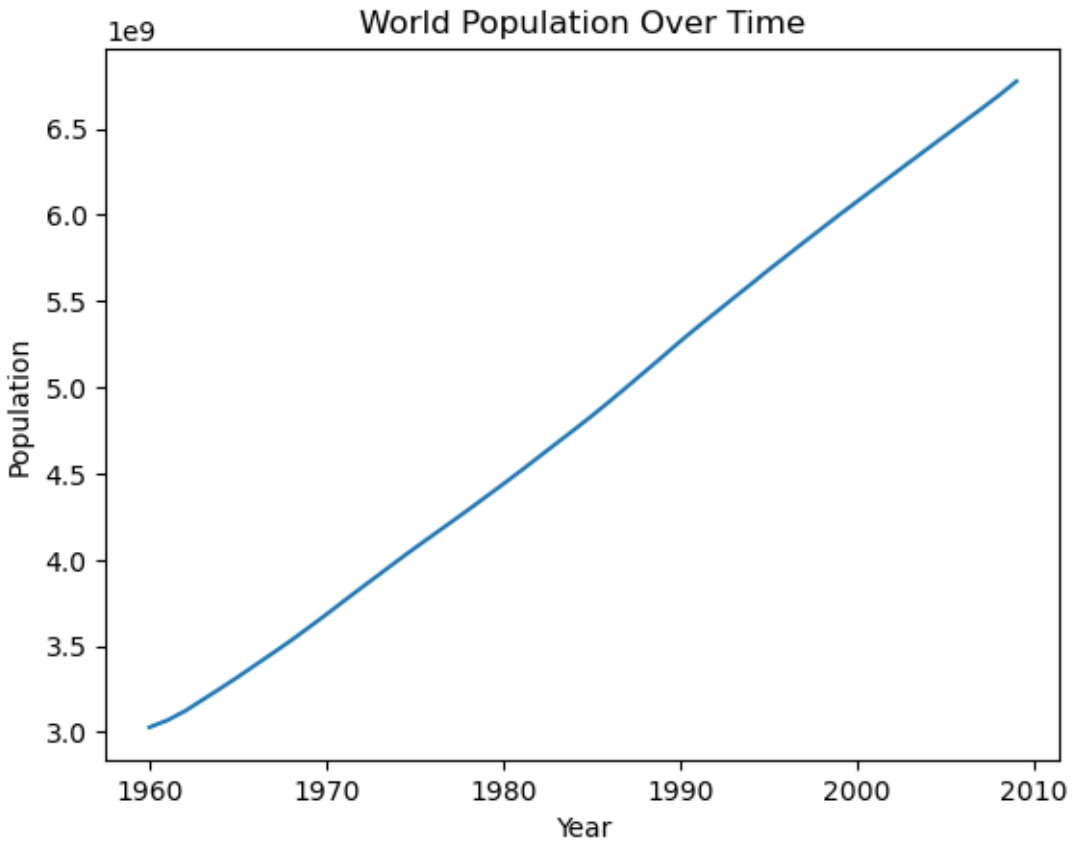
DSC540-T301_2237-1_Samanta_Rajib_week_1_2

June 18, 2023

[2]:
```
# Class : DSC540-T301 Data Preparation (2237-1)
# Name : Rajib Samanta
# Assignment : Week 1 & 2 Excercises
## Assignment:  1

import pandas as pd
import matplotlib.pyplot as plt
#import xlrd

# Read the Excel file
data = pd.read_excel('world-population.xlsm', sheet_name=None)

# prints all sheets
#print(df)
# Extract the necessary columns
years = data['world-population']['Year']
#years
population = data['world-population']['Population']
#population
# Create the line chart
plt.plot(years, population)

# Set the chart title and labels
plt.title('World Population Over Time')
plt.xlabel('Year')
plt.ylabel('Population')

# Display the chart
plt.show()
```



[3]:
```
# Assignment : 3. Complete the following activities:
```

[4]:
```
# a. Data Wrangling with Python: Activity 1 page 17
## : list of random numbers and then generate another list from the first one,␣
 ↪which only contains numbers that are divisible by three. Repeat the␣
 ↪experiment three times.
## Then, we will calculate the average difference of length between the two␣
 ↪lists.
```

[5]:
```
import random

num_experiments = 3
list_length = 100 # list length to generate 100 random number

# Initialize variables
total_diff_length = 0

# Perform the experiment Three times
for _ in range(num_experiments):
    # Generate a  list of random numbers between 1 to 10,000
```

```
    random_list = [random.randint(1, 10000) for _ in range(list_length)]

    # Filter the new list for numbers divisible by 3
    divisible_by_three_list = [num for num in random_list if num % 3 == 0]

    # Calculate the difference in length
    diff_length = len(random_list) - len(divisible_by_three_list)

    # Accumulate the difference in length for averaging later
    total_diff_length += diff_length

# Calculate the average difference in length
average_diff_length = total_diff_length / num_experiments

print(f"Average difference in length: {average_diff_length}")
```

```
Average difference in length: 67.0
```

[6]:
```
# Assignment : b. Data Wrangling with Python: Activity 2 page 31
```

[7]:
```
# initialize list variable
multi_lines= []
# assign the text copying from https://github.com/TrainingByPackt/
 ↪Data-Wrangling-with-Python/tree/master/Chapter01/Activity02
multi_lines= ["This eBook is for the use of anyone anywhere in the United␣
 ↪States and most other parts of the world at no cost and with almost no␣
 ↪restrictions whatsoever. You may copy it, give it away or re-use it under␣
 ↪the terms of the Project Gutenberg License included with this eBook or␣
 ↪online at www.gutenberg.org. If you are not located in the United States,␣
 ↪you will have to check the laws of the country where you are located before␣
 ↪using this eBook.\n"
"Title: Pride and prejudice\n"
"Author: Jane Austen\n"
"Release Date: November 12, 2022 [eBook #1342]\n"
"[Most recently updated: April 14, 2023]\n"
"\n"
"Language: English\n"
"\n"
"Produced by: Produced by: Chuck Greif and the Online Distributed Proofreading␣
 ↪Team at http://www.pgdp.net (This file was produced from images available at␣
 ↪The Internet Archive)"]

print(multi_lines)


#print(multi_lines)
```

```
# Join the lines into a single multiline string
#multi_lines_text = '\n'.join(multi_lines)
#print(multi_lines_text)

# Print the multiline text
#print("Multiline text:")
#print(multiline_text)"
# Prompt the user for multiline input
```

['This eBook is for the use of anyone anywhere in the United States and most other parts of the world at no cost and with almost no restrictions whatsoever. You may copy it, give it away or re-use it under the terms of the Project Gutenberg License included with this eBook or online at www.gutenberg.org. If you are not located in the United States, you will have to check the laws of the country where you are located before using this eBook.\nTitle: Pride and prejudice\nAuthor: Jane Austen\nRelease Date: November 12, 2022 [eBook #1342]\n[Most recently updated: April 14, 2023]\n\nLanguage: English\n\nProduced by: Produced by: Chuck Greif and the Online Distributed Proofreading Team at http://www.pgdp.net (This file was produced from images available at The Internet Archive)']

[8]:
```
# print Type
type(multi_lines)
```

[8]: list

[9]:
```
# Print length
len(multi_lines)
```

[9]: 1

[10]:
```
# Convert List to string
multi_lines_text = '\n'.join(multi_lines)
# "Remove all new lines and symbols using the replace function."
clean_text = multi_lines_text.replace('\n', '').replace('@', '').replace('#',␣
 ↪'').replace('!', '').replace('[', ' ').replace(']', ' ').replace('(', ' ').
 ↪replace(')', ' ').replace(',', '')
#multi_lines_text.replace('\n','')
print(clean_text)
```

This eBook is for the use of anyone anywhere in the United States and most other parts of the world at no cost and with almost no restrictions whatsoever. You may copy it give it away or re-use it under the terms of the Project Gutenberg License included with this eBook or online at www.gutenberg.org. If you are not located in the United States you will have to check the laws of the country where you are located before using this eBook.Title: Pride and prejudiceAuthor: Jane AustenRelease Date: November 12 2022 eBook 1342 Most recently updated: April 14 2023 Language: EnglishProduced by: Produced by: Chuck Greif and the

```
Online Distributed Proofreading Team at http://www.pgdp.net  This file was
produced from images available at The Internet Archive
```

[11]:
```
# Split the multiline text into words
word_list = clean_text.split()
print("List of words:")
print(word_list)
```

```
List of words:
['This', 'eBook', 'is', 'for', 'the', 'use', 'of', 'anyone', 'anywhere', 'in',
'the', 'United', 'States', 'and', 'most', 'other', 'parts', 'of', 'the',
'world', 'at', 'no', 'cost', 'and', 'with', 'almost', 'no', 'restrictions',
'whatsoever.', 'You', 'may', 'copy', 'it', 'give', 'it', 'away', 'or', 're-use',
'it', 'under', 'the', 'terms', 'of', 'the', 'Project', 'Gutenberg', 'License',
'included', 'with', 'this', 'eBook', 'or', 'online', 'at', 'www.gutenberg.org.',
'If', 'you', 'are', 'not', 'located', 'in', 'the', 'United', 'States', 'you',
'will', 'have', 'to', 'check', 'the', 'laws', 'of', 'the', 'country', 'where',
'you', 'are', 'located', 'before', 'using', 'this', 'eBook.Title:', 'Pride',
'and', 'prejudiceAuthor:', 'Jane', 'AustenRelease', 'Date:', 'November', '12',
'2022', 'eBook', '1342', 'Most', 'recently', 'updated:', 'April', '14', '2023',
'Language:', 'EnglishProduced', 'by:', 'Produced', 'by:', 'Chuck', 'Greif',
'and', 'the', 'Online', 'Distributed', 'Proofreading', 'Team', 'at',
'http://www.pgdp.net', 'This', 'file', 'was', 'produced', 'from', 'images',
'available', 'at', 'The', 'Internet', 'Archive']
```

[12]:
```
# "Create a list from this list that will contain only the unique words."
# Create a new list with unique words using set
unique_word_list = list(set(word_list))

print("List of unique words:")
print(unique_word_list)
```

```
List of unique words:
['and', 'AustenRelease', '12', 'was', '1342', 'laws', 'Jane', 'anywhere',
'Produced', 'Online', 'away', 'prejudiceAuthor:', '2023', 'located', 'terms',
'the', 'at', 'Date:', 'have', 'www.gutenberg.org.', 'for', 'April',
'restrictions', 'online', 'Gutenberg', 'anyone', 'check', 'EnglishProduced',
'are', 'This', 'Proofreading', 'it', 'you', 'recently', 'Language:', 'this',
'produced', 'to', 'almost', 'November', 'most', 'Archive', 'not', 'use',
'world', 'images', 'cost', 'States', 'updated:', 'Chuck', 'Greif', 'included',
'Distributed', 'License', 'Internet', 'United', 'will', 'http://www.pgdp.net',
'with', 'under', 'Team', 'country', 're-use', 'Pride', '2022', 'eBook.Title:',
'using', 'in', '14', 'no', 'of', 'whatsoever.', 'before', 'from', 'by:', 'or',
'available', 'Most', 'If', 'parts', 'give', 'eBook', 'The', 'Project', 'copy',
'You', 'where', 'file', 'is', 'other', 'may']
```

[13]:
```
# "Count the number of times the unique word has appeared in the list using the␣
 ↪key and value in dict."
```

```
# Create an empty dictionary to store word counts
word_counts = {}

# Count the occurrences of each unique word
for word in word_list:
    if word in word_counts:
        word_counts[word] += 1
    else:
        word_counts[word] = 1

print("Word counts:")
print(word_counts)
```

```
Word counts:
{'This': 2, 'eBook': 3, 'is': 1, 'for': 1, 'the': 9, 'use': 1, 'of': 4,
'anyone': 1, 'anywhere': 1, 'in': 2, 'United': 2, 'States': 2, 'and': 4, 'most':
1, 'other': 1, 'parts': 1, 'world': 1, 'at': 4, 'no': 2, 'cost': 1, 'with': 2,
'almost': 1, 'restrictions': 1, 'whatsoever.': 1, 'You': 1, 'may': 1, 'copy': 1,
'it': 3, 'give': 1, 'away': 1, 'or': 2, 're-use': 1, 'under': 1, 'terms': 1,
'Project': 1, 'Gutenberg': 1, 'License': 1, 'included': 1, 'this': 2, 'online':
1, 'www.gutenberg.org.': 1, 'If': 1, 'you': 3, 'are': 2, 'not': 1, 'located': 2,
'will': 1, 'have': 1, 'to': 1, 'check': 1, 'laws': 1, 'country': 1, 'where': 1,
'before': 1, 'using': 1, 'eBook.Title:': 1, 'Pride': 1, 'prejudiceAuthor:': 1,
'Jane': 1, 'AustenRelease': 1, 'Date:': 1, 'November': 1, '12': 1, '2022': 1,
'1342': 1, 'Most': 1, 'recently': 1, 'updated:': 1, 'April': 1, '14': 1, '2023':
1, 'Language:': 1, 'EnglishProduced': 1, 'by:': 2, 'Produced': 1, 'Chuck': 1,
'Greif': 1, 'Online': 1, 'Distributed': 1, 'Proofreading': 1, 'Team': 1,
'http://www.pgdp.net': 1, 'file': 1, 'was': 1, 'produced': 1, 'from': 1,
'images': 1, 'available': 1, 'The': 1, 'Internet': 1, 'Archive': 1}
```

[14]:
```
# "Find the top 25 words from the unique words that you have found using the␣
 ↪slice function."

# Sort the words based on their counts in descending order
sorted_words = sorted(word_counts, key=word_counts.get, reverse=True)

# Get the top 25 words
top_25_words = sorted_words[:25]

print("Top 25 words:")
print(top_25_words)
```

```
Top 25 words:
['the', 'of', 'and', 'at', 'eBook', 'it', 'you', 'This', 'in', 'United',
'States', 'no', 'with', 'or', 'this', 'are', 'located', 'by:', 'is', 'for',
'use', 'anyone', 'anywhere', 'most', 'other']
```

[15]:
```
# Assignment : c. Data Wrangling with Python: Activity 3 page 49
## "using permutations to generate all possible three-digit numbers that can be␣
 ↪generated using 0, 1, and 2. Then, loop over this iterator, and also use␣
 ↪isinstance and assert to make sure that the return types are tuples. Also,␣
 ↪use a single line of code involving dropwhile and lambda expressions to␣
 ↪convert all the tuples to lists while dropping any leading zeros (for␣
 ↪example, (0, 1, 2) becomes [1, 2]). Finally, write a function that takes a␣
 ↪list like before and returns the actual number contained in it."
from itertools import permutations, dropwhile

def get_number_from_list(num_list):
    num_str = ''.join(str(digit) for digit in num_list)
    return int(num_str)

digits = [0, 1, 2]
three_digit_numbers = permutations(digits, 3)
for num_tuple in three_digit_numbers:
    assert isinstance(num_tuple, tuple)
    # remove leading 0 using dropwhile and lambda function
    num_list = list(dropwhile(lambda x: x == 0 and num_tuple.index(x) == 0,␣
 ↪num_tuple))
    print(num_list)
```

```
[1, 2]
[2, 1]
[1, 0, 2]
[1, 2, 0]
[2, 0, 1]
[2, 1, 0]
```

[16]:
```
#"Check the actual type that dropwhile returns."
# Generate all possible three-digit numbers
three_digit_numbers = dropwhile(lambda x: x[0] == 0, permutations(digits, 3))
print(type(three_digit_numbers))
```

```
<class 'itertools.dropwhile'>
```

[17]:
```
## "Combine the preceding code into one block, and this time write a separate␣
 ↪function where you will pass the list generated from dropwhile,"

def get_whole_number(num_list):
    num_str = ''.join(map(str, num_list))
    return int(num_str)
```

[18]:
```
# Generate all possible three-digit numbers
three_digit_numbers = dropwhile(lambda x: x[0] == 0, permutations(digits, 3))

for num_tuple in three_digit_numbers:
```

```
    num_list = list(num_tuple)
    print(num_list)
    number = get_whole_number(num_list)
    print(number)
```

```
[1, 0, 2]
102
[1, 2, 0]
120
[2, 0, 1]
201
[2, 1, 0]
210
```

[19]:
```
# Assignment : d. Data Wrangling with Python: Activity 4 page 59
```

[20]:
```
# "Import zip_longest from itertools. Create a function to zip header, line and fillvalue=None."
from itertools import zip_longest

def zip_header_line(header, line, fillvalue=None):
    zipped_data = zip_longest(header, line, fillvalue=fillvalue)
    return list(zipped_data)
# Function to process each line and construct a dictionary
def process_line(line, headers):
    values = line.strip().split(',')
    data_dict = dict(zip(headers, values))
    # Fill up the dictionary with key-value pairs
    for header, value in zip(headers, values):
        if value == '':
            data_dict[header] = None
        else:
            data_dict[header] = value
    return data_dict
```

[21]:
```
# "Open the accompanying sales_record.csv file from the GitHub link by using r mode inside a with block and first check that it is opened."
import csv
#import urllib.request
import requests

url = "https://github.com/TrainingByPackt/Data-Wrangling-with-Python/blob/master/Lesson02/Activity04/sales_record.csv"
# Download the file using requests
response = requests.get(url)
# Check if the download was successful
if response.status_code == 200:
```

```
    # Open the file in "r" mode using a with block
    with open("sales_record.csv", "r") as file:
        # File is successfully opened
        print("File opened successfully.")
        # Perform further operations on the file
        # Read the first line
        reader = csv.reader(file)
        header_row = next(reader)
        #print(header_row)
        # "Read the first line and use string methods to generate a list of all␣
↪the column names."
        # Generate a list of column names
                # Convert list to string with comma separator
        column_names_string = ', '.join(header_row)
        print("List of columns: ")
        print(column_names_string)
        # Skip the header row as it is already printed
        next(file)
        for line in file:

            # Process each line
            #print(line)
            # "Read each line and pass that line to a function, along with the␣
↪list of the headers. The work of the function is to construct a dict out of␣
↪these two "
            # Construct Dictionary
            # Process each line and construct a dictionary
            data = process_line(line.strip(), header_row)

            # Print the dictionary
            #print(data)
        # Just priniting last line to avoid big file
        print(data)

else:
    # Download failed
    print("Failed to download the file.")
```

```
File opened successfully.
List of columns:
Region, Country, Item Type, Sales Channel, Order Priority, Order Date, Order ID,
Ship Date, Units Sold, Unit Price, Unit Cost, Total Revenue, Total Cost, Total
Profit
{'Region': 'Sub-Saharan Africa', 'Country': 'Tanzania', 'Item Type':
'Vegetables', 'Sales Channel': 'Online', 'Order Priority': 'L', 'Order Date':
'7/8/2011', 'Order ID': '996861922', 'Ship Date': '8/7/2011', 'Units Sold':
'2450', 'Unit Price': '154.06', 'Unit Cost': '90.93', 'Total Revenue':
```

```
'377447.00', 'Total Cost': '222778.50', 'Total Profit': '154668.50'}
```

[ ]: