CUNY SPS - MSDS CHALLENGE EXAM - 2020

Keshaw K Sahay
github: https://github.com/keshaws/CUNY_MSDS
rpubs: http://rpubs.com/keshaws/MSDS_Challenge_R

Programming Questions

Question 1 Correct all of the syntax errors in the function below to so that we determine whether or not the input word is an anagram. An anagram is a word that has the same value, even when it is reversed. Examples of anagrams are:

- aba
- civic
- dad

```
def is_anagram(word: str) -> bool
    """
    Returns whether or not a word is an anagram.

    Args:
        word: the word to test

    Returns:
        whether or not the word is an anagram
    """
    revered_wrd = word[0:1]
    return revered_word is word
```

Solution:

```
import  string

def is_anagram(word):
    # converting input word ito lower case
    word = word.lower()
    print(word)
    # reversing the word
    reversed_word = word[::-1]
    # initializing anagram boolean variable as False
    word_is_anagram = False
    # comparing input word with reversed word
    if word == reversed_word:
        word_is_anagram = True

    return word_is_anagram

# Driver Code
if __name__ == "__main__":
    print(is_anagram('Civic'))
```

Question 2

Complete the function below to return the temperature feel for a given list of temperatures. The function takes in a list of temperatures for the month as integer values, and should return the following value:

- If the average temperature is between 60 and 75 degrees, it should return "perfect!".
- If the average temperature is below 60 degrees, it should return "colder"
- If the average temperature is greater than 75, it should return "hotter"

Testcases:

Temperatures	Feel
[70, 65, 70, 68]	perfect!
[87, 90, 99, 88, 92, 94]	hotter
[30, 29, 31]	colder

Solution

```
def temp_cal(temp_list):
    # Checking input list length
    list_length = len(temp_list)
    # print(sum(temp_list))
    # Calculating average temp of list
    avg_temp = sum(temp_list)/list_length
    # print(avg_temp)
    # initializing temperature feel variable
    temp_feel = ''
    # operations for temperature feel
    if (avg_temp>=60 and avg_temp<=75):
        temp_feel = 'perfect!'
    elif avg_temp < 60:
        temp_feel = 'colder'
    elif avg_temp >75:
        temp_feel = 'hotter'
    return temp_feel

# Driver Code
if __name__ == "__main__":
    print(temp_cal([87,90,88,92,94]))
```

Question 3

Given a string of characters, return a dictionary of each character's frequency in the string.

Testcases:

String	Frequency
"aa"	{'a': 2}
"abba"	{'a': 2, 'b': 2}
"xyz	{'x': 1, 'y': 1, 'z': 1}

Solution

```
def char_frequency(str1):
    # initializing a dict
    dict = {}
    # operation for finding characters and counting the frequency
    for n in str1:
        keys = dict.keys()
        if n in keys:
            dict[n] += 1
        else:
            dict[n] = 1

    print(type(dict))
    return dict

# Driver Code
if __name__ == "__main__":
    print(char_frequency('abba'))
```

Question 4

Find if an array has a balance point. A balance point is defined as a point in the array, where the sum of all the values on the left are equal to the sum of all the values on the right. A few examples are:

- [1, 0, 1] is a **balanced** as the sum of numbers in yellow are equal to the sum of numbers in cyan.
- [1, 6, 5, 1, 2, 3, 1] is also a **balanced** array.
- [1, 1] is an **unbalanced** array.
- [1, 2, 6, 9] is an **unbalanced** array.
- [5, 2, 7, 0, 9] is an **unbalanced** array.

Testcases:

Numbers	Balanced
[1, 0, 1]	True
[1, 6, 5, 1, 2, 3, 1]	True
[1, 1]	False
[]	False
[1]	False

```
def arr_balance_point(arr):
    # Forming prefix sum array from 0
    if len(arr)>2:
        # Finding the length of array
        n = len(arr)
        prefixSum = [0] * n
        prefixSum[0] = arr[0]
        for i in range(1, n):
            prefixSum[i] = prefixSum[i - 1] + arr[i]

            # Forming suffix sum array from n-1
        suffixSum = [0] * n
        suffixSum[n - 1] = arr[n - 1]
        for i in range(n - 2, -1, -1):
            suffixSum[i] = suffixSum[i + 1] + arr[i]

        # Find the point where prefix and suffix sums are same.
        for i in range(1, n - 1, 1):
            if prefixSum[i] == suffixSum[i]:
                return True

        return False
    else:
        return False

# Driver Code
if __name__ == "__main__":
    # arr = [1, 4, 2, 5]
    arr =[1,6,5,1,2,3,1]
    # arr = [1,5,1]
    # arr = []
    print(arr_balance_point(arr))
```

SQL Questions

Given the table below, answer the following questions.

NAME	SALARY	DEPARTMENT	POSITION
JOHN	67000	HR	Manager
MATTHEW	87000	IT	Analyst
JANE	95000	IT	Manger
SARA	131000	IT	Architect

Question 1

Write a query to create the table above. The table should have the four columns, and one more **ID** column, which should be an integer that auto increments, and is the primary key.

Note: SQL Server 2018 is used

```
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[EMPLOYEE_TABLE](
		[ID] [int] IDENTITY(1,1) NOT NULL,
		[NAME] [varchar](50) NULL,
		[SALARY] [bigint] NULL,
		[DEPARTMENT] [varchar](50) NULL,
		[POSITION] [varchar](50) NULL,
 CONSTRAINT [PK_EMPLOYEE_TABLE] PRIMARY KEY CLUSTERED
(
		[ID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON
[PRIMARY]
) ON [PRIMARY]
GO
```

Question 2 Write a query to insert John's data into the table created in Question 1.

```
INSERT INTO [dbo].[EMPLOYEE_TABLE]
	([NAME]
	,[SALARY]
	,[DEPARTMENT]
	,[POSITION])
    VALUES
	('JOHN', 67000, 'HR','Manager')

GO
```

other records insert as below:

```
INSERT INTO [dbo].[EMPLOYEE_TABLE]
	([NAME]
	,[SALARY]
	,[DEPARTMENT]
	,[POSITION])
    VALUES
	('MATTHEW', 87000, 'IT','Analyst'),
		('JANE',95000, 'IT','Manager'),
		('SARA', 131000, 'IT','Architect')

GO
```

Question 3 Write a query to get the average salary of the IT department.

```
select AVG(Salary) from [dbo].[EMPLOYEE_TABLE] where department='IT'
```

Question 4 Write a query to update John's salary to 72,000

```
update [dbo].[EMPLOYEE_TABLE]
SET SALARY = 72000
Where Name='JOHN'
```

R Questions

Question 1

Complete the function below to return TRUE if a number is a prime, otherwise return FALSE:

```
is_prime <- function(p) {

      // Complete the function

}
```

http://rpubs.com/keshaws/MSDS_Challenge_R

```
```{r}
#creating R function to accept a number and return TRUE if a number is a prime, otherwise return FALSE:
is_prime <- function(num){

  if (num == 2) {
     TRUE
   } else if (any(num %% 2:(num-1) == 0)) {
     FALSE
   } else {
     TRUE
   }
}
```
```{r}
#Testing is prime function
test_prime_num <- is_prime(6)
print(test_prime_num)
print(is_prime(5))
```
```

Question 2

Given a list of the following words, write code in R to filter the list to fruits that start with a vowel:

Given	Expected
`list("apple", "pear", "orange", "banana", "elderberry", "strawberry")`	`list("apple", "orange", "elderberry")`

http://rpubs.com/keshaws/MSDS_Challenge_R

```
```{r message=FALSE, warning=FALSE}
library(tidyverse)
library(stringr)
```

```{r}
#Create a list of fruits as given in the question
fruit_list <- list('apple','pear','orange','banana','elderberry','strawberry')
paste0(fruit_list)
typeof(fruit_list)
```

```{r}
#Using grep function to find the index of fruit in the list starting with vowels
fruits_with_vowel_index <- c(grep("^[aeiouy]", tolower(fruit_list)))
paste0(fruits_with_vowel_index)
typeof(fruits_with_vowel_index)
```

```{r}
#filter the list of fruits using index obtained from above operation
fruits_start_with_vowles <- fruit_list[fruits_with_vowel_index]
paste0(fruits_start_with_vowles)
typeof(fruits_start_with_vowles)

```
```