```
# 1
l1 = [1,3,5,7,9]
print(l1)
l1[(len(l1)-1)],l1[0] = l1[0],l1[(len(l1)-1)]
print(l1)


# 2
lst = [1,2,3,4,5,6,7,8,]
lst1=[]
lst2=[]
for i in range(0,len(lst)):
  if i%2 ==0:
    lst1.append(lst[i])
  else:
    lst2.append(lst[i])

print(lst)
print(lst1)
print(lst2)


# 3
l1 = [2,5,2,3,4,6,3,9,5,4]
l2 = [3,4]

result = False
for i in range(len(l1)):
    if l1[i:i+len(l2)] == l2:
        result = True
        break  # Exit loop once subsequence is found

print(result)



# 4
l1 = [1,3,5,7,9,11,13]
l2 = [2,4,6]
l3 =[]
for i in range(len(l1)):
  l3.append(l1[i])
  for j in range(i,len(l2)):
    l3.append(l2[j])
    break

print(l3)


# 5

# N/A


# 6
tup = ('x','y','z')
lst = list(tup)
index_pos = int(input("Index: "))
element = (input("Enter the element: " ))
lst.insert(index_pos,element)
new_tup = tuple(lst)
print(new_tup)
```

```
# 7
tup = (1,1,1,1,1,1,1,1)
nth_element = 3
lst = list(tup)
for i in range(nth_element - 1, len(lst), nth_element):
  lst[i] = lst[i]*3

print(lst)


# 8
tup = ('d','c','b','a')
tup1 =()
for i in range(len(tup)-1,-1,-1):
  tup1 += (tup[i],)
print(tup1)


# 9

# N/A


# 10
tups =[(1, 2, 3), (4, 5, 6), (7, 8, 9), (10, 11, 12)]
for i in tups:
  print((i[0],i[-1]),end =" ")

#x = tuple((tup[0],tup[-1]) for tup in tups )
#print(x)


# 11

dct = {'Iphone':'Puppy','IQOO': 'Venkat', 'OnePlus': 'VK'}
sub_val = ['Venkat','VK']
extracted_keys = []
for keys,values in dct.items():
  if values in sub_val:
    extracted_keys.append(keys)
print(extracted_keys)


# 12
dict1= {1:'a',2:'b',3:'c',4:'d'}
x = dict1.keys()
y = dict1.values()
vals = []
keys =[]
for i in x:
  vals.append(i)
for i in y:
  keys.append(i)

new_dict = dict(zip(keys,vals))
print(new_dict)


# 13

# N/A
```

```
# 14
def partition_values_by_modulus(dictionary, modulus):
    return {i: [value for value in dictionary.values() if value % modulus == i] for i in range(modulus)}

# Example usage:
my_dict = {'a': 10, 'b': 7, 'c': 15, 'd': 23, 'e': 8}
modulus = 3
partitioned_dict = partition_values_by_modulus(my_dict, modulus)
print("Partitioned dictionary:", partitioned_dict)


# 15

# N/A


# 16
set1 = {1,2,3,4,5}
set2 = {4,5,6,7,8}
x = (set1 | set2) - (set1 & set2)
print(x)


# 17
set1 = {1,2,3,4,5,6,7}
set2 = {2,6}
set3=set()
for i in set2:
  for j in set1:
    if i == j:
      set3.add(i)

if set2 == set3:
  print(True)
else:
  print(False)
#x = set2<=set1
#print(x)


# 18
set_a = {1,2,3}
set_b = {'x','y','z'}
pairs =[]
for i in set_a:
  for j in set_b:
    if i !=j:
      pairs.append((i,j))
for i in pairs:
  print(i)


# 19
set_a = {2,4,6,8}
set_b = {1,2,3,1}
x = [i for i in set_a if i not in set_b]
print(set(x))
```

```
# 20
def find_subsets(input_set, divisor):
    subsets = []
    for num in input_set:
        if num % divisor == 0:
            subsets.append({num})
    return subsets

# Example usage:
input_set = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
divisor = 2
result = find_subsets(input_set, divisor)
print("Minimum set of subsets where each subset is divisible by", divisor, ":")
for subset in result:
    print(subset)



# 21
from itertools import zip_longest

def interleave(*lists):
    interleaved = []
    for elements in zip_longest(*lists):
        interleaved.extend([elem for elem in elements if elem is not None])
    return interleaved

list1 = [1, 2, 3]
list2 = ['a', 'b', 'c', 'd']
list3 = [True, False]
result = interleave(list1, list2, list3)
print("Interleaved list:", result)


# 22

lst_strs = ['LV','DSAI','HIVE']
str_len =[]
for i in lst_strs:
  str_len.append(len(i))

dct = dict(zip(lst_strs,str_len))
print(dct)

''' --- using fuction ---
def str_dict(keys,values):
  str_len =[]
  for i in lst_strs:
    str_len.append(len(i))
    dct = dict(zip(lst_strs,str_len))
  print(dct)
lst_strs = ['LV','DSAI','HIVE']
str_dict(lst_strs,str_len)
'''
```

```
# 23
def fib(n):
  a=0
  b=1
  if n <= 0:
    return 0
  elif n == 1:
    return a
  else:
    print(a)
    print(b)
    for i in range(2,n):
      c = a+b
      a=b
      b=c
      print(c)

fib(4)


# 24

l1 = [1,2,3,4,5,6]
n = 9
l2=[]
for i in range(len(l1)):
  for j in range(i,len(l1)):
    if l1[i]+l1[j] == n:
      l2.append((l1[i],l1[j]))

print(l2)



# 25

l1 = ['a','b','c','d','e']
k = 11
k = k % len(l1)
print(l1[k:]+l1[:k])




# 26

tups =((1,2,3),(4,5),(6,(7,9)))
sum = 0
for item in tups:
  for element in item:
    try:
      for sub_element in element:
        sum += sub_element
    except:
      sum += element
print(sum)


# 27

lst = [1,2,3,4,5,6,7,8,9]
n =3
tup = []
for i in range(0,len(lst),n):
  tup.append(tuple(lst[i:i+n]))
print(tuple(tup))
```

```
# 28

tups = ((1,2,3),(4,5,66),(7,8,9),(0,7,3))
max_sum = 0
max_tup = None
for item in tups:
  current_sum = 0
  for element in item:
    current_sum += element
  if current_sum > max_sum:
    max_sum = current_sum
    max_tup = item
print(max_tup)


# 29
tup1 =(1,2,3,4)
tup2=(5,6)
for i in tup1:
  if i in tup2:
    print(True)
    break
  print(False)
  break

'''
def common_elements(tup1, tup2):
    for element in tup2:
        if element in tup1:
            return True
    return False
tup1 = (1, 2, 3, 4)
tup2 = ( 5, 6)

result = common_elements(tup1, tup2)
print(result)
'''


# 30

tup1 = (1,2,3,4,5)
sum = 0
new_tup = []
for i in tup1:

  sum += i
  new_tup.append(sum)
print(tuple(new_tup))

     (1, 3, 6, 10, 15)


# 31

# N/A
```

```
# 32

# 33

dict1 = {'a':10,'b':20,'c':40,'d':30}
max_val = max(dict1.values())
for key,value in dict1.items():
  if value == max_val:
    print(key)




# 34

# N/A


# 35

dict1 = {1:"DSAI", 2: "LV",3:"THE HIVE"}
sd = dict(sorted(dict1.items(),key = lambda item : len(item[1])))
print(sd)


# 36

set1 ={1,2,3}
set2 = {3,4,5}
set3 = {5,6,7}
union_set = set1 | set2 | set3
print(union_set)


# 37

set1 = {1,2,2,3}
set2 = {3,5,6}
print(set1.isdisjoint(set2))


# 38

lst_sets= [ {1,2,3},{4,5,6},{7,8,9} ]
result = set()
for i in lst_sets:
  result = result.symmetric_difference(i)
print(result)
```

Question 1:

A weather forecasting company wants to store daily temperature data for different cities. Develop a Python program to prompt users to input temperature data for multiple cities and store it in appropriate variables.

```
temp_data = {}
no_of_cities = int(input("No. of different cities: "))
for i in range(no_of_cities):
  city = input("Enter the city name: ")
  temperature = input("Temperature : ")
  temp_data[city]=temperature
print("\nTemp data for each city:")
for i,j in temp_data.items():
  print(i,": ",j, "degree celsius")
```

Question 2: A gaming company is developing a character creation system for their new role-playing game. Develop a Python program that allows users to input characteristics such as name, race, class, and abilities for their characters.

```
char_role = {}

char_role['name']= input("Enter Character Name: ")
char_role['race'] = input("Enter Character Race: ")
char_role['class'] = input("Enter Character Class: ")

abilities =[]
ability = input("Ability: ")
while ability:
  abilities.append(ability)
  ability = input("Ability: ")

char_role['abilities'] = abilities
print("\nCharacter Created:")
print("Name:", char_role["name"])
print("Race:", char_role["race"])
print("Class:", char_role["class"])
print("Abilities:", ", ".join(char_role["abilities"]))
```

Question 3: You are organizing a school event where students will be asked to sign up using a form.Write a Python program that prompts the user to enter their name, age, and grade, and then prints a welcome message including their name and an appropriate message based on their age and grade.

```
name = input("Enter name: ")
age = int(input("Age: "))
grade = int(input("Grade secured: "))
#print("Welcome", name, "! your age is ",age, "& your grade is", grade)

if age<=12:
  if grade >=7:
    print(f"Welcome, {name}! Have a great time in elementary school!")
  else:
    print(f"Welcome, {name}! Enjoy your middle school years!")
elif age<=18:
  if grade >= 7:
    print(f"Welcome, {name}! Have a great time in High School!")
  else:
    print(f"Welcome, {name}! Make use of school exp!")
```

Question 4: You are managing inventory for a small store. Write a Python program that simulates adding items to a shopping cart. Allow the user to input the item name, quantity, and price, and then calculate the total cost of the items in the cart.

Question 5: A music streaming service wants to create personalized playlists for users based on their favorite genres and artists. Develop a Python program that uses lists and dictionaries to store user preferences and recommend songs accordingly.

Question 6: A logistics company needs to track the inventory levels of different products in multiple warehouses. Develop a Python program that uses nested dictionaries to represent warehouse inventory data and allows users to update and retrieve information.

Question 7: You are organizing a team-building activity for a group of students. Write a Python program that randomly assigns students to teams of 3. Ensure that each team has an equal number of members, and print out the list of teams with their members.

```
import random
students = ["Alice", "Bob", "Charlie", "David", "Emma", "Frank", "Grace", "Harry", "Ivy", "Jack", "Kelly", "Liam"]
random.shuffle(students)
teams = []
for i in range(0,len(students),3):
  teams.append(students[i:i+3])
for i, team in enumerate(teams, start=1):
  print(f"Team {i}: {', '.join(team)}")
```

Question 8: You are building a password manager application. Write a Python program that stores passwords for different accounts in a dictionary, where the keys are the account names and the values are the passwords. Allow the user to add, retrieve, and delete passwords.

```
acc_name = []
password =[]
while True:
  name = input("Enter acc name: ")
  if not name:
    break
  pw = input("Enter Password: ")
  if not pw:
    break
  acc_name.append(name)
  password.append(pw)
print(dict(zip(acc_name,password)))
```

Question 9: A fitness app wants to calculate the user's body mass index (BMI) based on their weight and height. Develop a Python program that prompts users to input their weight (in kilograms) and height (in meters) and calculates their BMI using arithmetic operators.

```
weight = float(input("Enter weight(in KG's): "))
height = float(input("Enter Height(in mts): "))

bmi = weight/(height ** 2)

print(bmi)
```

Question 10: A financial institution wants to calculate compound interest for different investment accounts. Develop a Python program that prompts users to input the principal amount, interest rate, and time period, and calculates the compound interest

using arithmetic operators.

```
p_amt = float(input("Enter prinipal amount: "))
rate = float(input("Rate of Interest: "))
time = float(input("Time Period? "))
compound_interest = p_amt * ((1 + rate / time) ** time)
print("Compound Interest is: ",compound_interest)
```

Question 11: You are designing a game where players roll two dice and sum the results. Write a Python program that simulates rolling two dice and calculates the sum. Allow the user to roll the dice multiple times and display the results.

```
import random
while True:
  roll_1 = random.randint(1,6)
  roll_2 = random.randint(1,6)
  total = roll_1 + roll_2
  print(f"you rolled {roll_1} and {roll_2}. Total: {total}")
  if input("Roll again? (y/n): ").lower() != 'y':
        break
```

Question 12: You are developing a program to calculate shipping costs for an online store. Write a Python program that prompts the user to enter the weight of their package and the distance it needs to be shipped. Calculate the shipping cost based on a flat rate per pound and a per-mile charge.

```
package_weight = float(input("Weight in pounds: "))
distance = float(input("Distance in miles: "))
rate_per_pound = 1.5
per_mile_charge = 0.1

flat_rate = package_weight * rate_per_pound
distance_charge = distance * per_mile_charge
print(flat_rate + distance_charge)
```

Question 13: An e-commerce platform wants to offer discounts on purchases during a sale event. Develop a Python program that applies different discount rates based on the total purchase amount and displays the final price after applying the discount.

```
total_purchase = float(input("Enter Total purchase amt: "))
if total_purchase >= 4000:
  total_purchase = total_purchase - (total_purchase * 0.2 )
  print(total_purchase)
elif total_purchase >= 2000:
  total_purchase = total_purchase - (total_purchase * 0.1 )
  print(total_purchase)
elif total_purchase >= 1000:
  total_purchase = total_purchase - (total_purchase * 0.05)
  print(total_purchase)
else:
  print(total_purchase)
```

Question 14: A game developer wants to implement a scoring system for a multiplayer game. Develop a Python program that calculates the total score for each player based on their performance in the game and determines the winner.

```
p1 = list(map(int,input("Enter Player-1 Scores (separated by space): ").split(" ")))
p2 = list(map(int,input("Enter Player-2 Scores (separated by space): ").split(" ")))

if sum(p1)> sum(p2):
  print("'Player-1' is the winner!")
else:
  print("'Player-2' is the winner!")
```

Question 15: A restaurant offers different discounts based on the time of day. Write a Python program that prompts the user for the current time and calculates the appropriate discount based on the following criteria: 10% discount for breakfast (before 10 AM), 20% discount for lunch (between 12 PM and 3 PM), and 15% discount for dinner (after 6 PM). Display the calculated discount to the user.

```
discounts = {'breakfast':10,'lunch':20,'dinner':15}
what_did_you_get = input("Enter (breakfast/lunch/dinner) :").lower()
price = int(input("Enter amount to be paid: "))
if what_did_you_get in discounts:
  print("Discount is: ",discounts[what_did_you_get],"%", " & Bill_amt: ", (price - price*(discounts[what_did_you_get]/100
else:
    print("Invalid option")
```

Question 16: You are organizing a tournament with a single-elimination format. Write a Python program that takes a list of participants and randomly generates the matchups for each round. Print out the matchups for each round until a winner is determined.

Question 17: A data analysis company wants to perform statistical analysis on a dataset containing sales data. Develop a Python program that defines functions for calculating the mean, median, and standard deviation of the dataset and uses them to analyze the data.

```
sales_data = [300,400,100,500,200,700]
mean = sum(sales_data)/len(sales_data)
print("Mean:", mean)

sorted_data = sorted(sales_data)
n = len(sales_data)
if n%2==0:
  print("Median:",(sorted_data[n//2 - 1] + sorted_data[n//2]) / 2)
else:
  print("Median:", sorted_data[n//2])

squared_diff = [(x - mean) ** 2 for x in sales_data]
variance = sum(squared_diff) / len(sales_data)
print("Standard Deviation:", variance ** 0.5)
```

Question 18: A social media platform wants to implement user authentication and authorization functionality. Develop a Python program that defines functions for user login, registration, and access control, and uses them to manage user accounts.

Question 19: You are building a calculator application with various mathematical functions. Write a Python program that defines functions for addition, subtraction, multiplication, and division. Allow the user to choose a function and input the numbers to perform the calculation.

```
def add(x, y):
    return x + y
def subtract(x, y):
    return x - y
def multiply(x, y):
    return x * y
def divide(x, y):
    if y == 0:
        return "Error! Division by zero is not allowed."
    return x / y

operation = {1:'Addition',2:'Subtraction',3:'Multiplication',4: 'Division'}
print(operation)
choice = int(input("Enter your choice (1/2/3/4): "))
if choice in (1,2,3,4):
    num1 = float(input("Enter first number: "))
    num2 = float(input("Enter second number: "))

    if choice == 1:
        print("Result:", add(num1, num2))
    elif choice == 2:
        print("Result:", subtract(num1, num2))
    elif choice == 3:
        print("Result:", multiply(num1, num2))
    elif choice == 4:
        print("Result:", divide(num1, num2))
else:
    print("Invalid choice")
```

Question 20: A company is organizing a team-building exercise where employees will participate in various activities. Write a Python program to randomly assign employees to teams for each activity, ensuring that each team has an equal number of members.

```
import random
emp_ids = [1,2,3,4,5,6,7,8]
random.shuffle(emp_ids)
teams = []
n = 4
for i in range(0,len(emp_ids),n):
  teams.append(emp_ids[i:i+n])
for i, team in enumerate(teams, start=1):
  team_str = ', '.join(str(employee_id) for employee_id in team)
  print(f"Team {i}: {team_str}")
```

Question 21: A ticket booking system encounters errors when processing user requests due to invalid input. Develop a Python program that handles errors such as invalid ticket quantities, incorrect payment information, and expired sessions.

Question 22: A data processing pipeline encounters errors when reading and writing data files due to file permissions and network issues. Develop a Python program that handles errors such as file not found, permission denied, and connection timeout.

Question 23: A bank is developing an ATM system. Develop a Python program that simulates withdrawing money from an ATM. Handle errors such as insufficient funds, incorrect PIN entry, and invalid withdrawal amounts. Display appropriate error messages to the user and allow them to retry or cancel the transaction.