European Countries

Project Description

Johanna Teichmann (18-625-483)

Katharina Stimpfl (16-169-199)

This project was developed as part of the course "Introduction to Programming" under the supervision of Dr. Mario Silic at the University of St. Gallen.

1. **Introduction**

Europe is not really a continent in its own right, because it is connected to Asia, and yet historically and culturally it can be seen as a continent in its own right. With a surface area of 10.5 million square kilometres, on which more than 700 million people live, Europe is one of the most densely populated places on earth. In the many different countries of Europe, a large number of languages are spoken, religions followed and other different cultural characteristics of all kinds cultivated (“Europe in general”, n.d.).

1. **What the program does**

The program presented here aims to provide more detailed information about these European countries. It allows to obtain specific data on one or more countries considered to be part of the European continent. Information such as capitol, area, population, population density, GDP and GDP per capita are available for each country. In addition, the program classifies all countries into geographical groups so that the user can easily see whether the country belongs to Northern, Eastern, Southeastern, Southern, Western or Middle Europe.

1. **How does it do it**

The User Interface was created with “Qt-Creator”, which is written in C++, so we have to translate the code for representing the different Labels, Buttons and Browsers into python code using modules like “qtpy” and “pyqt”. This is done by the python file “build.py”, which has to be compiled first.

The UI has some simple widgets like “comboBoxes”, “TextLabels”, or “TextBrowser” which are used for interaction between the user and the code running the application.

First we import the "Panda Module". It offers special functions and data structures for the manipulation of numerical tables.

Then we import data from our Excel file "GeographicalEurope" (see Appendix Picture 1).

The data is used to fill a “DataFrame”( <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html>), which is a specific structure offered by the python module “pandas”.

In order to facilitate the handling of the countries, we have assigned shorter country codes to them, so we know their index in the “DataFrame”.

Using this structure, we are able to acces the specific data. Therefore, the MainWindow() class which is representing the MainWindow, has functions like “onOKclicked()” which is reading the selected choice from the comboBox in the MainWindow and then returning the data connected to the Country of choice while using the “at”-function(Member of DataFrame) and our “country\_codes”-dictionary

This data is then shown in the “textBrowser” below.

The “Cancel”-Button will clear the TextBrowser, while the “Close”-Button closes the application.

1. **How to use it**

To use the program one has to download an IDE first, we recommend “Pycharm” from JetBrains.

If you are familiar with “GitHub”, you can easily import the code into your “IDE”.

Then you will have to compile the build.py file first and then the main.py file, so you can run the application.

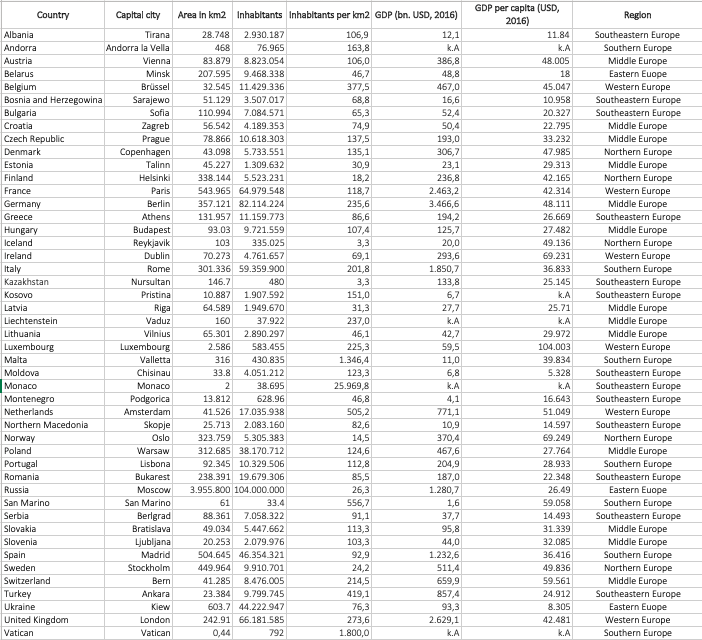
When the application is opened the user can easily choose what country he is interested in and what specific data he would like to be shown. There is also the possibility to get all the data regarding one specific country by clicking on “all data”.

1. **List of References**

Europe in general. (n.d.) Retrieved May 1, 2019, from http://www.europa-portal.info

1. **Appendix**

Picture 1 - Excel document, which serves as data base for the Python-program "European Countries".



On the following pages, please find the code of the Program “European Countries”:

#importing relevant modules  
import sys  
import pandas as pd #creating shortcut to panda module via nametag pd  
from qtpy import QtWidgets #using the qt-creator  
from ui.mainwindow import Ui\_MainWindow  
  
app = QtWidgets.QApplication(sys.argv)  
  
#loading data for operations and storing it in lists  
file = open("data/GeographicalEurope.csv", "r", newline="") #opens the file GeographicalEurope  
first = file.readline().strip().split(";") #get the first line from file  
  
index=[] #we need index for DataFrame structure(see below)  
for i in range(0, 47):  
 index.append(i)  
  
countries =[]  
capitol =[]  
space =[]  
inhabit =[]  
inhabits\_per\_km =[]  
bip =[]  
bip\_per\_capita =[]  
region =[]  
  
for line in file:  
 countries.append(line.strip().split(";")[0])  
 capitol.append(line.strip().split(";")[1])  
 space.append(line.strip().split(";")[2])  
 inhabit.append(line.strip().split(";")[3])  
 inhabits\_per\_km.append(line.strip().split(";")[4])  
 bip.append(line.strip().split(";")[5])  
 bip\_per\_capita.append(line.strip().split(";")[6])  
 region.append(line.strip().split(";")[7])  
  
dp = pd.DataFrame({first[0]: countries,  
 first[1]: capitol,  
 first[2]: space,  
 first[3]: inhabit,  
 first[4]: inhabits\_per\_km,  
 first[5]: bip,  
 first[6]: bip\_per\_capita,  
 first[7]: region},  
 index)  
#country codes for easier handeling when operating on DataFrame  
country\_codes ={"Albania" : 0,  
 "Andorra" : 1,  
 "Austria" : 2,  
 "Belarus" : 3,  
 "Belgium" : 4,  
 "Bosnia and Herzegowina" : 5,  
 "Bulgaria" : 6,  
 "Croatia" : 7,  
 "Czech Republic" : 8,  
 "Denmark" : 9,  
 "Estonia" : 10,  
 "Finland" : 11,  
 "France" : 12,  
 "Germany" : 13,  
 "Greece" : 14,  
 "Hungary" : 15,  
 "Iceland" : 16,  
 "Ireland" : 17,  
 "Italy" : 18,  
 "Kazakhstan" : 19,  
 "Kosovo" : 20,  
 "Latvia" : 21,  
 "Liechtenstein" : 22,  
 "Lithuania" : 23,  
 "Luxembourg" : 24,  
 "Malta" : 25,  
 "Moldova" : 26,  
 "Monaco" : 27,  
 "Montenegro" : 28,  
 "Netherlands" : 29,  
 "Northern Macedonia" : 30,  
 "Norway" : 31,  
 "Poland" : 32,  
 "Portugal " : 33,  
 "Romania" : 34,  
 "Russia" : 35,  
 "San Marino" : 36,  
 "Serbia" : 37,  
 "Slovakia" : 38,  
 "Slovenia" : 39,  
 "Spain" : 40,  
 "Sweden" : 41,  
 "Switzerland" : 42,  
 "Turkey" : 43,  
 "Ukraine" : 44,  
 "United Kingdom" : 45,  
 "Vatican" : 46,}  
file.close()  
  
#class MainWindow, setting up the ui (userinterface)  
class MainWindow(QtWidgets.QMainWindow):  
 def \_\_init\_\_(self, parent = None):  
 super().\_\_init\_\_(parent)  
  
 self.setWindowTitle("Wikipedia")  
  
 self.ui = Ui\_MainWindow()  
 self.ui.setupUi(self)  
  
 self.ui.eigenschaft\_combo.addItem("All Data")  
 #routine for filling combo boxes with choices for data  
 with open("data/GeographicalEurope.csv", "r", newline="") as file:  
 first\_line = file.readline().strip().split(";")  
 for choice in first\_line:  
 if choice == "ï»¿Country": #we need to skip the first Item, as its not useful  
 continue  
 self.ui.eigenschaft\_combo.addItem(choice)  
  
 #routine for filling combo boxes with choices of countries  
 with open("data/GeographicalEurope.csv", "r", newline="") as file:  
 file.readline()  
 for line in file:  
 line\_splitted = line.strip().split(";")  
 self.ui.land\_combo.addItem(line\_splitted[0])  
  
 #connecting the signals to slots, so their functionality is provided  
 self.ui.ok.clicked.connect(self.onOKclicked)  
 self.ui.abbr.clicked.connect(self.onAbbrclicked)  
 self.ui.close\_button.clicked.connect(self.onCloseclicked)  
  
 def onOKclicked(self): #shows data in textBrowser  
 country = self.ui.land\_combo.currentText()  
 self.ui.ausgabe\_label.setText(country +":")  
 attr = self.ui.eigenschaft\_combo.currentText() #attr = attribut  
 self.ui.label\_2.setText(attr +":")  
  
 if attr == "All Data":  
 self.ui.label\_2.setText("Capitol, Area in km^2, Inhabitants, Inhabitants per km^2, GDP (bn USD, 2016), GDP per capita (bn USD, 2016), Region:")  
 self.ui.textBrowser.setPlainText(dp.at[country\_codes[country], "Capitol"])  
 self.ui.textBrowser.append(dp.at[country\_codes[country], "Area in km2"])  
 self.ui.textBrowser.append(dp.at[country\_codes[country], "Inhabitants"])  
 self.ui.textBrowser.append(dp.at[country\_codes[country], "Inhabitants per km2"])  
 self.ui.textBrowser.append(dp.at[country\_codes[country], "GDP (bn USD, 2016)"])  
 self.ui.textBrowser.append(dp.at[country\_codes[country], "GDP per capita (bn USD, 2016)"])  
 self.ui.textBrowser.append(dp.at[country\_codes[country], "Region"])  
  
  
 elif attr == "Inhabitants":  
 self.ui.textBrowser.setPlainText(dp.at[country\_codes[country], attr])  
 elif attr == "Area in km2":  
 self.ui.textBrowser.setPlainText(dp.at[country\_codes[country], attr])  
 elif attr == "Capitol":  
 self.ui.textBrowser.setPlainText(dp.at[country\_codes[country], attr])  
 elif attr == "Inhabitants per km2":  
 self.ui.textBrowser.setPlainText(dp.at[country\_codes[country], attr])  
 elif attr == "GDP (bn USD, 2016)":  
 self.ui.textBrowser.setPlainText(dp.at[country\_codes[country], attr])  
 elif attr == "GDP per capita (bn USD, 2016)":  
 self.ui.textBrowser.setPlainText(dp.at[country\_codes[country], attr])  
 elif attr == "Region":  
 self.ui.textBrowser.setPlainText(dp.at[country\_codes[country], attr])  
  
 def onAbbrclicked(self): #clears textBrowser  
 self.ui.ausgabe\_label.clear()  
 self.ui.label\_2.clear()  
 self.ui.textBrowser.clear()  
  
 def onCloseclicked(self): #closes application  
 self.close()  
  
window = MainWindow()  
window.show()  
sys.exit(app.exec\_())