[SOLUTION] CS-465-AI-Lab-Midterm-CS A-Version[A]

March 9, 2023

Question 1: Write a function that takes a string and a substring as input and returns the number of occurrences of the substring in the string.

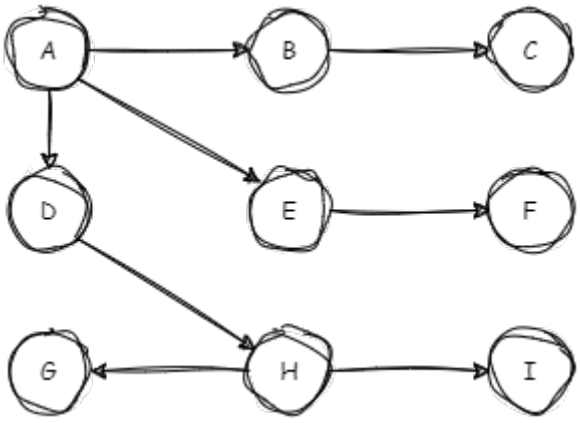
Input: “hello world”, “o” Output: 2

[1]:
```
def count_substring(string, sub):
    return string.count(sub)

print(count_substring("hello world", "o"))
```

```
2
```

Question 2: Write a code of Depth First Search (DFS) Algorithm. Test your code by traversing the following graph.



[2]:
```
graph = {
    "A" : ["B", "D", "E"],
    "B" : ["C"],
    "C" : [],
    "D" : ["H"],
    "E" : ["F"],
    "F" : [],
    "G" : [],
    "H" : ["G", "I"],
    "I" : []
```

```
}
```

[3]:
```
visited = set() # Set to keep track of visited nodes.

def dfs(visited, graph, node):
    if node not in visited:
        print(node, end=' ')
        visited.add(node)
        for neighbour in graph[node]:
            dfs(visited, graph, neighbour)

# Driver Code
dfs(visited, graph, 'A')
```

```
A B C D H G I E F
```

Question 3: You have data of car's sales in excel file as **"car_sales.csv"**. You are requested to show the data of only people above 40 years old and they have sold more than 4 cars.

[4]:
```
# importing the csv module
import csv

filename = "car_sales.csv"

# initialize Columns
fields = []

# initialize Rows
rows = []

# reading csv file
with open(filename, "r") as csv_file:

#   creating a csv reader object
    csv_reader = csv.reader(csv_file)

#   extracting field names through first row
    fields = next(csv_reader)

#   extracting each data row one by one
    for row in csv_reader:
        if int(row[2]) > 40 and int(row[6]) > 4:
            rows.append(row)

# printing the field names
print("\t".join(field for field in fields))
```

```
for row in rows:
#   parsing each column of a row
    for col in row:
        print(col + "\t", end = '')
    print()
```

```
ï»¿Sales ID     Name    Age     Gender  Years of sales experience       Number
of hours worked per day  Number of cars sold     Total value of cars sold
Total number of extras sold (insurance, parts, options, etc.)   Total value of
extras sold
9       Julia   62      F       40      5       5       102300  2       320
23      Gabriel 64      M       40      8       8       115750  1       245
27      John    45      M       19      5       5       34200   2       39
```

Question 4: Implement a class in Python to represent a bank account. Write initializer with default value(s). The class should have methods to deposit and withdraw money, and to check the balance.

•Run all methods to check your code.
•Create three instances of above class.
•Add above instances to new list.
•Iterate the above list.

[5]:
```
class BankAccount:
    def __init__(self, balance = 0):
        self.__balance = balance

    def get_balance(self):
        return self.__balance

    def deposit(self, amount):
        self.__balance += amount
        print(f"Deposited {amount} dollars. New balance is {self.get_balance()}␣
 ↪dollars.")

    def withdraw(self, amount):
        if self.get_balance() >= amount:
            self.__balance -= amount
            print(f"Withdrew {amount} dollars. New balance is {self.
 ↪get_balance()} dollars.")
        else:
            print("Sorry, insufficient funds.")

    def check_balance(self):
        print(f"Your balance is {self.get_balance()} dollars.")

    def __str__(self):
        return f"Bank Account [balance: {self.get_balance()}]"
```

[6]:
```
account = BankAccount()      # create a new account with balance zero
account.deposit(1000)        # deposit 1000 dollars
account.check_balance()      # check the balance (should be 1000 dollars)
account.withdraw(500)        # withdraw 500 dollars
account.check_balance()      # check the balance (should be 500 dollars)
account.withdraw(1000)       # try to withdraw more than the balance (should␣
 ↪print "Sorry, insufficient funds.")
account.check_balance()      # check the balance (should be 500 dollars)


obj1 = BankAccount(100)
obj2 = BankAccount(153.49)
obj3 = BankAccount(53)
```

```
Deposited 1000 dollars. New balance is 1000 dollars.
Your balance is 1000 dollars.
Withdrew 500 dollars. New balance is 500 dollars.
Your balance is 500 dollars.
Sorry, insufficient funds.
Your balance is 500 dollars.
```

[7]:
```
my_list = [obj1, obj2, obj3]

for i in my_list:
    print(i)
```

```
Bank Account [balance: 100]
Bank Account [balance: 153.49]
Bank Account [balance: 53]
```