# 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:** 3](#_Toc73706655)

[**Παραδοχές:** 3](#_Toc73706656)

[**Ο κώδικας:** 4](#_Toc73706657)

[**Screenshots απο την εκτέλεση της εφαρμογής:** 16](#_Toc73706658)

[**Τα γραφήματα** 17](#_Toc73706659)

[**Το σχήμα της βάσης δεδομένων** 18](#_Toc73706660)

[**Το περιεχόμενο της βάσης δεδομένων** 19](#_Toc73706661)

# **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.

1. 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 .

1. 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.

1. Συνάρτηση **db\_store**

Δημιουργία της βάσης δεδομένων (MySql) και INSERTS των tables.

1. 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.

To 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

238 *# Create the connection to the local mySQL database and test it*

239 **try**:

240 db\_connection = mysql.connector.connect(

241 host="localhost",

242 user="root",

243 password="toor"

244 )

245 **except** mysql.connector.Error **as** err:

246 **print**("DATABASE ERROR ", "ERROR Something went wrong:**\n** {}".format(err))

247 mb.showerror("DATABASE ERROR ", "ERROR Something went wrong:**\n** {}".format(err))

248 *# print("ERROR Something went wrong: {}".format(err))*

249 **return** 1 *# Return 1 if unable to connect to a database*

250

251 *# Create a cursor*

252 mycursor = db\_connection.cursor(buffered=True)

253

254 *# Create the database*

255 mycursor.execute("DROP DATABASE IF EXISTS arxes\_db;")

256 mycursor.execute("CREATE DATABASE arxes\_db;")

257 mycursor.execute("use arxes\_db;")

258

259 **for** k **in** range(len(list\_in)):

260

261 *# Get the dataframe from the list*

262 df = list\_in[k][0]

263 *# Get the user title from the list ( will be the name of the table )*

264 user\_title = list\_in[k][1]

265

266 *# First step is to create a table which will be named with the user provided name .*

267 table\_name = user\_title

268 sql = ("CREATE TABLE " +

269 table\_name +

270 "(id INT AUTO\_INCREMENT PRIMARY KEY," +

271 " country\_visitor\_type VARCHAR(255)," +

272 "`2016` INT," +

273 "`2017` INT," +

274 "`2018` INT," +

275 "`2019` INT)")

276 *# print(sql)*

277 mycursor.execute(sql)

278

279 *# For each row in our table*

280 **for** j **in** range(len(list\_in[k]) + 1):

281

282 *# Base sql insertion query string , concatenate stuff to it , in order to make the insertions queries*

283 sql\_insert = "INSERT INTO " + table\_name + \

284 " (`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 **if** i != 0:

293 temp = temp.rstrip(temp[-1])

294

295 sql\_insert += temp + "',"

296

297 *# Delete the last comma*

298 sql\_insert = sql\_insert.rstrip(sql\_insert[-1])

299 sql\_insert += ");"

300

301 *# # Show the SQL query*

302 *# print(sql\_insert)*

303

304 *# Execute it*

305 mycursor.execute(sql\_insert)

306

307 *# Save the changes to the database*

308 db\_connection.commit()

309

310 *# -------------------------------- CHART CREATOR --------------------------------------------------------*

311

312 *# This function takes 4 inputs:*

313 *# 1) A list that contains:*

314 *# 1.1) a pandas dataframe*

315 *# 1.2) The user title*

316 *# 1.3) The original file name to be used as the subplot title*

317 *# 2) The number of the subplot*

318 *# 3) The list of country codes*

319 *# 4) The list of country names*

320

321

322 **def** make\_charts\_2(list\_in, subplot\_number, country\_code, country\_name):

323

324 *# There will be 4 plots, in a 2 x 2 grid*

325 plot = plt.subplot(2, 2, subplot\_number)

326

327 *# Set the plot title as the original file name*

328 plot\_title = list\_in[2]

329

330 *# Set the title of the subplot*

331 plot.set\_title(plot\_title + "**\n**" + country\_name, fontsize=14)

332

333 *# set the subplot background color for better readability*

334 plot.set\_facecolor("gainsboro")

335

336 *# Set the label for the y axis*

337 plot.set\_ylabel('People', fontsize=14)

338

339 *# Set the label for the x axis*

340 plot.set\_xlabel('YEAR', fontsize=14)

341

342 *# Get the dataframe from the list*

343 df = list\_in[0]

344

345 *# Get the names of all the columns into a list*

346 *# ( will be used to title each bar for the bar plot )*

347 years = df.columns.tolist()

348 *# drop the first column from the list*

349 years.pop(0)

350

351 *# Return evenly spaced values based on the length of the list supplied*

352 *# example: For 4 years, x will be [ 0 1 2 3]*

353 x = np.arange(len(years)) *# the label locations*

354

355 *# Place ticks(labels) on the x axis, on the evenly spaced values*

356 plot.set\_xticks(x)

357

358 *# Source for labels text to attach to each tick is the years*

359 plot.set\_xticklabels(years)

360

361 *# ------------- Code to keep the correct country rows -------------*

362

363 *# Keep only the rows that have the country column ends with the country code we want*

364 *# example : keep only the rows in which the country column ends with 'EL'*

365 df1 = df[(df['COUNTRY'].str.endswith(country\_code))]

366

367 *# ------------- Code to keep the number of Foreign visitors -------------*

368

369 *# Keep only the row that have the country column BEGIN with FOR*

370 *# To keep the foreigners = non residents*

371 data\_foreign = df1[(df1['COUNTRY'].str.startswith('FOR'))]

372 *# Convert the dataframe into a list of lists*

373 data\_foreign = data\_foreign.values.tolist()

374 *# Keep the only item of the list*

375 data\_foreign = data\_foreign[0]

376 *# Delete the first item of the list, which is the country code and data type ( FOR|TOTAL)*

377 data\_foreign.pop(0)

378 *# Make the list of strings into a list of integers*

379 data\_foreign = [int(i) **for** i **in** data\_foreign]

380

381 *# ------------- Code to keep the total number of visitors -------------*

382

383 *# Keep in a dataframe only the row that has the country column that begins with TOTAL*

384 *# To keep the total number of visitors*

385 data\_total = df1[(df1['COUNTRY'].str.startswith('TOTAL'))]

386 *# Convert the dataframe into a list of lists*

387 data\_total = data\_total.values.tolist()

388 *# Keep the only item of the list*

389 data\_total = data\_total[0]

390 *# Delete the first item of the list, which is the country code and data type ( FOR|TOTAL)*

391 data\_total.pop(0)

392 *# Make the list of strings into a list of integers*

393 data\_total = [int(i) **for** i **in** data\_total]

394

395 width = 0.3 *# the width of the bars of the plot*

396

397 *# The two bars are created*

398 rect1 = plot.bar(x - width / 2, data\_foreign, width, label='Non Residents')

399 rect2 = plot.bar(x + width / 2, data\_total, width, label='Total')

400

401 *# The labels for the two bars are created*

402 plot.bar\_label(rect1, padding=5, fmt="**%d**", color='#1f77b4', backgroundcolor='0.8', rotation=10, size=9)

403 plot.bar\_label(rect2, padding=5, fmt='**%d**', color='#ff7f0e', backgroundcolor='0.8', rotation=10, size=9)

404

405 *# Create the formatting for the vertical axis*

406 *# This code was taken from stackoverflow*

407 *# https://stackoverflow.com/questions/40511476/how-to-properly-use-funcformatterfunc*

408 **def** millions(x, pos):

409 **return** '**%1.1f**M' % (x \* 1e-6)

410 *# Create the formatting for the vertical axis*

411 formatter = FuncFormatter(millions)

412

413 *# Set the formatting for the vertical axis*

414 plot.get\_yaxis().set\_major\_formatter(formatter)

415

416 *# Show a legend*

417 plot.legend()

418

419

420 *# This function takes as input*

421 *# 1) A list of lists, each list consists of 3 items*

422 *# 1.1) A pandas data frame*

423 *# 1.2) The user appointed name*

424 *# 1.3) The original name of the tsv file*

425 *# 2) The number of the subplot to be created*

426 *# 3) A list of country codes ['EL', 'ES']*

427 *# 4) A list of country names ['Greece', 'Spain']*

428

429

430 **def** make\_charts(in\_list, country\_codes, country\_names):

431

432 nigths\_list = in\_list[0]

433 arrivals\_list = in\_list[1]

434

435 make\_charts\_2(nigths\_list, 1, country\_codes[0], country\_names[0])

436 make\_charts\_2(nigths\_list, 3, country\_codes[1], country\_names[1])

437 make\_charts\_2(arrivals\_list, 2, country\_codes[0], country\_names[0])

438 make\_charts\_2(arrivals\_list, 4, country\_codes[1], country\_names[1])

439

440 plt.show()

441

442 *# ------------------------------ MAIN CODE -----------------------------------------------------------*

443

444 *# The list "URL\_list" contains lists that have 3 items each :*

445 *# 1) The url for each file we want to download*

446 *# 2) A name created by the user to easily distinguish the file*

447 *# 3) The original file name from eurostat*

448 *# This list will be passed to the downloader function, to download the data.*

449

450

451 URL\_list = [

452 *# list 1*

453 [

454 "https://ec.europa.eu/eurostat/estat-navtree-portlet-prod/BulkDownloadListing?file=data/tin00175.tsv.gz"

455 ,

456 "Nights"

457 ,

458 "Nights spent at tourist accommodation establishments by residents/non-residents"

459 ]

460 ,

461 *# list 2*

462 [

463 "https://ec.europa.eu/eurostat/estat-navtree-portlet-prod/BulkDownloadListing?file=data/tin00174.tsv.gz"

464 ,

465 "Arrivals"

466 ,

467 "Arrivals of residents/non-residents at tourist accommodation establishments"

468 ]

469 ]

470

471 *# Initialize tkInter*

472 root = tk.Tk()

473 *# root.withdraw()*

474

475 *# Tkinter window miscellaneous options*

476 *# root.iconbitmap("../Images/favicon.ico")*

477 root.title("Αρχές Γλωσσών Python 2021")

478 root.geometry("600x300+650+400") *# Width x Height + Padding left + Padding top*

479

480 *# Ask for a location to download the data into*

481 directory\_change()

482

483 *# The list "downloaded\_files" contains information about the files that the downloader has downloaded*

484 *# It Holds lists, that have 3 items each:*

485 *# 1) The filename of the .tsv file that was downloaded*

486 *# 3) The user created title*

487 *# 2) The original filename from the website*

488 downloaded\_files = []

489

490 **print**("------- DOWNLOADING -------")

491 **for** i **in** range(len(URL\_list)):

492 *# Feed each item of the URL list to the downloader*

493 temp = downloader(URL\_list[i])

494 **if** temp **is** **not** None:

495 downloaded\_files.append(temp)

496

497 *# IF no files have been downloaded, abort*

498 **if** **not** downloaded\_files:

499 **print**("No Data Downloaded, Exiting....")

500 mb.showerror("No Data Downloaded", "No Data. **\n** " + "Exiting...")

501 exit(0)

502

503 *# Ask user whether to continue with processing the data*

504 continue\_after\_download = mb.askquestion("Continue to data processing ?",

505 "You can stop here and just keep the downloaded files." "**\t\t**")

506 **if** continue\_after\_download == "no":

507 exit(0)

508

509 *# The list "cleaned\_files" contains the cleaned pandas dataframes + additional info*

510 *# Holds lists that have 3 items :*

511 *# 1) pandas dataframe that has been " cleaned "*

512 *# 2) The user created title*

513 *# 3) The original filename from the website*

514 cleaned\_files = []

515

516 **print**("------- PROCESSING DATA -------")

517 **for** i **in** range(len(downloaded\_files)):

518 cleaned\_files.append(data\_processor(downloaded\_files[i]))

519

520 **if** **not** cleaned\_files:

521 **print**("No Data, Exiting....")

522 mb.showerror("No Data", "No Data. **\n** " + "Exiting...")

523 exit(0)

524

525 **print**("------- STORING TO DATABASE -------")

526 db\_store(cleaned\_files)

527

528

529 *# The list of country codes the data will be plotted for*

530 *# global country\_code*

531 country\_codes = ['EL', 'ES']

532

533 *# The equivalent names for the above country codes*

534 *# global country\_name*

535 country\_names = ['Greece', 'Spain']

536

537 **print**("------- MAKING CHARTS -------")

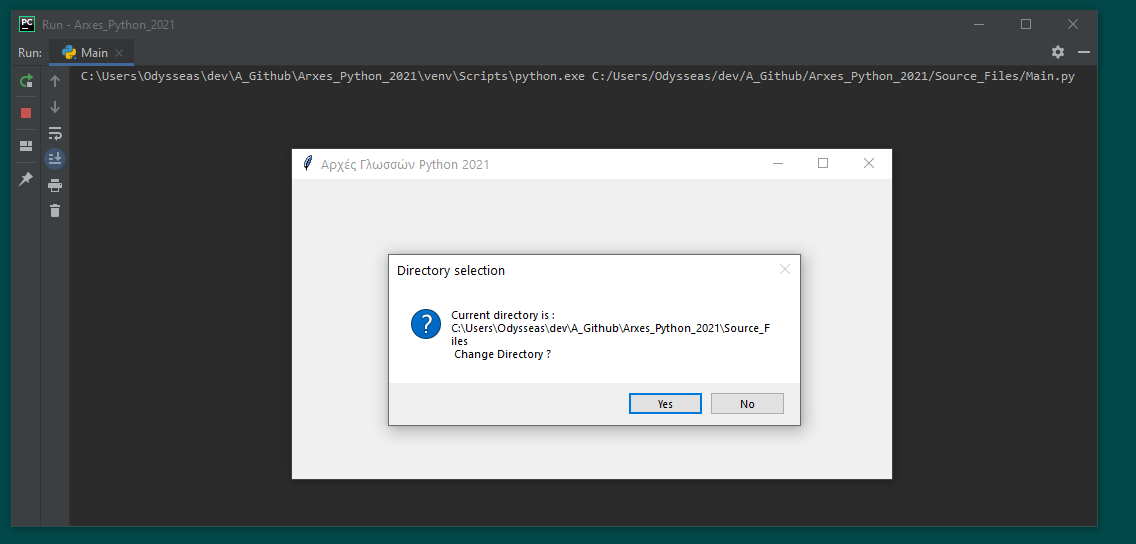
538 make\_charts(cleaned\_files, country\_codes, country\_names)

539

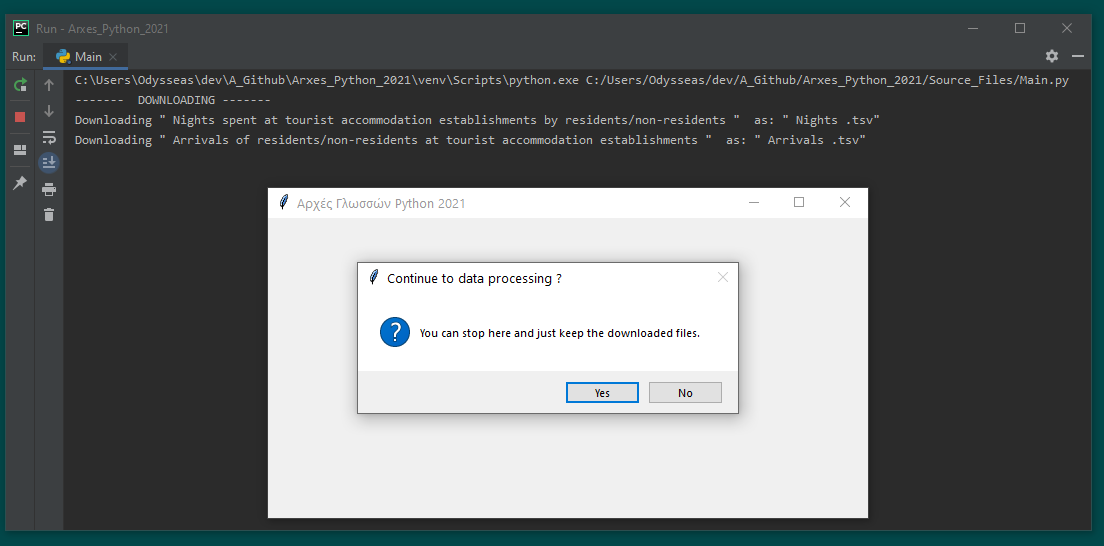
540 mb.showinfo("Done!**\t\t**")

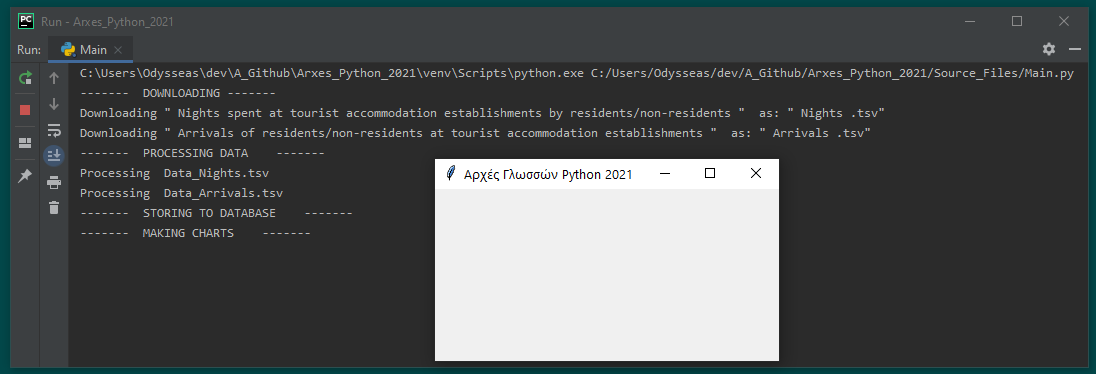
# **Screenshots απο την εκτέλεση της εφαρμογής:**

Αρχικά, η ερώτηση για την επιλογή directory



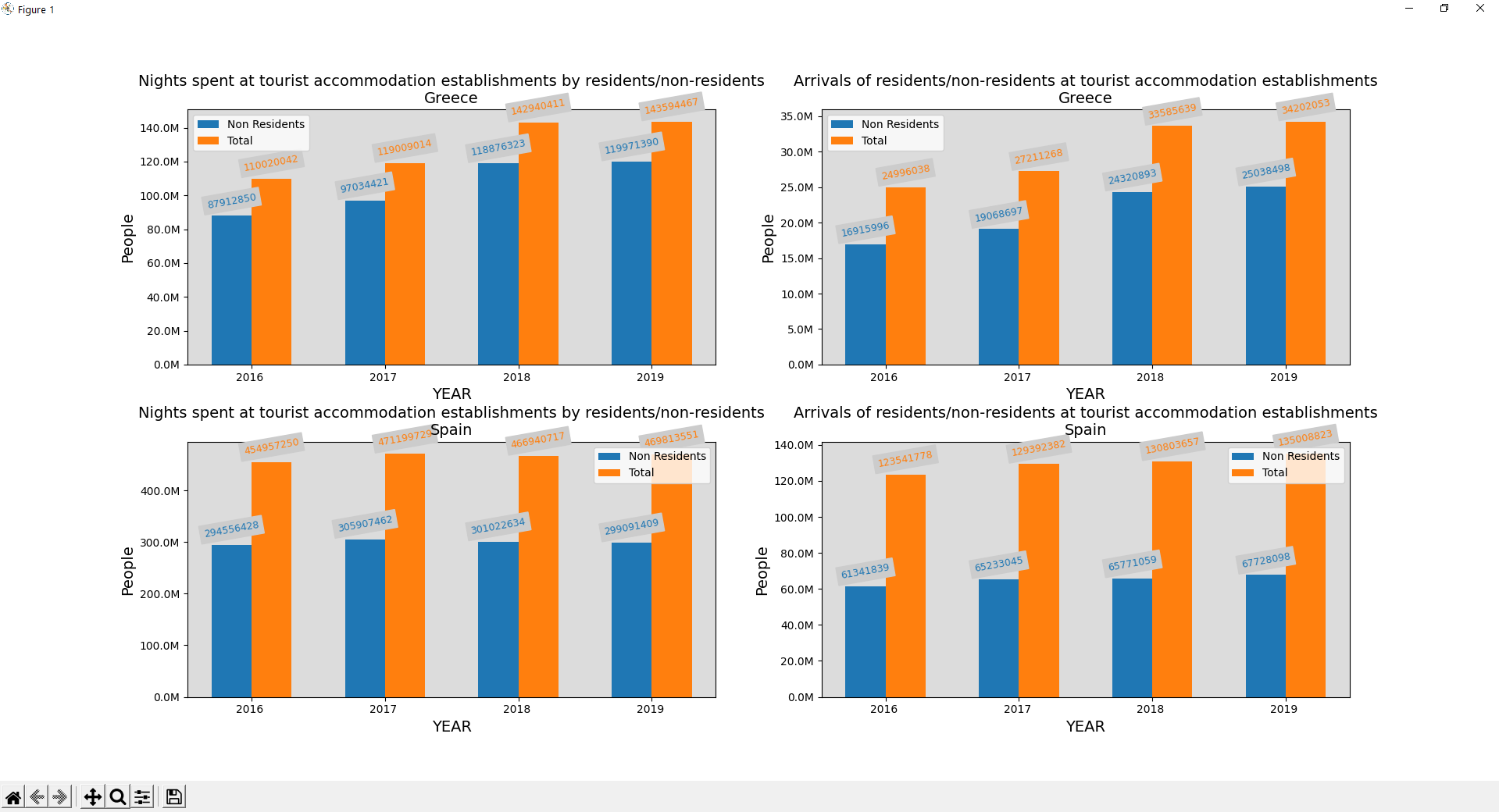
Ερώτηση αν ο χρήστης θέλει να συνεχίσει ή αν θέλει μόνο να κρατήσει τα κατεβασμένα αρχεία.

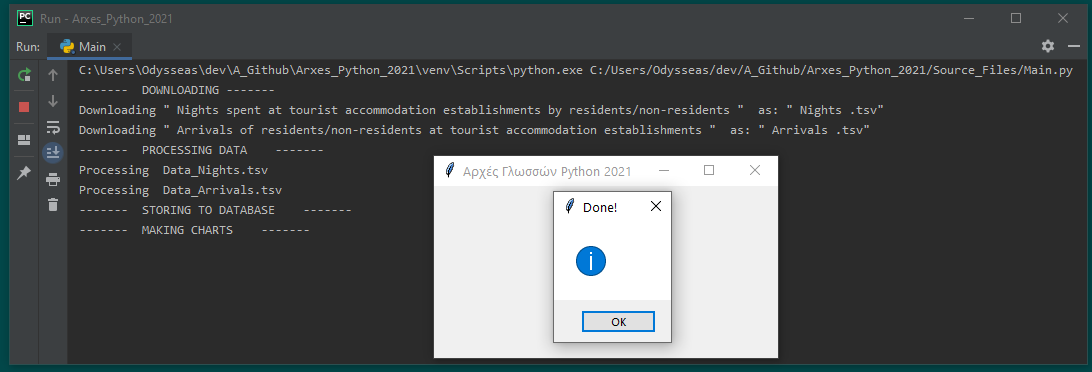


Screenshot απο το output της γραμμης εντολών κατα την επεξεργασία των δεδομένων και την αποθήκευση στη βάση δεδομένων.

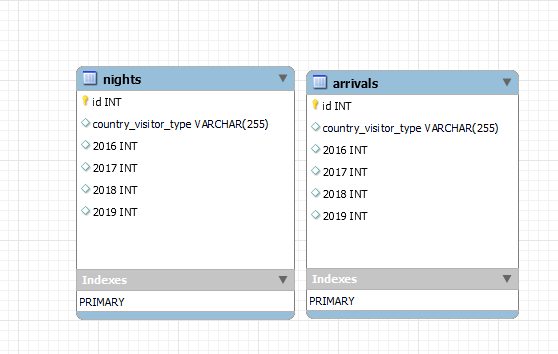
# **Τα γραφήματα**

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**

