Part I

Code :

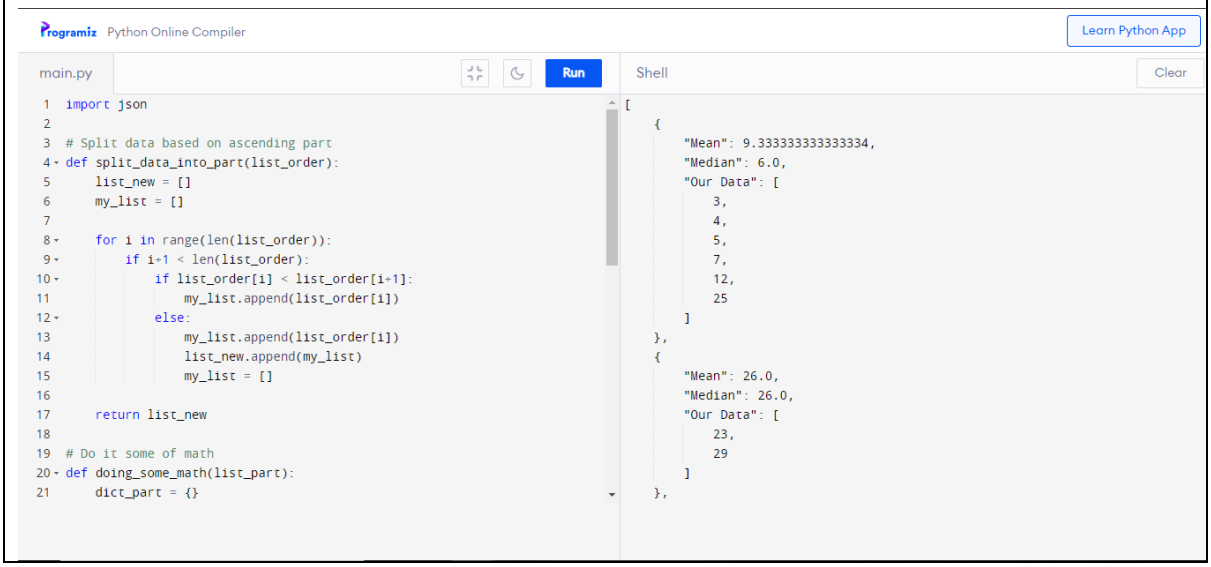
```
import json
# Split data based on ascending part
def split_data_into_part(list_order):
    list_new = []
    my_list = []

    for i in range(len(list_order)):
        if i+1 < len(list_order):
            if list_order[i] < list_order[i+1]:
                my_list.append(list_order[i])
            else:
                my_list.append(list_order[i])
                list_new.append(my_list)
                my_list = []

    return list_new

# Do it some of math
def doing_some_math(list_part):
    dict_part = {}
    median = 0;
    number_data = len(list_part);
    list_part = sorted(list_part)
    mean = sum(list_part) / number_data;

    if number_data % 2 != 0:
        median = list_part[number_data // 2]
        return {
            'Our Data' : list_part,
            'Mean' : mean,
            'Median' : median
        }
    else:
        median = (list_part[int((number_data-1)/2)] + list_part[int(number_data / 2)]) / 2
        return {
            'Our Data' : list_part,
            'Mean' : mean,
            'Median' : median
        }

output_data = [] # output
list_order = [3,4,5,7,12,25,23,29,28,27,31,32] # Input

list_order_part = split_data_into_part(list_order)
for list_math in list_order_part:
    output_data.append(doing_some_math(list_math))

# beautify to be json
```

```
print(json.dumps(output_data, sort_keys=True, indent=4))
```

Output (sample):


```
[
    {
        "Mean": 9.333333333333334,
        "Median": 6.0,
        "Our Data": [
            3,
            4,
            5,
            7,
            12,
            25
        ]
    },
    {
        "Mean": 26.0,
        "Median": 26.0,
        "Our Data": [
            23,
            29
        ]
    },
    {
        "Mean": 28.0,
        "Median": 28,
        "Our Data": [
            28
        ]
    }
]
```


Screenshoot



Part II

Code:

```
import requests
import json

input_data = [
   {"amount" : 15000.0, "currency" : "IDR"},
   {"amount" : 3.1, "currency" : "EUR"}
]
output_data = []

for data in input_data:
   amount = data["amount"]
   currency = data["currency"]
   response =
requests.get(f'https://api.frankfurter.app/latest?amount={amount}&from={currency}&to=USD')
   output_data.append(response.json()["rates"]["USD"])
   # print(json.dumps(response.json(), sort_keys=True, indent=4))

print("============== Input ===============")
print(json.dumps(input_data, sort_keys=True, indent=4))
print("============== Output ===============")
print(json.dumps(output_data, sort_keys=True, indent=4))
```

Output:

```
============== Input ===============
[
   {
      "amount": 15000.0,
      "currency": "IDR"
   },
   {
      "amount": 3.1,
      "currency": "EUR"
   }
]
============== Output ===============
[
   1.0642,
   3.5966
]
```

Screenshoot:

Programiz Python Online Compiler

Learn Python App

main.py

Run

```python
import requests
import json

input_data = [
    {"amount" : 15000.0, "currency" : "IDR"},
    {"amount" : 3.1, "currency" : "EUR"}
]
output_data = []

for data in input_data:
    amount = data["amount"]
    currency = data["currency"]
    response = requests.get(f'https://api.frankfurter.app/latest
        ?amount={amount}&from={currency}&to=USD')
    output_data.append(response.json()["rates"]["USD"])
    # print(json.dumps(response.json(), sort_keys=True, indent=4))

print("============== Input ===============")
print(json.dumps(input_data, sort_keys=True, indent=4))
print("============== Output ===============")
print(json.dumps(output_data, sort_keys=True, indent=4))
```

Shell

Clear

```
============== Input ===============
[
    {
        "amount": 15000.0,
        "currency": "IDR"
    },
    {
        "amount": 3.1,
        "currency": "EUR"
    }
]
============== Output ===============
[
    1.0642,
    3.5966
]
>
```

Part III

Code :

```
import itertools

list_eligible_money = [1000, 10000, 20000]
dict_lembar_uang = {
   "17000, 1",
   "23000, 4",
   "20000, 2",
   "15000, 6",
}

for data in dict_lembar_uang:
   list_split = data.split(",")
   harga_barang = int(list_split[0])
   lembar_uang = int(list_split[1])

   if (harga_barang > 0 and harga_barang <= 100000) and (lembar_uang > 0 and lembar_uang <=
10):
      if lembar_uang <= 1:
         index_search = [value for value in list_eligible_money if value <= harga_barang]
         print(f"Output : {[list_eligible_money[len(index_search)]]} dan lembar ({lembar_uang})
serta jumlah harga barang ({harga_barang})")
      else:
         ten_thousand = (harga_barang // 10000) * 10000
         thousand = ((harga_barang - ten_thousand))

         list_eligible_ten_thousand = [value for value in list_eligible_money if value <=
ten_thousand]
         list_eligible_thousand = [value for value in list_eligible_money if value <= thousand]
         list_put_ten_thousand = []
         list_put_thousand = []
```

```
        # Posibilities of ten thousand
        for L in range(0, len(list_eligible_ten_thousand)+1):
            for subset in itertools.combinations(list_eligible_ten_thousand, L):
                if sum(list(subset)) == ten_thousand:
                    list_put_ten_thousand.append(list(subset))

        count = 0
        for vals in list_eligible_ten_thousand:
            while sum([vals] * count) != ten_thousand:
                count+=1
            list_put_ten_thousand.append([vals] * count)
            count = 0

        # Posibilities of thousand
        for L in range(0, len(list_eligible_thousand)+1):
            for subset in itertools.combinations(list_eligible_thousand, L):
                if sum(list(subset)) == thousand:
                    list_put_thousand.append(list(subset))

        count = 0
        for vals in list_eligible_thousand:
            while sum([vals] * count) != thousand:
                count+=1
            list_put_thousand.append([vals] * count)
            count = 0

        output = []
        for i in range(len(list_put_ten_thousand)):
            for j in range(len(list_put_thousand)):
                if lembar_uang == (len(list_put_ten_thousand[i]) +len(list_put_thousand[j])):
                    if output != list_put_ten_thousand[i]+ list_put_thousand[j]:
                        output = list_put_ten_thousand[i]+ list_put_thousand[j]
        print(f"Output : {output} dan lembar ({lembar_uang}) serta jumlah harga barang
({harga_barang})")
    else:
        print("Melebihi maximum")
```

Output :

Output : [20000] dan lembar (1) serta jumlah harga barang (17000)
Output : [10000, 1000, 1000, 1000, 1000, 1000] dan lembar (6) serta jumlah harga barang (15000)
Output : [20000, 1000, 1000, 1000] dan lembar (4) serta jumlah harga barang (23000)
Output : [10000, 10000] dan lembar (2) serta jumlah harga barang (20000)

Screenshot :

Programiz Python Online Compiler

Learn Python App

main.py

Run

```python
list_eligible_money = [1000, 10000, 20000]
dict_lembar_uang = {
    "17000, 1",
    "23000, 4",
    "20000, 2",
    "15000, 6",
}

for data in dict_lembar_uang:
    list_split = data.split(",")
    harga_barang = int(list_split[0])
    lembar_uang = int(list_split[1])

    if (harga_barang > 0 and harga_barang <= 100000) and (lembar_uang > 0 and lembar_uang <= 10):
        if lembar_uang <= 1:
            index_search = [value for value in list_eligible_money if value <= harga_barang]
            print(f"Output : {[list_eligible_money[len(index_search)]]} dan lembar ({lembar_uang}) serta jumlah harga barang ({harga_barang})")
```

Shell

Clear

```
Output : [20000] dan lembar (1) serta jumlah harga barang (17000)
Output : [10000, 1000, 1000, 1000, 1000, 1000] dan lembar (6) serta jumlah harga barang (15000)
Output : [20000, 1000, 1000, 1000] dan lembar (4) serta jumlah harga barang (23000)
Output : [10000, 10000] dan lembar (2) serta jumlah harga barang (20000)
>
```

Part IV

Code :

```python
import sqlite3
from random import randrange

conn = sqlite3.connect('TestCase4.db')
c = conn.cursor()

c.execute(''' SELECT count(name) FROM sqlite_master WHERE type='table' AND name='STORE_BOOK' ''')
if c.fetchone()[0]==1 :
    print('Table STORE_BOOK exists.')
else:
    conn.execute('''
        CREATE TABLE STORE_BOOK(
            ID_BOOK          INT PRIMARY KEY NOT NULL,
            NAME_BOOK        TEXT    NOT NULL,
            DESCRIPTION_BOOK TEXT    NOT NULL,
            AUTHOR_BOOK      TEXT    NOT NULL,
            LIMIT_BOOK       INT
        );
    ''')
    print('Created Table STORE_BOOK.')

c.execute(''' SELECT count(name) FROM sqlite_master WHERE type='table' AND name='USERS' ''')
if c.fetchone()[0]==1 :
    print('Table USERS exists.')
else:
    conn.execute('''
        CREATE TABLE USERS(
            ID_USER          INT PRIMARY KEY NOT NULL,
            NAME_USER        TEXT    NOT NULL,
            ADDRESS_USER     TEXT    NOT NULL
```

```
        );
    ''')
    print('Created Table USERS.')

c.execute(''' SELECT count(name) FROM sqlite_master WHERE type='table' AND
name='PEMINJAMAN' ''')
if c.fetchone()[0]==1 :
    print('Table PEMINJAMAN exists.')
else:
    conn.execute('''
        CREATE TABLE PEMINJAMAN(
            ID_PEMINJAMAN     INTEGER PRIMARY KEY AUTOINCREMENT,
            ID_USER_PEMINJAM  INT    NOT NULL,
            ID_BOOK           INT    NOT NULL,
            STATUS_PINJAMAN   TEXT   NOT NULL,
            Timestamp DATETIME DEFAULT CURRENT_TIMESTAMP
        );
    ''')
    print('Created Table PEMINJAMAN.')

SESSION_ID_USER = 0
SESSION_NAME_USER = ""

while True:
    print("System Start... Welcome to Perpustakaan")
    print("Pilih Masukkan Perpustakaan")
    print("1. Login")
    print("2. Register")
    print("3. Lihat Buku")
    print("4. Peminjaman")
    pilihan = int(input())

    print()
    if pilihan == 0:
        print("Insert Seluruh Buku .. hanya untuk admin")
        conn.execute("INSERT INTO STORE_BOOK
(ID_BOOK,NAME_BOOK,DESCRIPTION_BOOK,AUTHOR_BOOK,LIMIT_BOOK) VALUES (111, 'Harry
Potter (Cursed of Black)', 'Bla bla bla...', 'Bla bla bla...', 2)")
        conn.execute("INSERT INTO STORE_BOOK
(ID_BOOK,NAME_BOOK,DESCRIPTION_BOOK,AUTHOR_BOOK,LIMIT_BOOK) VALUES (222,
'Learning Programming', 'Bla bla bla...', 'Bla bla bla...', 1)")
        conn.execute("INSERT INTO STORE_BOOK
(ID_BOOK,NAME_BOOK,DESCRIPTION_BOOK,AUTHOR_BOOK,LIMIT_BOOK) VALUES (333, 'Hand
Writing Book', 'Bla bla bla...', 'Bla bla bla...', 3)")
        conn.commit()
    elif pilihan == 1:
        print("=== Login Data ===")
        print("Masukkan ID User : ")
        id_user = int(input())
        cursor = conn.execute("SELECT NAME_USER FROM USERS WHERE ID_USER =
{}".format(id_user))
```

```
        for row in cursor.fetchall():
            SESSION_ID_USER = id_user
            SESSION_NAME_USER = row[0]
            print("Welcome ",row[0])
    elif pilihan == 2:
        print("=== Register Data ===")
        generated_id = randrange(100000, 1000000)
        print("Masukkan Nama : ")
        nama = input()
        print("Masukkan Alamat : ")
        alamat = input()
        print("Save your ID here... ",generated_id)
        script = "INSERT INTO USERS (ID_USER, NAME_USER, ADDRESS_USER) VALUES (?,?,?)"
        conn.execute(script, (generated_id, nama, alamat))
        conn.commit()
    elif pilihan == 3:
        print("Buku Perpustakaan yang tersediaa...")
        cursor = conn.execute("SELECT * FROM STORE_BOOK")
        for row in cursor.fetchall():
            print(f"ID BUKU : {row[0]} -- NAME : {row[1]} -- DESKRIPSI : {row[2]} -- BUKU TERSISA :
{row[4]}")
    elif pilihan == 4:
        print("Masukkan ID Buku yang akan dipinjam")
        id_buku = int(input())
        nama_buku = ""
        sisa_buku = 0
        cursor = conn.execute("SELECT ID_BOOK, NAME_BOOK ,LIMIT_BOOK FROM STORE_BOOK
WHERE ID_BOOK = {}".format(id_buku))
        for row in cursor.fetchall():
            print(f"Buku yang anda pilih : {row[1]} -- SISA : {row[2]}")
            nama_buku = row[1]
            sisa_buku = row[2]
        print("Anda yakin akan meminjam ? YES/NO ")
        persetujuan = str(input())
        if persetujuan == 'YES':
#           if SESSION_ID_USER == 0:
#               print("Anda guest, mohon login")
#           else:
            sisa_buku -= 1
            cursor = conn.execute("UPDATE STORE_BOOK set LIMIT_BOOK = {} WHERE ID_BOOK =
{}".format(sisa_buku, id_buku))

            script_peminjam = "INSERT INTO PEMINJAMAN (ID_USER_PEMINJAM, ID_BOOK,
STATUS_PINJAMAN) VALUES (?,?,?)"
            conn.execute(script_peminjam, (SESSION_ID_USER, id_buku, "PINJAM"))
            print("Anda telah meminjam")

    print()

conn.close()
```