Canada provinces and cities

Data Analysis performed by
Joel Sandé

Preface

- Data of theses stats have been drawn from:

http://www.downloadexcelfiles.com/wo_en/download-list-cities-canada#.WpmQS-jOWM8

For more information, please visit :
www.downloadexcelfiles.com

Preface

- I want this powerpoint, highlight, from a practical case, the power of analysis of the Python language and its ability to generate results in visual form.

- I will leave a link to download the csv file from where the data is drawn. The code will be available upon request from sandejoel@yahoo.ca

(Free if you are a student taking my course, or a government representative).

Overview

- Introduction
- Analysis Methodology
- Small requests
- Complex requests
- Conclusion

Analysis Methodology

- By habit, I like to make small queries (Functions) for warm-up: So we will start with this one.

- In general, when these queries are established, for the future, when we have to do with larger queries, just go search them one by one or even combine them to facilitate the task during a large query .

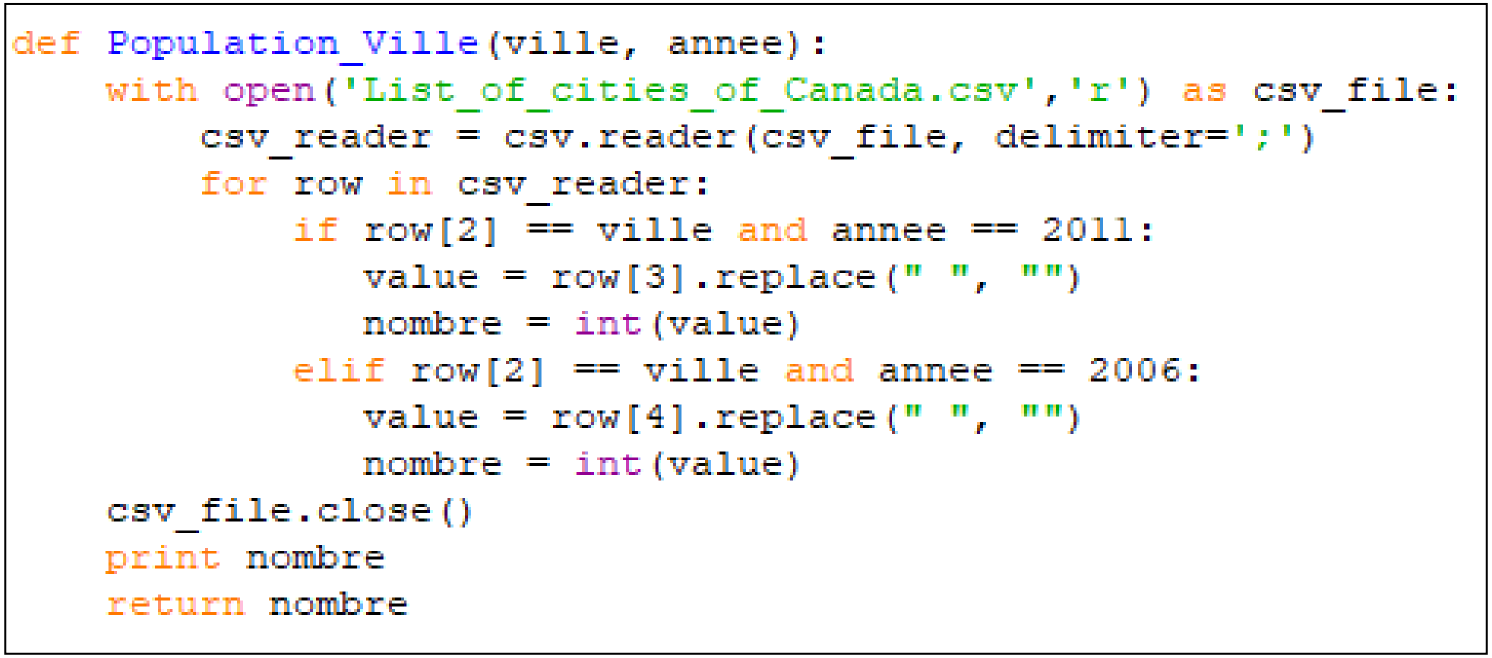
- It becomes child's play, and the code is easier to maintain. We will see in the end a case where this is not necessary.

- Here we go …

Canada's 10 Most Populated Cities in 2011 with their respective populations.

Small Requests

The population of a city for a given year



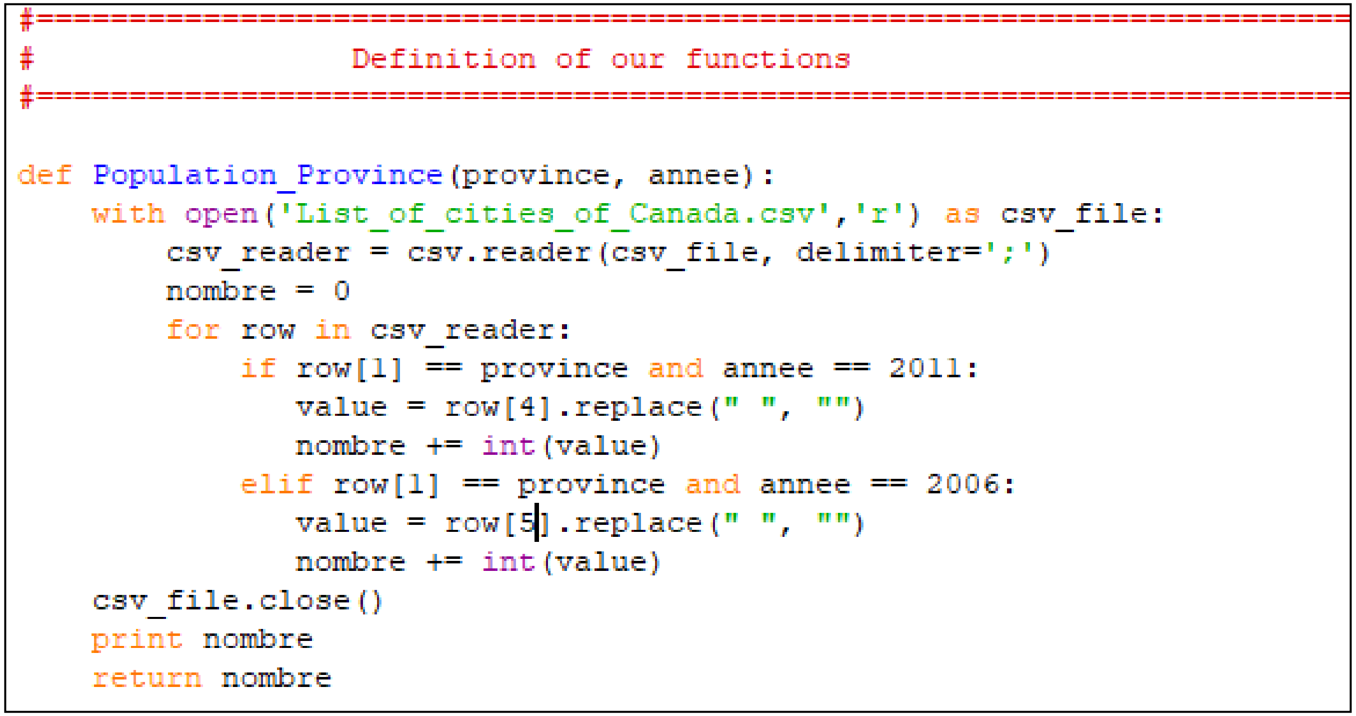
```
def Population_Ville(ville, annee):
    with open('List_of_cities_of_Canada.csv','r') as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=';')
        for row in csv_reader:
            if row[2] == ville and annee == 2011:
               value = row[3].replace(" ", "")
               nombre = int(value)
            elif row[2] == ville and annee == 2006:
               value = row[4].replace(" ", "")
               nombre = int(value)
    csv_file.close()
    print nombre
    return nombre
```

Population_Ville('Calgary', 2006)

The python`s answer is 988812

Petite Requête

The population of a province for a given year

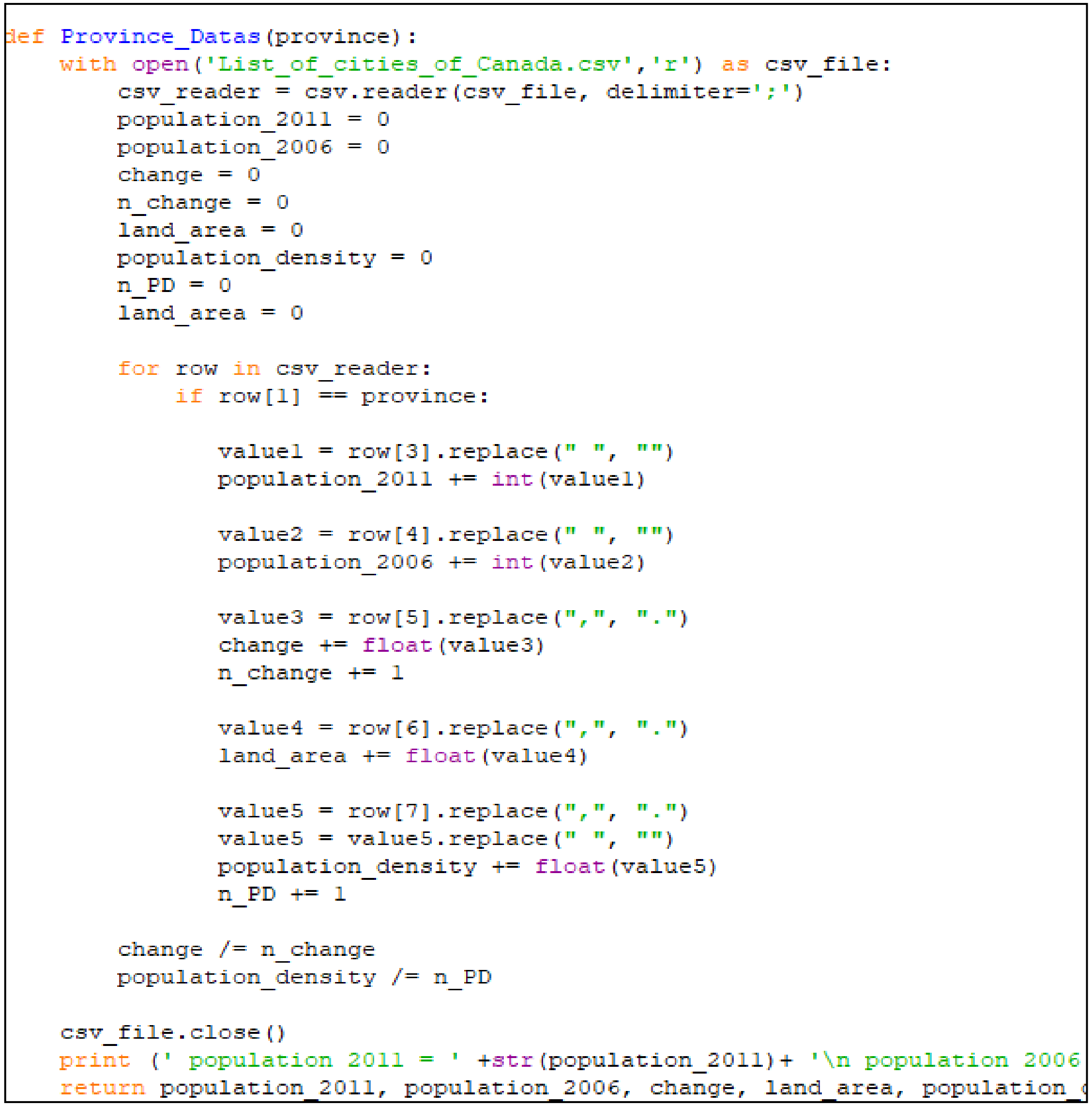


```
#=======================================================================
#                 Definition of our functions
#=======================================================================

def Population_Province(province, annee):
    with open('List_of_cities_of_Canada.csv','r') as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=';')
        nombre = 0
        for row in csv_reader:
            if row[1] == province and annee == 2011:
               value = row[4].replace(" ", "")
               nombre += int(value)
            elif row[1] == province and annee == 2006:
               value = row[5].replace(" ", "")
               nombre += int(value)
    csv_file.close()
    print nombre
    return nombre
```

Population_Province('Alberta', 2011)

The python`s answer is 2207686

Petite Requête



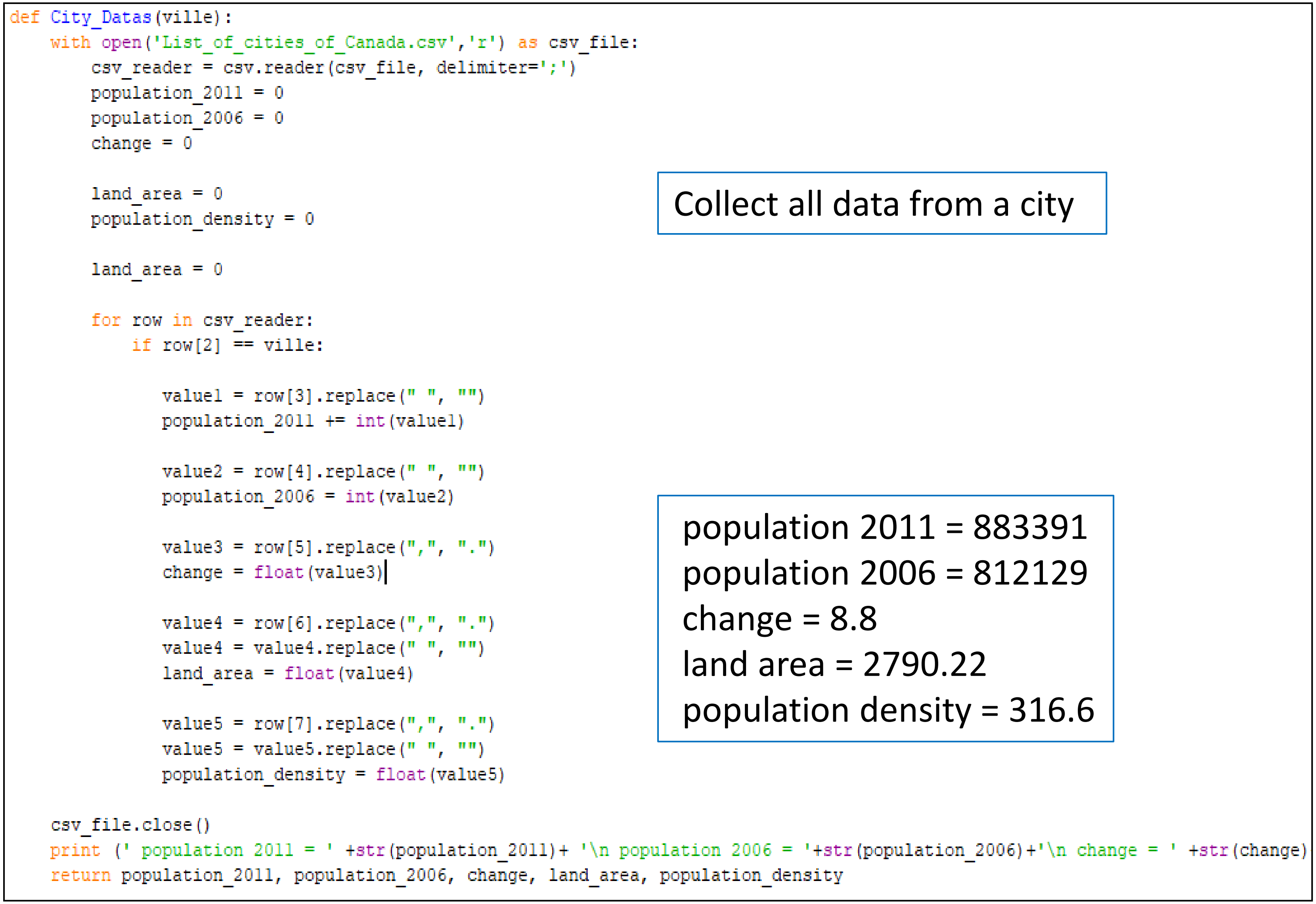
```
def Province_Datas(province):
    with open('List_of_cities_of_Canada.csv','r') as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=';')
        population_2011 = 0
        population_2006 = 0
        change = 0
        n_change = 0
        land_area = 0
        population_density = 0
        n_PD = 0
        land_area = 0

        for row in csv_reader:
            if row[1] == province:

                value1 = row[3].replace(" ", "")
                population_2011 += int(value1)

                value2 = row[4].replace(" ", "")
                population_2006 += int(value2)

                value3 = row[5].replace(",", ".")
                change += float(value3)
                n_change += 1

                value4 = row[6].replace(",", ".")
                land_area += float(value4)

                value5 = row[7].replace(",", ".")
                value5 = value5.replace(" ", "")
                population_density += float(value5)
                n_PD += 1

        change /= n_change
        population_density /= n_PD

    csv_file.close()
    print (' population 2011 = ' +str(population_2011)+ '\n population 2006
    return population_2011, population_2006, change, land_area, population_
```

Collect all data from a province

Province_Datas('Alberta')

```
population 2011 = 2473572
population 2006 = 2207686
change = 18.7166666667
land area = 2335.86
population density =
756.622222222
```

Small requests

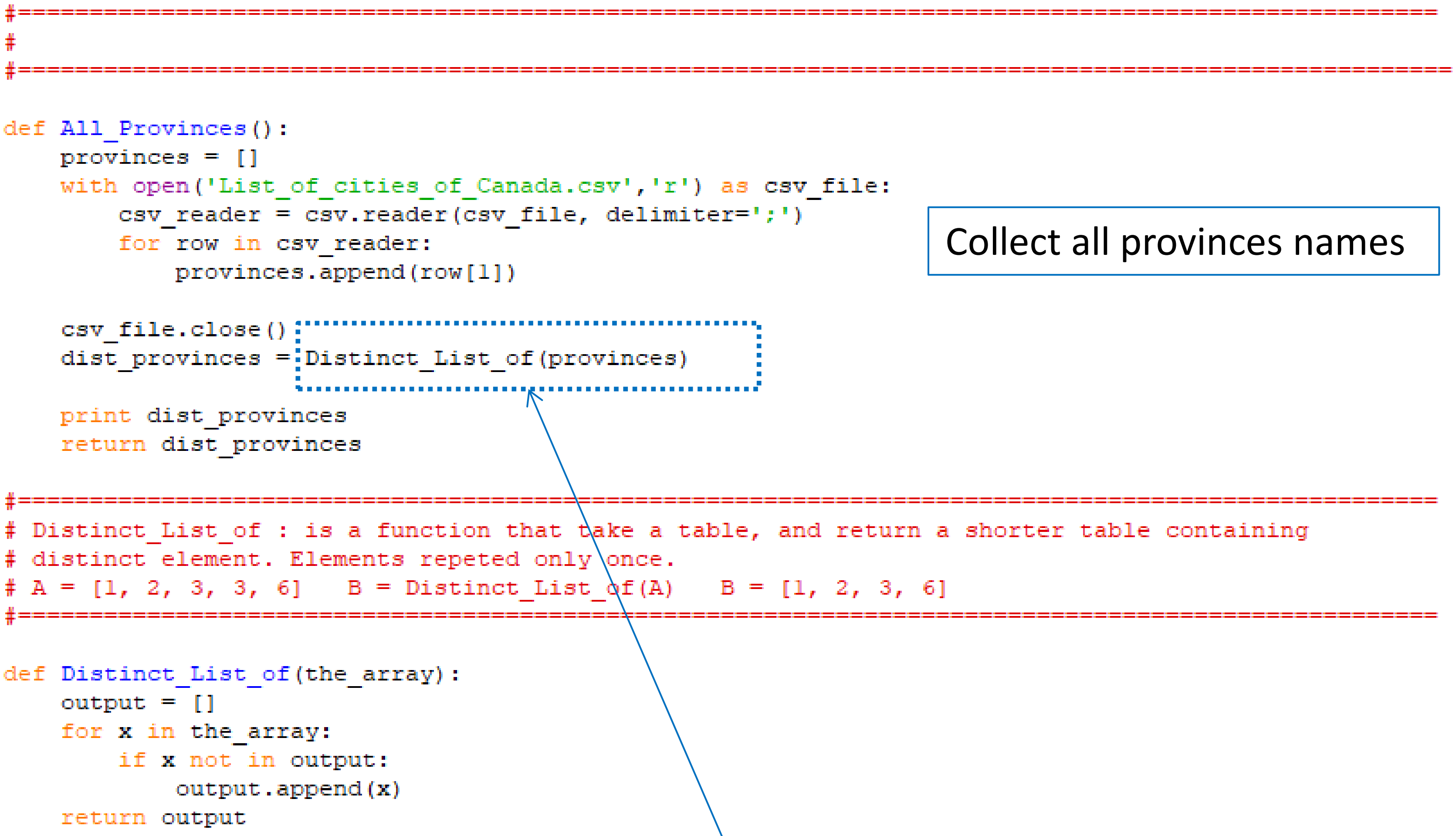


```
def City_Datas(ville):
    with open('List_of_cities_of_Canada.csv','r') as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=';')
        population_2011 = 0
        population_2006 = 0
        change = 0

        land_area = 0
        population_density = 0

        land_area = 0

        for row in csv_reader:
            if row[2] == ville:

               value1 = row[3].replace(" ", "")
               population_2011 += int(value1)

               value2 = row[4].replace(" ", "")
               population_2006 = int(value2)

               value3 = row[5].replace(",", ".")
               change = float(value3)

               value4 = row[6].replace(",", ".")
               value4 = value4.replace(" ", "")
               land_area = float(value4)

               value5 = row[7].replace(",", ".")
               value5 = value5.replace(" ", "")
               population_density = float(value5)

    csv_file.close()
    print (' population 2011 = ' +str(population_2011)+ '\n population 2006 = '+str(population_2006)+'\n change = ' +str(change)
    return population_2011, population_2006, change, land_area, population_density
```

Collect all data from a city

population 2011 = 883391
population 2006 = 812129
change = 8.8
land area = 2790.22
population density = 316.6

Petite Requête



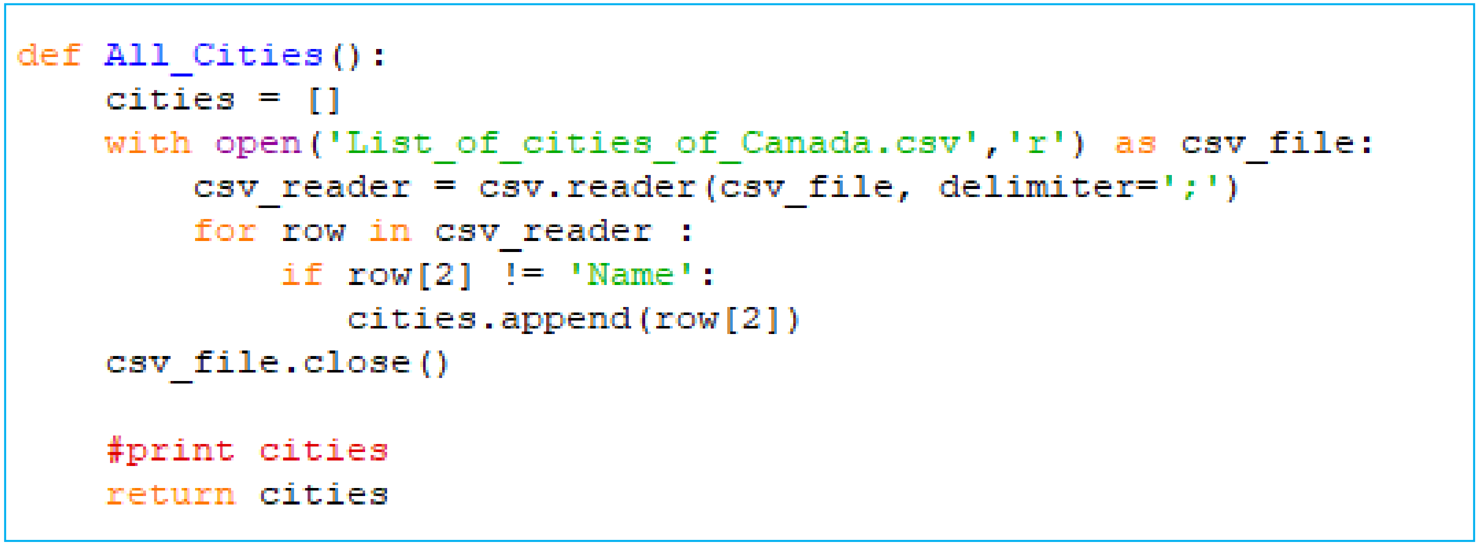
```
#==================================================================================================
#
#====================================================================================================

def All_Provinces():
    provinces = []
    with open('List_of_cities_of_Canada.csv','r') as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=';')
        for row in csv_reader:
            provinces.append(row[1])

    csv_file.close()
    dist_provinces = Distinct_List_of(provinces)

    print dist_provinces
    return dist_provinces

#==================================================================================================
# Distinct_List_of : is a function that take a table, and return a shorter table containing
# distinct element. Elements repeted only once.
# A = [1, 2, 3, 3, 6]    B = Distinct_List_of(A)    B = [1, 2, 3, 6]
#==================================================================================================

def Distinct_List_of(the_array):
    output = []
    for x in the_array:
        if x not in output:
            output.append(x)
    return output
```

Collect all provinces names

Small function that serves me to make a separate selection.
NB: This function is not necessary in the recovery of the names of cities because there is no repetition

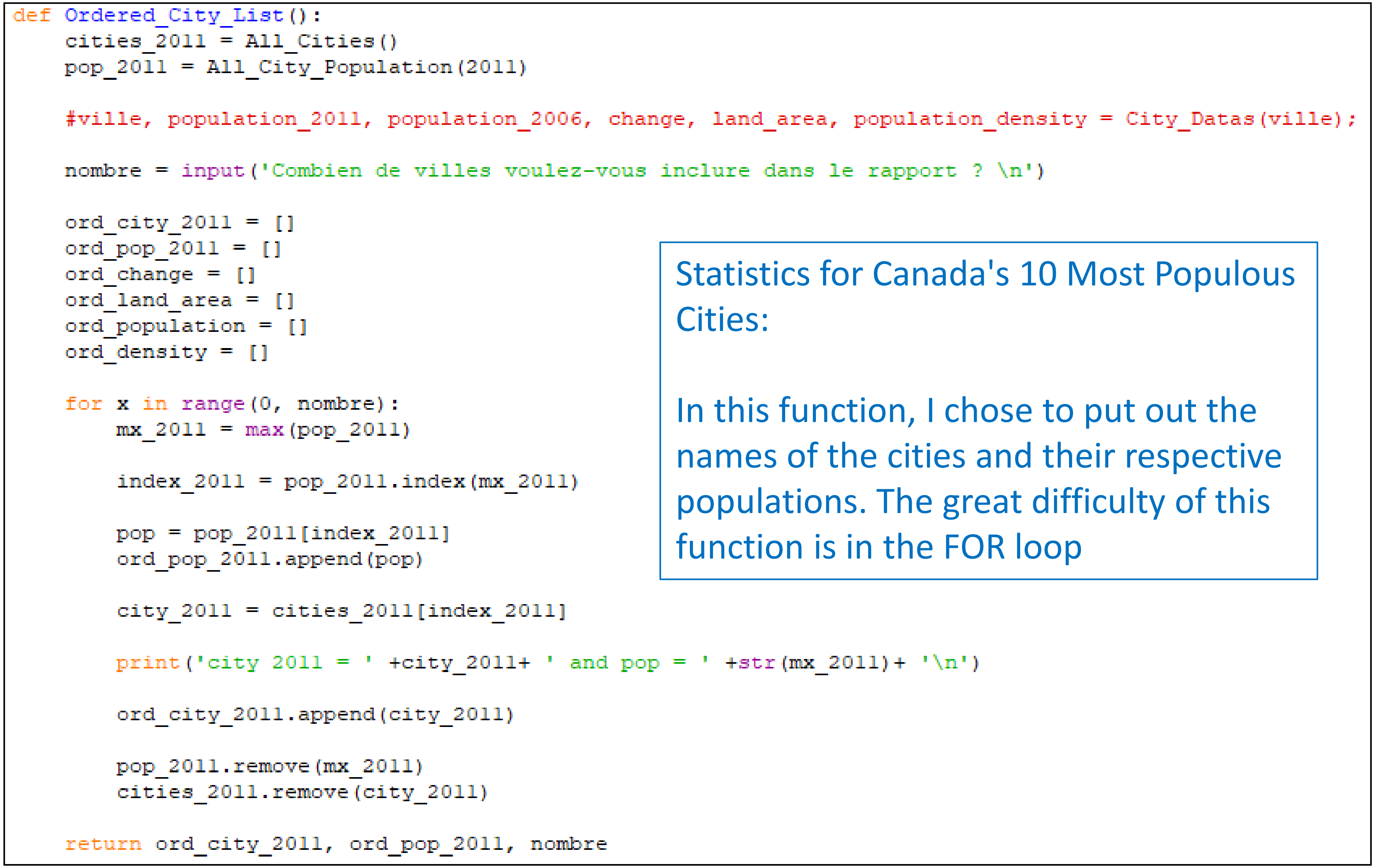
Petite Requête

Collect all cities names



```
def All_Cities():
    cities = []
    with open('List_of_cities_of_Canada.csv','r') as csv_file:
        csv_reader = csv.reader(csv_file, delimiter=';')
        for row in csv_reader :
            if row[2] != 'Name':
                cities.append(row[2])
    csv_file.close()

    #print cities
    return cities
```

Grosses Requêtes

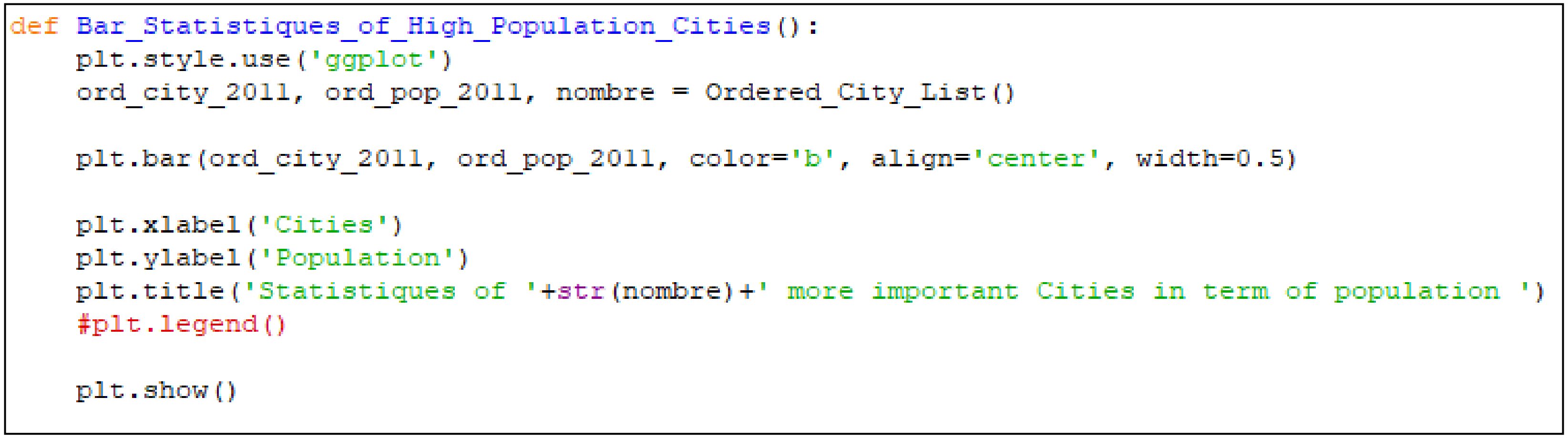


```
def Ordered_City_List():
    cities_2011 = All_Cities()
    pop_2011 = All_City_Population(2011)

    #ville, population_2011, population_2006, change, land_area, population_density = City_Datas(ville);

    nombre = input('Combien de villes voulez-vous inclure dans le rapport ? \n')

    ord_city_2011 = []
    ord_pop_2011 = []
    ord_change = []
    ord_land_area = []
    ord_population = []
    ord_density = []

    for x in range(0, nombre):
        mx_2011 = max(pop_2011)

        index_2011 = pop_2011.index(mx_2011)

        pop = pop_2011[index_2011]
        ord_pop_2011.append(pop)

        city_2011 = cities_2011[index_2011]

        print('city 2011 = ' +city_2011+ ' and pop = ' +str(mx_2011)+ '\n')

        ord_city_2011.append(city_2011)

        pop_2011.remove(mx_2011)
        cities_2011.remove(city_2011)

    return ord_city_2011, ord_pop_2011, nombre
```

Statistics for Canada's 10 Most Populous Cities:

In this function, I chose to put out the names of the cities and their respective populations. The great difficulty of this function is in the FOR loop

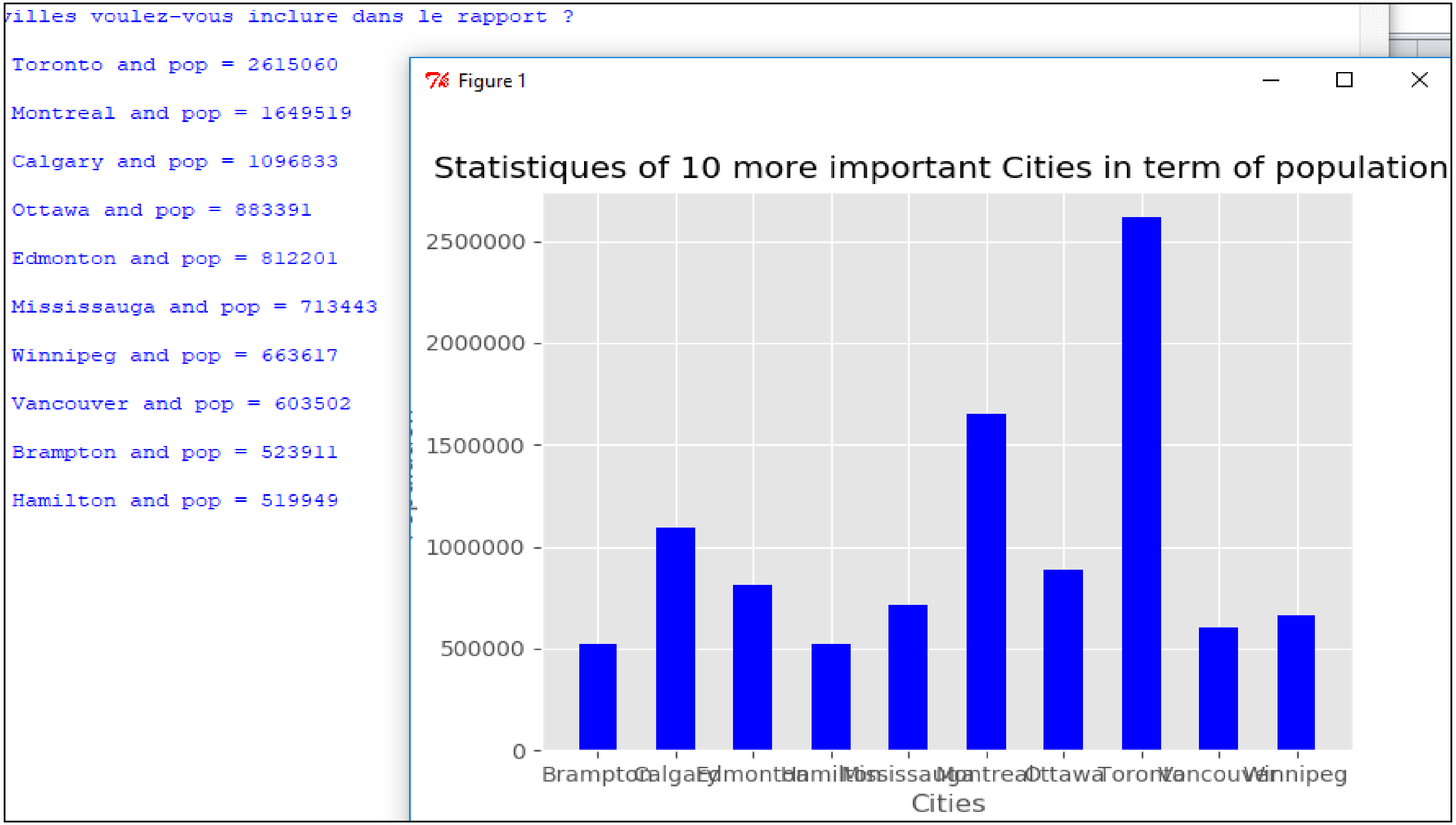
Visualization



```python
def Bar_Statistiques_of_High_Population_Cities():
    plt.style.use('ggplot')
    ord_city_2011, ord_pop_2011, nombre = Ordered_City_List()

    plt.bar(ord_city_2011, ord_pop_2011, color='b', align='center', width=0.5)

    plt.xlabel('Cities')
    plt.ylabel('Population')
    plt.title('Statistiques of '+str(nombre)+' more important Cities in term of population ')
    #plt.legend()

    plt.show()
```

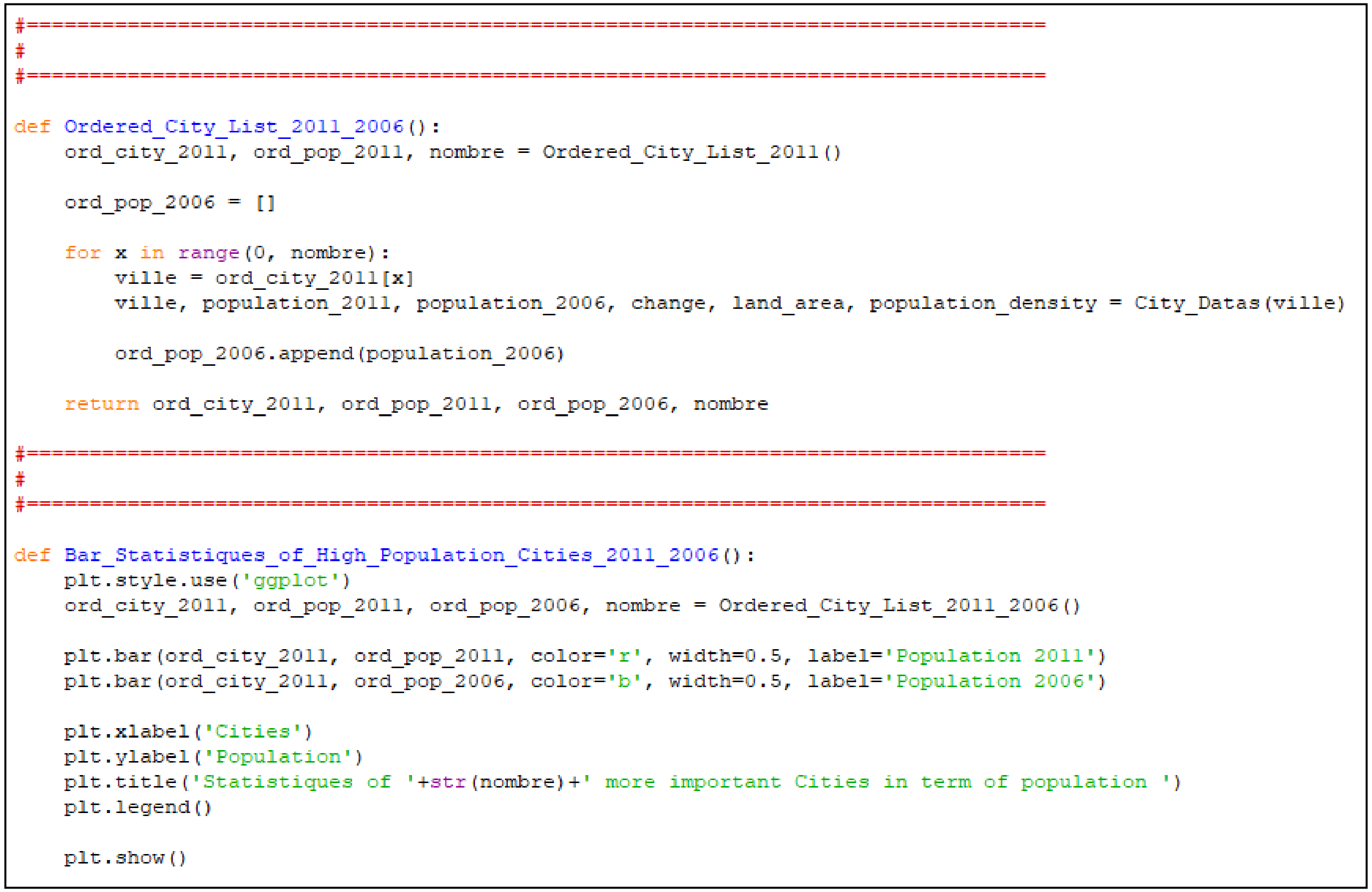
Grosse Requête

Les 10 villes les plus peuplées du Canada en 2011 avec leurs populations respectives



```
villes voulez-vous inclure dans le rapport ?

Toronto and pop = 2615060

Montreal and pop = 1649519

Calgary and pop = 1096833

Ottawa and pop = 883391

Edmonton and pop = 812201

Mississauga and pop = 713443

Winnipeg and pop = 663617

Vancouver and pop = 603502

Brampton and pop = 523911

Hamilton and pop = 519949
```

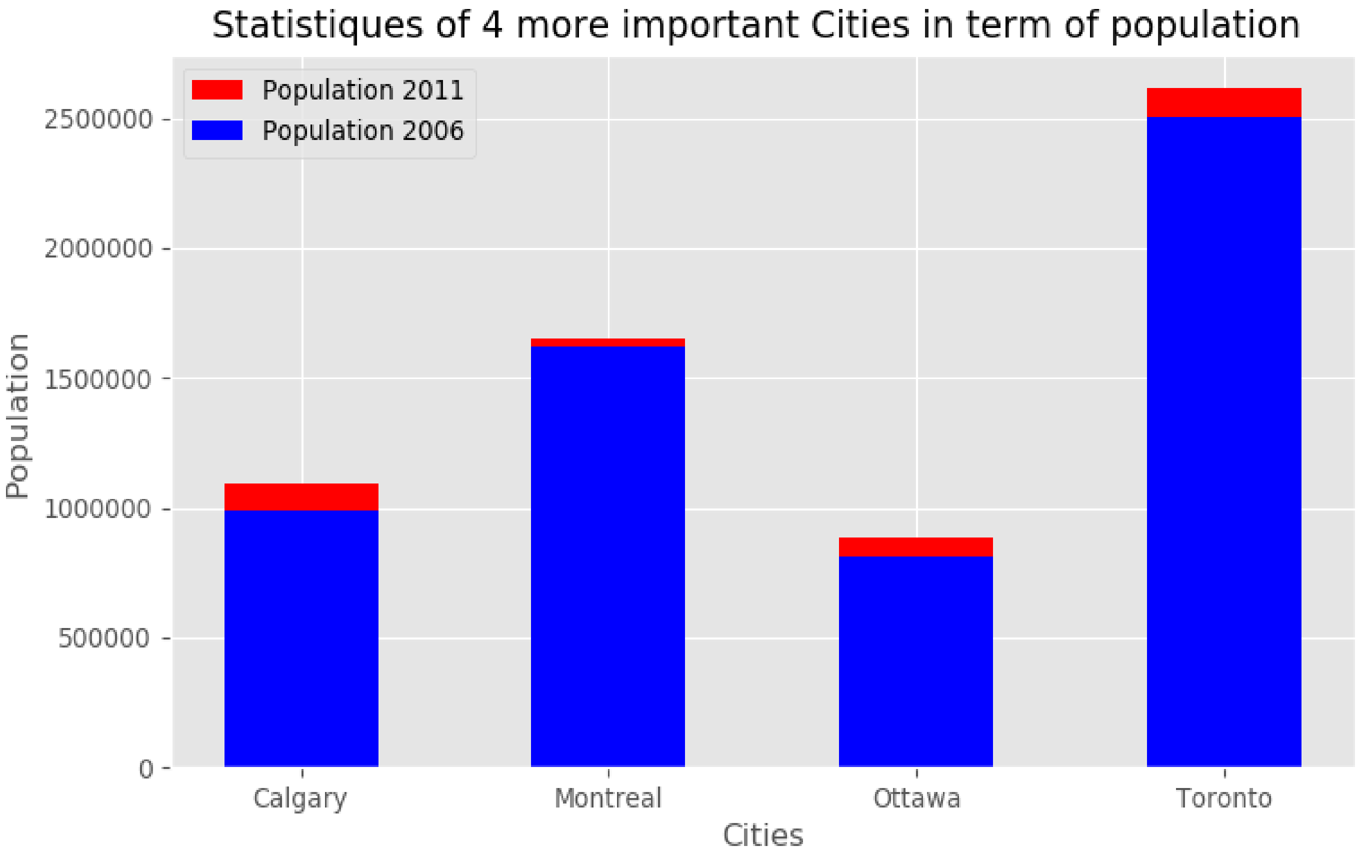
A small and smart manipulation is needed to perform the following request



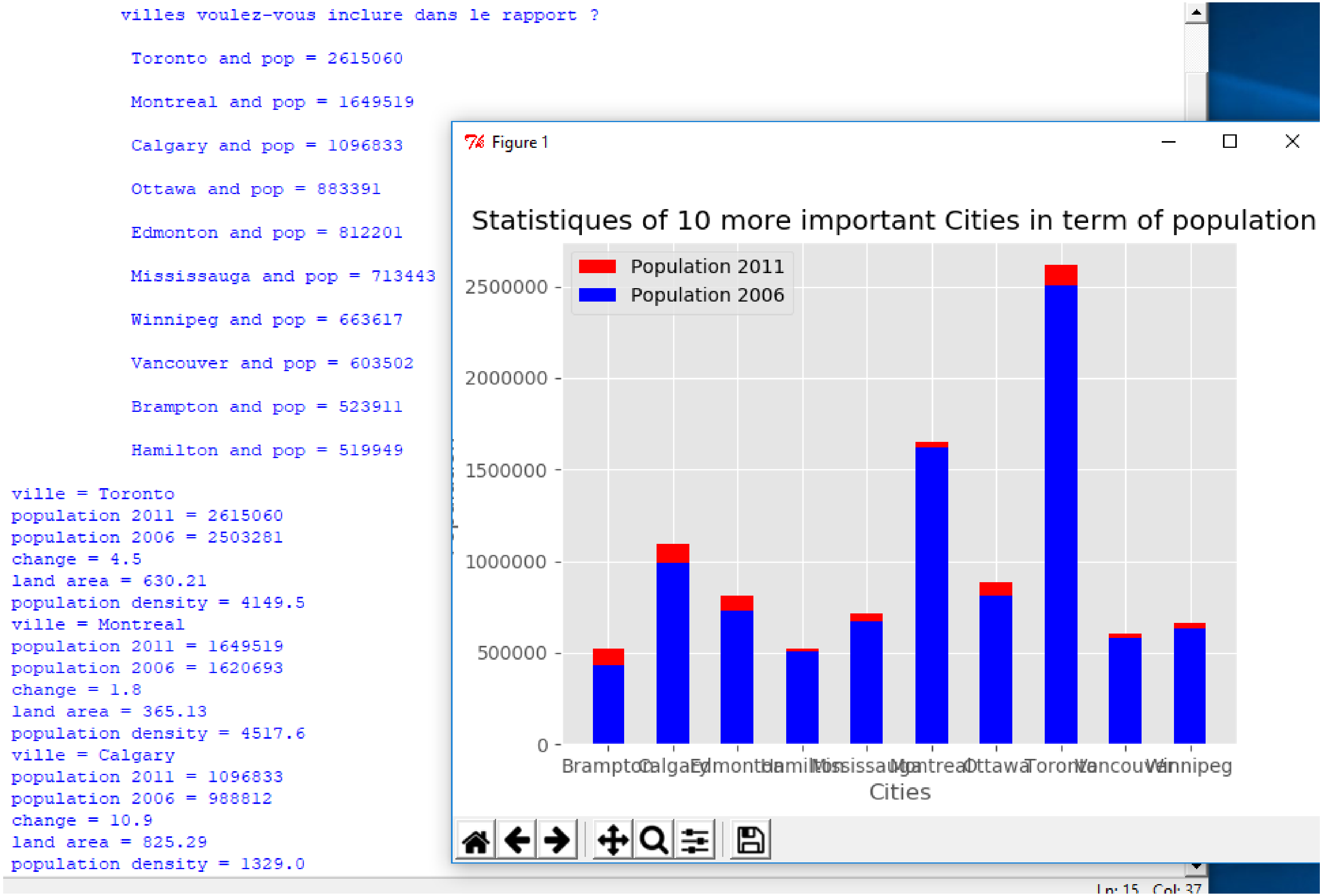
```
#=================================================================================
#
#=================================================================================

def Ordered_City_List_2011_2006():
    ord_city_2011, ord_pop_2011, nombre = Ordered_City_List_2011()

    ord_pop_2006 = []

    for x in range(0, nombre):
        ville = ord_city_2011[x]
        ville, population_2011, population_2006, change, land_area, population_density = City_Datas(ville)

        ord_pop_2006.append(population_2006)

    return ord_city_2011, ord_pop_2011, ord_pop_2006, nombre

#=================================================================================
#
#=================================================================================

def Bar_Statistiques_of_High_Population_Cities_2011_2006():
    plt.style.use('ggplot')
    ord_city_2011, ord_pop_2011, ord_pop_2006, nombre = Ordered_City_List_2011_2006()

    plt.bar(ord_city_2011, ord_pop_2011, color='r', width=0.5, label='Population 2011')
    plt.bar(ord_city_2011, ord_pop_2006, color='b', width=0.5, label='Population 2006')

    plt.xlabel('Cities')
    plt.ylabel('Population')
    plt.title('Statistiques of '+str(nombre)+' more important Cities in term of population ')
    plt.legend()

    plt.show()
```

Complex request

Statistics for Canada's 4 Most Popular Cities



Complex Request



```
villes voulez-vous inclure dans le rapport ?

 Toronto and pop = 2615060

 Montreal and pop = 1649519

 Calgary and pop = 1096833

 Ottawa and pop = 883391

 Edmonton and pop = 812201

 Mississauga and pop = 713443

 Winnipeg and pop = 663617

 Vancouver and pop = 603502

 Brampton and pop = 523911

 Hamilton and pop = 519949

ville = Toronto
population 2011 = 2615060
population 2006 = 2503281
change = 4.5
land area = 630.21
population density = 4149.5
ville = Montreal
population 2011 = 1649519
population 2006 = 1620693
change = 1.8
land area = 365.13
population density = 4517.6
ville = Calgary
population 2011 = 1096833
population 2006 = 988812
change = 10.9
land area = 825.29
population density = 1329.0
```

Conclusion

- Populations continue to grow and grow.