1. In the event of an error in Python, I want to make sure that certain code can be executed instead of shutting down. How can I implement this code? (Please write in as much detail as possible)

Answer: debugging

Errors displayed in the terminal vary depending on how you structured your program. As for a specific error in code it is a must to know what commonly causes an error, it can be the case; most of the functions and variables need to be written strictly in lower, upper case or Camel case. Symbol; error in using semi-colon, tilde, commas, and enclosing symbols like curly braces, brackets, parenthesis. Indentions; proper use of tabs after declaring a function or when creating loops it is important to always check the indentions as it tells what line of code is inside the other and vice versa. Debugging needs to be performed from time to time before the program gets longer and longer to avoid crashing of program.

1. When text data is loaded, newline characters are inserted in front for various reasons such as blanks, line breaks, and tabs. How would you design an algorithm to remove all newline characters from these text data?

Answer: by utilizing Regular Expression

Regular Expression allows you to create a program way easier by creating patterns out of ‘re module’ so it is needed to be imported, you can create shorter and cleaner programs especially when you are extracting data from a bunch of text files. From what is being asked in the question, create a function called ‘cleaned\_text’ where the updated text data will be formatted. You will need to use ‘re.findall’ before declaring the pattern to go through the loaded data. Blanks, line breaks and tabs are whitespaces in a line, and you can omit it with ‘\s’. Locate words before the whitespaces needed to remove for example ‘ The’ you may create ‘\s\W’ pattern and all whitespace before words that start with upper case will be selected.

text = [

“ ‘;lkhgasdfghj[poiuytrewqwo”,

“ lkjhgfdsxcvbnmoiuytrddfgh,”,

]

updated = []

import re

def cleaned\_text(text)

cleaned\_text(text)

3. How to select the terms with the most occurrences and their frequency in order from the total overlapping terminology? The extracted terms are as follows. (Case insensitive when checking for term duplication.)

**terms = ["apple", "ApPle", "APPLE", "banana", "BANana", "banana", "BaNaNa", … "Kiwi", "KIWI", "KiWi", "kIwI", "KIWi"]**

The list contains more than 10000 terms.

terms = ["apple", "ApPle", "APPLE", "banana", "BANana", "banana", "BaNaNa", … "Kiwi", "KIWI", "KiWi", "kIwI", "KIWi"]

select = []

def most\_occurrence(terms);

for i in terms;

if ‘apple’, ‘banana’ and ‘kiwi’ < 4;

select.append

print(“Terms with the most occurrences is”, select)

else:

print(“

def

enumerate.

There is a list that contains multiple sentences. When you want to remove all the bullet points in that sentence, how can you remove them all? The list consists of the following:

**sentences = [**

**"1. Test sentence",**

**"2) Test sentence",**

**"a. A test sentence",**

**…**

**"a-5) test sentence",**

**"6-14) Test sentences",**

Answer: by applying RegEx in the program

Locate differences in every character, as I said on the other problem RegEx allows you to extract data easily by using patterns,

sentences = [

“1. Test sentence”,

“2) Test sentence”,

“a. A test sentence”,

…

“a-5) test sentence”,

“6-14) Test sentences”,

]

import re

def removed\_bullets(sentences);

“\d.\sT”

“\d)\sT”

“\w.\sA”

“\w-\d)\st”

“\d-(2))\sT”

There's a list A of hundreds of sentences and a list B of hundreds of terms. We are developing a function that extracts the term B when it is included in sentence A. However, as in the example below, there is an issue of incorrect extraction when there is another term between terms. Considering these issues, how can we proceed when extracting terms B from A sentences? (Note: The terms in glossary list B are in lower case only, but the sentences in sentence list A are both uppercase and lowercase.)

**"""**

**A = [**

**"Suddenly my earphones disappear",**

**"The sound echoed through the hollow cave",**

**...**

**"test"**

**]**

**B = [**

**"disappear",**

**"low",**

**...**

**"sample"**

**]**

**------------------------------------------------------------------------**

**Case1)**

**Example in case of incorrect extraction)**

**sample sentence : Suddenly my Earphones disappear**

**Extracted word  : ear**

**Example in case of correct extraction)**

**sample sentence : Suddenly my Earphones disappear**

**Extracted word  : disappear**

**------------------------------------------------------------------------**

**Case2)**

**Example in case of incorrect extraction)**

**sample sentence : The sound echoed through the hollow cave**

**Extracted word  : low**

**Example in case of correct extraction)**

**sample sentence : The sound echoed through the hollow cave**

**Extracted word  : hollow**

**------------------------------------------------------------------------**

**"""**

first = [

“Suddenly my earphones disappear”,

"The sound echoed through the hollow cave",

…

“test’

]

sec = [

“disappear”,

“low”,

…

“sample”

]

#def extracted\_term(first, sec);

if sec in first;

extracted\_term()