```
###Assignment 4
###The program will set and get country along with information such as
population, are and population density. Also you can modify, save and edit
country list along with thier information

#the following class is named Country and it will set and get country
informatioon

class Country: #creating a country class

   def __init__(self,name,pop,area,continent): #creating construct and accpting
name, population, area and content prameter
       self._name = name # setting construct to class variables. This
description applies for line 10 to 13
       self._pop = pop #setting self._pop
       self._area = area #setting up self._area
       self._continent = continent #setting up self._continet variable
       self._popDensity = round((self._pop/self._area), 2) #calculating
population density which is equal to population /area

   def __repr__(self): #setting up repesentation method
       return self._name + " is in " + self._continent + " with a population
density of " + str(self._popDensity) + "pop is " +str(self._pop) + " area " +
str(self._area) #will produce a genrilzed description of the instant
        #China is in Asia with a population density of 4.56

   def setPopulation(self,pop): #define population
       self._pop = pop #setting self.pop to equal pop which is provided upon
calling

   def setPopDensity(self): #define setPop Density method
       self._popDensity = self._pop/self._area #setting self._popDensity by
applying the equation of population/area

   def getName(self): #creating the getName method
       return self._name #returing the name that was set in the construct

   def getArea(self):
       return self._area #it return the area of the country

   def getPopulation(self): #definging the function
       return self._pop #returing the population

   def getContinent(self): #define the function
       return self._continent #returns the inforatmoin to the user

   def getPopDensity(self): #definging the function
       return self._popDensity #it return the population density to the user

class CountryCatalogue:  #creating a class

    def __init__(self, file):
        try: #it tries the following line incase the name of the file was faults
            self._file = open(file, "r") #open the file
            self._catalogue = {} #creating a dictionary type list
            self._cDictionary = {} #same as upove

            self._cFile = open("continent.txt", "r") #opening continenet.txt
file
            self._cFile.readline() #reading and ignoring the first line
            for line in self._cFile: #read everyline after the first line and
then perform the following actiosn
                line = line.split(",") #spliting and convering text into lists
                self._cDictionary[line[0]]=line[1].replace("\n","")
```

```
            #print(self._cDictionary)

            self._file.readline() #read the first line
            for line in self._file: #for everyline in the self._file do the
follwing
                line = line.replace(",","").split("|") #replace and split the
line into lists
                lineC =
Country(line[0],int(line[1]),float(line[2]),self._cDictionary[line[0]])
#creating an object using the Country class
                self._catalogue[line[0]]=lineC
            self._cFile.close() #close self._cFile
            self._file.close() #close self._file
            #print(self._catalogue)

        except IOError : #creat an exception incase the user input the wrong
file name
            print("Error: file was not found.")
            sys.exit() #it exit the program

        except ValueError :#create an aexcpetion incase the user ad unreadable
file
            print("Error: invalid file.")
            sys.exit() #it exits the program

        except RuntimeError as error :
            print("Error:", str(error))
            sys.exit() #it exits the program
        #print(self._catalogue)

    def addCountry(self, name, pop, area, continent): #defining the function
        if name.title() in self._cDictionary: #check if name exist in the
cDictionary
            print("Country does not exist, Please renter the information") #it
alerts that country already exist
            name = str(input("Please input country's name: ")).title()  #it asks
for country name
            pop = int(input("Please input {} population: ".format(name))) #it
asks for population
            area = float(input("Please input {} area: ".format(name))) #it asks
for area of the country
            continent = str(input("Please input {} continent:
".format(name))).title() #it asks for the continent name
            self.addCountry(name.title(), pop, area, continent.title()) #it
calls it self to check and add the country
        else:#if name does not exist
            self._catalogue[name] = Country(name.title(), int(pop), float(area),
continent.title()) #it creates an object
            self._cDictionary[name] = continent #it adds the name and continent
to cDictionary
            print("{} has successfully been added to the list".format(name)) #it
alerts that the name was sucessfull
            #print(self._cDictionary)
            #print(self._catalogue)

    def deletCountry(self, name): #defien a function
        if name in self._cDictionary: #for every item in cDictionary
            self._cDictionary.pop(name) #remove the name from cDictionary
            self._catalogue.pop(name) #remove the name from catalogue
            print("country has been remvoed") #it alerts upon completion

        else:
            print("name does not exist") #it alerts if country does not exist
```

```
    def findCountry(self, name): #define a function
        if name in self._cDictionary: #for every  itme on the list
            print(name, " Summary: ",self._catalogue[name]) #it prints summary
of countyr name
            print(name, " Area: ",self._catalogue[name].getArea()) #it prints
out the area
            print(name, " Population: ", self._catalogue[name].getPopulation())
#it prints out the population
        else: #otherwise
            print("Country does not exist") #it alerts that the county does
exist


    def filterCountriesByContinent(self, name): #define a function
        countries = set() #define a set
        for cat in self._catalogue: #for every item in catalogue
            if self._catalogue[cat].getContinent() == name: #if name matches to
alist itme
                countries.add(cat) #it adds the country

        print("the following countries belong to the continent of ", name, " are
", countries) #it prints out the contires list


    def printCountryCatalogue(self):
        for cat in self._catalogue:
            print(self._catalogue[cat])

    def setPopulationOfASelectedCountry(self,name, pop): #defining the method
that takes two variables
        if name.title() in self._catalogue: #if name exist in the list
            self._catalogue[name].setPopulation(pop) #set the population number
            self._catalogue[name].setPopDensity() #set the population density
            print(name, " population has been modified to ",
self._catalogue[name].getPopulation()) #print the population
            print(name, " population density is ",
self._catalogue[name].getPopDensity()) #print the population density
            #print(self._catalogue[name]) #print self._catalogue[name]
        else:
            print("You can not change the population because {} does not exist
in the database".format(name))

    def findCountryWithLargestPop(self): #defien the function
        initPop = 0 #define initial population
        countryName = "" #define country name
        for cat in self._catalogue: #for each item in self._catalogue
            if self._catalogue[cat].getPopulation() > initPop: #if country
populatrion is bigger than init
                initPop = self._catalogue[cat].getPopulation() #set initPop to
equal to new population
                countryName = self._catalogue[cat].getName() #set country to
equal the contry with the larger population
        print("The country with the largest population is ", countryName,": ",
initPop) #print country and population

    def findCountryWithSmallestArea(self): #define the function
        lowestTrue = True #set lowstTrue to True
        country = "" #set country to equal nothing
        lowestArea = 0 #set lowerstArea to equal 0
        for cat in self._catalogue: #for each item on the list
            if lowestTrue: #if true
                lowestArea = self._catalogue[cat].getArea() #set lowestARea to
```

```
the lowestArea
                lowestTrue = False #change lowestTrue to false
            if self._catalogue[cat].getArea() < lowestArea: #if the current is
lowest area is lower than the curnet lowestArea
                lowestArea = self._catalogue[cat].getArea() #set lowestArea to
the new area
                country = self._catalogue[cat].getName() #set country to the
current country with the lowest area
        print(country, " has the smallest area of ", lowestArea) #print the
country and area

    def filterCountriesByPopDensity(self, min, max):
        countries = set() #it create a set
        for cat in self._catalogue: #for every item in the list
            if self._catalogue[cat].getPopDensity()>=min and
self._catalogue[cat].getPopDensity()<=max:
                countries.add(self._catalogue[cat].getName()) #it adds to the
list
        print("The following countries population density falls between {} and
{}".format(min,max), " are " ,str(countries).replace("{",
"").replace("'","").replace("}","")) #it prints out the list

    def findMostPopulousContinent(self): #define the function
        northAmerica={} #define a dict for north america
        naTotal=0 #define north america to equal 0
        southAmerica = {} #define a dict for south america
        saTotal=0 #define south america to equal 0
        asia={} ##define a dict for aisa
        aTotal=0 #define asia to equal 0
        africa={} #define a dict for africa
        afTotal = 0  #define africa to equal 0
        europe = {} #define a dict for europe
        eTotal = 0  #define europe to equal 0
        australia={} #define a dict for australia
        auTotal = 0  #define australia to equal 0
        other = {}
        oTotal = 0
        for cat in self._catalogue: #for every item on the list
            if self._catalogue[cat].getContinent() == 'North America': #if item
is equal to North America
                northAmerica[cat]=self._catalogue[cat].getPopulation() #save the
contry name and the population number
                naTotal = naTotal + self._catalogue[cat].getPopulation()
#incrument the total with the population
            elif self._catalogue[cat].getContinent() == 'Asia':  #if item is
equal to North America
                asia[cat]=self._catalogue[cat].getPopulation()   #save the
contry name and the population num
                aTotal = aTotal + self._catalogue[cat].getPopulation()
#incrument the total with the population
            elif self._catalogue[cat].getContinent() == 'South America':   #if
item is equal to south america
                southAmerica[cat]=self._catalogue[cat].getPopulation()  #save
the contry name and the population num
                saTotal = saTotal +
self._catalogue[cat].getPopulation()#incrument the total with the population
            elif self._catalogue[cat].getContinent() == 'Africa':  #if item is
equal to africa
                africa[cat]=self._catalogue[cat].getPopulation()  #save the
contry name and the population num
                afTotal = afTotal + self._catalogue[cat].getPopulation()
#incrument the total with the population
            elif self._catalogue[cat].getContinent() == 'Europe':  #if item is
equal to europe
```

```
                europe[cat]=self._catalogue[cat].getPopulation()  #save the
contry name and the population num
                eTotal = eTotal + self._catalogue[cat].getPopulation()
#incrument the total with the population
            elif self._catalogue[cat].getContinent() == 'Australia':  #if item
is equal to australia
                australia[cat]=self._catalogue[cat].getPopulation()   #save the
contry name and the population num
                auTotal = auTotal + self._catalogue[cat].getPopulation()
#incrument the total with the population
            else:
                other[cat]=self._catalogue[cat].getPopulation()         ##if
item is equal to something else
                oTotal = oTotal + self._catalogue[cat].getPopulation()   #other
content
        test = {"North America": naTotal, "South America":saTotal,
"Asia":aTotal,"Africa":afTotal, "Europe": eTotal}
        total = 0
        continent = ""
        for i in test:
            if test[i]> total:
                total = test[i]
                continent = i
        print(continent, "is the most populated continent with total of ",
total)
        if continent =="North America": #if continent is equal to North America
            self.displayList(northAmerica) #calculate the information using
displayList
        elif continent =='South America': #if continent is equal to South
America
            self.displayList(southAmerica) #calculate the information using
displayList
        elif continent.title() =='Europe': #if continent is equal to Europe
            self.displayList(europe) #calculate the information using
displayList
        elif continent.title()=='Asia': #if continent is equal to Asia
            self.displayList(asia) #calculate the information using displayList
        elif continent.title() =='Africa': #if continent is equal to Africa
            self.displayList(africa) #calculate the information using
displayList
        elif continent.title() =='Australia': #if continent is equal to
Australia
            self.displayList(australia) #calculate the information using
displayList
        else:
            self.displayList(other) #if continent is equal to other


    def saveCountryCatalogue(self, fileName): #define a function
        file = open(fileName,'w') #it opens fileName and then write
        writeFile = False #set writeFile to false
        for i in sorted(self._catalogue): #for every item that is sorted in a
list
            output = self._catalogue[i].getName()+" | "+
self._catalogue[i].getContinent()+" | "+ str(self._catalogue[i].getPopulation())
+" | "+ str(self._catalogue[i].getPopDensity()) +"\n" #it genrate a string
            file.write(output) #it writes to a file
        print("the file was saved") #it alert the file was saved
        file.close() #it closes the file

    def displayList(self, list): #define the method that takes a list as an
argument
        for i in list: #for each item on the list
            print(i,": ", list[i],end=" \n") #print the itme and end with a new
```

```
line
```