Assignment 2

Student Name: Eryk Gloginski **(L00157413)**

Course: BSc Computing

Module: Automation

Lecturer: Saim Ghafoor

Submission Date: 5/14/2022

Task 1:

I first declare the necessary variables required to store the data after which I open the "**Sample for task1.csv**" file where I read all the csv data. I then iterate over the items in "**list_file**", convert each item into a string and then split it and add it to an appropriate list.

```
import csv
# declare the variable that will hold all things
list_file = []
# declare variables to store the string input into
str_packets = []
str_tx = []
str_rx = []
# read the csv file and put everything in list_file
with open('Sample for task 1.csv', newline='') as f:
    reader = csv.reader(f)
    list_file = list(reader)
# iterate over items in the list and add them to their appropriate lists
for item in list_file:
    tempItem = ''.join(item)
    temp = tempItem.split(',')
    str_packets.append(temp[2])
    str_tx.append(temp[4])
    str_rx.append(temp[6])
```

Afterwards, I remove the first-row elements and declare a new list to store only int values, I will calculate the "**Tx Packets(TCP Streams)**" and "**Rx Packets(PDR %)**" using the formula provided with this

assignment.

```
19  # pop the first element of each list because that is just the title for column
20  str_packets.pop(0)
21  str_tx.pop(0)
22  str_rx.pop(0)
23  # declare variables for final strings
24  int_packets = []
25  int_tx = []
26  int_rx = []
27  # convert each item in each string
28  for val in str_packets:
29      int_packets.append(int(val))
30  for val in str_tx:
31      int_tx.append(int(val))
32  for val in str_rx:
33      int_rx.append(int(val))
34  # declare final list
35  tx_calc = []
36  rx_calc = []
37  # multiply each by each and put into final list
38  for i in range(len(int_packets)):
39      tx_calc.append(round((int_tx[i] / int_packets[i]) * 100))
40      rx_calc.append(round((int_rx[i] / int_packets[i]) * 100))
```

Lastly, I save the file into "**Task2.tr**" which I will use for plotting with GNUPLOT on a Linux virtual machine.

```
41  # save into file
42  textfile = open('Task2.tr', 'w')
43  for i in range(len(tx_calc)):
44      text = str(rx_calc[i]) + '   ' + str(tx_calc[i])
45      textfile.write(text)
46      textfile.write('\n')
47  textfile.close()
```

This is the output of running this code:

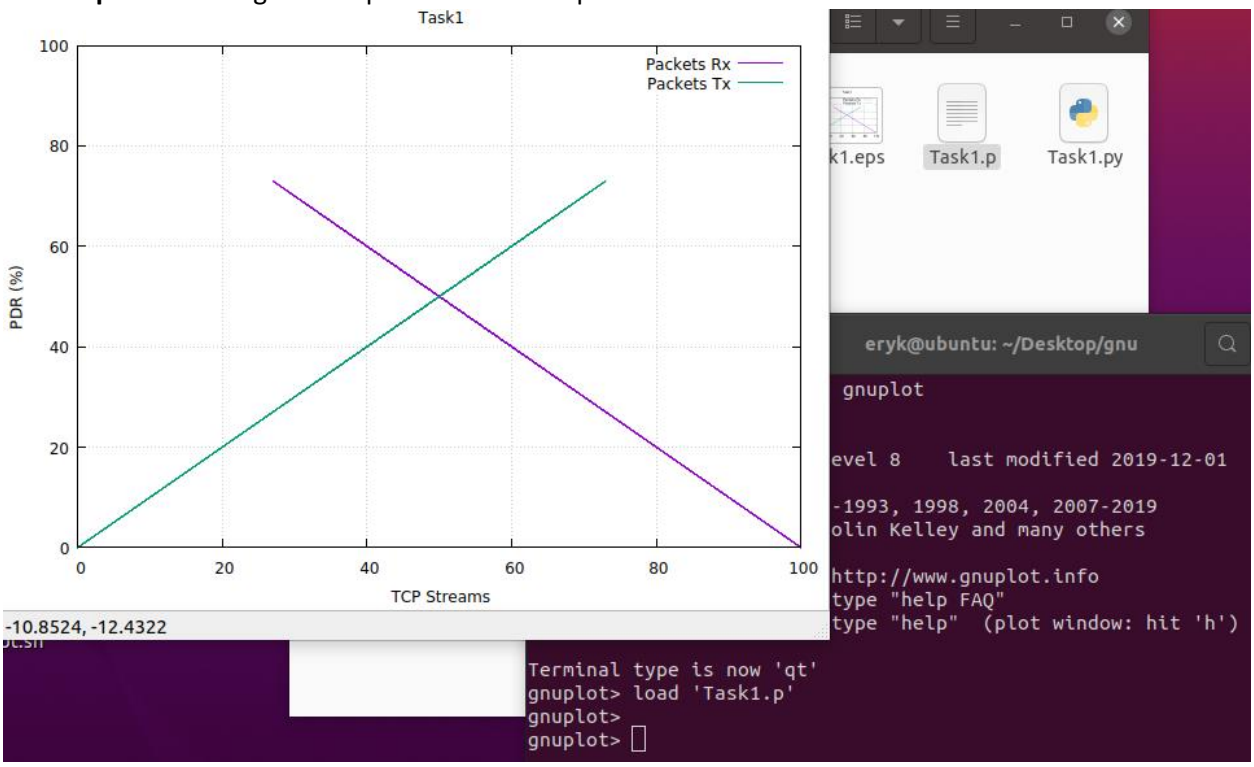
CA2 > Task1.tr

```
559   33  67
560   50  50
561   50  50
562   33  67
563   33  67
564   33  67
565   33  67
566   33  67
567
```

After switching to my Linux virtual machine, I typed up a simple **GNUPLOT** "**Task1.p**" file where I plot the contents of the "**Task1.tr**" file as instructed in the assignment briefing for Task 1.

```
set title "Task1"
set xlabel "TCP Streams"
set ylabel "PDR (%)"

set xrange [0:100]
set yrange [0:100]

set grid

set boxwidth 10

set key top right

plot "/home/eryk/Desktop/gnu/Task1.tr" using 1:2 with line title "Packets Rx", "/home/eryk/Desktop/gnu/Task1.tr" using 2:2 with line title "Packets Tx" ;
replot

set term post eps enhan color 'Helvetica, 31'
set out "Task1.eps" ; replot
```

The **output** of running this simple **GNUPLOT** script is:



Task 2:

Firstly, I import the required libraries: **requests**, **bs4(BeautifulSoup)** and **pandas**. I take input for the job title and number of pages that I would like to scrape from the website "**https://www.irishjobs.ie/**". I make an **extract** method to return the content of the website using a "**BeautifulSoup**" html parser.

```
1   import requests
2   from bs4 import BeautifulSoup
3   import pandas as pd
4   # take in variables
5   jobTitleIn = input('>Enter desired job title: ')
6   pagesAmount = int(input('>Enter amount of pages to scrape: '))
7   # declare a list to store the job objects
8   jobList = []
9   # method to extract the soup page data
10  def extract(jobtitle, page):
11      # referer url and user agent for our browser
12      headers  = {'Referer' : 'https://www.irishjobs.ie/', 'User-Agent': 'Mozilla/5.0 (Windows
13      # formatted string with job title and page number
14      website = f'https://www.irishjobs.ie/ShowResults.aspx?Keywords={jobtitle}&Page={page}'
15      r = requests.get(website, headers = headers)
16      # save parsed html content into soup variable and return it
17      soup = BeautifulSoup(r.content, 'html.parser')
18      return soup
```

In the **transform** method using the extracted content, I find all the "**divs**" with the class "**module job-result**" and I save it to a variable called "**divs**" which I will use to iterate over and find all the required job titles, companies, dates posted, post links and descriptions. These I save to a temporary job object which I then append to the "**jobList**".

```
19  # method to transform the soup page data
20  def transform(soup):
21      divs = soup.find_all('div', class_ = 'module job-result')
22      for item in divs:
23          # get data into variables
24          title = item.find('h2').find('a').text
25          company = item.find('h3').find('a').text
26          date = item.find('li', class_ = 'updated-time').text
27          link = item.find('h2').find('a').get("href")
28          description = item.find('p').find('span').text.replace('\n', '')
29          # job object that holds variables
30          job = {
31              'Job Title': title,
32              'Company': company,
33              'Date Posted': date,
34              'Post Link': 'https://www.irishjobs.ie' + link,
35              'Description': description }
36          # add job object to jobList
37          jobList.append(job)
38      return
```

After that, I loop over the number of pages that I have specified for a specific job title, I **extract** the content of the page number and I use the **transform** method to scrap the required data and append it to the "**jobList**". Once the loop is complete, I convert the "**jobList**" as a **DataFrame** object called "**df**", I declare the filename as a variable and then I put the **jobList DataFrame** object to a **csv** file.

```
39    # loop to iterate over amount of pages
40    for i in range(1, (pagesAmount + 1)):
41        print(f'Scraping Page: {int(i)}')
42        pageData = extract(jobTitleIn, i)
43        transform(pageData)
44    # save jobList as a data frame and save data frame to csv format
45    df = pd.DataFrame(jobList)
46    filename = 'Task2.csv'
47    df.to_csv(filename)
48    print(f'Successfully Scrapped {len(jobList)} jobs! ')
```

This is the **output** of the code:

```
>Enter desired job title: Python
>Enter amount of pages to scrape: 4
Scraping Page: 1
Scraping Page: 2
Scraping Page: 3
Scraping Page: 4
Successfully Scrapped 97 jobs!
PS I:\Year 2\Semester 4\Automation\Scripts\CA2>
```

	A	B	C	D	E	F	G
1		Job Title	Company	Date Posted	Post Link	Description	
2	0	Python (Rates Risk) D	Brightwater	Updated 08/05/2022	https://www.irish	Some of the key	
3	1	Python Developer	Reperio Human Capital Ltd	Updated 08/05/2022	https://www.irish	I am on the sear	
4	2	Senior Python Develo	Spring	Updated 07/05/2022	https://www.irish	My client a lead	
5	3	Python Developer	HedgeServ	Updated 05/05/2022	https://www.irish	The Back Office	
6	4	Python Software Engi	Reperio Human Capital Ltd	Updated 06/05/2022	https://www.irish	I am working wi	
7	5	Senior Python Develo	EOLAS â€“ IT RECRUITMENT	Updated 06/05/2022	https://www.irish	Python Develop	
8	6	Senior Python Backen	Reperio Human Capital Ltd	Updated 06/05/2022	https://www.irish	3+ years using P	
9	7	Python Developer - M	Reperio Human Capital Ltd	Updated 06/05/2022	https://www.irish	Python Develop	
10	8	Backend Developer (F	Reperio Human Capital Ltd	Updated 06/05/2022	https://www.irish	Backend Engine	
11	9	Python Developer	Reperio Human Capital Ltd	Updated 05/05/2022	https://www.irish	A global tech co	
12	10	Python Engineer	Stelfox	Updated 05/05/2022	https://www.irish	You can join one	
13	11	Python Developer	Berkley Recruitment Group	Updated 04/05/2022	https://www.irish	My client, a cutt	
14	12	Senior Python Develo	EOLAS â€“ IT RECRUITMENT	Updated 04/05/2022	https://www.irish	Our Dublin base	
15	13	Mid/Senior Python De	Reperio Human Capital Ltd	Updated 04/05/2022	https://www.irish	Using extensive	
16	14	Senior Python Develo	FRS Recruitment	Updated 02/05/2022	https://www.irish	I am recruiting f	
17	15	Senior Python Develo	Harvey Nash	Updated 30/04/2022	https://www.irish	Harvey Nash are	
18	16	Python Developer	Computer Futures	Updated 30/04/2022	https://www.irish	My client is look	

Task 3:

Firstly, I import the required libraries: **openpyxl**, **pathlip(Path)**, **smtplib**, **ssl**, **email.message(EmailMessage)**. I declare the lists to store the emails, may dues, numbers. I declare a path to the "**task3list.xlsx**" called "**list_file**" and I load the openpyxl workbook and call it "**excel_file**". I then make a reference to the **currently active sheet** and iterate over all 3rd and 8th columns saving them into the "**email**" and "**may dues**" lists.

```
1   import openpyxl
2   from pathlib import Path
3   import smtplib, ssl
4   from email.message import EmailMessage
5   # declare arrays to store emails and the dues and their indexes
6   emails = []
7   may_dues = []
8   nums = []
9   # set path of the xlsx file, change the path if program elsewhere
10  list_file = Path('I:\Year 2\Semester 4\Automation\Scripts\CA2', 'task3list.xlsx')
11  # read from the excel file
12  excel_file = openpyxl.load_workbook(list_file)
13  # take the excel file into a variable
14  sheet = excel_file.active
15  # add emails and the dues into an array
16  for row in sheet.rows:
17      emails.append(row[3].value)
18      may_dues.append(row[8].value)
```

Since the first initial 3 and last 2 rows in the excel file are only empty, I must **remove** that useless data of "**None**" objects including the 4th row which contains the labels for each **column**. After that I check which elements in the "**may dues**" list contain the **None** type object, and I save their index to the "**nums**" list. I also declare the "**receivers**" list and I iterate over the amount of **None** type objects found in the "**may dues**" list while appending them to the "**receivers**" list by their index.

```
19  # remove first 4 useless rows that are not needed
20  for x in range(3, -1, -1):
21      emails.pop(x)
22      may_dues.pop(x)
23  # remove last 2 useless rows (because of the line 18 and 19 added)
24  for x in range(0, 2):
25      emails.pop()
26      may_dues.pop()
27  # get index of None in may_dues
28  nums = [i for i in range(len(may_dues)) if may_dues[i] == None]
29  # declare receivers
30  receivers = []
31  # add receivers
32  for i in range(len(nums)):
33      receivers.append(emails[nums[i]])
```

Lastly, I iterate over the amount of **None** type objects in the "**nums**" list where I declare an **EmailMessage** object, of which I set the content to a basic message. I set the subject, the author and the iterated receiver's email, as well as the default ssl context. For this part, I am using the Google **smtp relay** at port **587** where I need to have **access for less secure apps enabled**. Once I have that part complete, I start the **tls connection** where I login using my **email** and **password**. Finally, I am able send the message to the receivers.

```
35  for i in range(len(nums)):
36      message = EmailMessage()
37      # set content, subject, from(login email) and to
38      message.set_content("""
39      This is an automated message.
40      You have not paid your dues yet.
41      Please do so.
42
43      Thank you for reading this message!
44      """)
45      message['Subject'] = 'Dues Not Paid!'
46      # change this to your login email, used below to login
47      message['From'] = 'l00157413@gmail.com'
48      message['To'] = emails[nums[i]]
49      # create ssl context
50      context = ssl.create_default_context()
51      # login into gmail account(THIS IS IMPORTANT: make sure that access for less secure apps
52      # is turned off using this link: https://myaccount.google.com/lesssecureapps )
53      with smtplib.SMTP('smtp.gmail.com', port = 587) as smtp:
54          smtp.starttls(context = context)
55          # login(change the !ZAQ!2wsx to your own password, google recommended)
56          smtp.login(message['From'], '!ZAQ!2wsx')
57          # finally send message
58          smtp.send_message(message)
```

The **output** of the code depends on the Google smtp relay, but the email message should be received within a minute. This is the message that I get on my own personal email server: