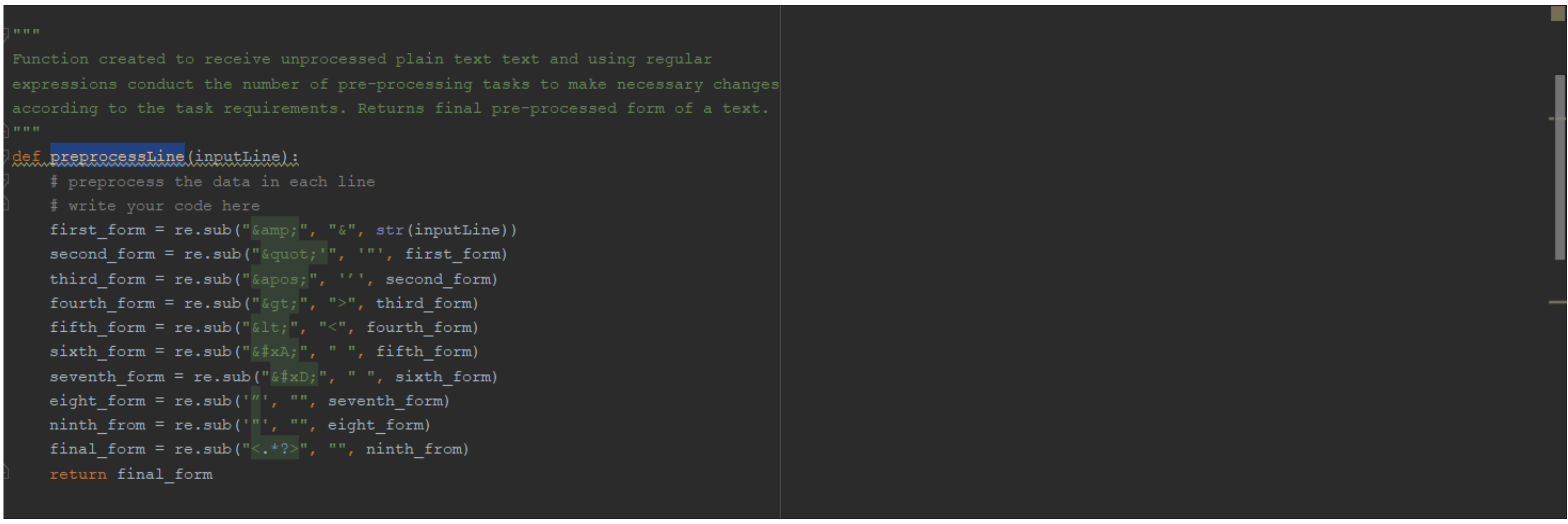
Documentation on the FIT9133 Assignment #2
Building a Text Parser
Semester 1 2019

Student Name: Semen Mordovin

Student ID:29403413

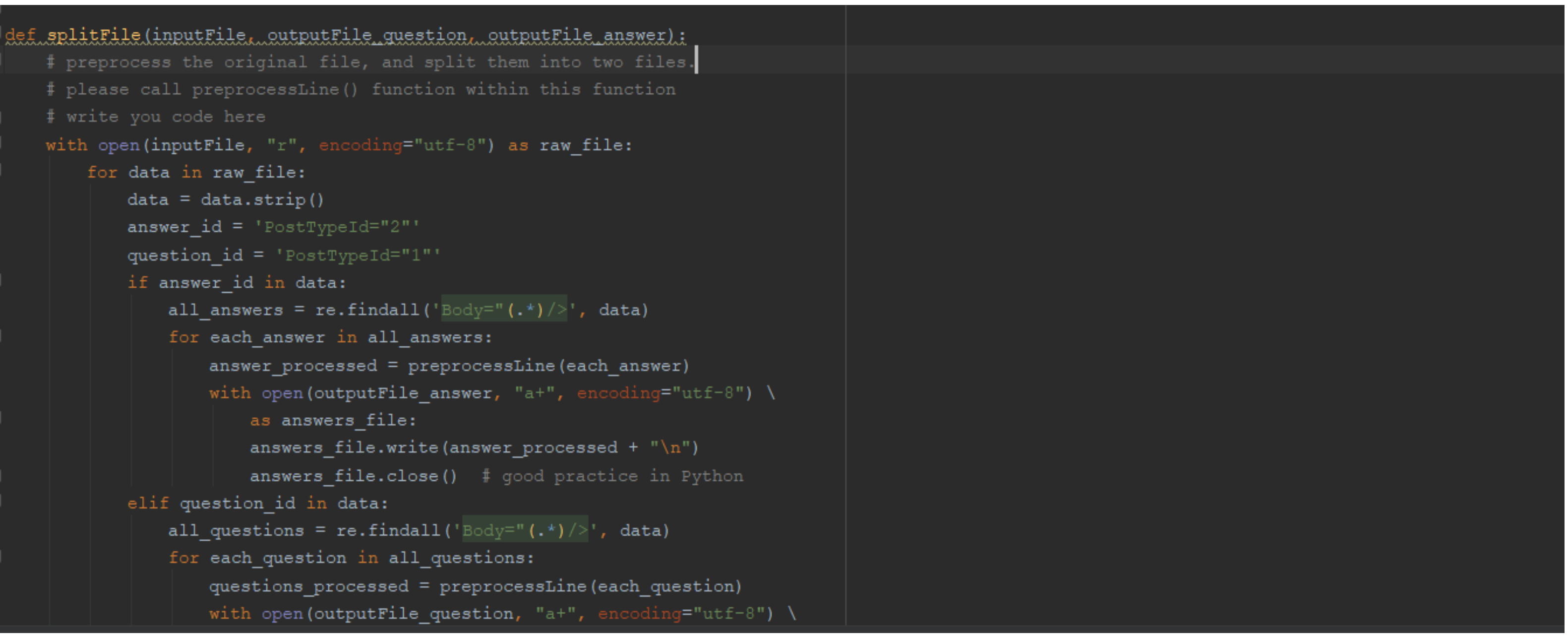
Task 1: Handling with File Contents and Preprocessing

In this task the main idea was is to receive .xml file, preprocess data in it and create to clean separate files with data we need: Answer posts and Question post. This is how it looks like. This image is a representation of preprocessing function, which, get rid of a “trash” in file:



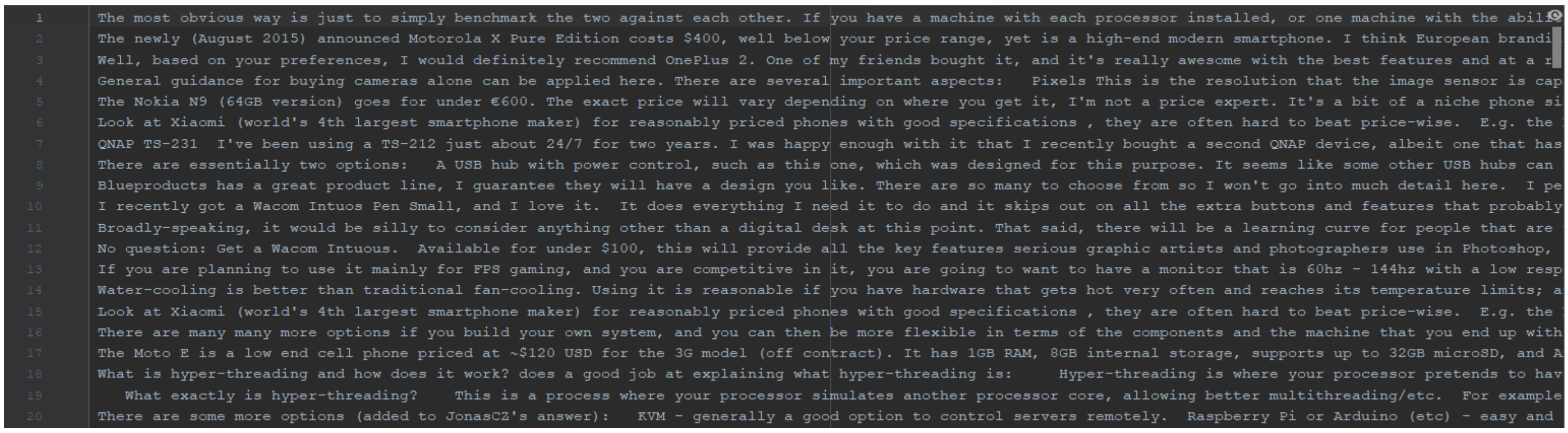
```python
"""
Function created to receive unprocessed plain text text and using regular
expressions conduct the number of pre-processing tasks to make necessary changes
according to the task requirements. Returns final pre-processed form of a text.
"""
def preprocessLine(inputLine):
    # preprocess the data in each line
    # write your code here
    first_form = re.sub("&amp;", "&", str(inputLine))
    second_form = re.sub("&quot;'", '"', first_form)
    third_form = re.sub("&apos;", '′', second_form)
    fourth_form = re.sub("&gt;", ">", third_form)
    fifth_form = re.sub("&lt;", "<", fourth_form)
    sixth_form = re.sub("&#xA;", " ", fifth_form)
    seventh_form = re.sub("&#xD;", " ", sixth_form)
    eight_form = re.sub('″', "", seventh_form)
    ninth_from = re.sub('"', "", eight_form)
    final_form = re.sub("<.*?>", "", ninth_from)
    return final_form
```

This part of a Task 1 is dedicated to read file from a file, look for posts that are represents Answers and Questions:



```python
def splitFile(inputFile, outputFile_question, outputFile_answer):
    # preprocess the original file, and split them into two files.
    # please call preprocessLine() function within this function
    # write you code here
    with open(inputFile, "r", encoding="utf-8") as raw_file:
        for data in raw_file:
            data = data.strip()
            answer_id = 'PostTypeId="2"'
            question_id = 'PostTypeId="1"'
            if answer_id in data:
                all_answers = re.findall('Body="(.*)/>', data)
                for each_answer in all_answers:
                    answer_processed = preprocessLine(each_answer)
                    with open(outputFile_answer, "a+", encoding="utf-8") \
                        as answers_file:
                        answers_file.write(answer_processed + "\n")
                        answers_file.close()  # good practice in Python
            elif question_id in data:
                all_questions = re.findall('Body="(.*)/>', data)
                for each_question in all_questions:
                    questions_processed = preprocessLine(each_question)
                    with open(outputFile_question, "a+", encoding="utf-8") \
```

This is how the output of this program looks like after you Run it:

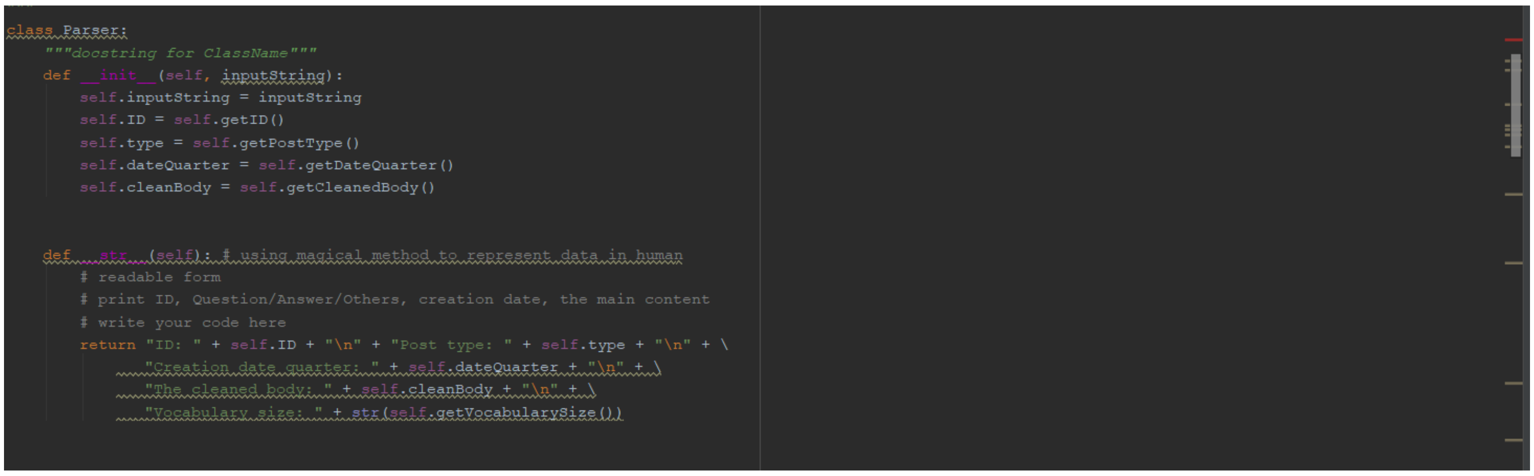


```
The most obvious way is just to simply benchmark the two against each other. If you have a machine with each processor installed, or one machine with the abili
The newly (August 2015) announced Motorola X Pure Edition costs $400, well below your price range, yet is a high-end modern smartphone. I think European brandi
Well, based on your preferences, I would definitely recommend OnePlus 2. One of my friends bought it, and it's really awesome with the best features and at a r
General guidance for buying cameras alone can be applied here. There are several important aspects:   Pixels This is the resolution that the image sensor is cap
The Nokia N9 (64GB version) goes for under €600. The exact price will vary depending on where you get it, I'm not a price expert. It's a bit of a niche phone si
Look at Xiaomi (world's 4th largest smartphone maker) for reasonably priced phones with good specifications , they are often hard to beat price-wise.  E.g. the
QNAP TS-231  I've been using a TS-212 just about 24/7 for two years. I was happy enough with it that I recently bought a second QNAP device, albeit one that has
There are essentially two options:   A USB hub with power control, such as this one, which was designed for this purpose. It seems like some other USB hubs can
Blueproducts has a great product line, I guarantee they will have a design you like. There are so many to choose from so I won't go into much detail here.  I pe
I recently got a Wacom Intuos Pen Small, and I love it.  It does everything I need it to do and it skips out on all the extra buttons and features that probably
Broadly-speaking, it would be silly to consider anything other than a digital desk at this point. That said, there will be a learning curve for people that are
No question: Get a Wacom Intuous.  Available for under $100, this will provide all the key features serious graphic artists and photographers use in Photoshop,
If you are planning to use it mainly for FPS gaming, and you are competitive in it, you are going to want to have a monitor that is 60hz - 144hz with a low resp
Water-cooling is better than traditional fan-cooling. Using it is reasonable if you have hardware that gets hot very often and reaches its temperature limits; a
Look at Xiaomi (world's 4th largest smartphone maker) for reasonably priced phones with good specifications , they are often hard to beat price-wise.  E.g. the
There are many many more options if you build your own system, and you can then be more flexible in terms of the components and the machine that you end up with
The Moto E is a low end cell phone priced at ~$120 USD for the 3G model (off contract). It has 1GB RAM, 8GB internal storage, supports up to 32GB microSD, and A
What is hyper-threading and how does it work? does a good job at explaining what hyper-threading is:     Hyper-threading is where your processor pretends to hav
   What exactly is hyper-threading?    This is a process where your processor simulates another processor core, allowing better multithreading/etc.  For example
There are some more options (added to JonasCZ's answer):   KVM - generally a good option to control servers remotely.  Raspberry Pi or Arduino (etc) - easy and
```

The two files created with answers and questions. Each line represented with clean data, preprocessed by our program. Then we moving to the next step – building class for this text analysis.

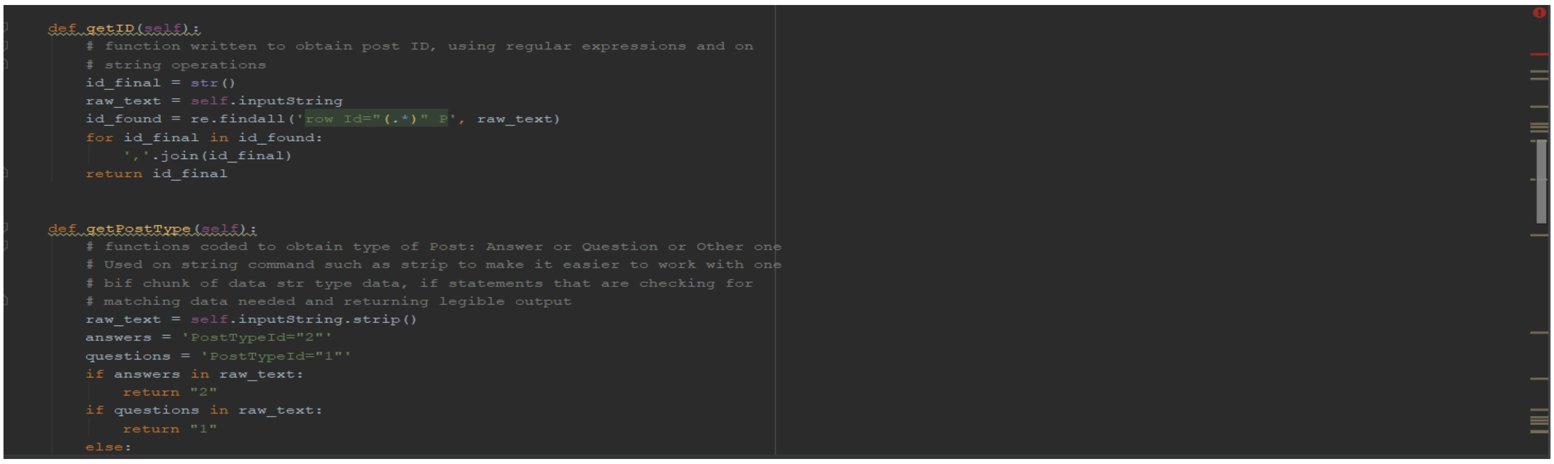
Task 2: Building a Class for Data Analysis

Having been given certain name of a Class and functions we develop a logic for: String representation of a program, getting the ID of the post, type of post, its Creation Date, the cleaned text of post and count the number of Unique Words, or showing so called Vocabulary Size. This is how code looks like:



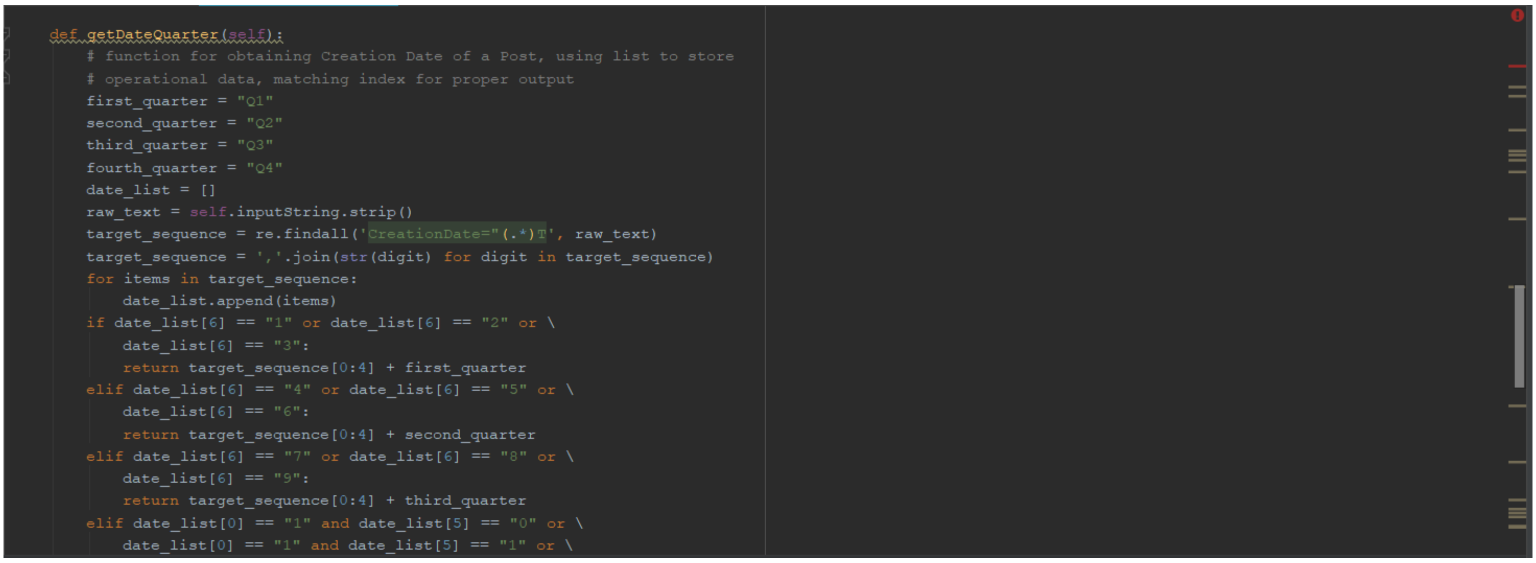
```python
class Parser:
    """docstring for ClassName"""
    def __init__(self, inputString):
        self.inputString = inputString
        self.ID = self.getID()
        self.type = self.getPostType()
        self.dateQuarter = self.getDateQuarter()
        self.cleanBody = self.getCleanedBody()


    def __str__(self): # using magical method to represent data in human
        # readable form
        # print ID, Question/Answer/Others, creation date, the main content
        # write your code here
        return "ID: " + self.ID + "\n" + "Post type: " + self.type + "\n" + \
            "Creation date quarter: " + self.dateQuarter + "\n" + \
            "The cleaned body: " + self.cleanBody + "\n" + \
            "Vocabulary size: " + str(self.getVocabularySize())
```

There are tow functions: Magical method string converting data in human readable output and main constructor of a Class Parser.



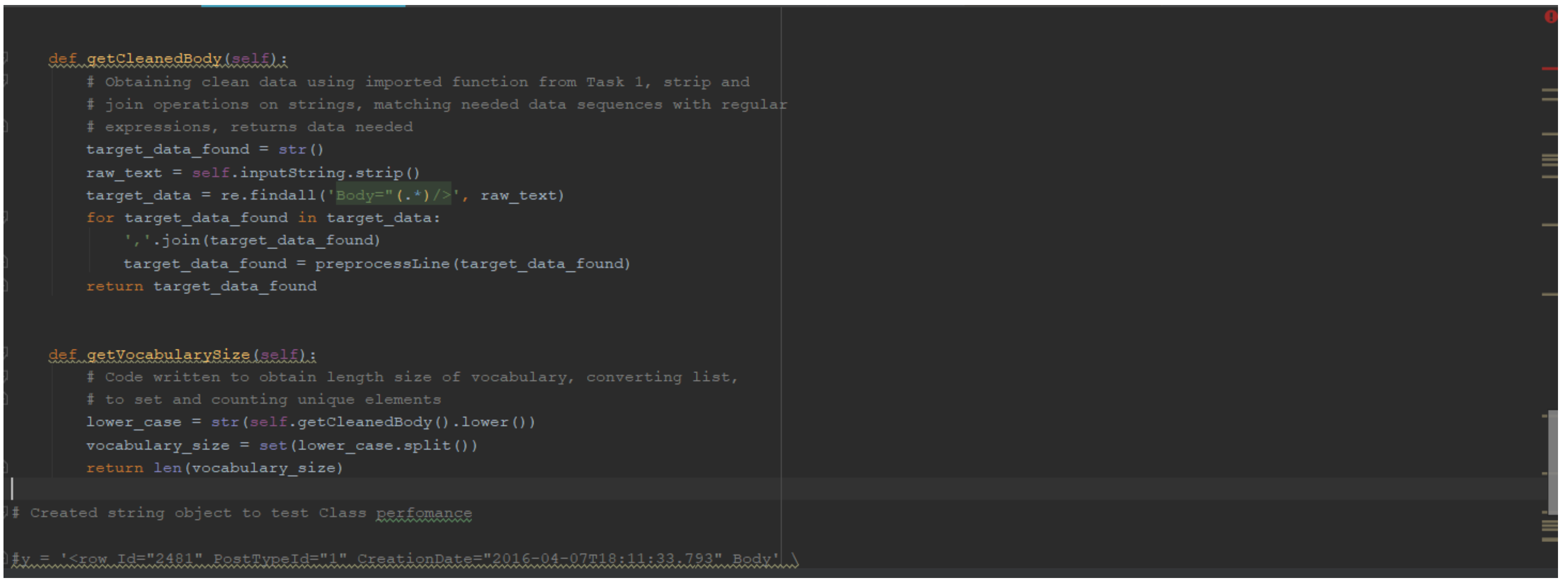
```python
def getID(self):
    # function written to obtain post ID, using regular expressions and on
    # string operations
    id_final = str()
    raw_text = self.inputString
    id_found = re.findall('row Id="(.*)" P', raw_text)
    for id_final in id_found:
        ','.join(id_final)
    return id_final


def getPostType(self):
    # functions coded to obtain type of Post: Answer or Question or Other one
    # Used on string command such as strip to make it easier to work with one
    # bif chunk of data str type data, if statements that are checking for
    # matching data needed and returning legible output
    raw_text = self.inputString.strip()
    answers = 'PostTypeId="2"'
    questions = 'PostTypeId="1"'
    if answers in raw_text:
        return "2"
    if questions in raw_text:
        return "1"
    else:
```

There goes a function to obtain ID of post and function getting Type of each Post.



```python
def getDateQuarter(self):
    # function for obtaining Creation Date of a Post, using list to store
    # operational data, matching index for proper output
    first_quarter = "Q1"
    second_quarter = "Q2"
    third_quarter = "Q3"
    fourth_quarter = "Q4"
    date_list = []
    raw_text = self.inputString.strip()
    target_sequence = re.findall('CreationDate="(.*)T', raw_text)
    target_sequence = ','.join(str(digit) for digit in target_sequence)
    for items in target_sequence:
        date_list.append(items)
    if date_list[6] == "1" or date_list[6] == "2" or \
        date_list[6] == "3":
        return target_sequence[0:4] + first_quarter
    elif date_list[6] == "4" or date_list[6] == "5" or \
        date_list[6] == "6":
        return target_sequence[0:4] + second_quarter
    elif date_list[6] == "7" or date_list[6] == "8" or \
        date_list[6] == "9":
        return target_sequence[0:4] + third_quarter
    elif date_list[0] == "1" and date_list[5] == "0" or \
        date_list[0] == "1" and date_list[5] == "1" or \
```

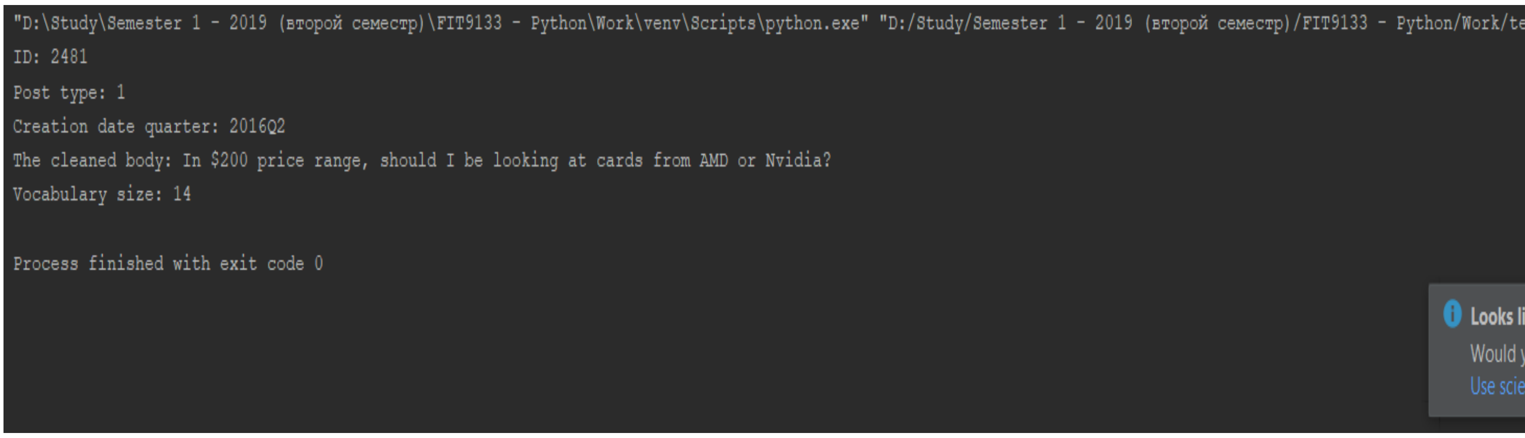
This function gets Creation Date



```python
    def getCleanedBody(self):
        # Obtaining clean data using imported function from Task 1, strip and
        # join operations on strings, matching needed data sequences with regular
        # expressions, returns data needed
        target_data_found = str()
        raw_text = self.inputString.strip()
        target_data = re.findall('Body="(.*)/>', raw_text)
        for target_data_found in target_data:
            ','.join(target_data_found)
            target_data_found = preprocessLine(target_data_found)
        return target_data_found


    def getVocabularySize(self):
        # Code written to obtain length size of vocabulary, converting list,
        # to set and counting unique elements
        lower_case = str(self.getCleanedBody().lower())
        vocabulary_size = set(lower_case.split())
        return len(vocabulary_size)

# Created string object to test Class perfomance

#y = '<row Id="2481" PostTypeId="1" CreationDate="2016-04-07T18:11:33.793" Body' \
```

These two functions clean the text and count Vocabulary Size.

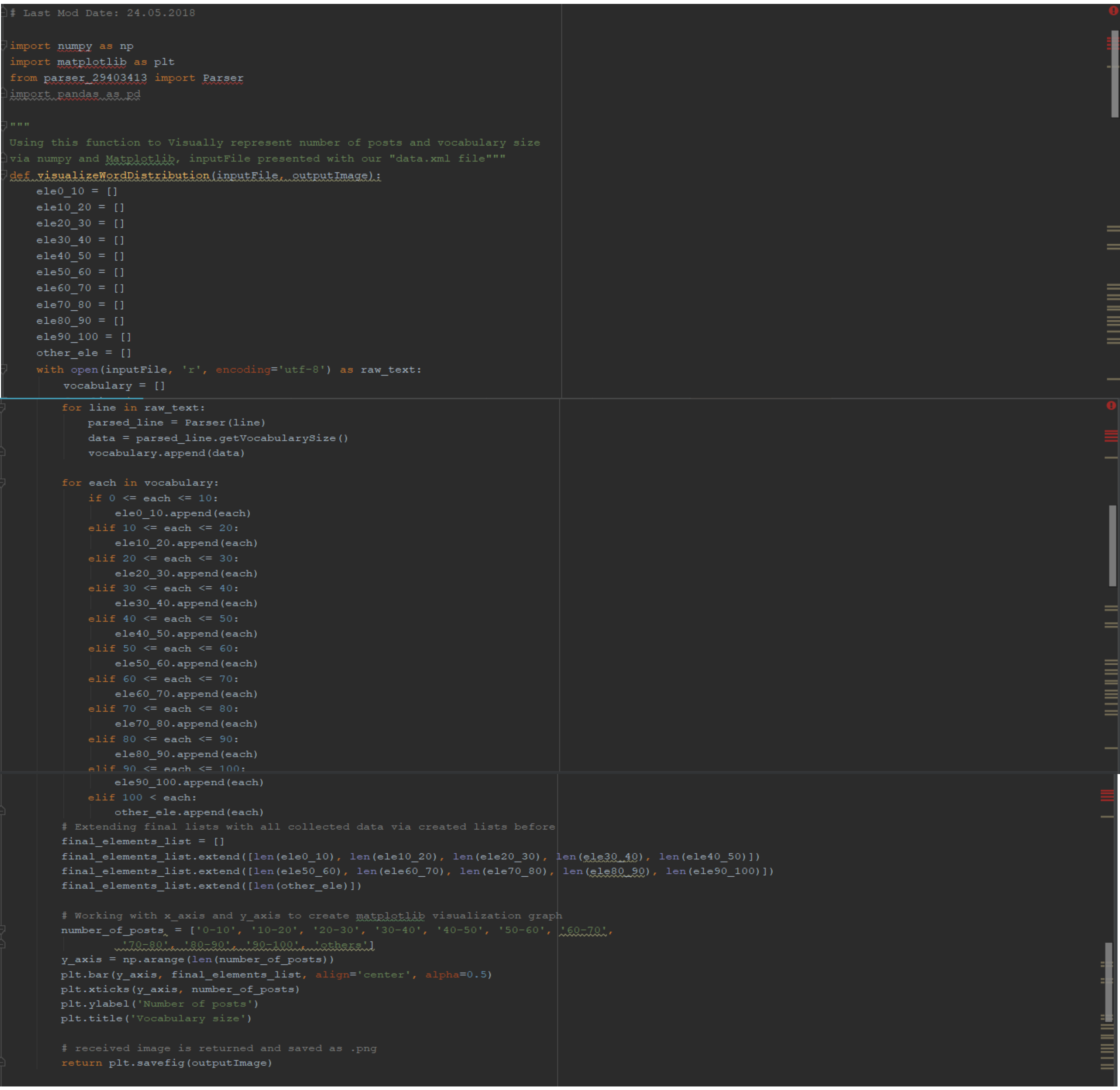
And there goes the output of Task 2:



```
"D:\Study\Semester 1 - 2019 (второй семестр)\FIT9133 - Python\Work\venv\Scripts\python.exe" "D:/Study/Semester 1 - 2019 (второй семестр)/FIT9133 - Python/Work/te
ID: 2481
Post type: 1
Creation date quarter: 2016Q2
The cleaned body: In $200 price range, should I be looking at cards from AMD or Nvidia?
Vocabulary size: 14

Process finished with exit code 0
```

Task 3: Analyzing the File for Data Visualization

Task three is aimed at Visualization part of the whole task, we have to represent, graphically a statistic of number of posts and Vocabulary Size for each one. This function is concerned with this task:



```python
# Last Mod Date: 24.05.2018

import numpy as np
import matplotlib as plt
from parser_29403413 import Parser
import pandas as pd

"""
Using this function to Visually represent number of posts and vocabulary size
via numpy and Matplotlib, inputFile presented with our "data.xml file"""
def visualizeWordDistribution(inputFile, outputImage):
    ele0_10 = []
    ele10_20 = []
    ele20_30 = []
    ele30_40 = []
    ele40_50 = []
    ele50_60 = []
    ele60_70 = []
    ele70_80 = []
    ele80_90 = []
    ele90_100 = []
    other_ele = []
    with open(inputFile, 'r', encoding='utf-8') as raw_text:
        vocabulary = []
        for line in raw_text:
            parsed_line = Parser(line)
            data = parsed_line.getVocabularySize()
            vocabulary.append(data)

        for each in vocabulary:
            if 0 <= each <= 10:
                ele0_10.append(each)
            elif 10 <= each <= 20:
                ele10_20.append(each)
            elif 20 <= each <= 30:
                ele20_30.append(each)
            elif 30 <= each <= 40:
                ele30_40.append(each)
            elif 40 <= each <= 50:
                ele40_50.append(each)
            elif 50 <= each <= 60:
                ele50_60.append(each)
            elif 60 <= each <= 70:
                ele60_70.append(each)
            elif 70 <= each <= 80:
                ele70_80.append(each)
            elif 80 <= each <= 90:
                ele80_90.append(each)
            elif 90 <= each <= 100:
                ele90_100.append(each)
            elif 100 < each:
                other_ele.append(each)
        # Extending final lists with all collected data via created lists before
        final_elements_list = []
        final_elements_list.extend([len(ele0_10), len(ele10_20), len(ele20_30), len(ele30_40), len(ele40_50)])
        final_elements_list.extend([len(ele50_60), len(ele60_70), len(ele70_80), len(ele80_90), len(ele90_100)])
        final_elements_list.extend([len(other_ele)])

        # Working with x_axis and y_axis to create matplotlib visualization graph
        number_of_posts = ['0-10', '10-20', '20-30', '30-40', '40-50', '50-60', '60-70',
                '70-80', '80-90', '90-100', 'others']
        y_axis = np.arange(len(number_of_posts))
        plt.bar(y_axis, final_elements_list, align='center', alpha=0.5)
        plt.xticks(y_axis, number_of_posts)
        plt.ylabel('Number of posts')
        plt.title('Vocabulary size')

        # received image is returned and saved as .png
        return plt.savefig(outputImage)
```

As it mentioned before, tt should create a .png file with graphical representation.