Τμήμα Μηχανικών Η/Υ & Πληροφορικής

Πανεπιστήμιο Πατρών

Μάθημα Αρχών Γλωσσών Προγραμματισμού και Μεταφραστών

Προαιρετική Εργασία Εξαμήνου σε Γλώσσα Python

Κατσαρός Γεώργιος

E-mail: st1093386@ceid.upatras.gr

AM: 1093386

Έτος: 2ο

Εαρινό Εξάμηνο 2023-2024

Περιεχόμενα:

Κώδικας Python:

Ο κώδικας είναι γενικά οργανωμένος σε 3 αρχεία, main.py, QtUi.py και functions.py. Αυτό έχει γίνει έτσι ώστε να είναι πιό οργανομένος.

main.py

```
# The main code file that just calls the PyQt Application

import sys
from PySide6 import QtCore, QtWidgets, QtGui

import QtUi as ui
import functions as fun

app = QtWidgets.QApplication([])

if __name__ == "__main__":
    widget = ui.MyWidget()
    widget.setFixedSize(800, 600)
    widget.show()
    sys.exit(app.exec())
```

Αυτό το αρχείο δεν κάνει κάτι περίπλοκο. Συνοπτικά, αρχικοποιεί μια PySide6 Application, δημιουργεί και διαμορφώνει ένα κύριο widget που αντιπροσωπεύει το παράθυρο της εφαρμογής και ξεκινά το κύκλο συμβάντων της εφαρμογής για την εκτέλεση του GUI.

Δημιουργία μιας περίπτωσης QApplication:
app = QtWidgets.QApplication([]): Αρχικοποιεί ένα αντικείμενο QApplication, το οποίο είναι απαραίτητο για κάθε εφαρμογή Qt. Αυτό το αντικείμενο διαχειρίζεται τους πόρους και τις ρυθμίσεις που αφορούν όλη την εφαρμογή.

Κύρια λογική της εφαρμογής:
widget = ui.MyWidget(): Δημιουργεί μια περίπτωση της κλάσης MyWidget που ορίζεται στο αρχειο QtUi.py και αντιπροσοπεύει το παράθυρο της εφαρμογής.

widget.setFixedSize(800, 600): Ορίζει ένα σταθερό μέγεθος για το παράθυρο της εφαρμογής σε 800x600px. Ο λογος που το παράθυρο έχει σταθερό μέγεθος είναι περισσότερο αισθητικός παρά λειτουργικός ωστε να διασφαλίζεται μια ομοιόμορφη εμφάνιση και ο κώδικας για τα μεγέθη και τις τοποθετήσεις τον αντικειμένων να παραμείνει απλός.

widget.show(): Εμφανίζει το παράθυρο στην οθόνη.

Έναρξη του κύκλου συμβάντων της εφαρμογής:
sys.exit(app.exec()): Εκκινεί το βρόχο συμβάντων της Qt καλώντας την app.exec(), η οποία διατηρεί την εφαρμογή σε λειτουργία και ανταποκρίνεται στην είσοδο του χρήστη μέχρι την έξοδο. Η sys.exit() διασφαλίζει ότι η εφαρμογή τερματίζεται ολοκληρωμένα.

QtUi.py

Αυτό το αρχείο καθορίζει την εμφάνιση και τη συμπεριφορά των στοιχείων του GUI της εφαρμογής.
Οι βασικές λειτουργίες που κανει ειναι:

1. **Ορισμός της κλάσης MyWidget**

```
# The configuration of the GUI elements
from PySide6 import QtCore, QtWidgets, QtGui
import matplotlib.pyplot as plt
import functions as fun

class MyWidget(QtWidgets.QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Python Project 2024")
```

2. **Δήλωση κουμπιών**

```
        #Declaration of buttons
        ReadFileButton = QtWidgets.QPushButton("Read File")
        CalcBasicButton = QtWidgets.QPushButton("Basic Stats")
        CalcMonthButton = QtWidgets.QPushButton("Month Stats")
        CalcSeasonButton = QtWidgets.QPushButton("Season Stats")
        CalcRoomTypeButton = QtWidgets.QPushButton("Room Type Stats")
        CalcVisitorTypeButton = QtWidgets.QPushButton("Visitor Type Stats")
        CalcTrendButton = QtWidgets.QPushButton("Trends")
```

3. **Δήλωση της κονσόλας εξόδου**

```
        #Declaration of outputConsole Label
        ConsoleText = QtWidgets.QLabel("Please Click Read File To Start." )
        ConsoleText.setStyleSheet("QLabel {color: black; background-color:
silver; border: 1px solid gray; border-radius: 2px;}")
```

4. **Δήλωση των Layouts:**

```
        #Declaration of the application layouts
        outerLayout = QtWidgets.QHBoxLayout(self)
        menuLayout = QtWidgets.QVBoxLayout()
        outputLayout = QtWidgets.QVBoxLayout()
```

Το *outerLayout* είναι μια οριζόντια διάταξη QHBoxLayout που εφαρμόζεται σε ολο το παράθυρο με το ‘(self)’.

Το *menuLayout* είναι μια κάθετη διάταξη QVBoxLayout.
Το *outputLayout* είναι επίσης μια κάθετη διάταξη QVBoxLayout.
Το πώς αυτά συνδυάζονται μεταξύ τους θα αναλυθεί παρακάτω.

5. **Ρύθμιση της απεικόνισης των γραφημάτων εξόδου**

```
        #Setup of the output graph display, initialization as a blank white
screen
        pic = QtWidgets.QLabel(self, alignment=QtCore.Qt.AlignCenter)
        fig= plt.figure()
        fig.savefig(".blank.png")
        pixmap = QtGui.QPixmap(".blank.png")
        pic.setPixmap(pixmap)
        outputLayout.addWidget(pic)
        pic.show()
```

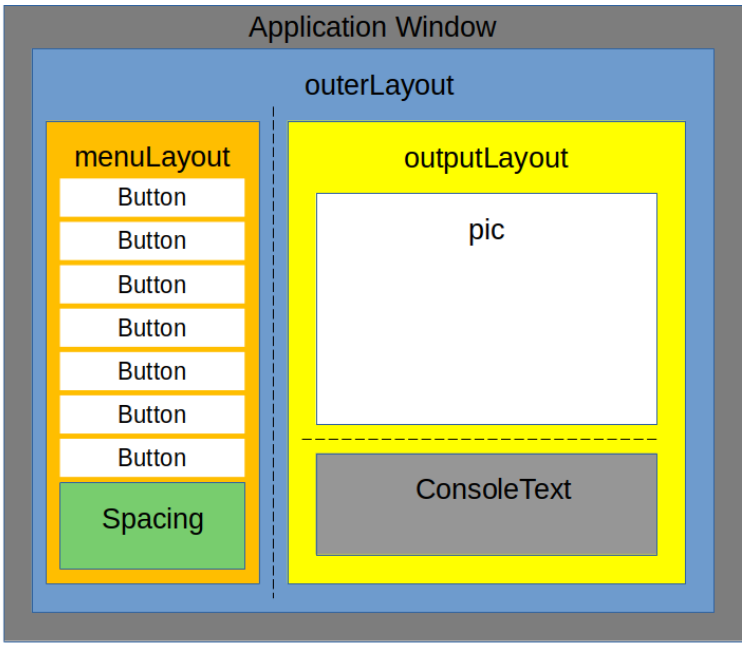
Απο την δημιουργία του παραθύρου δημιουργείται και ένα κενό γράφημα το οποίο αποθηκεύεται ως εικόνα και ύστερα αυτή η εικόνα προβάλλεται στο outputLayout. Στην συνέχεια, όποιο γράφημα θέλουμε να προβάλλεται θα φορτώνεται ως εικόνα στην θέση αυτη.

6. **Τοποθέτηση των widgets στα αντίστοιχα layouts**

```
        #Placement of widgets to the coresponding layouts
        outputLayout.addWidget(ConsoleText)

        menuLayout.addWidget(ReadFileButton)
        menuLayout.addWidget(CalcBasicButton)
        menuLayout.addWidget(CalcMonthButton)
        menuLayout.addWidget(CalcSeasonButton)
        menuLayout.addWidget(CalcRoomTypeButton)
        menuLayout.addWidget(CalcVisitorTypeButton)
        menuLayout.addWidget(CalcTrendButton)

        menuLayout.insertSpacing(-1,500)
        outerLayout.addLayout(menuLayout)
        outerLayout.addLayout(outputLayout)
```

Τα κουμπιά προστίθενται στο menuLayout.
Τα ConsoleText προστίθεται στο outputLayout.
Τα menuLayout και outputLayout προστίθενται στο outerLayout.

Έτσι προκύπτει η διάταξη του παρακάτω διαγράμματος:



7. **Δήλωση συναρτήσεων κουμπιών:**
   a. **Κουμπί Ανάγνωσης Αρχείου:**

```
def readFileButton_func():
    ConsoleText.setText("Reading File...")
    ConsoleText.repaint()
    QtWidgets.QApplication.processEvents()
    try:
        fun.readFile('hotel_booking.csv')
    except FileNotFoundError:
        ConsoleText.setText("File does not exist!")
    else:
        ConsoleText.setText("File Read Successfully!")
```

ConsoleText.setText("Reading File…"): Η κονσόλα εξόδου ενημερώνει τον χρήστη πως το αρχείο διαβάζεται.

ConsoleText.repaint():Η κονσόλα εξόδου ενημερώνεται.

QtWidgets.QApplication.processEvents(): Η εφαρμογή εκτελεί την ενημέρωση της κονσόλας εξόδου

Αν δεν βρεθεί αρχείο με όνομα ‘hotel_booking.csv’ τοτε ο χρήστης ενημερώνεται μέσω της κονσόλας εξόδου και αντίστοιχα μόλις ολοκληρωθεί η ανάγνωση ο χρήστης ενημερώνεται πως αυτή ήταν επιτυχής.

b. Κουμπί Υπολογισμού Βασικών Στατιστικών:

```
      def calcBasicsButton_func():
            try:
                  fun.calcBasics()
            except NameError:
                  ConsoleText.setText("No File Selected! Please Read A
File First.")
            else:
                  ConsoleText.setText("Basic Stats Calculations
Successful! Showing Graph.")
                  pixmap = QtGui.QPixmap(".basicStats.png")
                  pic.setPixmap(pixmap)
                  pic.setScaledContents(True)
                  pic.show()
```

Εδω η εικόνα της οθόνης εξόδου ανανεώνεται να προβάλει το '.basicStats.png' που είναι το παραγόμενο γράφημα που έχει αποθηκευτεί στον φάκελο του αρχείου.
Να σημειωθεί πως τα παραγόμενα γραφήματα αποθηκεύονται ως Hidden Files και για αυτο υπάρχει η τελεία στην αρχή του ονόματος.

c. Αντίστοιχα Τα Υπόλοιπα Κουμπιά:

```
            def calcMonthsButton_func():
            try:
                  fun.calcMonthStats()
            except NameError:
                  ConsoleText.setText("No File Selected! Please Read A
File First.")
            else:
                  ConsoleText.setText("Month Stats Calculation
Successful! Showing Graph.")
                  pixmap= QtGui.QPixmap(".monthStats.png")
                  pic.setPixmap(pixmap)
                  pic.setScaledContents(True)
                  pic.show()

      def calcSeasonButton_func():
            try:
                  fun.calcSeasonStats()
            except NameError:
                  ConsoleText.setText("No File Selected! Please Read A
File First.")
            else:
                  ConsoleText.setText("Season Stats Calculation
Succeessful! Showing Graph.")
                  pixmap = QtGui.QPixmap(".seasonStats.png")
                  pic.setPixmap(pixmap)
```

```
                pic.setScaledContents(True)
                pic.show()

        def calcRoomTypeButton_func():
                try:
                        fun.calcRoomTypeStats()
                except NameError:
                        ConsoleText.setText("No File Selected! Please Read A
File First.")
                else:
                        ConsoleText.setText("Room Type Calculation Successful!
Showing Graph.")
                        pixmap = QtGui.QPixmap(".roomTypeStats.png")
                        pic.setPixmap(pixmap)
                        pic.setScaledContents(True)
                        pic.show()

        def calcVisitorTypeButton_func():
                try:
                        fun.calcVisitorTypeStats()
                except NameError:
                        ConsoleText.setText("No File Selected! Please Read A
File First.")
                else:
                        ConsoleText.setText("Visitor Type Stats Calculation
Successful! Showing Graph.")
                        pixmap = QtGui.QPixmap(".visitorTypeStats.png")
                        pic.setPixmap(pixmap)
                        pic.setScaledContents(True)
                        pic.show()

        def calcTrendButton_func():
                ConsoleText.setText("Calculating Trend... Please Wait.")
                ConsoleText.repaint()
                QtWidgets.QApplication.processEvents() #Force repaint of
output Console
                try:
                        fun.calcTrend()
                except NameError:
                        ConsoleText.setText("No File Selected! Please Read A
File First.")
                else:
                        ConsoleText.setText("Trend Calculation Successful!
Showing Graph.")
                        pixmap = QtGui.QPixmap(".trend.png")
                        pic.setPixmap(pixmap)
```

```
                pic.setScaledContents(True)
                pic.show()
```

8. **Event Handling:**

```
    #Calling of above functions at the event of a clicked button
    ReadFileButton.clicked.connect(readFileButton_func)
    CalcBasicButton.clicked.connect(calcBasicsButton_func)
    CalcMonthButton.clicked.connect(calcMonthsButton_func)
    CalcSeasonButton.clicked.connect(calcSeasonButton_func)
    CalcRoomTypeButton.clicked.connect(calcRoomTypeButton_func)
    CalcVisitorTypeButton.clicked.connect(calcVisitorTypeButton_func)
    CalcTrendButton.clicked.connect(calcTrendButton_func)
```

---

functions.py

Το αρχείο αυτό παρέχει ένα σύνολο συναρτήσεων για την επεξεργασία των δεδομένων. Τα διαγράμματα που προκύπτουν αποθηκεύονται ως αρχεία εικόνας για χρήση στο GUI.

```
#   Declaration of functions

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import linregress
```

1. **readFile(fileName):**

```
def readFile(fileName):
    global csvFile
    global df
    df = pd.read_csv(fileName)
    #Add collumn 'date' with the arrival date in format YYYY/Month/D
    df['date'] = df['arrival_date_year'].astype(str) + '/' +
df['arrival_date_month'].astype(str) + '/' +
df['arrival_date_day_of_month'].astype(str)
    df['date'] = pd.to_datetime(df['date'], format='%Y/%B/%d')
    #Set 'date' collumn as the index of the dataframe
    df = df.set_index('date')
    #Create dictionary of dataframe
    csvFile = df.to_dict(orient='records')
```

Αρχείο διαβάζεται και αποθηκεύεται σε ενα data frame *'df'*.
Στο *'df'* προστίθεται η στήλη *'date'* που αποτελεί συνδυασμό των στηλών *'arrival_date_year', 'arrival_date_month', 'arrival_date_day_of_month*'.
Η στήλη *'date'* μετατρέπεται σε datetime format YYYY/MM/DD και ορίζεται ως το index του dataframe.

Ο όρος *'%Y'* αντιστοιχεί στην χρονολογία, το *'%B'* είναι ο μήνας γραμμένος ολογράφως και το *'%d'* η ημέρα του μήνα.
Δηλαδή μία γραμμή της στήλης *'date':* '2016/July/01' θα μετατραπεί σε '2016/07/01'.
Τέλος το dataframe *'df'* αποθηκεύεται ως dictionary *'csvFile'*.

2. **Υπολογισμός Διανυκτερεύσεων:**

Υπολογίζει τον μέσο αριθμό διανυκτερεύσεων σε ξενοδοχεία θέρετρων και πόλεων.

```
def calcStayInNightsAvg():
      global resortNightsAvg
      global cityNightsAvg
      resortNightsAvg = 0
      cityNightsAvg = 0
      resortCounter = 0
      cityCounter = 0
      #Count all StayIns for each hotel type
      for lines in csvFile:
      if lines['hotel'] == "Resort Hotel":
            resortNightsAvg += int(lines['stays_in_weekend_nights']) +
int(lines['stays_in_week_nights'])
            resortCounter += 1
      else:
            cityNightsAvg += int(lines['stays_in_weekend_nights']) +
int(lines['stays_in_week_nights'])
            cityCounter += 1
      #Calculate Precentage
      resortNightsAvg = round(resortNightsAvg/resortCounter, 2)
      cityNightsAvg = round(cityNightsAvg/cityCounter, 2)
```

3. **Υπολογισμός Ακυρώσεων:**

Υπολογίζει το ποσοστό ακύρωσης για ξενοδοχεία σε θέρετρα και πόλεις.

```
def calcCansellationPrc():
      global resortCancellationPrc
      global cityCancellationPrc
      resortCancellationPrc = 0
      cityCancellationPrc = 0
      resortCounter = 0
      cityCounter = 0
      #Count all cansellations for each hotel type
      for lines in csvFile:
      if lines['hotel'] == "Resort Hotel":
            resortCancellationPrc += int(lines['is_canceled'])
            resortCounter += 1
      else:
            cityCancellationPrc += int(lines['is_canceled'])
            cityCounter += 1
      #Calculate Precentage
      resortCancellationPrc = round(float((resortCancellationPrc /
```

```
resortCounter) * 100), 2)
      cityCancellationPrc = round(float((cityCancellationPrc /
cityCounter) * 100), 2)
```

4. **Υπολογισμός Βασικών Στατιστικών στοιχείων:**

```
def calcBasics():
      calcStayInNightsAvg()
      calcCansellationPrc()

      #Create double plot of the above data
      fig = plt.figure()
      axes = fig.subplots(2)

      #Create first plot about cancellations
      axes[0].barh(["Resort", "City"], [resortCancellationPrc,
cityCancellationPrc], color=['tab:orange', 'tab:red'])
      axes[0].text(resortCancellationPrc/2, 0,
str(resortCancellationPrc) + "%", color = 'white')
      axes[0].text(cityCancellationPrc/2, 1, str(cityCancellationPrc) +
"%", color = 'white')
      axes[0].title.set_text("Hotel Cancellation Precentage:")

      #Create second plot about Stayins
      axes[1].barh(["Resort", "City"], [resortNightsAvg, cityNightsAvg],
color=['tab:orange', 'tab:red'])
      axes[1].text(resortNightsAvg/2, 0, str(resortNightsAvg), color =
'white')
      axes[1].text(cityNightsAvg/2, 1, str(cityNightsAvg), color =
'white')
      axes[1].title.set_text("Hotel Night Stay-Ins Average:")

      #Save plot
      fig.tight_layout()
      fig.savefig(".basicStats.png")
      plt.close()
```

Αρχίκα, καλούνται οι συναρτήσεις *calcStayInNightsAvg()* και *calcCansellationPrc()* και μετά απεικονίζονται σε ένα διπλό γράφημα το οποίο αποθηκεύται ως *'.basicStats.png'*

5. **Υπολογισμός Διανυκτερεύσεων ανα μήνα:**

```
def calcMonthStats():
      global totalMonthDict
      totalMonthDict = {
      "January" : 0,
      "February" : 0,
```

```
      "March" : 0,
      "April" : 0,
      "May" : 0,
      "June" : 0,
      "July" : 0,
      "August" : 0,
      "September" : 0,
      "October" : 0,
      "November" : 0,
      "December" : 0

      }

      resortMonthDict = totalMonthDict.copy()
      cityMonthDict = totalMonthDict.copy()

      #Count reservations for each month and the total reservations
      for lines in csvFile:
      if lines['hotel'] == "Resort Hotel":
            resortMonthDict[lines['arrival_date_month']] += 1
            totalMonthDict[lines['arrival_date_month']] += 1
      else:
            cityMonthDict[lines['arrival_date_month']] += 1
            totalMonthDict[lines['arrival_date_month']] += 1

      #Create plot
      #Plot total reservations
      plt.plot(list(totalMonthDict.keys()),
list(totalMonthDict.values()), marker='.', label = 'Total',
color='Blue')
      #Plot Resort Reservations
      plt.plot(list(resortMonthDict.keys()),
list(resortMonthDict.values()), marker='.', label = 'Resort Hotels',
color='Orange')
      #Plot City reservations
      plt.plot(list(cityMonthDict.keys()), list(cityMonthDict.values()),
marker='.', label = 'City Hotels', color='Red')
      #Plot Decorations:
      plt.fill_between(list(totalMonthDict.keys()),
list(totalMonthDict.values()), color='blue', alpha=.1)
      plt.fill_between(list(cityMonthDict.keys()),
list(cityMonthDict.values()), color='red', alpha=.1)
      plt.fill_between(list(resortMonthDict.keys()),
list(resortMonthDict.values()), color='orange', alpha=.1)
      plt.grid()
      plt.legend()
```

```
      plt.title("Reservations Per Month:")
      plt.xticks(rotation=30)
      plt.tight_layout()
      #Save plot
      plt.savefig(".monthStats.png")
      plt.close()
```

Αρχικοποιεί τα dictionaries για την καταμέτρηση των μηνιαίων κρατήσεων για το σύνολο, το θέρετρο και τα ξενοδοχεία πόλης.
Διατρέχει τα δεδομένα και μετράει τις κρατήσεις για κάθε μήνα.
Δημιουργεί ένα διάγραμμα που δείχνει τον αριθμό των κρατήσεων ανά μήνα για το σύνολο, το θέρετρο και τα ξενοδοχεία της πόλης.
Αποθηκεύει το διάγραμμα ως *'.monthStats.png'*

6. Υπολογισμός Κρατήσεων Ανα Εποχή:

```
def calcSeasonStats():
      totalSeasonDict = {
      "Winter" : 0,
      "Spring" : 0,
      "Summer" : 0,
      "Autumn" : 0
      }

      #Calculate reservations for each month
      calcMonthStats()

      #Season reservation = The sum of the reservation for each month of
this season
      totalSeasonDict['Winter'] = totalMonthDict['December'] +
totalMonthDict['January'] + totalMonthDict['February']
      totalSeasonDict['Spring'] = totalMonthDict['March'] +
totalMonthDict['April'] + totalMonthDict['May']
      totalSeasonDict['Summer'] = totalMonthDict['June'] +
totalMonthDict['July'] + totalMonthDict['August']
      totalSeasonDict['Autumn'] = totalMonthDict['September'] +
totalMonthDict['October'] + totalMonthDict['November']

      #Pie chart with labels, values, precentages and decorations
      plt.pie(list(totalSeasonDict.values()), startangle=-90,
labels=list(totalSeasonDict.keys()), explode=[0,0,0.1,0], shadow=True,
autopct='%0.0d%%')
      plt.title("Reservations Per Season:")
      #Save Pie chart
      plt.savefig(".seasonStats.png")
      plt.close()
```

Καλεί την calcMonthStats() για να υπολογίσει τις μηνιαίες κρατήσεις.
Αθροίζει τις μηνιαίες κρατήσεις σε εποχιακές κρατήσεις.

Δημιουργεί ένα διάγραμμα πίτας που δείχνει το ποσοστό των κρατήσεων για κάθε εποχή. Ο όρος *'autopct='%0.0d%%'* υπολογίζει αυτόματα τα ποσοστά για να εμφανιστούν στην πίτα.'
Αποθηκεύει το διάγραμμα ως αρχείο εικόνας *'.seasonStats.png'*.

7. **Υπολογισμός Κρατήσεων Ανα Τύπο Δωματείου:**

```
def calcRoomTypeStats():
      roomTypeDict = {
      "A" : 0,
      "B" : 0,
      "C" : 0,
      "D" : 0,
      "E" : 0,
      "F" : 0,
      "G" : 0,
      "H" : 0,
      "L" : 0,
      "P" : 0
      }
      counter = 0
      #Count reservations for each room type:
      for lines in csvFile:
      roomTypeDict[lines['reserved_room_type']] += 1
      counter += 1

      #Pie Chart with values, Labels of the top 3 room types and
decorations
      plt.tight_layout()
      plt.pie(list(roomTypeDict.values()), startangle=90, labels = ["A",
"", "", "D", "E", "", "", "", "", ""], labeldistance=.6)
      # Create labels for the legend with format: label: xx.xx%
      labels = [f'{l}: {round(s/counter*100, 2)}%' for l, s in
zip(list(roomTypeDict.keys()),roomTypeDict.values())]
      plt.legend(shadow=True, labels=labels, bbox_to_anchor=(0.05,0.8),
loc='center right')
      plt.title("Room Type Reservations:")
      #save pie chart
      plt.savefig(".roomTypeStats.png")
      plt.close()
```

8. **Υπολογισμός κρατήσεων που αφορούν οικογένειες, ζευγάρια ή μεμονωμένους ταξιδιώτες:**

```
def calcVisitorTypeStats():
      visitorTypeDict = {
      "Families" : 0,
      "Couples" : 0,
```

```
      "Alone Travelers" : 0
      }

      #count reservations for each visitor type
      for lines in csvFile:
      if lines['adults'] == 1 and lines['children'] == 0:
            visitorTypeDict['Alone Travelers'] += 1
      elif lines['adults'] == 2 and lines['children'] == 0:
            visitorTypeDict['Couples'] += 1
      elif lines['adults'] >= 1 and lines['children'] >= 0:
            visitorTypeDict['Families'] += 1

      #Pie chart with labels, number of reservations, precentage and
decorations
      plt.tight_layout()
      plt.pie(list(visitorTypeDict.values()),labels=[f'{l}\n{s}' for l,
s in zip(list(visitorTypeDict.keys()), visitorTypeDict.values())],
autopct='%1.1f%%', shadow=True, explode=([0.1, 0.1, 0.2]))
      plt.title("Type of Visitors:")
      #save pie chart
      plt.savefig(".visitorTypeStats.png")
      plt.close()
```

Εδώ γίνεται η υπόθεση πώς αν η κράτηση είναι για δύο ενήλικες τότε έχουμε να κάνουμε για ζευγάρι. Αν η κράτηση είναι για έναν ή περισσότερους ενήλικες και τουλάχιστον ένα παιδί ή για πάνω από δύο ενήλικες τότε έχουμε να κάνουμε με οικογένια.

9. **Υπολογισμός Τάσεων με την πάροδο του χρόνου:**

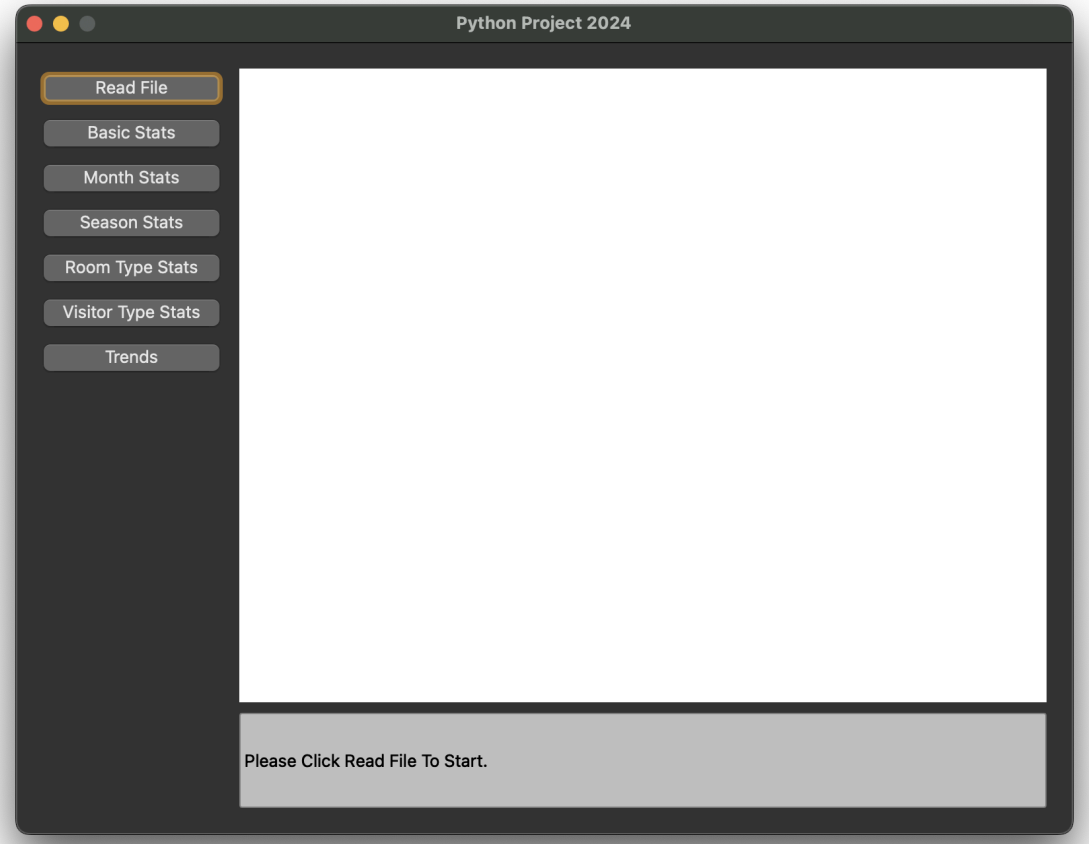
```
def calcTrend():
      #calc total Reservation for each month of each year
      df['reservations'] = 1  #Add a new collumn with the constant 1
      #Make a series with only each month and the total number of
reservations for this month = sum of reservations = 1+1+...+1
      resampledDF = df.resample('ME').sum()['reservations']
      #x = [1, 2, ..., 26]
      x = range(26)
      y = resampledDF.tolist()

      #Calculate Linear Regression
      slope, intercept, r_value, p_value, std_err = linregress(x,y)
      regression_line = slope * x + intercept #Array with values of the
trend line for each month

      #Make type series to type dataFrame
      resampledDF = resampledDF.to_frame()

      #Add collumn 'Trend' with values of trend line
```

```
    resampledDF['Trend'] = regression_line
    #Plot reservations per month AND the trend line
    resampledDF.plot()
    #Plot Decorations
    plt.grid()
    plt.legend()
    plt.title("Trend of Reservations over time:")
    plt.tight_layout()
    #save plot
    plt.savefig(".trend.png")
    plt.close()
```

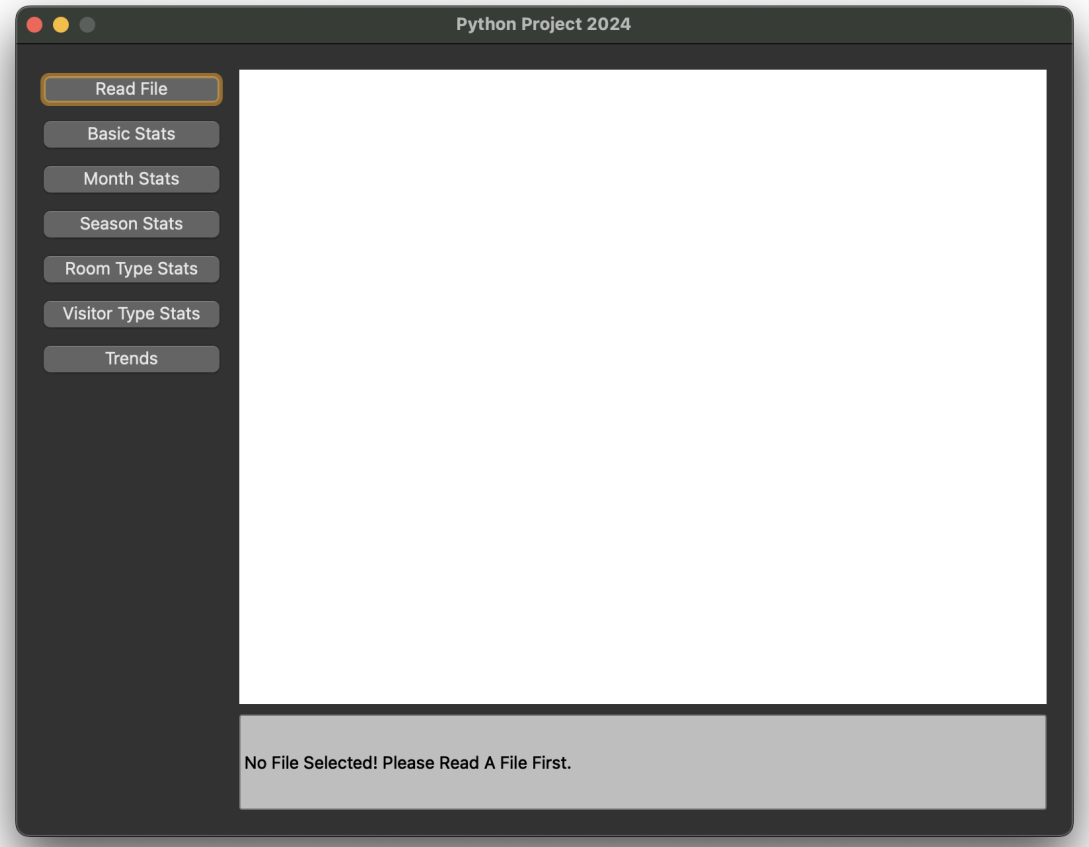
Προσθέτει μια στήλη κρατήσεων στο DataFrame *df* με σταθερή τιμή 1 για σκοπούς μέτρησης.
Κανει resample τα δεδομένα ανά τέλος μήνα και αθροίζει τις κρατήσεις για να λάβει μηνιαίες μετρήσεις κρατήσεων στο *resampledDF.*
Εκτελεί τον αλγόριθμο γραμμικής παλινδρόμησης στις μηνιαίες μετρήσεις κρατήσεων για τον υπολογισμό της γραμμής τάσης.
Δημιουργεί ένα διάγραμμα που δείχνει τις μηνιαίες κρατήσεις και τη γραμμή τάσης.
Αποθηκεύει τη γραφική παράσταση ως αρχείο εικόνας *'.trend.png'*.

Screenshots παραδειγμάτων

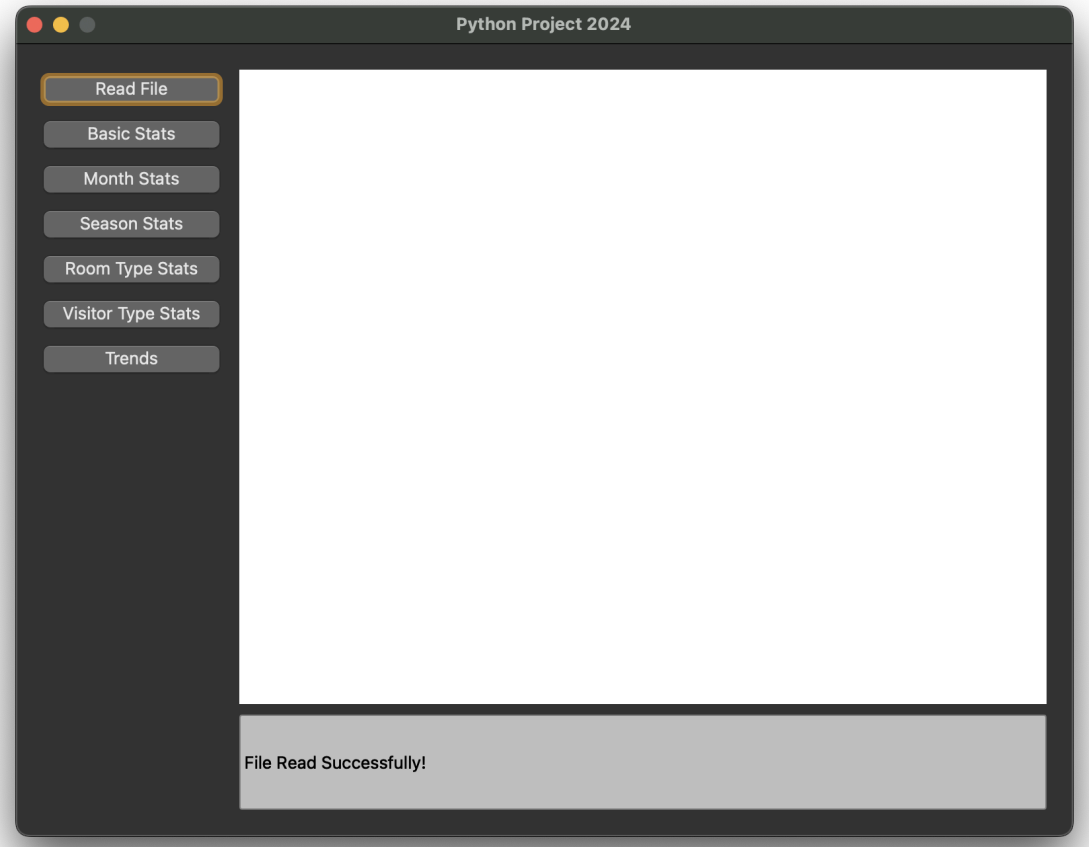
Η εφαρμογή μόλις έχει ανοίξει:



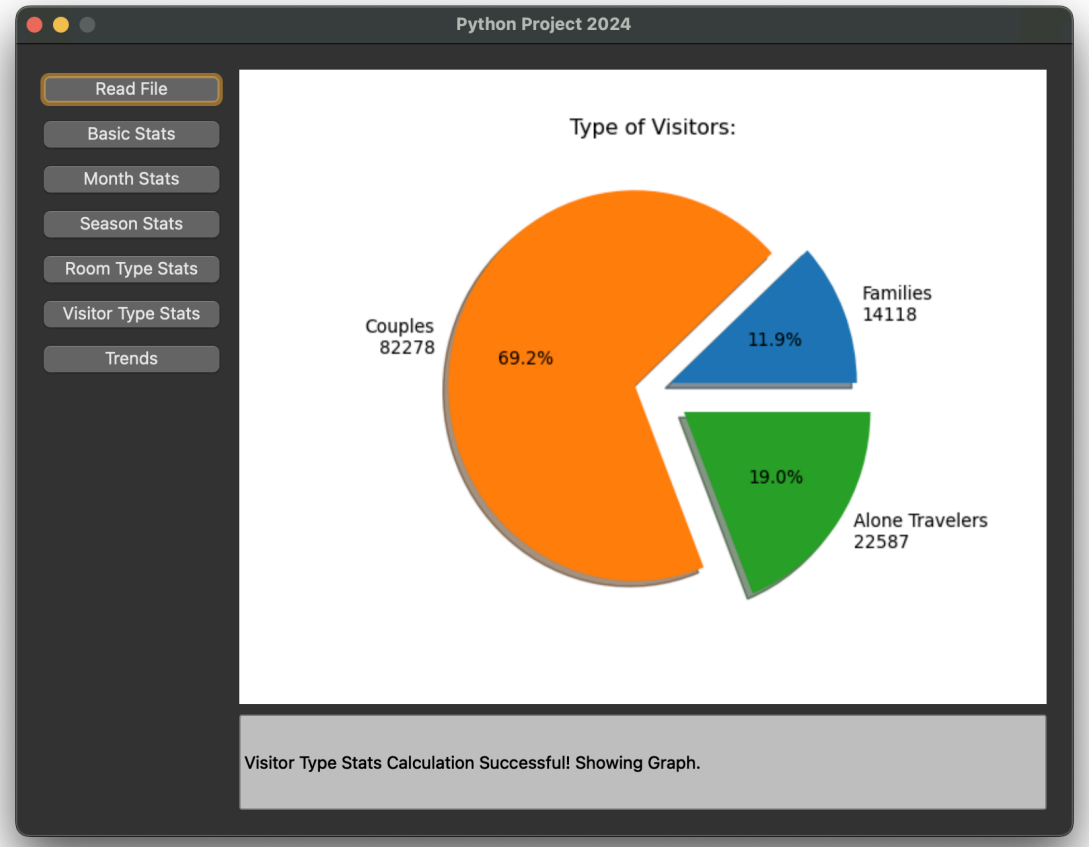
Η κονσόλα εξόδου αν επιλεγεί μια ενέργεια χωρίς να έχει πατηθεί το κουμπί Read File:



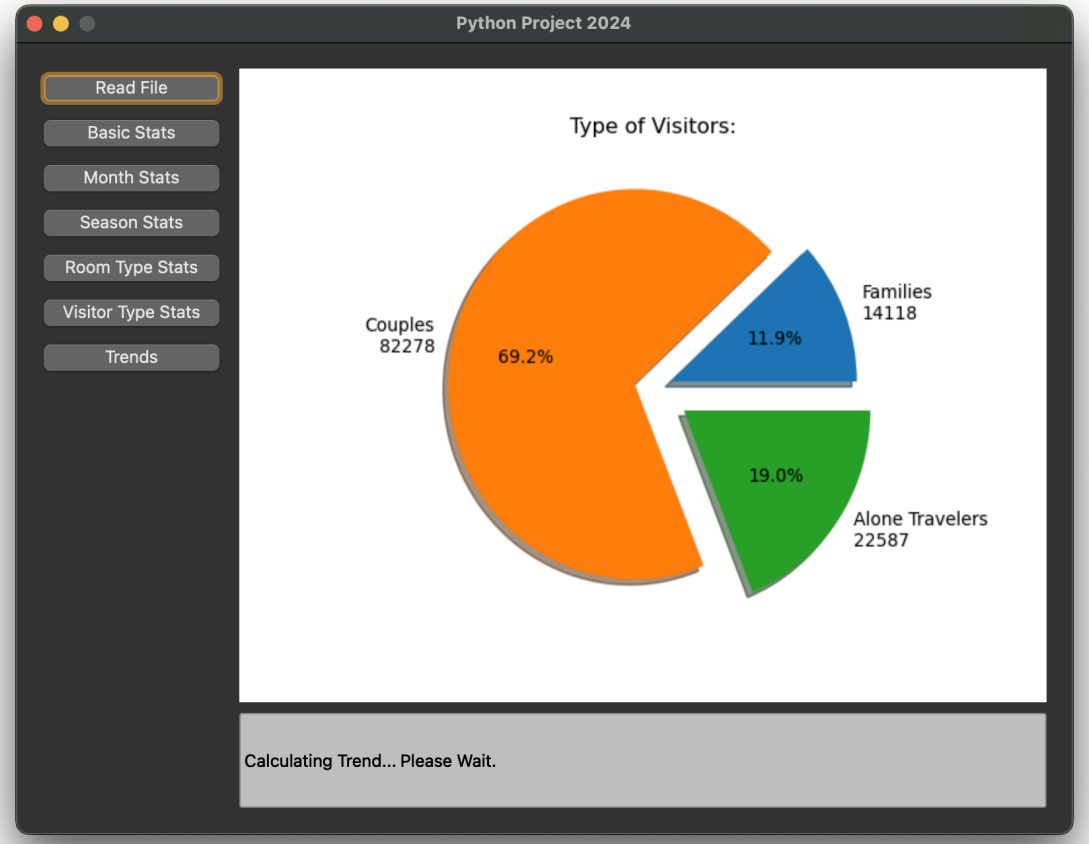
Η κονσόλα εξόδου οταν πατηθεί το κουμπί Read File:



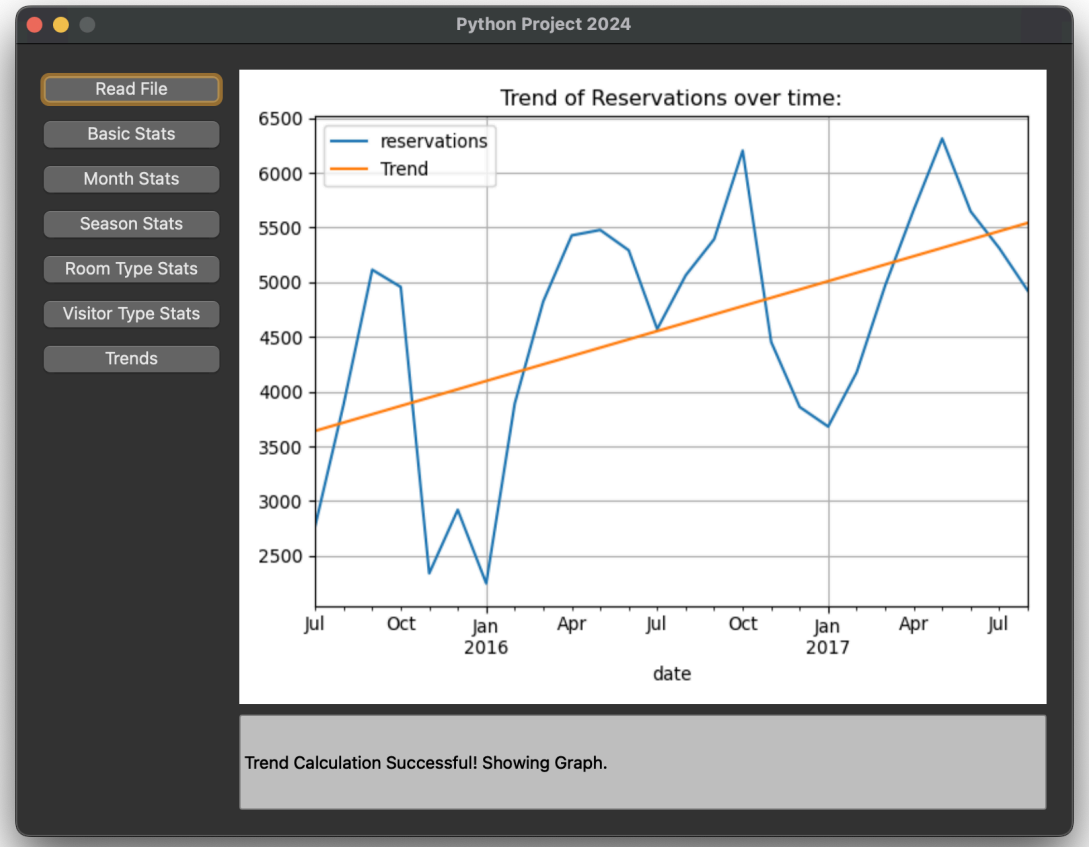
Η κονσόλα εξόδου και η εικόνα εξόδου οταν πατηθεί το κουμπί Visitor Type Stats:



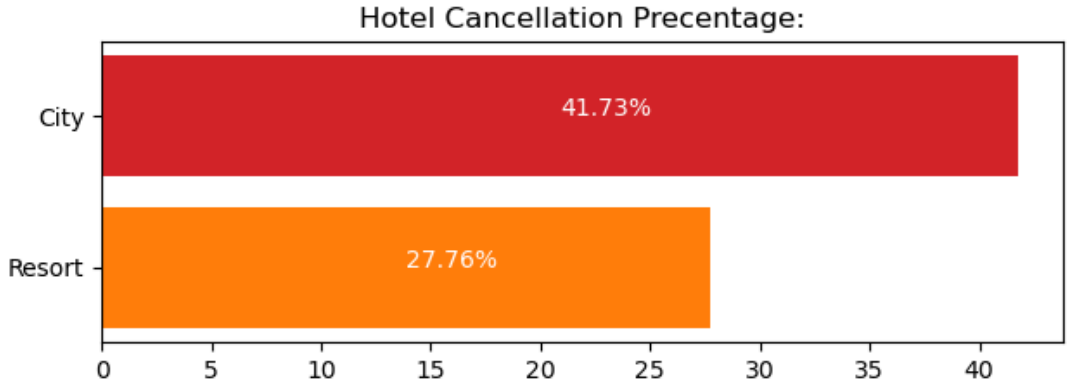
Η κονσόλα εξόδου όταν ύστερα από το Visitor Type Stats πατηθεί το κουμπί Trends. Η εφαρμογή παραμένει έτσι μέχρι να ολοκληρωθούν οι υπολογισμοί και για αυτό η κονσόλα εξόδου αναγράφει: *“Calculating Trend… Please Wait.”*

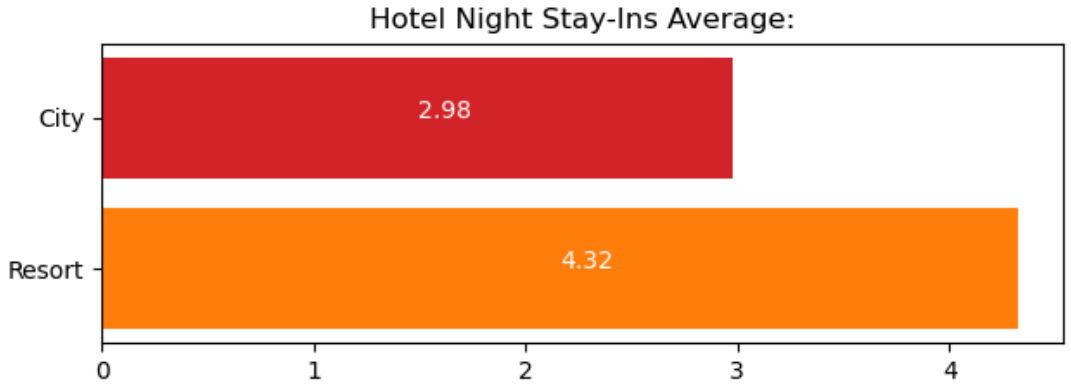


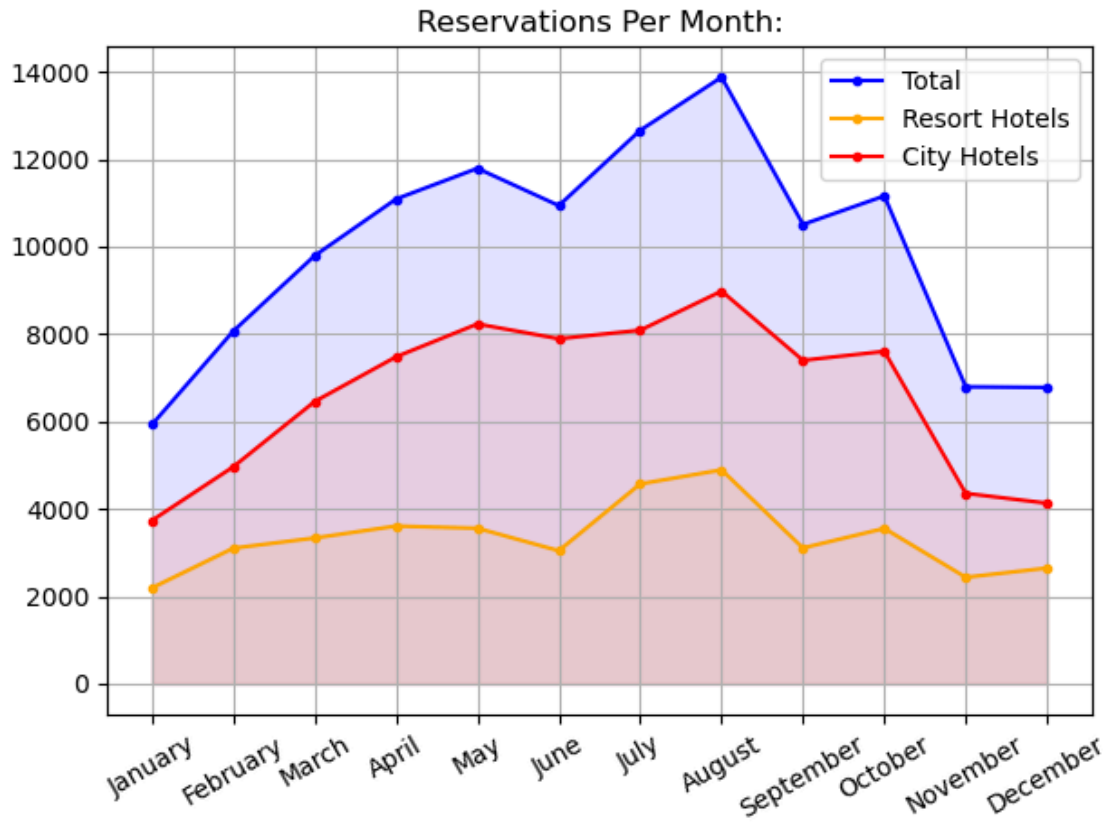
Μόλις ολοκληρωθούν οι υπολογισμοί, ενημερώνεται η εικόνα εξόδου και η κονσόλα εξόδου:

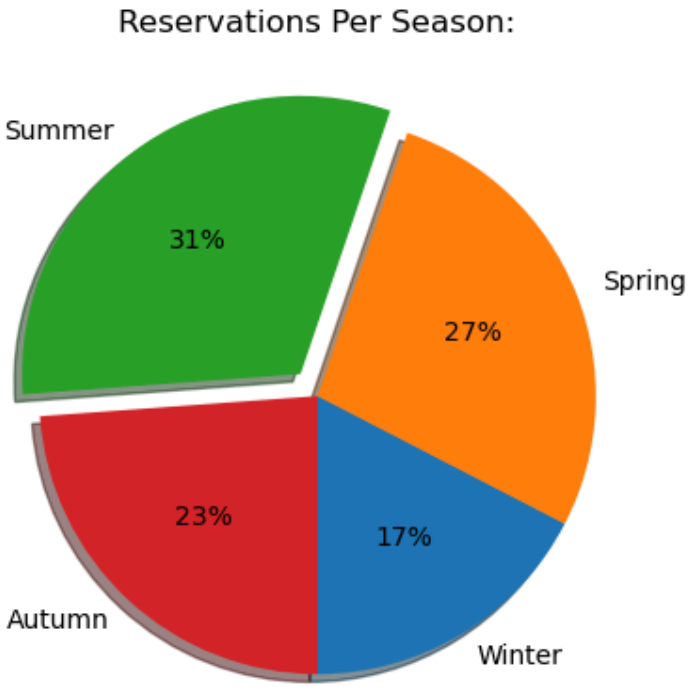


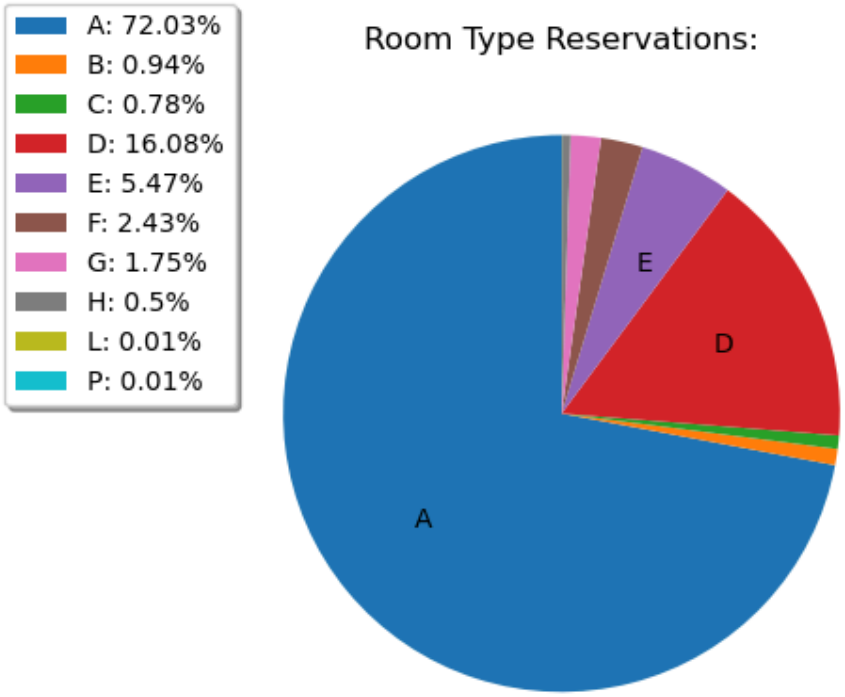
Τα ζητούμενα γραφήματα

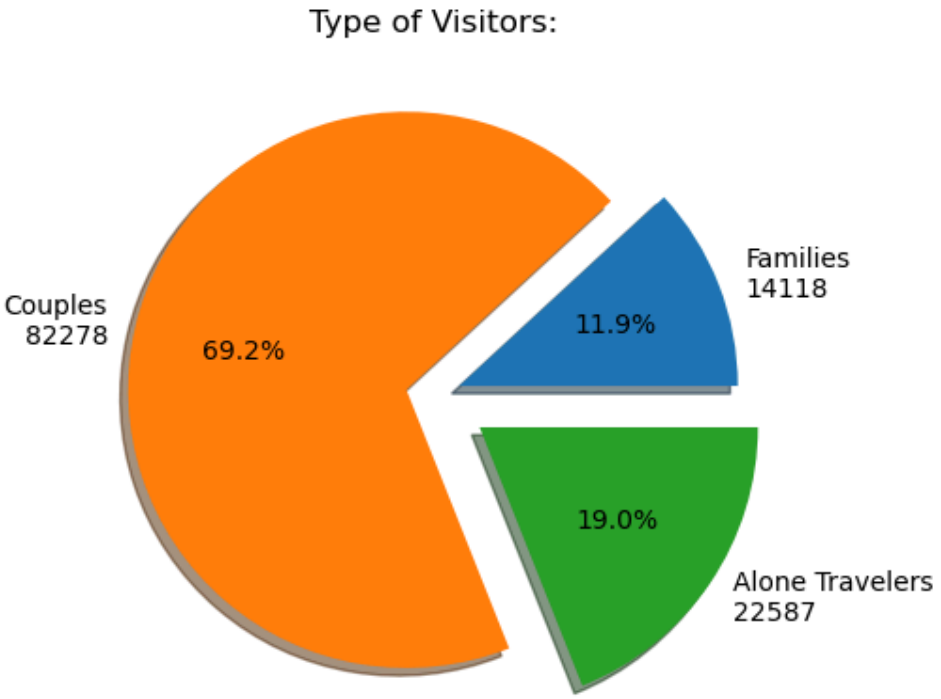


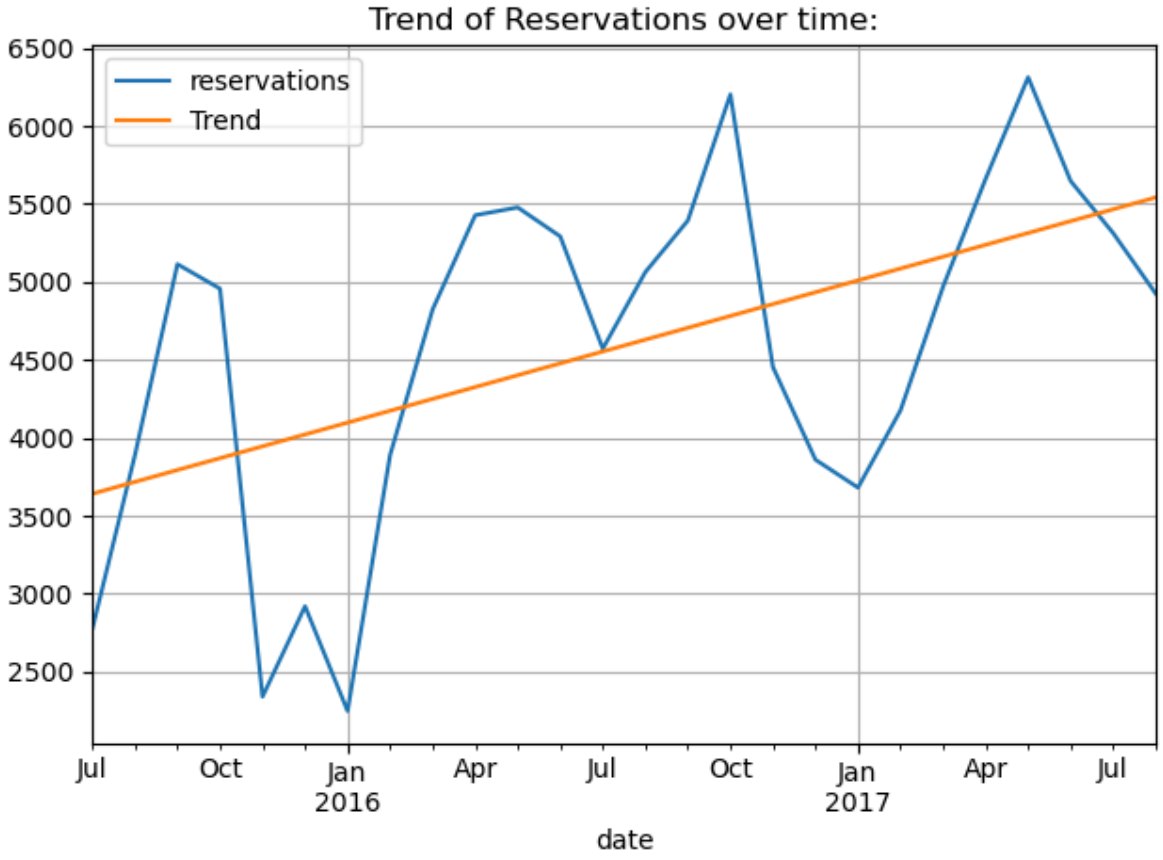





Reservations Per Season:
Summer
31%
Spring
27%
23%
17%
Autumn
Winter


A: 72.03%
B: 0.94%
C: 0.78%
D: 16.08%
E: 5.47%
F: 2.43%
G: 1.75%
H: 0.5%
L: 0.01%
P: 0.01%
Room Type Reservations:
E
D
A


Type of Visitors:
Couples
82278
69.2%
Families
14118
11.9%
19.0%
Alone Travelers
22587


Trend of Reservations over time:
reservations
Trend
6500
6000
5500
5000
4500
4000
3500
3000
2500
Jul
Oct
Jan
2016
Apr
Jul
Oct
Jan
2017
Apr
Jul
date

Σχόλια - Παραδοχές

Η εφαρμογή έχει υλοποιηθεί σε Python version: 3.12.3.
Έχει δοκιμαστεί επιτυχώς σε Linux Ubuntu 20.04.6 και Mac Os Sonoma 14.2.1

Για εκτέλεση:

python main.py

Το αρχείο των δεδομένων θα πρέπει να έχει το όνομα *hotel_booking.csv* και να βρίσκεται στο working directory απο το οποίο καλούμε την εφαρμογή.

Επειδή τα γραφήματα αποθηκεύονται στο working directory πριν εμφανιστούν στην οθόνη, θα πρέπει ο χρήστης που καλεί την εφαρμογή να έχει read και write δικαιώματα στο working directory, αλλιώς δεν θα εμφανιστούν τα γραφήματα.

Για την υλοποίηση της εφαρμογής έχουν χρησιμοποιήθει οι βιβλιοθήκες:

sys: v3.12.3,

PySide6: v6.7.0
https://doc.qt.io/qtforpython-6/quickstart.html
Install with: pip install pyside6

Matplotlib: v3.8.4
https://matplotlib.org/
Install with: pip install matplotlib

Pandas: v2.2.2
https://pandas.pydata.org/
Install with: pip install pandas

Numpy: v1.26.4
https://numpy.org/
Install with: pip install numpy

Scipy: v1.13.0
https://scipy.org/
Install with: pip install scipy

Για παλαιότερες εκδόσεις αυτών των βιβλιοθηκών μπορει να υπάρχουν προβλήματα.