Assignment_week_9 & 10_Raghuwanshi_Prashant_DSC540

November 5, 2021

Assignment: Week 9 & Week 10 Exercise, Advanced Data Gathering and Visualization

Name: Prashant Raghuwanshi

Date: 11/03/2021

Course: DSC540-T301 Data Preparation (2221-1) Data Wrangling with Python: Activity 9, Extract top 100 ebooks from gutenberg

[1]:
```
# Import necessary libraries including regex, and beautifulsoup
import urllib.request, urllib.parse, urllib.error
import requests
from bs4 import BeautifulSoup
import ssl
import re
# Import libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

[2]:
```
# check SSL certificate errors
ctx = ssl.create_default_context()
ctx.check_hostname = False
ctx.verify_mode = ssl.CERT_NONE
```

[3]:
```
# Read the HTML from the URL and pass on to BeautifulSoup
top100url = 'https://www.gutenberg.org/browse/scores/top'
response = requests.get(top100url)
```

[4]:
```
# Write a small function to check the status of web request
def status_check(r):
    if r.status_code==200:
        print("Success!")
        return 1
    else:
        print("Failed!")
        return -1
```

[5]:
```
status_check(response)
```

```
Success!
```

[5]: 1

[6]:
```
# Decode the response and pass on to `BeautifulSoup` for HTML parsing
contents = response.content.decode(response.encoding)
```

[7]:
```
soup = BeautifulSoup(contents, 'html.parser')
```

[8]:
```
# Find all the _href_ tags and store them in the list of links. Check how the␣
↪list looks like - print first 30 elements
# Empty list to hold all the http links in the HTML page
lst_links=[]
```

[9]:
```
# Find all the href tags and store them in the list of links
for link in soup.find_all('a'):
    #print(link.get('href'))
    lst_links.append(link.get('href'))
```

[10]:
```
lst_links[:30]
```

[10]:
```
['/',
 '/about/',
 '/about/',
 '/policy/collection_development.html',
 '/about/contact_information.html',
 '/about/background/',
 '/policy/permission.html',
 '/policy/privacy_policy.html',
 '/policy/terms_of_use.html',
 '/ebooks/',
 '/ebooks/',
 '/ebooks/bookshelf/',
 '/browse/scores/top',
 '/ebooks/offline_catalogs.html',
 '/help/',
 '/help/',
 '/help/copyright.html',
 '/help/errata.html',
 '/help/file_formats.html',
 '/help/faq.html',
 '/policy/',
 '/help/public_domain_ebook_submission.html',
 '/help/submitting_your_own_work.html',
 '/help/mobile.html',
 '/attic/',
```

```
 '/donate/',
 '/donate/',
 '#books-last1',
 '#authors-last1',
 '#books-last7']
```

Initialize empty list to hold the file numbers

[11]:
```
booknum=[]
```

[12]:
```
for i in range(19,119):
    link=lst_links[i]
    link=link.strip()
    # Regular expression to find the numeric digits in the link (href) string
    n=re.findall('[0-9]+',link)
    if len(n)==1:
        # Append the filenumber casted as integer
        booknum.append(int(n[0]))
```

Print the file numbers

[13]:
```
print ("\nThe file numbers for the top 100 ebooks on Gutenberg are shown␣
 ↪below\n"+"-"*70)
print(booknum)
```

```
The file numbers for the top 100 ebooks on Gutenberg are shown below
----------------------------------------------------------------------
[1, 1, 7, 7, 30, 30, 84, 1342, 25344, 11, 46, 345, 2701, 2542, 64317, 1080, 844,
174, 43, 5200, 1952, 219, 1661, 98, 1260, 205, 1232, 41, 1727, 1250, 160, 23,
76, 2591, 7370, 4980, 408, 3207, 6130, 74, 2554, 55, 2852, 1400, 514, 16, 32992,
120, 2814, 4300, 2600, 1184, 16328, 66655, 2148, 45, 203, 34901, 5740, 63256,
215, 996, 3825, 158, 768, 66654, 58585, 35, 902, 1597, 1497, 66658, 36, 66659,
779, 43453, 42884, 829, 2500, 22381, 11030, 244, 2680, 3600, 28054, 66663, 1998,
135, 1001, 3296, 61, 1524]
```

You will notice lot of empty spaces/blanks here and there. Ignore them. They are part of HTML page markup and its whimsical nature!

[14]:
```
print(soup.text[:2000])
```

```
Top 100 | Project Gutenberg
```

Menu

About

About Project Gutenberg
Collection Development
Contact Us
History & Philosophy
Permissions & License
Privacy Policy
Terms of Use

Search and Browse

Book Search
Bookshelves
Frequently Downloaded
Offline Catalogs

Help

All help topics →
Copyright Procedures
Errata, Fixes and Bug Reports
File Formats
Frequently Asked Questions
Policies →
Public Domain eBook Submission
Submitting Your Own Work
Tablets, Phones and eReaders
The Attic →

Donate

Donation

Frequently Viewed or Downloaded
These listings are based on the number of times each eBook gets downloaded.
Multiple downloads from the same Internet address on the same day count as

```
one download, and addresses that download more than 100 eBooks in a day are
considered robots and are not counted.

Downloaded Books
2021-11-04171208
last 7 days1179647
last 30 days4946139



Top 100 EBooks yesterday
Top 100 Authors yesterday
Top 100 EBooks last 7 days
Top 100 Authors last 7 days
Top 100 EBooks last 30 days
Top 100 Authors last 30 days


Top 100 EBooks yesterday

Frankenstein; Or, The Modern Prometheus by Mary Wollstonecraft Shelley (3989)
Pride and Prejudice by Jane Austen (2071)
The Scarlet Letter by Nathaniel Hawthorne (1578)
Alice's Adventures in Wonderland by Lewis Carroll (1038)
A Christmas Carol in Prose; Being a Ghost Story of Christmas by Charles Dickens
(1010)
Dracula by Bram Stoker (967)
Moby Dick; Or, The Whale by Herman Melville (847)
A Doll's House : a play by Henrik Ibsen (837)
The Great Gatsby by F. Scott  Fitzgerald (788)
A Modest Proposal by Jonathan Swift (788)
The Importance of Being Earnest: A Trivial Comedy for Serious People by Oscar
Wilde (786)
The Picture of Dorian Gray by Oscar Wilde (754)
The Strange Case of Dr. Jekyll and Mr. Hyde by Robert Louis Stevenson (717)
Metamorphosis by Franz Kafka (703)
The Yellow Wallpaper by Charlotte Perkins Gilman (660)
```

[15]:
```
# Temp empty list of Ebook names
lst_titles_temp=[]
```

[16]:
```
# Use `splitlines()` method of the `soup.text`. It splits the lines of the text␣
 ↪of the `soup` object.
start_idx=soup.text.splitlines().index('Top 100 EBooks yesterday')
```

[17]:
```
# Loop 1-100 to add the strings of next 100 lines to this temporary list.
for i in range(100):
    lst_titles_temp.append(soup.text.splitlines()[start_idx+2+i])
```

[18]:
```
# Use `match` and `span` to find indices and use them
lst_titles=[]
for i in range(100):
    id1,id2=re.match('^[a-zA-Z ]*',lst_titles_temp[i]).span()
    lst_titles.append(lst_titles_temp[i][id1:id2])
```

[19]:
```
# print the list of title
for l in lst_titles:
    print(l)
```

```
Top
Top
Top
Top


Top

Frankenstein
Pride and Prejudice by Jane Austen
The Scarlet Letter by Nathaniel Hawthorne
Alice
A Christmas Carol in Prose
Dracula by Bram Stoker
Moby Dick
A Doll
The Great Gatsby by F
A Modest Proposal by Jonathan Swift
The Importance of Being Earnest
The Picture of Dorian Gray by Oscar Wilde
The Strange Case of Dr
Metamorphosis by Franz Kafka
The Yellow Wallpaper by Charlotte Perkins Gilman
Heart of Darkness by Joseph Conrad
The Adventures of Sherlock Holmes by Arthur Conan Doyle
A Tale of Two Cities by Charles Dickens
Jane Eyre
Walden
The Prince by Niccol
The Legend of Sleepy Hollow by Washington Irving
The Odyssey by Homer
Anthem by Ayn Rand
The Awakening
Narrative of the Life of Frederick Douglass
```

Adventures of Huckleberry Finn by Mark Twain
Grimms
Second Treatise of Government by John Locke
Old Granny Fox by Thornton W
The Souls of Black Folk by W
Leviathan by Thomas Hobbes
The Iliad by Homer
The Adventures of Tom Sawyer
Crime and Punishment by Fyodor Dostoyevsky
The Wonderful Wizard of Oz by L
The Hound of the Baskervilles by Arthur Conan Doyle
Great Expectations by Charles Dickens
Little Women by Louisa May Alcott
Peter Pan by J
The Youngest Girl in the School by Evelyn Sharp
Treasure Island by Robert Louis Stevenson
Dubliners by James Joyce
Ulysses by James Joyce
War and Peace by graf Leo Tolstoy
The Count of Monte Cristo
Beowulf
A New Story Book for Children by Fanny Fern
The Works of Edgar Allan Poe
Anne of Green Gables by L
Uncle Tom
On Liberty by John Stuart Mill
Tractatus Logico
The American Diary of a Japanese Girl by Yon
The Call of the Wild by Jack London
Don Quixote by Miguel de Cervantes Saavedra
Pygmalion by Bernard Shaw
Emma by Jane Austen
Wuthering Heights by Emily Bront
All
The Prophet by Kahlil Gibran
The Time Machine by H
The Happy Prince
Andersen
The Republic by Plato
Australasia Triumphant
The War of the Worlds by H
Chambers
The Tragical History of Doctor Faustus by Christopher Marlowe
A Pickle for the Knowing Ones by Timothy Dexter
The Philippine Islands
Gulliver
Siddhartha by Hermann Hesse
Myths and Legends of Ancient Greece and Rome by E

```
Incidents in the Life of a Slave Girl
A Study in Scarlet by Arthur Conan Doyle
Meditations by Emperor of Rome Marcus Aurelius
Essays of Michel de Montaigne
The Brothers Karamazov by Fyodor Dostoyevsky
Ancient history from the monuments
Thus Spake Zarathustra
Les Mis
Divine Comedy
The Confessions of St
The Communist Manifesto by Friedrich Engels and Karl Marx
Hamlet
The History of the Peloponnesian War by Thucydides
The Life and Adventures of Robinson Crusoe by Daniel Defoe
The Turn of the Screw by Henry James
Carmilla by Joseph Sheridan Le Fanu
The Jungle by Upton Sinclair
The King in Yellow by Robert W
```

Data Wrangling with Python: Activity 10, Build your own movie database by reading from an API

[20]:
```
import json
```

[21]:
```
import urllib.request, urllib.parse, urllib.error
import json
```

[22]:
```
with open('apikey.json') as f:
    keys = json.load(f)
    omdbapi = keys['api_key']
```

[23]:
```
serviceurl = 'http://www.omdbapi.com/?'
apikey = '&apikey='+omdbapi
```

[24]:
```
# Write a utility function `print_json` to print nicely the movie data from a␣
 ↪JSON file (which we will get from the portal)
# Here are the keys of a JSON file,
#'Title', 'Year', 'Rated', 'Released', 'Runtime', 'Genre', 'Director',␣
 ↪'Writer', 'Actors', 'Plot', 'Language','Country', 'Awards', 'Ratings',␣
 ↪'Metascore', 'imdbRating', 'imdbVotes', 'imdbID'

def print_json(json_data):
    list_keys=['Title', 'Year', 'Rated', 'Released', 'Runtime', 'Genre',␣
 ↪'Director', 'Writer',
               'Actors', 'Plot', 'Language', 'Country', 'Awards', 'Ratings',
               'Metascore', 'imdbRating', 'imdbVotes', 'imdbID']
    print("-"*50)
    for k in list_keys:
```

```
        if k in list(json_data.keys()):
            print(f"{k}: {json_data[k]}")
    print("-"*50)
```

[25]:
```
# Write a utility function to download a poster of the movie based on the␣
↪information from the jason dataset and save in your local folder
def save_poster(json_data):
    import os
    title = json_data['Title']
    poster_url = json_data['Poster']
    # Splits the poster url by '.' and picks up the last string as file␣
↪extension
    poster_file_extension=poster_url.split('.')[-1]
    # Reads the image file from web
    poster_data = urllib.request.urlopen(poster_url).read()

    savelocation=os.getcwd()+'\\'+'Posters'+'\\'
    # Creates new directory if the directory does not exist. Otherwise, just␣
↪use the existing path.
    if not os.path.isdir(savelocation):
        os.mkdir(savelocation)

    filename=savelocation+str(title)+'.'+poster_file_extension
    f=open(filename,'wb')
    f.write(poster_data)
    f.close()
```

[26]:
```
# Write a utility function `search_movie` to search a movie by its name, print␣
↪the downloaded JSON data (use the `print_json` function for this) and save␣
↪the movie poster in the local folder (use `save_poster` function for this)

def search_movie(title):
    try:
        url = serviceurl + urllib.parse.urlencode({'t': str(title)})+apikey
        print(f'Retrieving the data of "{title}" now... ')
        print(url)
        uh = urllib.request.urlopen(url)
        data = uh.read()
        json_data=json.loads(data)

        if json_data['Response']=='True':
            print_json(json_data)
            # Asks user whether to download the poster of the movie
            if json_data['Poster']!='N/A':
                save_poster(json_data)
        else:
            print("Error encountered: ",json_data['Error'])
```

```
    except urllib.error.URLError as e:
        print(f"ERROR: {e.reason}")
```

[27]:
```
# `search_movie` function by entering *Titanic*
search_movie("Titanic")
```

```
Retrieving the data of "Titanic" now…
http://www.omdbapi.com/?t=Titanic&apikey=1ad6b76c
------------------------------------------------
Title: Titanic
Year: 1997
Rated: PG-13
Released: 19 Dec 1997
Runtime: 194 min
Genre: Drama, Romance
Director: James Cameron
Writer: James Cameron
Actors: Leonardo DiCaprio, Kate Winslet, Billy Zane
Plot: A seventeen-year-old aristocrat falls in love with a kind but poor artist
aboard the luxurious, ill-fated R.M.S. Titanic.
Language: English, Swedish, Italian, French
Country: United States, Mexico, Australia
Awards: Won 11 Oscars. 125 wins & 83 nominations total
Ratings: [{'Source': 'Internet Movie Database', 'Value': '7.8/10'}, {'Source':
'Rotten Tomatoes', 'Value': '89%'}, {'Source': 'Metacritic', 'Value': '75/100'}]
Metascore: 75
imdbRating: 7.8
imdbVotes: 1,098,236
imdbID: tt0120338
------------------------------------------------
```

[28]:
```
#search_movie function by entering "*Random_error*"
search_movie("Random_error")
```

```
Retrieving the data of "Random_error" now…
http://www.omdbapi.com/?t=Random_error&apikey=1ad6b76c
Error encountered:  Movie not found!
```

I have found the folder name poster with titanic jpg picture in my working directory

3) Connect to the Twitter API and do a simple data pull

[29]:
```
import twitter
api = twitter.Api(consumer_key='BvAZWpZjb87Bv4OSFjsOdxwvy',
  consumer_secret='HYQ1FPShH4B0lISBPmYLzcHGAnsc2GNOI89PpvVwbzRoPlhidU',
    access_token_key='1453146427224383489-yAo1gc4dyrEdkQIja5ZYZm7JFoagM6',
    access_token_secret='E0k0QzZUC8Ily8ZqKAkme6T97jreyzriHLEyyBz1asm9V')
```

[30]:
```
print(api.VerifyCredentials())
```

```
{"created_at": "Tue Oct 26 23:49:09 +0000 2021", "default_profile": true,
"default_profile_image": true, "id": 1453146427224383489, "id_str":
"1453146427224383489", "name": "Prashant Raghuwanshi",
"profile_background_color": "F5F8FA", "profile_image_url":
"http://abs.twimg.com/sticky/default_profile_images/default_profile_normal.png",
"profile_image_url_https": "https://abs.twimg.com/sticky/default_profile_images/
default_profile_normal.png", "profile_link_color": "1DA1F2",
"profile_sidebar_border_color": "C0DEED", "profile_sidebar_fill_color":
"DDEEF6", "profile_text_color": "333333", "profile_use_background_image": true,
"screen_name": "praghuw1o1", "withheld_in_countries": []}
```

[31]:
```
# Download User Timeline
statuses = api.GetUserTimeline(screen_name='Michael Grogan')
print([s.text for s in statuses])
```

```
[]
```

[32]:
```
api.GetSearch(term='Bellevue University', since=2021-11-1, count=10)
```

[32]:
```
[Status(ID=1456712167668043776, ScreenName=BillyHeyen, Created=Fri Nov 05
19:56:59 +0000 2021, Text='RT @BillyHeyen: Bellevue’s Kearston Lunsford is
heading to Tiffin University as a runner \u200d '),
 Status(ID=1456697889258881027, ScreenName=StatisticsViews, Created=Fri Nov 05
19:00:15 +0000 2021, Text='Adjunct Instructor-Data Science - Bellevue, NE -
Bellevue University https://t.co/4KTbysZj9S'),
 Status(ID=1456667648717172747, ScreenName=BellevueU, Created=Fri Nov 05
17:00:05 +0000 2021, Text='Happy #NationalCareerDevelopmentMonth! Every Bellevue
University student is pre-registered on Handshake, an app des…
https://t.co/ohNCPTjGmE'),
 Status(ID=1456648204980736012, ScreenName=MUCoachGilbert, Created=Fri Nov 05
15:42:49 +0000 2021, Text='RT @Midland_WBB: GAMEDAY for everyone!!\n       •
Varsity will play Bellevue University in Bellevue, Nebraska with a tip-off of
3PM!…'),
 Status(ID=1456643865952219140, ScreenName=CSM_Flames, Created=Fri Nov 05
15:25:35 +0000 2021, Text='On the road: Two CSM teams are on the road today.
Basketball plays Dickinson State at 1 p.m. in the Bellevue Univer…
https://t.co/UlgjiOXZHu'),
 Status(ID=1456629805366059021, ScreenName=JonHanc57133297, Created=Fri Nov 05
14:29:43 +0000 2021, Text='RT @Midland_WBB: GAMEDAY for everyone!!\n       •
Varsity will play Bellevue University in Bellevue, Nebraska with a tip-off of
3PM!…'),
 Status(ID=1456609857566646276, ScreenName=BillyHeyen, Created=Fri Nov 05
13:10:27 +0000 2021, Text='Bellevue’s Kearston Lunsford is heading to Tiffin
University as a runner \u200d https://t.co/cTkyy9Ehus'),
 Status(ID=1456596397910855680, ScreenName=Midland_WBB, Created=Fri Nov 05
12:16:58 +0000 2021, Text='GAMEDAY for everyone!!\n       • Varsity will play
```

```
Bellevue University in Bellevue, Nebraska with a tip-off of 3PM!…
https://t.co/HqxKfRptFA'),
 Status(ID=1456538610895708167, ScreenName=SmajioE, Created=Fri Nov 05 08:27:20
+0000 2021, Text='Bellevue University diploma, Order a fake degree and
transcript. Buy a best Bellevue University degree.… https://t.co/E6Rers6Hvh'),
 Status(ID=1456403937725091861, ScreenName=CHSMusings, Created=Thu Nov 04
23:32:12 +0000 2021, Text='Ole @bruceforseattle played football at University of
Washington back in late 1970s and Bruce will bring back some…
https://t.co/dpWrDN4f2p')]
```

[33]:
```
tweets = api.GetSearch(term='Bellevue University', since=2021-11-1, count=10)
all_tweets = [tweet.text for tweet in tweets]
all_tweets[:5]
```

[33]:
```
['RT @BillyHeyen: Bellevue's Kearston Lunsford is heading to Tiffin University
as a runner  \u200d ',
 'Adjunct Instructor-Data Science - Bellevue, NE - Bellevue University
https://t.co/4KTbysZj9S',
 'Happy #NationalCareerDevelopmentMonth! Every Bellevue University student is
pre-registered on Handshake, an app des… https://t.co/ohNCPTjGmE',
 'RT @Midland_WBB: GAMEDAY for everyone!!\n       • Varsity will play Bellevue
University in Bellevue, Nebraska with a tip-off of 3PM!…',
 'On the road: Two CSM teams are on the road today. Basketball plays Dickinson
State at 1 p.m. in the Bellevue Univer… https://t.co/UlgjiOXZHu']
```

Using one of the datasets provided in Weeks 7 & 8, or a dataset of your own,

choose 3 of the following visualizations to complete.

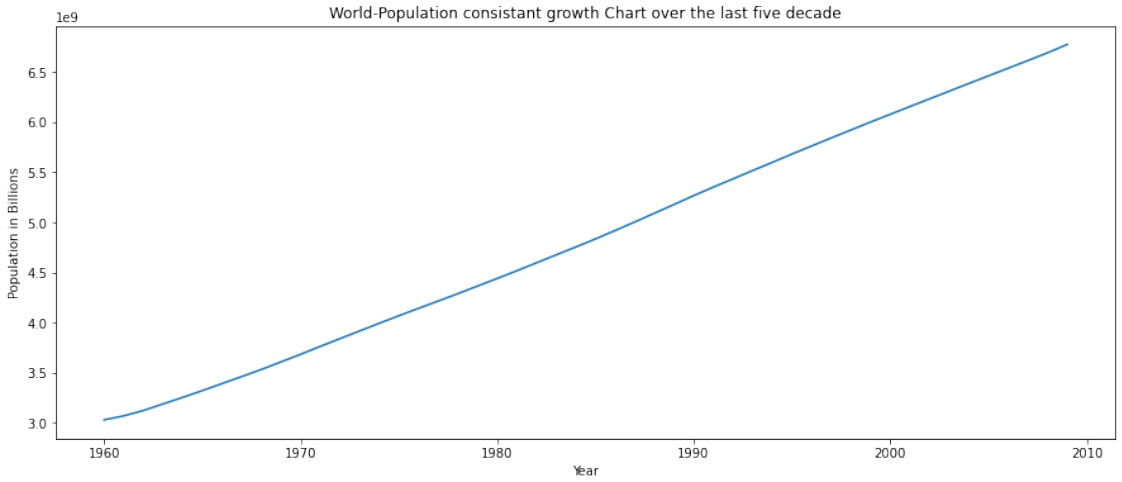
[34]:
```
# Import the sample data file for analysis scenario by using different charts
world_pop_df = pd.read_excel (r'C:
 ↪\Users\dell\Documents\Machine_learning_assigments\world-population.xlsx',␣
 ↪sheet_name='world-population')
# Show the dataframe data values
world_pop_df.head(5)
```

[34]:

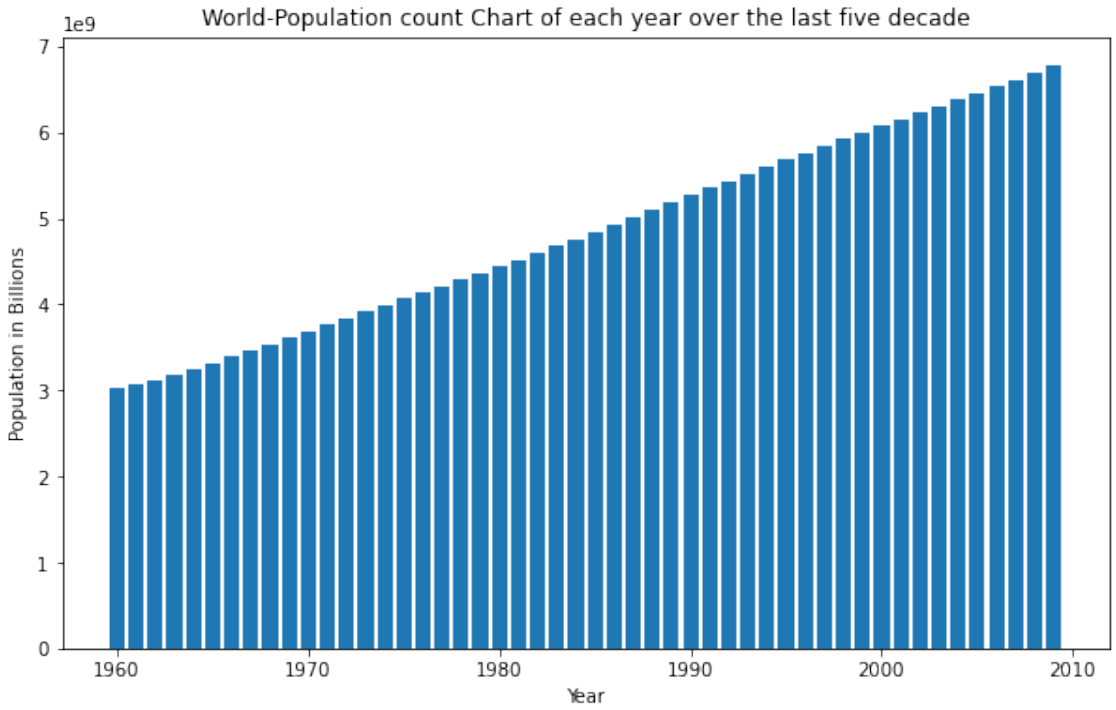
	Year	Population
0	1960	3028654024
1	1961	3068356747
2	1962	3121963107
3	1963	3187471383
4	1964	3253112403

[35]:
```
# Creatting line chart by using world population data to find the answer of␣
 ↪below question

## how is the population of world is changing over the decades
```

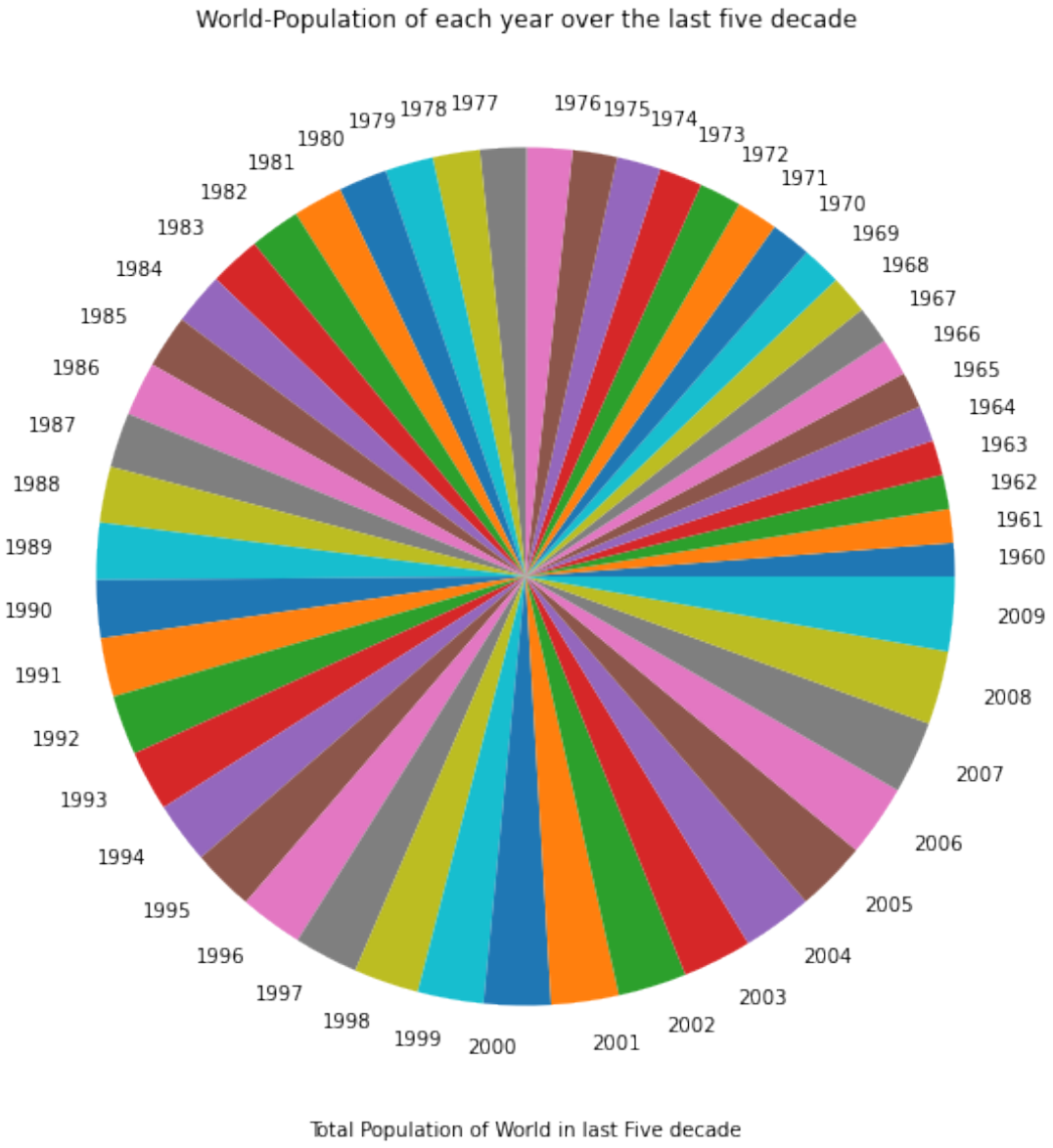
```
## Plot Line Graph
plt.figure(figsize=(15,6)) # choosing the relavant size of ploted figure
plt.plot(world_pop_df['Year'], world_pop_df['Population'])
## Setting lables
plt.ylabel("Population in Billions")
plt.xlabel("Year")
## Setting titles
plt.title("World-Population consistant growth Chart over the last five decade")
plt.show()
```



[36]:
```
# Creatting Bar chart by using world population data to find the answer of below question

## how much the population of world at each years over last five decades

## Plot Line Graph
plt.figure(figsize=(10,6)) # choosing the relavant size of ploted figure
plt.bar(world_pop_df['Year'], world_pop_df['Population'])
## Setting lables
plt.ylabel("Population in Billions")
plt.xlabel("Year")
## Setting titles
plt.title("World-Population count Chart of each year over the last five decade")
plt.show()
```



[37]:
```
# Creatting pie chart by using world population data to find the answer of␣
 ↪below question

## Contribution of each years of world population in last five decade long␣
 ↪population growth.

## Plot Line Graph
plt.figure(figsize=(10,10)) # choosing the relavant size of ploted figure
plt.pie(world_pop_df['Population'], labels = world_pop_df['Year'])
## Setting lables
plt.xlabel("Total Population of World in last Five decade")
## Setting titles
plt.title("World-Population of each year over the last five decade")
plt.show()
```


World-Population of each year over the last five decade
1976
1975
1974
1973
1972
1971
1970
1969
1968
1967
1966
1965
1964
1963
1962
1961
1960
2009
2008
2007
2006
2005
2004
2003
2002
2001
2000
1999
1998
1997
1996
1995
1994
1993
1992
1991
1990
1989
1988
1987
1986
1985
1984
1983
1982
1981
1980
1979
1978
1977
Total Population of World in last Five decade