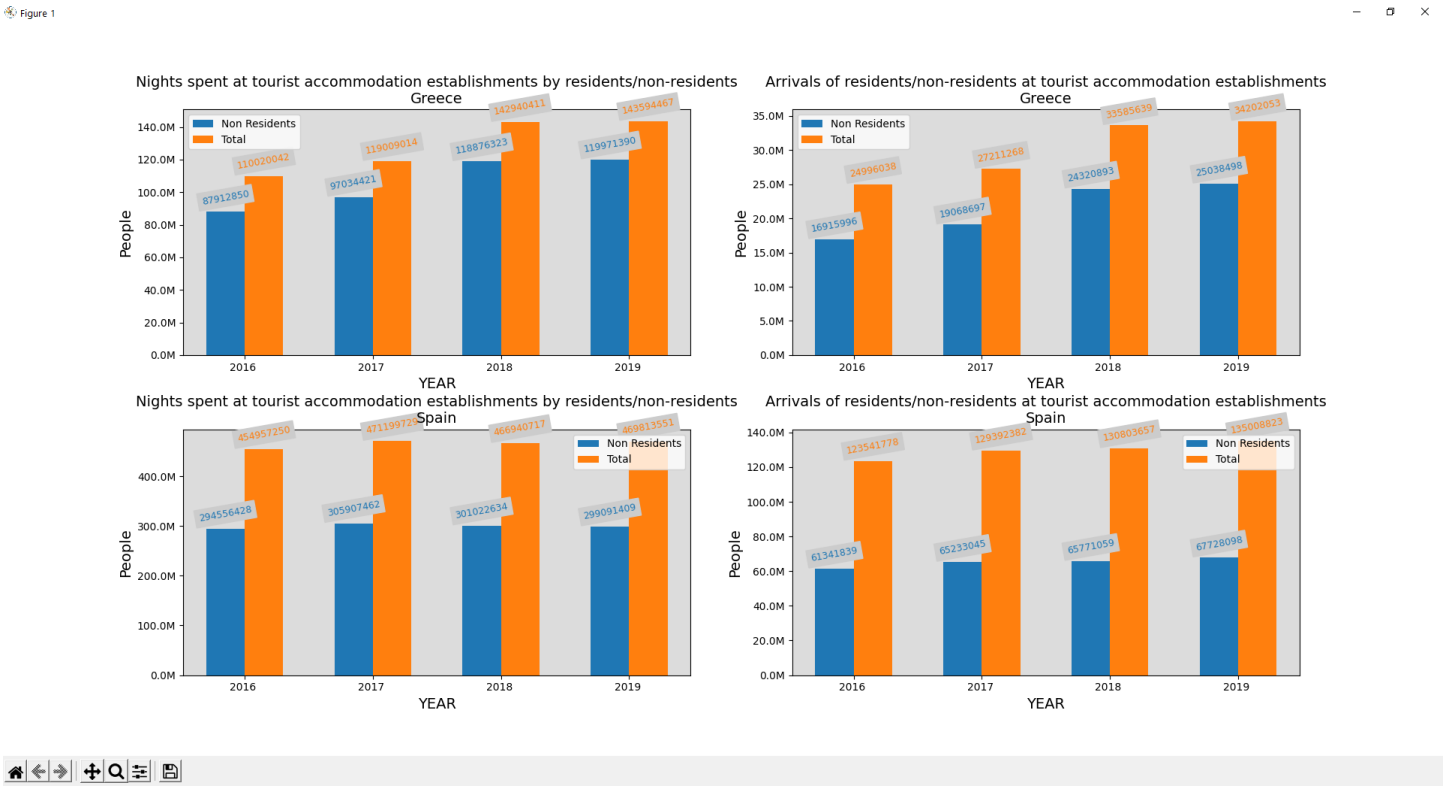


Screenshot απο το output της γραμμης εντολών κατα την επεξεργασία των δεδομένων και την αποθήκευση στη βάση δεδομένων.

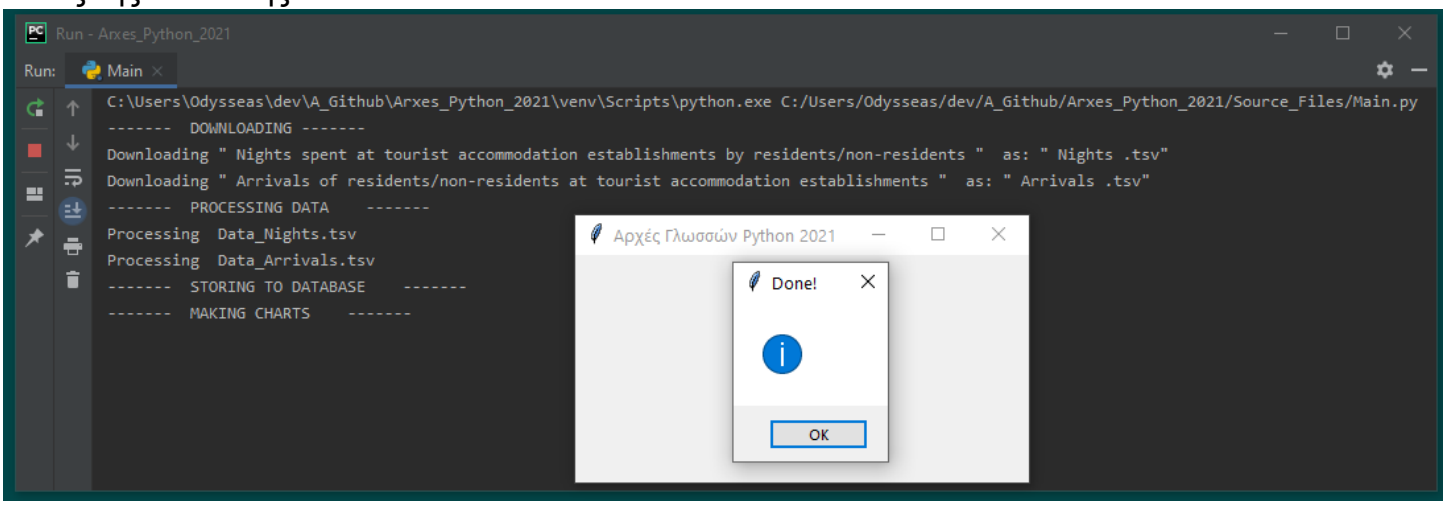


Τα γραφήματα

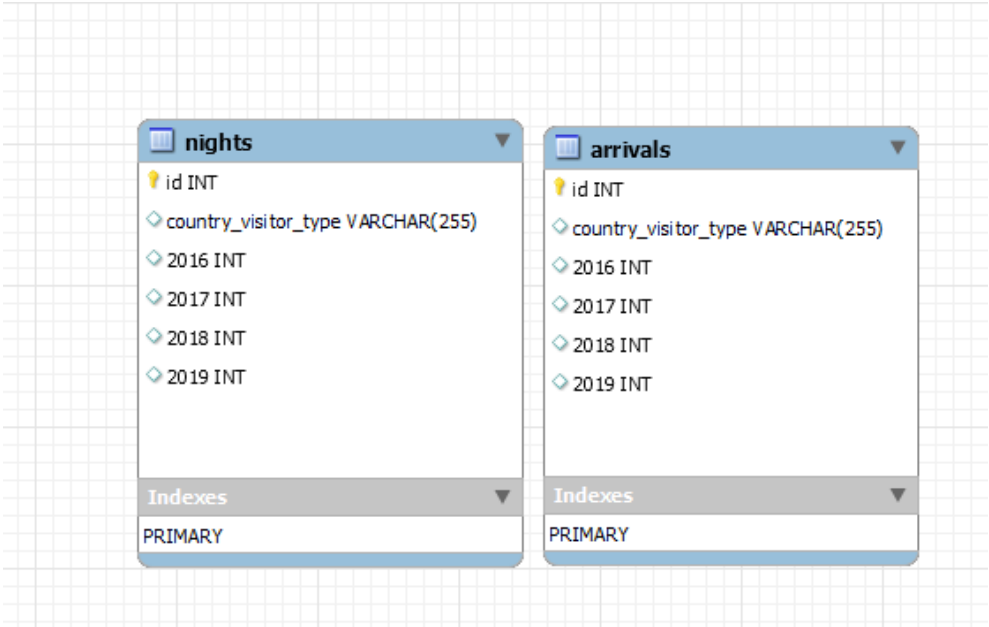
Υπάρχουν σε υψηλότερη ανάλυση, σε μορφή png , μαζί με την παρούσα αναφορά.



Τέλος της εκτέλεσης



Το σχήμα της βάσης δεδομένων



Το περιεχόμενο της βάσης δεδομένων