Worksheet - 0

Basics

Exercise on Functions

Task - 1

```
def lengthConversion():
  """
    used to convert m to ft or vice versa
  """
  length = input("Enter m or ft.")
  if(length == "m"):
    meter = float(input("Enter length in meter"))
    print(f"{meter}m = {meter * 3.28}ft.")
  elif(length == "ft"):
    feet = float(input("Enter length in feet"))
    print(f"{feet}ft = {feet / 3.28}m")
  else:
    raise ValueError("Invalid Length")
def weightConversion():
  '''
    used to convert kg to lbs or vice versa
  '''
  weight = input("Enter kg or lbs: ")
  if(weight == "kg"):
    kg = float(input("Enter weight in kg: "))
    print(f"{kg}kg => {kg * 2.20}lbs")
  elif(weight == "lbs"):
    lbs = float(input("Enter weight in lbs: "))
    print(f"{lbs}lbs => {lbs / 2.20}kg")
  else:
    raise ValueError("Invalid Weight")
def volumeConversion():
  '''
    used to convert litre to gallon or vice versa
  '''
  volume = input("Enter l or gal: ")
  if(volume == "l"):
    litre = float(input("Enter Litres: "))
    print(f"{litre}l => {litre * 0.264}gallons")
  elif(volume == "gal"):
    gal = float(input("Enter Gallon: "))
    print(f"{gal}gallons => {gal / 0.264}litres")
  else:
    raise ValueError("Invalid Unit")



while True:
  typeOfConv = input("Choose type of Conversion (length or weight or volume): ")
  try:
    if(typeOfConv == "length"):
      lengthConversion()
    elif(typeOfConv == "weight"):
      weightConversion()
    elif(typeOfConv == "volume"):
      volumeConversion()
    if(typeOfConv in ["length", "weight", "volume"]):
      break;
    else:
      print("Invalid Input!!!(Enter: length or weight or volume)")
      continue;
  except ValueError as ve:
    print(f"Error: {ve}")
    continue;
```

```
Choose type of Conversion (length or weight or volume): length
Enter m or ft.m
Enter length in meterdf
Error: could not convert string to float: 'df'
Choose type of Conversion (length or weight or volume): length
Enter m or ft.m
Enter length in meter39
39.0m = 127.91999999999999ft.
```

Task - 2

```
def sumIt(nums):
  summ = 0
  for i in nums:
    summ += i
  return summ
def avgIt(nums):
  summ = sumIt(nums)
  return summ/len(nums)

def maxIt(nums):
  max = nums[0]
  for i in nums:
    if i > max:
      max = i
  return max

def minIt(nums):
  min = nums[0]
  for i in nums:
    if i < min:
      min = i
  return min


while True:
    task = input("Choose an task (sum, average, max, min): ").strip().lower()

    try:
        if task not in ["sum", "average", "max", "min"]:
            print("Invalid Input!!! (Enter: sum, average, max, min)")
            continue

        nums = input("Enter a list of numbers separated by spaces: ").strip().split()
        nums = [float(num) for num in nums]

        if len(nums) == 0:
            raise ValueError("The list cannot be empty.")

        if task == "sum":
            result = sumIt(nums)
        elif task == "average":
            result = avgIt(nums)
        elif task == "max":
            result = maxIt(nums)
        elif task == "min":
            result = minIt(nums)

        print(f"The result of {task} task is: {result}")
        break

    except ValueError as ve:
        print(f"Error: {ve} Numeric Values only")
        continue
```

```
Choose an task (sum, average, max, min): min
Enter a list of numbers separated by spaces: 2 3 4 4 5 0
The result of min task is: 0.0
```

Exercise on List Manipulation

```
def extract(nums):
    """1. Extracts every other value of list"""
    return nums[::2]

def slicee(nums, start, end):
    """2. Slice the list from start to end index"""
    return nums[start:end+1]

def reverse(nums):
    """3. reverses the list"""
    return nums[::-1]

def remove(nums):
    """4. Removes the first and last numbers of the list"""
    return nums[1:-1]

def getfirstn(nums, n):
    """5. Gets the first n number of items from the list"""
    return nums[:n]

def getlastn(nums, n):
    """6. Gets the last n number of items from the list """
    return nums[-n:]

def reverseskip(nums):
  """7. """
  return nums[-2::-2]


while True:
    task = input("Choose a task (extract, slice, reverse, remove, getfirstn, getlastn, reverseskip): ").strip().lower()

    try:
        if task not in ["extract", "slice", "reverse", "remove", "getfirstn", "getlastn", "reverseskip"]:
            print("Invalid Input!!! (Enter: extract, slice, reverse, remove, getfirstn, getlastn, reverseskip)")
            continue

        nums = input("Enter a list of numbers separated by spaces: ").strip().split()
        nums = [float(num) for num in nums]

        if len(nums) == 0:
            raise ValueError("The list cannot be empty.")

        if task == "extract":
            result = extract(nums)
        elif task == "slice":
            start = int(input("Enter start index: "))
            end = int(input("Enter end index: "))
            result = slice(nums, start, end)
        elif task == "reverse":
            result = reverse(nums)
        elif task == "remove":
            result = remove(nums)
        elif task == "getfirstn":
            n = int(input("Enter N: "))
            result = getfirstn(nums, n)
        elif task == "getlastn":
            n = int(input("Enter N: "))
            result = getlastn(nums, n)
        elif task == "reverseskip":
            result = reverseskip(nums)

        print(f"The result of {task} task is: {result}")
        break

    except ValueError as ve:
        print(f"Error: {ve}. Please enter valid numeric values.")
        continue
```

```
Choose a task (extract, slice, reverse, remove, getfirstn, getlastn, reverseskip): reverseskip
Enter a list of numbers separated by spaces: 1 2 3 4 5 6 7
The result of reverseskip task is: [6.0, 4.0, 2.0]
```

Exercise on Nested List

```
def flatten(lst):
    """It makes a nested list into a single list(makes it into a single dimension)"""
```

```
        flat_list = []
        for sublist in lst:
            if isinstance(sublist, list):
                flat_list.extend(flatten(sublist))
            else:
                flat_list.append(sublist)
        return flat_list

    def accessnestedelement(lst, indices):
        """Access the Nested List elements"""
        element = lst
        for index in indices:
            element = element[index]
        return element

    def sumnested(lst):
        """Sums all the elements of the nested List"""
        total = 0
        for item in lst:
            if isinstance(item, list):
                total += sumnested(item)
            else:
                total += item
        return total

    def removeelement(lst, elem):
        """Removes specific element of the list"""
        modified_list = []
        for item in lst:
            if isinstance(item, list):
                modified_list.append(removeelement(item, elem))
            elif item != elem:
                modified_list.append(item)
        return [sublist for sublist in modified_list if sublist]

    def findmax(lst):
        """Finds the maximum value element of the list"""
        max_value = float('-inf')
        for item in lst:
            if isinstance(item, list):
                max_value = max(max_value, findmax(item))
            else:
                max_value = max(max_value, item)
        return max_value

    def countoccurrences(lst, elem):
        """Count the number of time an element appears in the list"""
        count = 0
        for item in lst:
            if isinstance(item, list):
                count += countoccurrences(item, elem)
            elif item == elem:
                count += 1
        return count

    def flatten2(lst):
        """ """
        flat_list = []
        for item in lst:
            if isinstance(item, list):
                flat_list.extend(flatten2(item))
            else:
                flat_list.append(item)
        return flat_list

    def averagenested(lst):
        """ """
        flat_list = flatten(lst)
        return sum(flat_list) / len(flat_list) if flat_list else 0


    while True:
        task = input("Choose a task (flatten, accessnestedelement, sumnested, removeelement, findmax, countoccurrences, flatten2, averagenested)

        try:
            if task not in ["flatten", "accessnestedelement", "sumnested", "removeelement", "findmax", "countoccurrences", "flatten2", "averager
                print("Invalid Input!!! Please choose a valid task.")
                continue

            nested_list = eval(input("Enter a nested list (use square brackets []): "))  # Using eval to accept nested lists

            if not isinstance(nested_list, list):
                raise ValueError("Input must be a list")

            if task == "flatten":
                result = flatten(nested_list)
            elif task == "accessnestedelement":
                indices = list(map(int, input("Enter indices separated by spaces: ").strip().split()))
                result = accessnestedelement(nested_list, indices)
            elif task == "sumnested":
                result = sumnested(nested_list)
            elif task == "removeelement":
                elem = eval(input("Enter element to remove: "))  # Using eval to accept numbers and strings
                result = removeelement(nested_list, elem)
            elif task == "findmax":
                result = findmax(nested_list)
            elif task == "countoccurrences":
                elem = eval(input("Enter element to count: "))
                result = countoccurrences(nested_list, elem)
            elif task == "flatten2":
                result = flatten2(nested_list)
            elif task == "averagenested":
                result = averagenested(nested_list)

            print(f"The result of {task} task is: {result}")
            break

        except (ValueError, TypeError, IndexError) as e:
            print(f"Error: {e}. Please enter valid input.")
            continue
```

```
Choose a task (flatten, accessnestedelement, sumnested, removeelement, findmax, countoccurrences, flatten2, averagenested): flatten
Enter a nested list (use square brackets []): [[1, 3, 4 ,5, 6], [2]]
The result of flatten task is: [1, 3, 4, 5, 6, 2]
```

Numpy

Importing Necessary Libraries

```
import numpy as np
import time
```

Problem - 1: Array Creation

[ ] ↳ 15 cells hidden

Problem - 2: Array Manipulation: Numerical Ranges and Array Indexing

[ ] ↳ 18 cells hidden

Problem - 3: Array Operations

[ ] ↳ 15 cells hidden

Problem - 4: Matrix Operations

[ ] ↳ 9 cells hidden

Numpy Speed

[ ] ↳ 1 cell hidden