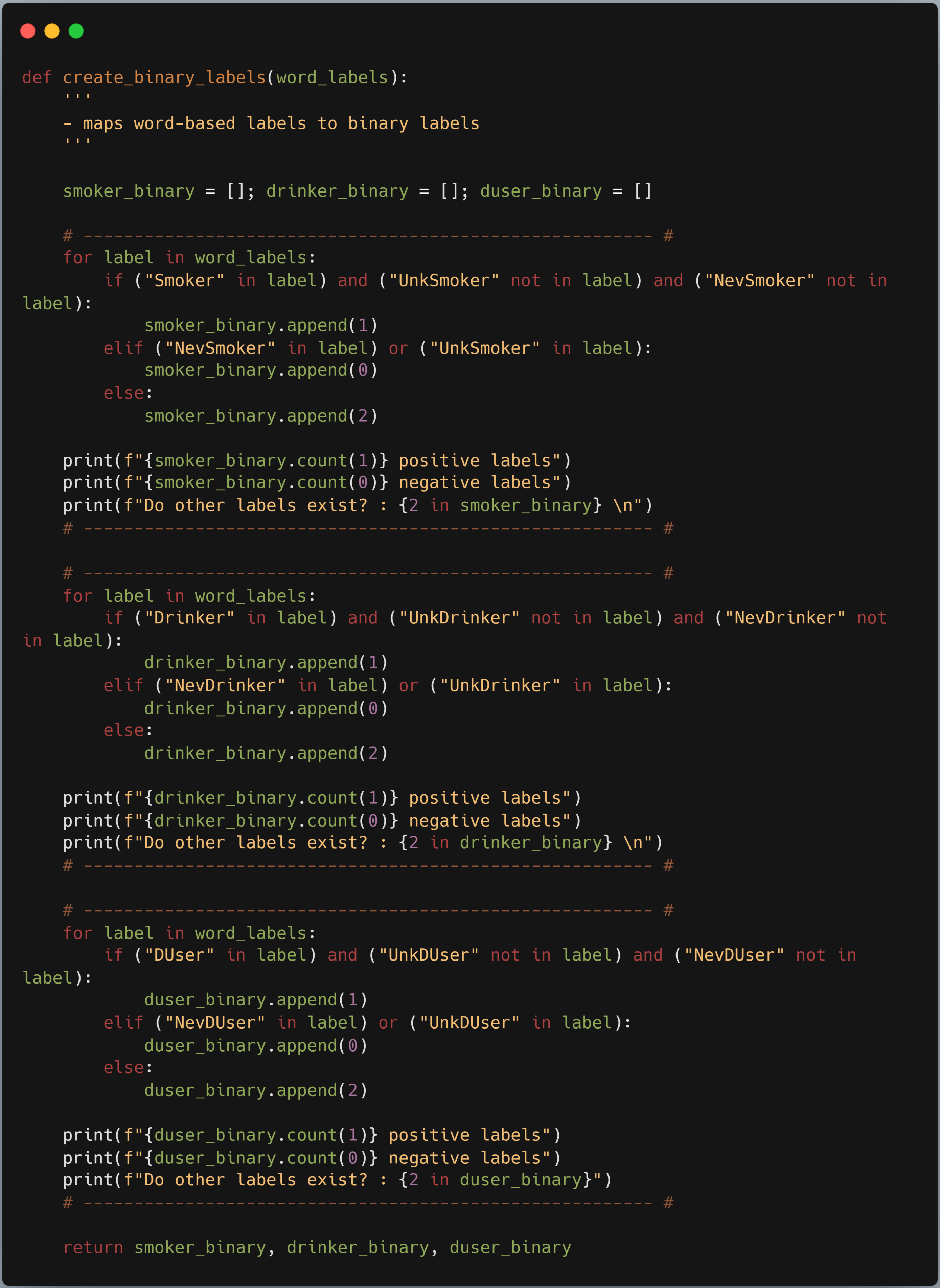
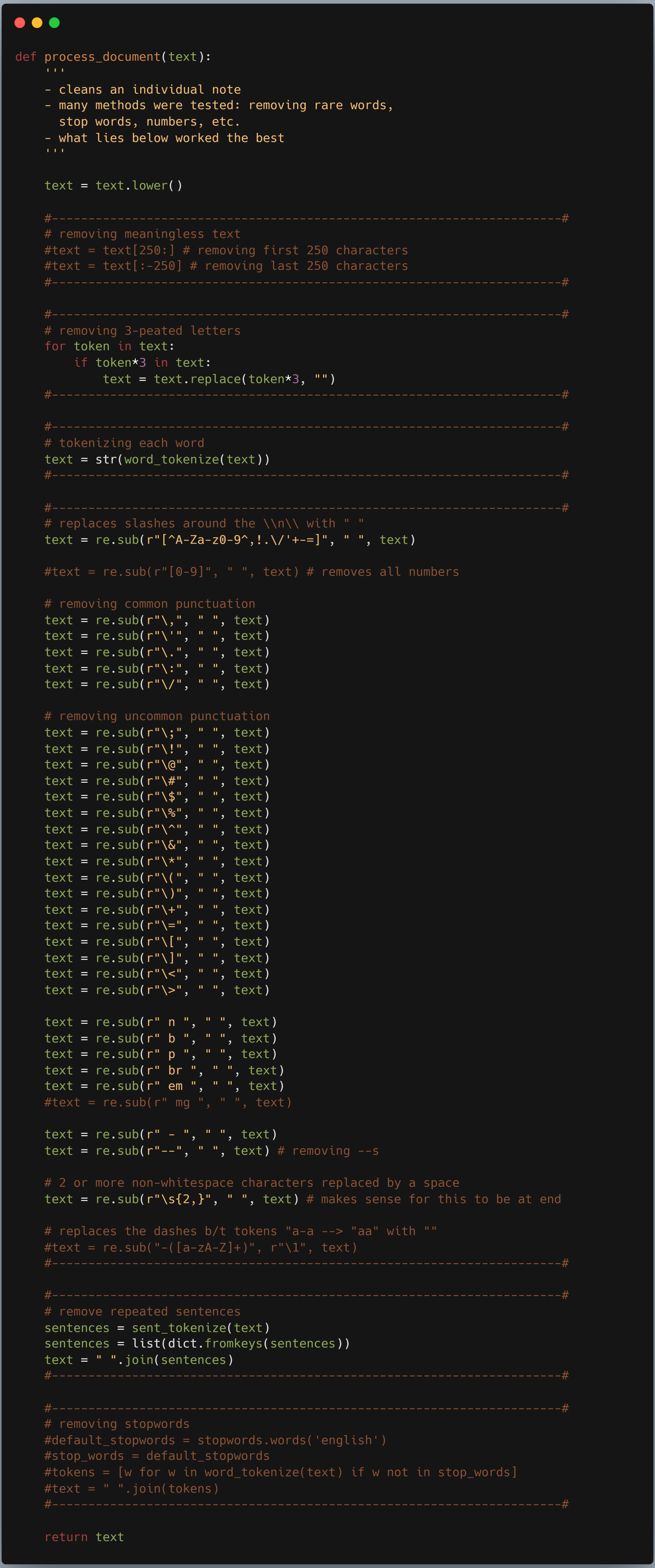
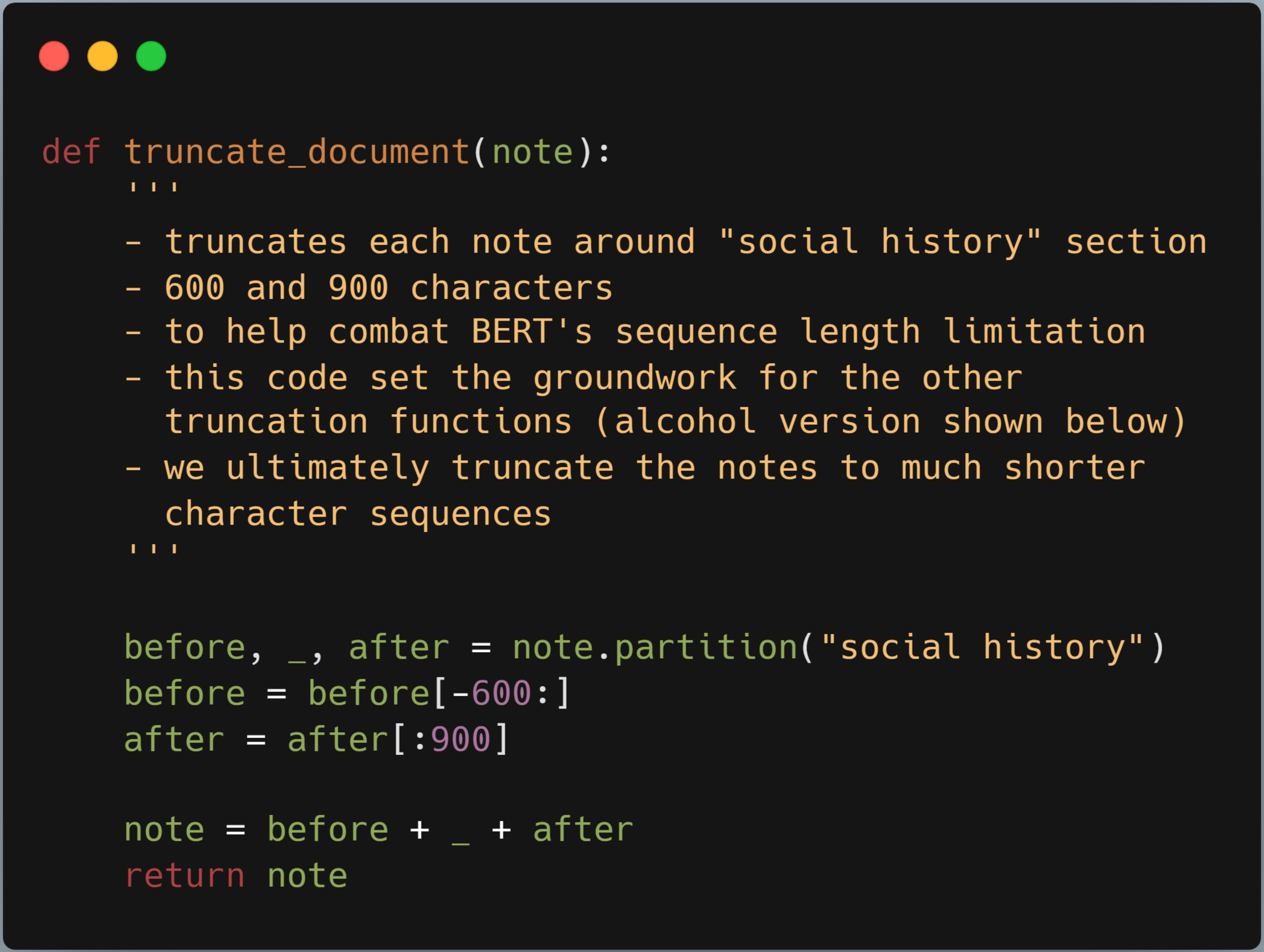


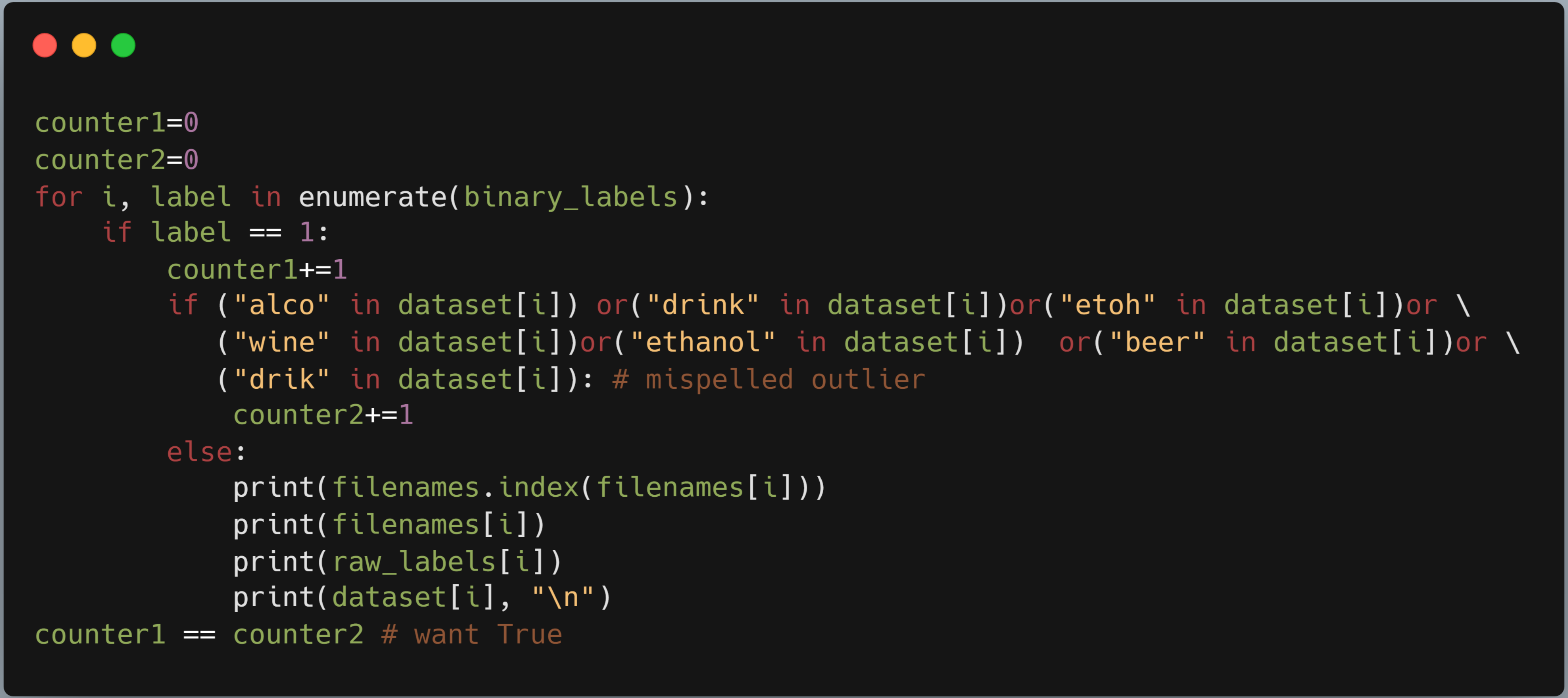
```python
def create_binary_labels(word_labels):
    '''
    - maps word-based labels to binary labels
    '''

    smoker_binary = []; drinker_binary = []; duser_binary = []

    # ------------------------------------------------------- #
    for label in word_labels:
        if ("Smoker" in label) and ("UnkSmoker" not in label) and ("NevSmoker" not in 
label):
            smoker_binary.append(1)
        elif ("NevSmoker" in label) or ("UnkSmoker" in label):
            smoker_binary.append(0)
        else:
            smoker_binary.append(2)

    print(f"{smoker_binary.count(1)} positive labels")
    print(f"{smoker_binary.count(0)} negative labels")
    print(f"Do other labels exist? : {2 in smoker_binary} \n")
    # ------------------------------------------------------- #

    # ------------------------------------------------------- #
    for label in word_labels:
        if ("Drinker" in label) and ("UnkDrinker" not in label) and ("NevDrinker" not 
in label):
            drinker_binary.append(1)
        elif ("NevDrinker" in label) or ("UnkDrinker" in label):
            drinker_binary.append(0)
        else:
            drinker_binary.append(2)

    print(f"{drinker_binary.count(1)} positive labels")
    print(f"{drinker_binary.count(0)} negative labels")
    print(f"Do other labels exist? : {2 in drinker_binary} \n")
    # ------------------------------------------------------- #

    # ------------------------------------------------------- #
    for label in word_labels:
        if ("DUser" in label) and ("UnkDUser" not in label) and ("NevDUser" not in 
label):
            duser_binary.append(1)
        elif ("NevDUser" in label) or ("UnkDUser" in label):
            duser_binary.append(0)
        else:
            duser_binary.append(2)

    print(f"{duser_binary.count(1)} positive labels")
    print(f"{duser_binary.count(0)} negative labels")
    print(f"Do other labels exist? : {2 in duser_binary}")
    # ------------------------------------------------------- #

    return smoker_binary, drinker_binary, duser_binary
```



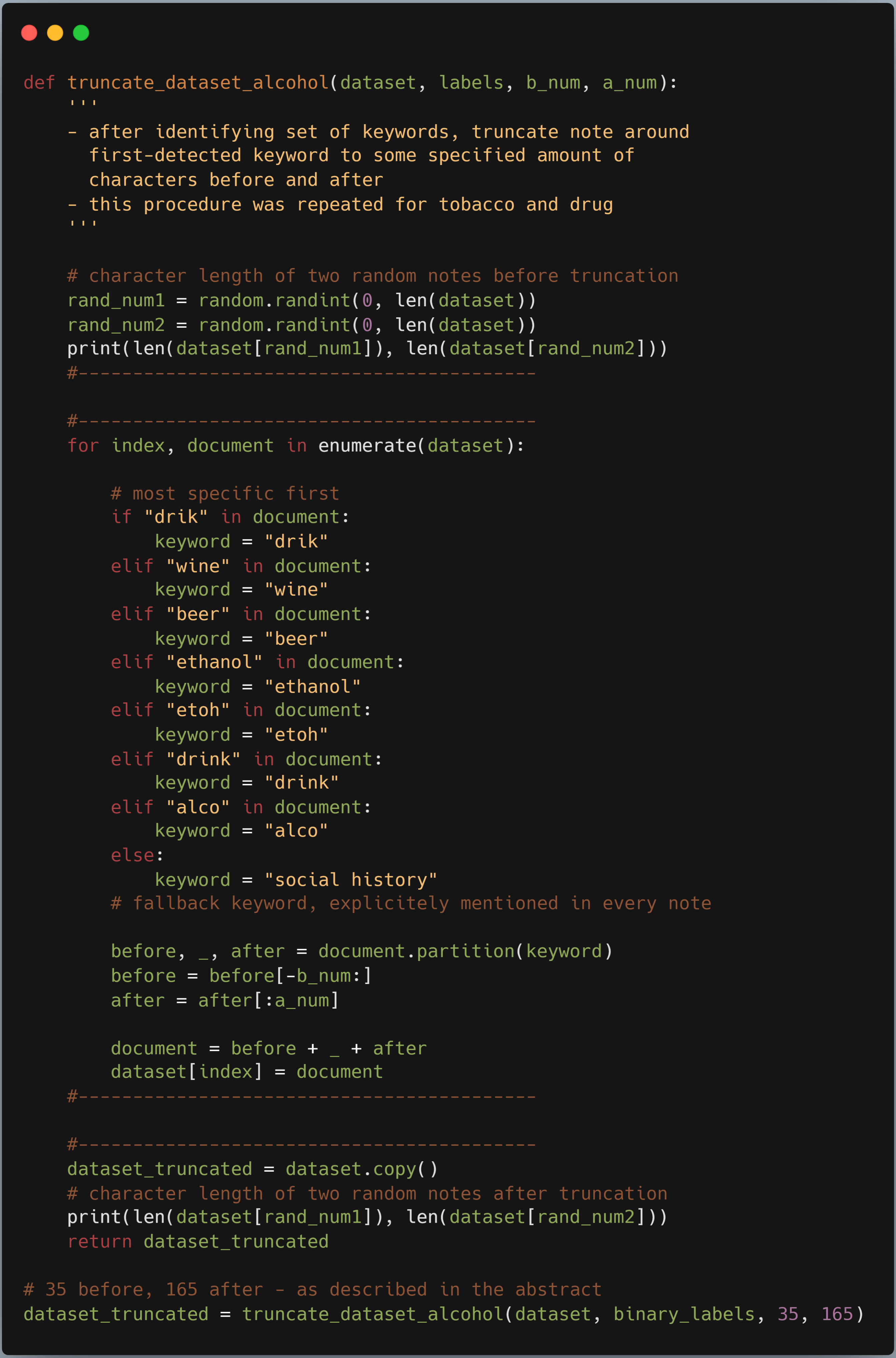
```python
def process_document(text):
    '''
    - cleans an individual note
    - many methods were tested: removing rare words,
      stop words, numbers, etc.
    - what lies below worked the best
    '''

    text = text.lower()

    #-----------------------------------------------------------------------#
    # removing meaningless text
    #text = text[250:] # removing first 250 characters
    #text = text[:-250] # removing last 250 characters
    #-----------------------------------------------------------------------#

    #-----------------------------------------------------------------------#
    # removing 3-peated letters
    for token in text:
        if token*3 in text:
            text = text.replace(token*3, "")
    #-----------------------------------------------------------------------#

    #-----------------------------------------------------------------------#
    # tokenizing each word
    text = str(word_tokenize(text))
    #-----------------------------------------------------------------------#

    #-----------------------------------------------------------------------#
    # replaces slashes around the \\n\\ with " "
    text = re.sub(r"[^A-Za-z0-9^,!.\/'+-=]", " ", text)

    #text = re.sub(r"[0-9]", " ", text) # removes all numbers

    # removing common punctuation
    text = re.sub(r"\,", " ", text)
    text = re.sub(r"\'", " ", text)
    text = re.sub(r"\.", " ", text)
    text = re.sub(r"\:", " ", text)
    text = re.sub(r"\/", " ", text)

    # removing uncommon punctuation
    text = re.sub(r"\;", " ", text)
    text = re.sub(r"\!", " ", text)
    text = re.sub(r"\@", " ", text)
    text = re.sub(r"\#", " ", text)
    text = re.sub(r"\$", " ", text)
    text = re.sub(r"\%", " ", text)
    text = re.sub(r"\^", " ", text)
    text = re.sub(r"\&", " ", text)
    text = re.sub(r"\*", " ", text)
    text = re.sub(r"\(", " ", text)
    text = re.sub(r"\)", " ", text)
    text = re.sub(r"\+", " ", text)
    text = re.sub(r"\=", " ", text)
    text = re.sub(r"\[", " ", text)
    text = re.sub(r"\]", " ", text)
    text = re.sub(r"\<", " ", text)
    text = re.sub(r"\>", " ", text)

    text = re.sub(r" n ", " ", text)
    text = re.sub(r" b ", " ", text)
    text = re.sub(r" p ", " ", text)
    text = re.sub(r" br ", " ", text)
    text = re.sub(r" em ", " ", text)
    #text = re.sub(r" mg ", " ", text)

    text = re.sub(r" - ", " ", text)
    text = re.sub(r"--", " ", text) # removing --s

    # 2 or more non-whitespace characters replaced by a space
    text = re.sub(r"\s{2,}", " ", text) # makes sense for this to be at end

    # replaces the dashes b/t tokens "a-a --> "aa" with ""
    #text = re.sub("-([a-zA-Z]+)", r"\1", text)
    #-----------------------------------------------------------------------#

    #-----------------------------------------------------------------------#
    # remove repeated sentences
    sentences = sent_tokenize(text)
    sentences = list(dict.fromkeys(sentences))
    text = " ".join(sentences)
    #-----------------------------------------------------------------------#

    #-----------------------------------------------------------------------#
    # removing stopwords
    #default_stopwords = stopwords.words('english')
    #stop_words = default_stopwords
    #tokens = [w for w in word_tokenize(text) if w not in stop_words]
    #text = " ".join(tokens)
    #-----------------------------------------------------------------------#

    return text
```



```python
def truncate_document(note):
    '''
    - truncates each note around "social history" section
    - 600 and 900 characters
    - to help combat BERT's sequence length limitation
    - this code set the groundwork for the other
      truncation functions (alcohol version shown below)
    - we ultimately truncate the notes to much shorter
      character sequences
    '''

    before, _, after = note.partition("social history")
    before = before[-600:]
    after = after[:900]

    note = before + _ + after
    return note
```



```python
counter1=0
counter2=0
for i, label in enumerate(binary_labels):
    if label == 1:
        counter1+=1
        if ("alco" in dataset[i]) or("drink" in dataset[i])or("etoh" in dataset[i])or \
           ("wine" in dataset[i])or("ethanol" in dataset[i])  or("beer" in dataset[i])or \
           ("drik" in dataset[i]): # mispelled outlier
            counter2+=1
        else:
            print(filenames.index(filenames[i]))
            print(filenames[i])
            print(raw_labels[i])
            print(dataset[i], "\n")
counter1 == counter2 # want True
```



```python
def truncate_dataset_alcohol(dataset, labels, b_num, a_num):
    '''
    - after identifying set of keywords, truncate note around
      first-detected keyword to some specified amount of
      characters before and after
    - this procedure was repeated for tobacco and drug
    '''

    # character length of two random notes before truncation
    rand_num1 = random.randint(0, len(dataset))
    rand_num2 = random.randint(0, len(dataset))
    print(len(dataset[rand_num1]), len(dataset[rand_num2]))
    #-----------------------------------------

    #-----------------------------------------
    for index, document in enumerate(dataset):

        # most specific first
        if "drik" in document:
            keyword = "drik"
        elif "wine" in document:
            keyword = "wine"
        elif "beer" in document:
            keyword = "beer"
        elif "ethanol" in document:
            keyword = "ethanol"
        elif "etoh" in document:
            keyword = "etoh"
        elif "drink" in document:
            keyword = "drink"
        elif "alco" in document:
            keyword = "alco"
        else:
            keyword = "social history"
        # fallback keyword, explicitely mentioned in every note

        before, _, after = document.partition(keyword)
        before = before[-b_num:]
        after = after[:a_num]

        document = before + _ + after
        dataset[index] = document
    #-----------------------------------------

    #-----------------------------------------
    dataset_truncated = dataset.copy()
    # character length of two random notes after truncation
    print(len(dataset[rand_num1]), len(dataset[rand_num2]))
    return dataset_truncated

# 35 before, 165 after - as described in the abstract
dataset_truncated = truncate_dataset_alcohol(dataset, binary_labels, 35, 165)
```