SEAS 6414

Spring 2024

Assignment 2

Dr. Adewale Akinfaderin

Last week I submitted the code followed by the execution. This was too much separation. For this week, I came up with a way to put them together so you can see the problem, the code, and then the execution. I put the problem description and the code into multiline Python strings. Then I printed the description, printed the code, and then executed the code using the exec() function.

Let me know if you want to see the source file. This is the executed file and has everything in it. But I am happy to share the source file. This assignment is in GitHub at https://github.com/OwlSaver/GWU.

Execution

```
################################################################################
# Problem 1
################################################################################

Problem:

Work with Python list comprehensions to perform the following tasks:
- Create a list of the first 20 positive integers.
- Utilize list comprehension to generate a new list containing the squares of all
  the numbers in the original list. Print the resulting list.
- Using the list created in the previous step, employ list comprehension once again
  to form a new list. This time, include only the squares of the odd numbers from
  the original list. Print this new list.
Note: The focus of this question is on using list comprehensions effectively.

Code:

import math
aList = range(20)
squareList = [x * x for x in aList]
print(squareList)
oddSquares = [x for x in squareList if math.sqrt(x) % 2 == 1]
print(oddSquares)

Execution:

[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289, 324, 361]
[1, 9, 25, 49, 81, 121, 169, 225, 289, 361]
```

```
################################################################################
# Problem 2
################################################################################

Problem:

Apply Python dictionary comprehensions for text analysis with the following steps:
- Given a sentence, split it into individual words. You may utilize the simple
  string split method available in Python.
- Use dictionary comprehension to construct a dictionary. The keys should be
  the words obtained from the sentence, and the values should be the lengths of
  these respective words.
- Output the resulting dictionary.
Note: Focus on demonstrating your ability to use dictionary comprehensions in
      Python for processing and analyzing text data.

Code:

import re
aSentence = "A sentence of words, for all to see."
wordList = re.findall(r"\w+|[^\w\s]", aSentence)
wordDict = {x: len(x) for x in wordList}
print(wordDict)

Execution:

{'A': 1, 'sentence': 8, 'of': 2, 'words': 5, ',': 1, 'for': 3, 'all': 3, 'to': 2, 'see': 3, '.': 1}

################################################################################
# Problem 3
################################################################################

Problem:

Develop a Python function to count occurrences of a substring within a given string:
- Take two strings as input: 'str1' (the main string) and 'str2' (the substring to
  search for).
- Write a function to calculate how many times 'str2' occurs within 'str1'.
- For example, if 'str1 = "coding is cool"' and 'str2 = "co"', the function should
  return 2 as the output.
Note: The focus is on string manipulation and search algorithms in Python. Consider
      edge cases, such as overlapping occurrences of the substring.

Code:

def countOLSubs(aString, aSubString):
    import re
    retval = len(re.findall(r'(?=' + aSubString + ')', aString))
    return retval
aTestString = "coding is cool"
sTestSubString = "co"
coss = countOLSubs(aTestString, sTestSubString)
print(f"There are {coss} occurences of '{sTestSubString}' in '{aTestString}'.")
aTestString = "This is a strange way to say how are we doing?"
sTestSubString = "is"
coss = countOLSubs(aTestString, sTestSubString)
print(f"There are {coss} occurences of '{sTestSubString}' in '{aTestString}'.")
aTestString = "aaaaaaaaaa"
sTestSubString = "aa"
coss = countOLSubs(aTestString, sTestSubString)
print(f"There are {coss} occurrences of '{sTestSubString}' in '{aTestString}'.")

Execution:

There are 2 c of 'co' in 'coding is cool'.
There are 2 occurrences of 'is' in 'This is a strange way to say how are we doing?'.
There are 9 occurrences of 'aa' in 'aaaaaaaaaa'.
```

```
################################################################################
# Problem 4
################################################################################

Problem:

Implement a Python function using list comprehension:
- Accept a string as input.
- Using list comprehension, create a list of strings where each string is formed by
  removing one character at a time from the original string. Each element in the
  list should represent the original string minus one of its characters.
- As an example, for the input string "Wale", the expected output is the list
  ["ale", "Wle", "W ae", "W al"].
Note: It is mandatory to use list comprehension for this task to demonstrate your
      proficiency in this Python feature. Ensure that all characters of the input string are
      considered, including repetitive ones.

Code:

def removeOneChar(aString):
    listOfStrings = [aString] * len(aString)
    listOfNewStrings = [x[:ind] + x[ind + 1:] for ind, x in enumerate(listOfStrings)]
    return listOfNewStrings
aTestString = "Wale"
aTestList  = removeOneChar(aTestString)
print(f"The string '{aTestString}' generated the list '{aTestList}'.")
aTestString = "look"
aTestList  = removeOneChar(aTestString)
print(f"The string '{aTestString}' generated the list '{aTestList}'.")

Execution:

The string 'Wale' generated the list '['ale', 'Wle', 'Wae', 'Wal']'.
The string 'look' generated the list '['ook', 'lok', 'lok', 'loo']'.

################################################################################
# Problem 5
################################################################################

Problem:

Work with dictionaries in Python to compute average scores:
Given the dictionary D containing students' names as keys and lists of their scores
as values, create a new dictionary where each key is a student's name and the
corresponding value is their average score.

The dictionary D is defined as follows:
D = {"Jake" : [99, 87, 91, 77], "Charlie" : [100, 100, 99], "Ellen" : [95, 70, 85, 100, 100]}

Task:
- Iterate through the dictionary D and calculate the average score for each student.
- Store the results in a new dictionary where the keys are the names of the
  students and the values are their respective average scores.

Note: This problem aims to enhance your skills in handling dictionaries, iterating
      over them, and performing calculations on their values in Python.

Code:

D = {"Jake" : [99, 87, 91, 77], "Charlie" : [100, 100, 99], "Ellen" : [95, 70, 85, 100, 100]}
C = {key : sum(value) / len(value) for key, value in D.items()}
print(C)

Execution:

{'Jake': 88.5, 'Charlie': 99.66666666666667, 'Ellen': 90.0}
```

```
################################################################################################
# Problem 6
################################################################################################

Problem:

Develop a Python program to filter and store specific words from a sentence. Follow
these steps:
- Split the given sentence into individual words.
- Use the strip method to remove punctuation from each word.
- Utilize the append method to add words to a new list, but only include those
  that are longer than 5 letters.

Given Sentence: "Tom enjoyed reading books on philosophy. He often pondered
the deeper meanings of life, especially during quiet nights."

Requirements:
- Create a list of all words in the sentence that exceed 5 letters in length.
- Ensure punctuation is excluded when determining the length of each word.

This exercise aims to enhance your skills in string manipulation, list handling, and
conditional logic in Python.

Code:

sentence = "Tom enjoyed reading books on philosophy. He often pondered the deeper meanings of life, especially
during quiet nights."
words1 = sentence.split()
outWords = []
for word in words1:
    cleanWord = word.strip(".,")
    if len(cleanWord) > 5:
        outWords.append(cleanWord)
print(outWords)

Execution:

['enjoyed', 'reading', 'philosophy', 'pondered', 'deeper', 'meanings', 'especially', 'during', 'nights']
```

```
################################################################################################
# Problem 7
################################################################################################

Problem:

Write a Python program to count words starting with a vowel in a sentence. The
program should be adaptable to any given sentence. Employ the split, replace,
and lower methods for this task.
Sample Sentence: "During her summer vacation, Alice explored various historical
sites and enjoyed local cuisines."
Task Objectives:
- Process the sentence to identify and count words beginning with a vowel (a, e,
  i, o, u).
- Ensure the program can handle different sentences and is not case-sensitive.
- Use string methods split to separate words, replace to remove punctuation,
  and lower to standardize all characters to lowercase.
Note: This exercise focuses on enhancing your string processing skills and understanding of basic Python string
methods.

Code:

def countVowels(aSentence):
    retval = 0
    wordList = aSentence.split()
    for word in wordList:
        if word.replace(",","").replace(".","").replace("?","").lower()[0] in "aeiou":
            retval += 1
    return retval
S = "During her summer vacation, Alice explored various historical sites and enjoyed local cuisines."
print(f"The sentence '{S}' has {countVowels(S)} words that begin with a vowel.")
S = "What time is it in Japan when it is 8:00 AM in Albania?"
print(f"The sentence '{S}' has {countVowels(S)} words that begin with a vowel.")
S = "Eager elephants eagerly enjoy eating enormous, exquisite, exotic apples every evening."
print(f"The sentence '{S}' has {countVowels(S)} words that begin with a vowel.")

Execution:

The sentence 'During her summer vacation, Alice explored various historical sites and enjoyed local cuisines.'
has 4 words that begin with a vowel.
The sentence 'What time is it in Japan when it is 8:00 AM in Albania?' has 8 words that begin with a vowel.
The sentence 'Eager elephants eagerly enjoy eating enormous, exquisite, exotic apples every evening.' has 11
words that begin with a vowel.

################################################################################################
# Problem 8
################################################################################################

Problem:
Use list comprehension to generate an n x n identity matrix.
Task:
- Write a one-line Python code using list comprehension to create an nxn identity
  matrix.
- The identity matrix should be represented as a list of lists, where each inner list
  corresponds to a row in the matrix.
- The diagonal elements of the matrix should be 1, and all other elements should be 0.

Code:

print([[1 if i == j else 0 for i in range(8)] for j in range(8)])

Execution:

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

```
################################################################################################
# Problem 9
################################################################################################

Problem:

Implement functions to generate the Look-and-Say sequence, which follows a pattern
of reading off the numbers of digits in groups of the same digit:

1, 11, 21, 1211, 111221, 312211, 13112221, . . .

Functions to Implement:
- Parse String(term): Accepts a term from the sequence as a string and returns
  a list of lists. Each sub-list contains two elements: the digit and the count of
  its consecutive appearances. For example, 'Parse String('1211')' should return
  [[1, 1], [1, 2], [2, 1]], representing "one 1, one 2, and two 1s".
- Next Term(term): Accepts a term of the sequence as a string and returns the
  next term. This function should utilize 'Parse String' internally. For example,
  'Next Term('1211')' should output '111221'.
- Read It(init, n): Given an initial term 'init', this function should print the
  next n - 1 terms of the sequence.

Note: These functions will help in understanding the concept of sequence generation,
      string parsing, and nested data structures in Python

Code:

def parseString(aTerm):
    rcnt = 0
    retval = []
    for i in range(len(aTerm)):
        rcnt = rcnt + 1
        if i == len(aTerm) - 1:
            retval.append([rcnt, aTerm[i]])
        else:
            if aTerm[i] == aTerm[i+1]:
                pass
            else:
                retval.append([rcnt, aTerm[i]])
                rcnt = 0
    return retval

def nextTerm(aTerm):
    retval = ''.join(str(x) for x in [a for b in parseString(aTerm) for a in b])
    return retval

def readIt(init, n):
    newTerm = init
    for i in range(n - 1):
        newTerm = nextTerm(newTerm)
        print(newTerm)

readIt("1",7)
print(" ")
readIt("843111332",7)
print(" ")

Execution:

Starting with 1 generating the next 6 terms of the Look-and-Say sequence:
11
21
1211
111221
312211
13112221

Starting with 843111332 generating the next 6 terms of the Look-and-Say sequence:
181413312312
1118111411231112131112
3118311421121331121113311 2
1321181321141221121123211231232112
1113122118111312211411222112211213122112113111213122112
3113112221183113112221142132212221121113112221121113311211131122211 2
```