# 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](#_Toc73706293)

[**Παραδοχές** 3](#_Toc73706294)

[**Ο κώδικας** 4](#_Toc73706295)

[**Screenshots απο την εκτέλεση της εφαρμογής** 16](#_Toc73706296)

[**Το σχήμα της βάσης δεδομένων** 18](#_Toc73706297)

[**Το περιεχόμενο της βάσης δεδομένων** 19](#_Toc73706298)

# **Σύντομη περιγραφή του κώδικα:**

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

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 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** '**%1.1f**M' % (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)

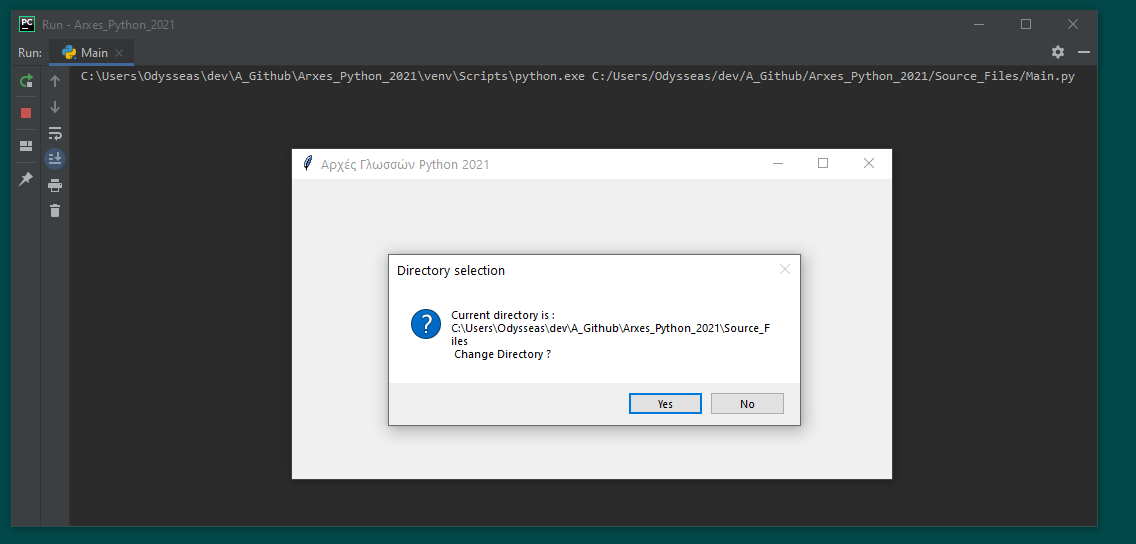
538

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

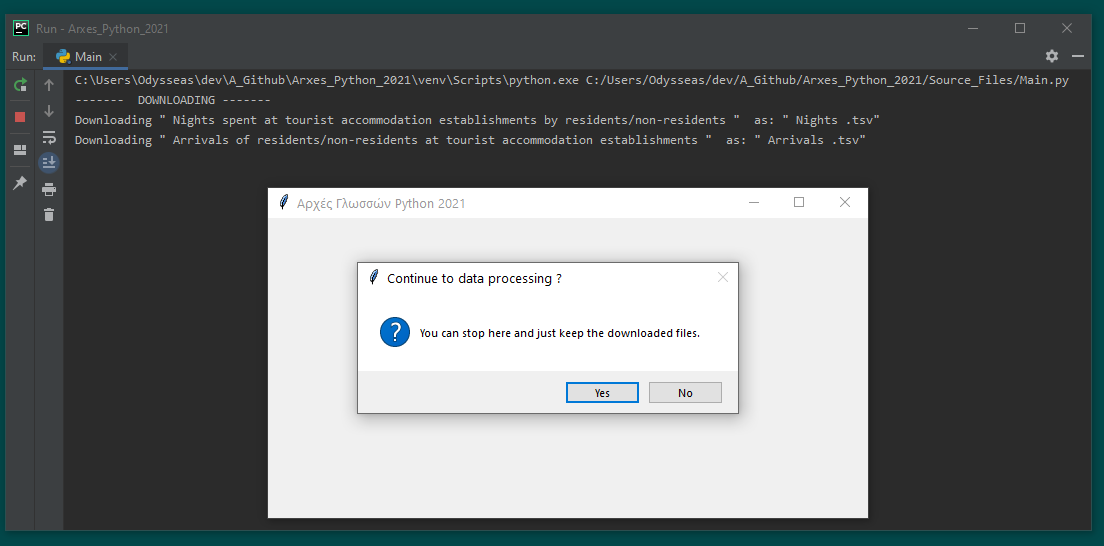
540

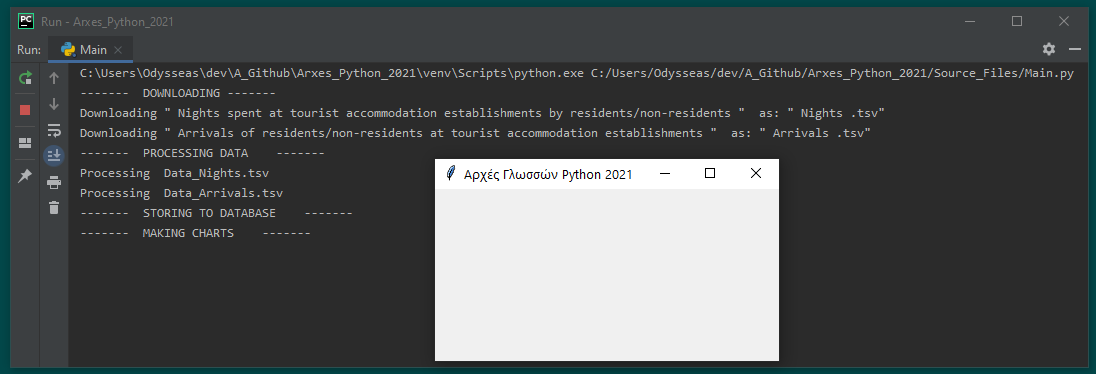
# **Screenshots απο την εκτέλεση της εφαρμογής:**

Αρχικά, η ερώτηση για την επιλογή directory

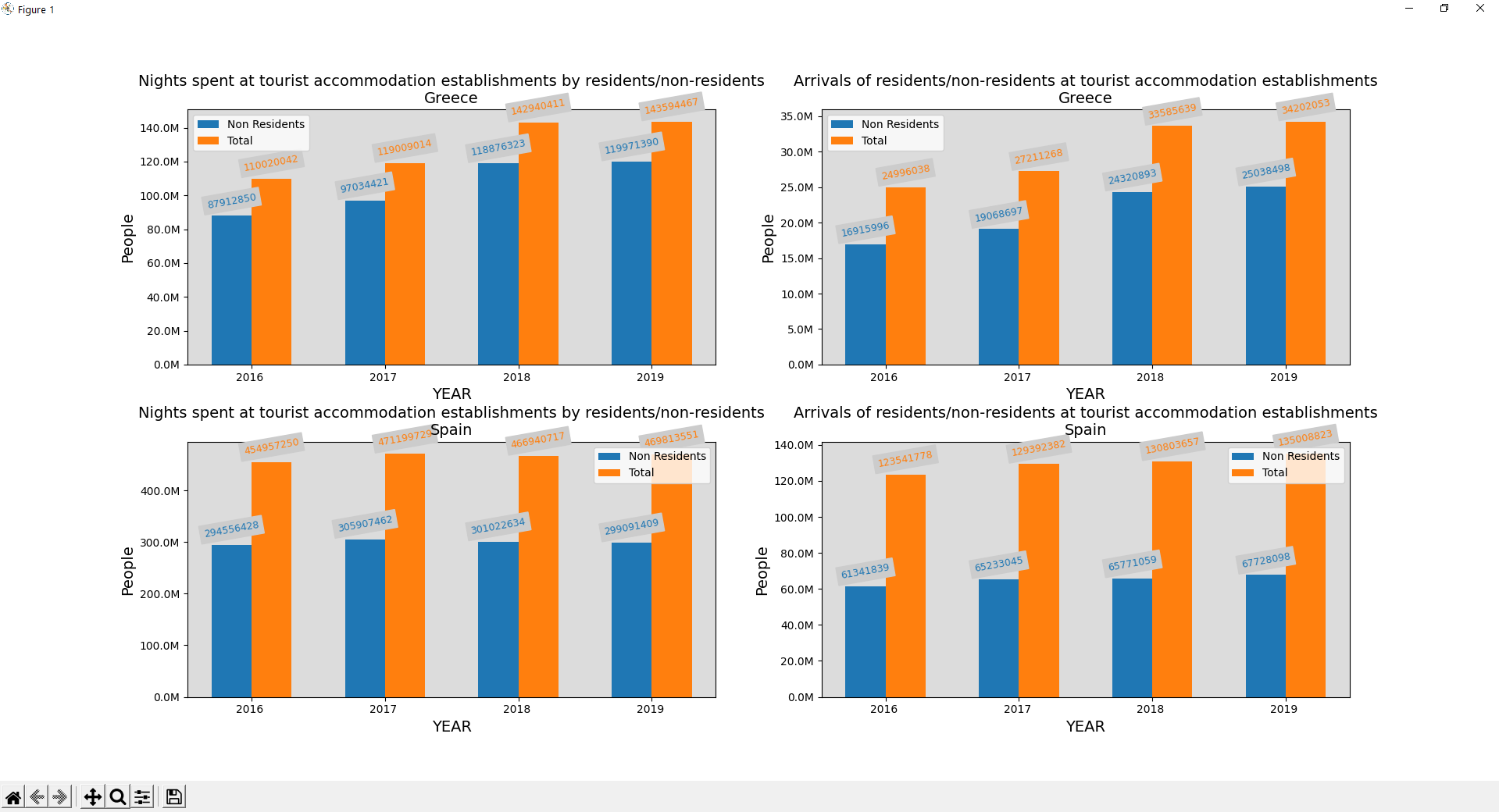


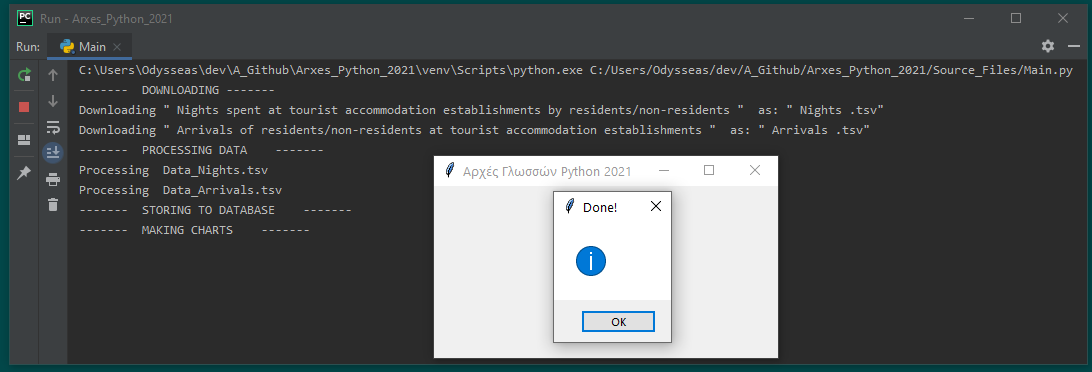
Ερώτηση αν ο χρήστης θέλει να συνεχίσει ή αν θέλει μόνο να κρατήσει τα κατεβασμένα αρχεία.



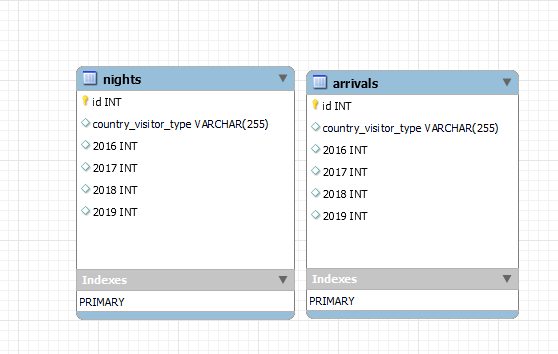
Screenshot απο το output της γραμμης εντολών κατα την επεξεργασία των δεδομένων και την αποθήκευση στη βάση δεδομένων.

Τα γραφήματα όπως παράγονται απο το matplotlib. ( 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**

