AI Slip Solution

Write a python program to print multiplication table for given number.

```
# Multiplication table (from 1 to 10) in Python

num = 12

# To take input from the user
# num = int(input("Display multiplication table of? "))

# Iterate 10 times from i = 1 to 10
for i in range(1, 11):
   print(num, 'x', i, '=', num*i)
```

Write a program to implement a water jug problem.

```
print("Water Jug Problem")
x=int(input("Enter X:"))
y=int(input("Enter Y:"))
while True:

    rno=int(input("Enter the Rule No"))

    if rno==1:
        if x<4:
            x=4

    if rno==2:
        if y<3:
            y=3

    if rno==5:
        if x>0:
            x=0

    if rno==6:
        if y>0:
```

```
        y=0

if rno==7:
    if x+y>= 4 and y>0:
        x,y=4,y-(4-x)

if rno==8:
    if x+y>=3 and x>0:
        x,y=x-(3-y),3

if rno==9:
    if x+y<=4 and y>0:
        x,y=x+y,0

if rno==10:
    if x+y<=3 and x>0:
        x,y=0,x+y

print("X =" ,x)
print("Y =" ,y)
if (x==2):
    print(" The result is a Goal state")
break
```

Write a python program that accepts a sentence and calculate the number of uppercase letters and lowercase letters.

```
def string_test(s):
    d={"UPPER_CASE":0, "LOWER_CASE":0}
    for c in s:
        if c.isupper():
            d["UPPER_CASE"]+=1
        elif c.islower():
            d["LOWER_CASE"]+=1
        else:
            pass
    print ("Original String : ", s)
    print ("No. of Upper case characters : ", d["UPPER_CASE"])
    print ("No. of Lower case Characters : ", d["LOWER_CASE"])

string_test('The quick Brown Fox')
```

Write a program to implement Depth First Search algorithm.

```
graph1 = {
   'A' : ['B','S'],
   'B' : ['A'],
   'C' : ['D','E','F','S'],
   'D' : ['C'],
   'E' : ['C','H'],
   'F' : ['C','G'],
   'G' : ['F','S'],
   'H' : ['E','G'],
   'S' : ['A','C','G']
}

def dfs(graph, node, visited):
   if node not in visited:
      visited.append(node)
      for k in graph[node]:
         dfs(graph,k, visited)
   return visited

visited = dfs(graph1,'A', [])
print(visited)
```

Write a program to find Factorial of the given number.

```
# change the value for a different result
num = 7

# To take input from the user
#num = int(input("Enter a number: "))

factorial = 1

# check if the number is negative, positive or zero
if num < 0:
   print("Sorry, factorial does not exist for negative numbers")
elif num == 0:
```

```
  print("The factorial of 0 is 1")
else:
  for i in range(1,num + 1):
     factorial = factorial*i
  print("The factorial of",num,"is",factorial)
```

Write a program to implement Breadth first Search.

```
graph = {
  'A' : ['B','C'],
  'B' : ['D', 'E'],
  'C' : ['F'],
  'D' : [],
  'E' : ['F'],
  'F' : []
}

visited = [] # List to keep track of visited nodes.
queue = []     #Initialise a queue

def bfs(visited, graph, node):
  visited.append(node)
  queue.append(node)

  while queue:
    s = queue.pop(0)
    print (s, end = " ")

    for neighbour in graph[s]:
      if neighbour not in visited:
        visited.append(neighbour)
        queue.append(neighbour)

# Driver Code
bfs(visited, graph, 'A')         #execution starts here bfs(visited[],Graph,'A') PC
jump to line 12
```

Explanation

- **Lines 3-10**: The illustrated graph is represented using an *adjacency list*. An easy way to do this in Python is to use a *dictionary data structure*, where each vertex has a stored list of its adjacent nodes.

- **Line 12**: visited is a list that is used to keep track of visited nodes.
- **Line 13**: queue is a list that is used to keep track of nodes currently in the queue.
- **Line 29**: The arguments of the bfs function are the visited list, the graph in the form of a dictionary, and the starting node A.
- **Lines 15-26**: bfs follows the algorithm described above:
    1. It checks and appends the starting node to the visited list and the queue.
    2. Then, while the queue contains elements, it keeps taking out nodes from the queue, appends the neighbors of that node to the queue if they are unvisited, and marks them as visited.
    3. This continues until the queue is empty.

Write a program that accepts a sentence and calculate the number of letters and digits.

```
str = input("Input a string: ")
d=l=0
for c in str:
    if c.isdigit():
        d=d+1
    elif c.isalpha():
        l=l+1
    else:
        pass
print("Letters", l)
print("Digits", d)
```

Write a program to implement Simple Chatbot.

```
print("Simple Question and Answering Program")
print("========================================")
print(" You may ask any one of these questions")
print("Hi")
print("How are you?")
print("Are you working?")
print("What is your name?")
print("what did you do yesterday?")
print("Quit")
while True:

    question = input("Enter one question from above list:")
    question = question.lower()
    if question in ['hi']:
        print("Hello")
    elif question in ['how are you?','how do you do?']:
        print("I am fine")
    elif question in ['are you working?','are you doing any job?']:
        print("yes. I'am working in KLU")
    elif question in ['what is your name?']:
        print("My name is Emilia")
        name=input("Enter your name?")
        print("Nice name and Nice meeting you",name)
```

```
    elif question in ['what did you do yesterday?']:
        print("I saw Bahubali 5 times")
    elif question in ['quit']:
            break
    else:
                print("I don't understand what you said")
```

Write a python program to count the number of characters in a string without using any
built-in function.

```
string=raw_input("Enter string:")
count=0
for i in string:
     count=count+1
print("Length of the string is:")
print(count)
```

Write a program to illustrate different Set Operations.

```
# define three sets
E = {0, 2, 4, 6, 8};
N = {1, 2, 3, 4, 5};
# set union
print("Union of E and N is",E | N)
# set intersection
print("Intersection of E and N is",E & N)
# set difference
print("Difference of E and N is",E - N)
# set symmetric difference
print("Symmetric difference of E and N is",E ^ N)
```

Write a python program to find the length of a set.

```
setObject = {'a', 'b', 'c'}
length = len(setObject)
print(f,Length of this set is {length}.')
```

Write a program to implement List Operations.

```
myList = ["Earth", "revolves", "around", "Sun"]
print(myList)
print(len(myList))
print(myList[0])
print(myList[3])
#Slice elements from a list
print(myList[1:3])
#Use negative index
print(myList[-1])
#print(myList[4])
#add element to a list
myList.insert(0,"The")
print(myList)
print(len(myList))
myList.insert(4,"the")
print(myList)
#append an element to a list
myList.append("continuously")
print(myList)
print(len(myList))
#When use extend the argument should be another list
#the elements of that list will be added
#to the current list as individual elements
myList.extend(["for", "sure"])
print(myList)
print(len(myList))
#you can append a sublist to the current list using append
myList.append(["The","earth","rotates"])
print(myList)
print(len(myList))
#delete a element in the list using element
myList.remove("The")
```

```
#delete a element in the list using index
myList.remove(myList[3])
print(myList)
```

Write a program to check if a given no is odd or even.

```
num = int (input ("Enter any number to test whether it is odd or even: ")

if (num % 2) == 0:

        print ("The number is even")

else:

        print ("The provided number is odd")
```

Write a program to solve water jug problem : 3 jugs with capacity 12 gallon , 8 gallon and 5 gallon are given with unlimited water supply and target to achieve is 6 gallon
of water in the second jug .

```
from collections import deque
def Solution(a, b, target):
        m = {}
        isSolvable = False
        path = []

        q = deque()

        #Initializing with jugs being empty
        q.append((0, 0))
```

```
while (len(q) > 0):

        # Current state
        u = q.popleft()
        if ((u[0], u[1]) in m):
                continue
        if ((u[0] > a or u[1] > b or
                u[0] < 0 or u[1] < 0)):
                continue
        path.append([u[0], u[1]])

        m[(u[0], u[1])] = 1

        if (u[0] == target or u[1] == target):
                isSolvable = True

                if (u[0] == target):
                        if (u[1] != 0):
                                path.append([u[0], 0])
                else:
                        if (u[0] != 0):

                                path.append([0, u[1]])

                sz = len(path)
                for i in range(sz):
                        print("(", path[i][0], ",",
                                path[i][1], ")")
                break

        q.append([u[0], b]) # Fill Jug2
        q.append([a, u[1]]) # Fill Jug1

        for ap in range(max(a, b) + 1):
                c = u[0] + ap
                d = u[1] - ap
```

```
                    if (c == a or (d == 0 and d >= 0)):
                        q.append([c, d])

                    c = u[0] - ap
                    d = u[1] + ap

                    if ((c == 0 and c >= 0) or d == b):
                        q.append([c, d])

                q.append([a, 0])

                q.append([0, b])

        if (not isSolvable):
            print("Solution not possible")

if __name__ == '__main__':

        Jug1, Jug2, target = 4, 3, 2
        print("Path from initial state "
            "to solution state ::")

        Solution(Jug1, Jug2, target)
```

Write a program to check if a given number is Armstrong or not.

```
num = int(input("Enter a number: "))
sum = 0
temp = num

while temp > 0:
   digit = temp % 10
   sum += digit ** 3
   temp //= 10

if num == sum:
   print(num,"is an Armstrong number")
else:
   print(num,"is not an Armstrong number")
```

Write a program to take input from user as, 3 subject marks and find out average of marks
and percentage of marks.

```
a = int(input("Enter the marks of first subject: "))
b = int(input("Enter the marks of second subject: "))
c = int(input("Enter the marks of third subject: "))
total = a+b+c
avg = total/3
print("Total marks: ",total)
print("Average marks: ",avg)
```

Write a program to implement KNN algorithm.

```
import numpy as np

import pandas as pd

from matplotlib import pyplot as plt

from sklearn.datasets import load_breast_cancer

from sklearn.metrics import confusion_matrix

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model_selection import train_test_split

import seaborn as sns

sns.set()

breast_cancer = load_breast_cancer()

X = pd.DataFrame(breast_cancer.data, columns=breast_cancer.feature_names)

X = X[['mean area', 'mean compactness']]

y = pd.Categorical.from_codes(breast_cancer.target,
breast_cancer.target_names)

y = pd.get_dummies(y, drop_first=True)

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)

knn = KNeighborsClassifier(n_neighbors=5, metric='euclidean')

knn.fit(X_train, y_train)

y_pred = knn.predict(X_test)

sns.scatterplot( x='mean area',  y='mean compactness', hue='benign',
data=X_test.join(y_test, how='outer'))

plt.scatter(  X_test['mean area'],  X_test['mean compactness'],
c=y_pred,cmap='coolwarm',alpha=0.7)

confusion_matrix(y_test, y_pred)
```

Write a Program to check whether the given no is prime or not

```
number = int(input("Enter any number: "))

if number > 1:

   for i in range(2, number):

      if (number % i) == 0:

         print(number, "is not a prime number")

         break

   else:

      print(number, "is a prime number")

else:

   print(number, "is not a prime number")
```

Write a program to implement Regression algorithm.
(Download salary_data.csv dataset).

#Importing libraries

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

 # Importing the dataset

```
dataset = pd.read_csv('salary_data.csv')
X = dataset.iloc[:, :-1].values #get a copy of dataset exclude last column
y = dataset.iloc[:, 1].values #get array of dataset in column 1st
```

 # Splitting the dataset into the Training set and Test set

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1/3,
random_state=0)
```

```
# Fitting Simple Linear Regression to the Training set
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

# Visualizing the Training set results
viz_train = plt
viz_train.scatter(X_train, y_train, color='red')
viz_train.plot(X_train, regressor.predict(X_train), color='blue')
viz_train.title('Salary VS Experience (Training set)')
viz_train.xlabel('Year of Experience')
viz_train.ylabel('Salary')
viz_train.show()

# Visualizing the Test set results
viz_test = plt
viz_test.scatter(X_test, y_test, color='red')
viz_test.plot(X_train, regressor.predict(X_train), color='blue')
viz_test.title('Salary VS Experience (Test set)')
viz_test.xlabel('Year of Experience')
viz_test.ylabel('Salary')
viz_test.show()

# Predicting the result of 5 Years Experience
y_pred  =regressor.predict(np.array([5]).reshape(1, 1))
```

Write a program which accepts the user's first and last name and print them in reverse order
with a space between them.

```
def user_name():
first_name = input('Please enter first name:')
last_name = input('Please enter last name:')
print(last_name,first_name)
```

Write a python program to display the first and last colours from the following list.
color_list = ["Red","Green","White" ,"Black"]

```
color_list = ["Red","Green","White" ,"Black"]
print( "%s %s"%(color_list[0],color_list[-1]))
```

Write a program that prints out all the elements of the list that are less than 20.
a = [5, 10, 15, 7, 25]

```
        For i in a:
        If i<20:
        print(i);
```

Write a Python program which accepts a sequence of comma-separated numbers from user
and generate a list and a tuple with those numbers.

```
values = input("Input some comma seprated numbers : ")
list = values.split(",")
tuple = tuple(list)
print('List : ',list)
print('Tuple : ',tuple)
```

Write a program to check whether a number is palindrome or not.

```
n=int(input("Enter number:"))
temp=n
rev=0
while(n>0):
    dig=n%10
    rev=rev*10+dig
    n=n//10
if(temp==rev):
    print("The number is a palindrome!")
else:
    print("The number isn't a palindrome!")
```

Write a Python script to concatenate following dictionaries to create a new one.
Sample Dictionary:
dic1= {1:10, 2:20}
dic2= {3:30, 4:40}
dic3= {5:50, 6:60}

Sol:

```
dic1={1:10, 2:20}
dic2={3:30, 4:40}
dic3={5:50,6:60}
dic4 = {}
for d in (dic1, dic2, dic3): dic4.update(d)
print(dic4)
```

Write a python program to calculate LCM of given 2 numbers.(Accept input from User)

```
# defining a function to calculate LCM
def calculate_lcm(x, y):
   # selecting the greater number
   if x > y:
       greater = x
   else:
       greater = y
   while(True):
       if((greater % x == 0) and (greater % y == 0)):
           lcm = greater
           break
       greater += 1
   return lcm


# taking input from users
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))
# printing the result for the users
print("The L.C.M. of", num1,"and", num2,"is", calculate_lcm(num1, num2))
```

Write a python program to accept and convert string in uppercase or vice versa.

```
tx1 = input("Please Enter your Own Text : ")

tx2 = tx1.upper()

print("\nOriginal  =  ", tx1)
print("Result =  ", tx2)
```

Write a python program to find the repeated items of a tuple.

```
#create a tuple

tuplex = 2, 4, 5, 6, 2, 3, 4, 4, 7

print(tuplex)

#return the number of times it appears in the tuple.

count = tuplex.count(4)

print(count)
```

Write a python program to check if a given key already exists in a dictionary. If a key exists, replace it with another key/value pair.

```
d = {1: 10, 2: 20, 3: 30, 4: 40, 5: 50, 6: 60}

def is_key_present(x):

  if x in d:
```

```
    print('Key is present in the dictionary')
else:
    print('Key is not present in the dictionary')
is_key_present(5)
is_key_present(9)
```

Write a Python program to find the greatest common divisor (gcd) of two integers.

```
# Python code to demonstrate the working of gcd()
# importing "math" for mathematical operations
import math
print("The gcd of 60 and 48 is : ", end="")
print(math.gcd(60, 48))
```

Write a program to check whether the given no is perfect number or not

```
num=int(input("Enter the number: "))
sum_v=0
for i in range(1,num):
```

```
    if (num%i==0):
        sum_v=sum_v+i
if(sum_v==num):
    print("The entered number is a perfect number")
else:
    print("The entered number is not a perfect number")
```