

FDS

Name- Saish Baviskar

Department-AI & DS

Section-SE A , Batch – A4

Roll No-23155

Experiment No. 1: In a second-year computer engineering class, group A students playcricket, and group B

students play

badminton and group C students play football.

Write a Python program using functions to compute the following:

- a)** List of students who play both cricket and badminton.
- b)** List of students who play either cricket or badminton but not both.
- c)** Number of students who play neither cricket nor badminton.
- d)** Number of students who play cricket and football but not badminton.

(NOTE: While realizing the group, duplicate entries should be avoided. Do not use SET built-in functions)

Function for removing duplicate entries from the group

```
def removeDuplicate(d):
```

```
    lst=[]
```

```
    for i in d:
```

```
        if i not in lst:
```

```
            lst.append(i)
```

```
    return lst
```

```
#<.....->
```

```
# Function for finding intersection between two sets (A&B)
```

```
def intersection(lst1,lst2):
```

```
    lst3=[]
```

```
    for val in lst1:
```

```
        if val in lst2:
```

```
            lst3.append(val)
```

```
    return lst3
```

```
#<.....>
```

```
# Function for finding union of two sets (A|B)
```

```
def union(lst1,lst2):
```

```
    lst3=lst1.copy()
```

```
    for val in lst2:
```

```
        if val not in lst3:
```

```
            lst3.append(val)
```

```
    return lst3
```

```
#<.....->
```

```
# Function for finding difference between two sets (A-B)
```

```
def diff(lst1,lst2):
```

```
    lst3=[]
```

```
    for val in lst1:
```

```
    if val not in lst2:
        lst3.append(val)
    return lst3
```

```
#<.....>
```

```
# Function for finding symmetric difference of two sets ( $A \Delta B$ )
```

```
def sym_diff(lst1, lst2):
    lst3=[]
    D1=diff(lst1, lst2)
    print("Difference between Cricket and Badminton (C-B) is : ", D1)
    D2=diff(lst2, lst1)
    print("Difference between Badminton and Cricket (B-C) is : ", D2)
    lst3=union(D1, D2)
    return lst3
```

```
#<.....->
```

```
# Functon for finding List of students who play both cricket and badminton
```

```
def CB(lst1, lst2):
    lst3=intersection(lst1, lst2)
    print("\n\nList of students who play both cricket and badminton is : ", lst3)
    return len(lst3)
```

```
#<.....>
```

```
# Function for finding List of students who play either cricket or badminton but not both
```

```
def eCeB(lst1,lst2):  
    lst3=sym_diff(lst1,lst2)  
    print("\nList of students who play either cricket or badminton but not both is :",lst3)  
    return len(lst3)
```

```
#<.....->
```

```
# Function for finding Number of students who play neither cricket nor badminton
```

```
def nCnB(lst1,lst2,lst3):  
    lst4=diff(lst1,union(lst2,lst3))  
    print("\n\nList of students who play neither cricket nor badminton is :",lst4)  
    return len(lst4)
```

```
#<.....>
```

```
# Function for finding Number of students who play cricket and football but not badminton
```

```
def CBnF(lst1,lst2,lst3):  
    lst4=diff(intersection(lst1,lst2),lst3)  
    print("\n\nList of students who play cricket and football but not badminton is :",lst4)  
    return len(lst4)
```

```
#<.....->
```

```
# Main function
```

```
# Creating an empty list for SE COMP
```

```

SEComp = []

n = int(input("\nEnter number of students in SE COMP: "))

print("Enter the names of",n,"students (Please press ENTER after entering each students name) :")

for i in range(0, n):

    ele = input()

    SEComp.append(ele) # adding the element

print("Original list of students in SEComp : " + str(SEComp))

```

#<.....>

Creating an empty list for Cricket

```

Cricket = []

n = int(input("\nEnter number of students who play cricket : "))

print("Enter the names of",n,"students who play cricket (Please press ENTER after entering each students name) :")

for i in range(0, n):

    ele = input()

    Cricket.append(ele) # adding the element

print("Original list of students playing cricket is : " + str(Cricket))

Cricket=removeDuplicate(Cricket)

print("The list of students playing cricket after removing duplicates : " + str(Cricket))

```

#<.....-->

Creating an empty list for Football

```

Football = []

n = int(input("\nEnter number of students who play football : "))

```

```
print("Enter the name of",n,"students who play football (Please press ENTER after entering each students name) :")
```

```
for i in range(0, n):
```

```
    ele = input()
```

```
    Football.append(ele) # adding the element
```

```
print("Original list of students playing football :"+str(Football))
```

```
Football=removeDuplicate(Football)
```

```
print("The list of students playing football after removing duplicates : "+str(Football))
```

```
#<.....->
```

```
# Creating an empty list for Badminton
```

```
Badminton = []
```

```
n = int(input("\n\nEnter number of students who play badminton : "))
```

```
print("Enter the name of",n,"students who play badminton (Please press ENTER after entering each students name) :")
```

```
for i in range(0, n):
```

```
    ele = input()
```

```
    Badminton.append(ele) # adding the element
```

```
print("Original list of students playing badminton :"+str(Badminton))
```

```
Badminton=removeDuplicate(Badminton)
```

```
print("The list of students playing badminton after removing duplicates : "+str(Badminton))
```

```
#<.....>
```

```
flag=1
```

```
while flag==1:
```

```
    print("\n\n.....MENU.....\n")
```

```
    print("1. List of students who play both cricket and badminton")
```

```

print("2. List of students who play either cricket or badminton but not both")
print("3. List of students who play neither cricket nor badminton")
print("4. Number of students who play cricket and football but not badminton")
print("5. Exit\n")
ch=int(input("Enter your Choice (from 1 to 5) :"))

if ch==1:
    print("Number of students who play both cricket and badminton : ",
CB(Cricket,Badminton))
    a = input("\n\nDo you want to continue (yes/no) :")
    if a == "yes":
        flag = 1
    else:
        flag = 0
    print("Thanks for using this program!")

elif ch==2:
    print("Number of students who play either cricket or badminton but not both : ",
eCeB(Cricket, Badminton))
    a = input("\n\nDo you want to continue (yes/no) :")
    if a == "yes":
        flag = 1
    else:
        flag = 0
    print("Thanks for using this program!")

elif ch==3:
    print("Number of students who play neither cricket nor badminton : ",
nCnB(SEComp,Cricket,Badminton))
    a = input("\n\nDo you want to continue (yes/no) :")

```

```
if a == "yes":
```

```
    flag = 1
```

```
else:
```

```
    flag = 0
```

```
    print("Thanks for using this program!")
```

```
elif ch==4:
```

```
    print("Number of students who play cricket and football but not badminton : ",  
CBnF(Cricket,Football,Badminton))
```

```
    a = input("\n\nDo you want to continue (yes/no) :")
```

```
    if a == "yes":
```

```
        flag = 1
```

```
    else:
```

```
        flag = 0
```

```
        print("Thanks for using this program!")
```

```
elif ch==5:
```

```
    flag=0
```

```
    print("Thanks for using this program!")
```

```
else:
```

```
    print("!!Wrong Choice!! ")
```

```
    a=input("\n\nDo you want to continue (yes/no) :")
```

```
    if a=="yes":
```

```
        flag=1
```

```
    else:
```

```
        flag=0
```

```
        print("Thanks for using this program!")
```

```
#<-----END OF PROGRAM----->
```


OUTPUT:-

```
(base) ubuntu@ubuntu-optiplex-5000:~$ python SETTHEORY.py

Enter number of students in SE COMP: 5
Enter the names of 5 students (Please press ENTER after entering each students name) :
RAJ
VIKRANT
ATHARV
SAM
RAM
Original list of students in SEComp : ['RAJ', 'VIKRANT', 'ATHARV', 'SAM', 'RAM']

Enter number of students who play cricket : 3
Enter the names of 3 students who play cricket (Please press ENTER after entering each students name) :
RAJ
ATHARV
SAM
Original list of students playing cricket is :['RAJ', 'ATHARV', 'SAM']
The list of students playing cricket after removing duplicates : ['RAJ', 'ATHARV', 'SAM']

Enter number of students who play football : 2
Enter the name of 2 students who play football (Please press ENTER after entering each students name) :
VIKRANT
RAM
Original list of students playing football :['VIKRANT', 'RAM']
The list of students playing football after removing duplicates : ['VIKRANT', 'RAM']

Enter number of students who play badminton : 3
Enter the name of 3 students who play badminton (Please press ENTER after entering each students name) :
VIKRANT
RAJ
ATHARV
Original list of students playing badminton :['VIKRANT', 'RAJ', 'ATHARV']
The list of students playing badminton after removing duplicates : ['VIKRANT', 'RAJ', 'ATHARV']

-----MENU-----
1. List of students who play both cricket and badminton
2. List of students who play either cricket or badminton but not both
3. List of students who play neither cricket nor badminton
4. Number of students who play cricket and football but not badminton
5. Exit

Enter your Choice (from 1 to 5) :1

List of students who play both cricket and badminton is :  ['RAJ', 'ATHARV']
Number of students who play both cricket and badminton :  2
```

Do you want to continue (yes/no) :yes

-----MENU-----

1. List of students who play both cricket and badminton
2. List of students who play either cricket or badminton but not both
3. List of students who play neither cricket nor badminton
4. Number of students who play cricket and football but not badminton
5. Exit

Enter your Choice (from 1 to 5) :2

Difference between Cricket and Badminton (C-B) is : ['SAM']

Difference between Badminton and Cricket (B-C) is : ['VIKRANT']

List of students who play either cricket or badminton but not both is : ['SAM', 'VIKRANT']

Number of students who play either cricket or badminton but not both : 2

Do you want to continue (yes/no) :yes

-----MENU-----

1. List of students who play both cricket and badminton
2. List of students who play either cricket or badminton but not both
3. List of students who play neither cricket nor badminton
4. Number of students who play cricket and football but not badminton
5. Exit

Enter your Choice (from 1 to 5) :3

List of students who play neither cricket nor badminton is : ['RAM']

Number of students who play neither cricket nor badminton : 1

Do you want to continue (yes/no) :yes

-----MENU-----

1. List of students who play both cricket and badminton
2. List of students who play either cricket or badminton but not both
3. List of students who play neither cricket nor badminton
4. Number of students who play cricket and football but not badminton
5. Exit

-----MENU-----

1. List of students who play both cricket and badminton
2. List of students who play either cricket or badminton but not both
3. List of students who play neither cricket nor badminton
4. Number of students who play cricket and football but not badminton
5. Exit

Enter your Choice (from 1 to 5) :4

List of students who play cricket and football but not badminton is : []

Number of students who play cricket and football but not badminton : 0

Do you want to continue (yes/no) :yes

-----MENU-----

1. List of students who play both cricket and badminton
2. List of students who play either cricket or badminton but not both
3. List of students who play neither cricket nor badminton
4. Number of students who play cricket and football but not badminton
5. Exit

Enter your Choice (from 1 to 5) :5

Thanks for using this program!

(base) ubuntu@ubuntu-optiplex-5000:~\$

FDS

Name- Saish Baviskar

Department-AI & DS

Section-SE A , Batch – A4

Roll No-23155

Experiment Number 2 : Write a python program to store marks stored in subject "Fundamentals of Data Structure" by N students in the class. Write functions to compute the following:

1. The average score of the class.
2. Highest score and lowest score of the class.
3. Count of students who were absent for the test.
4. Display mark with highest frequency.

Function for average score of the class

```
def average(listofmarks):
```

```
    sum=0
```

```
    count=0
```

```
    for i in range(len(listofmarks)):
```

```
        if listofmarks[i]!=-999:
```

```
            sum+=listofmarks[i]
```

```
            count+=1
```

```
    avg=sum/count
```

```
    print("Total Marks : ", sum)
```

```
    print("Average Marks : {:.2f}".format(avg))
```

```
#<.....->
```

Function for Highest score in the test for the class

```
def Maximum(listofmarks):  
    for i in range(len(listofmarks)):  
        if listofmarks[i]!=-999:  
            Max=listofmarks[0]  
            break  
    for i in range(1,len(listofmarks)):  
        if listofmarks[i]>Max:  
            Max=listofmarks[i]  
    return(Max)
```

#<.....-.....>

Function for Lowest score in the test for the class

```
def Minimum(listofmarks):  
    for i in range(len(listofmarks)):  
        if listofmarks[i]!=-999:  
            Min=listofmarks[0]  
            break  
    for i in range(1,len(listofmarks)):  
        if listofmarks[i]<Min:  
            Min=listofmarks[i]  
    return(Min)
```

#<.....>

Function for counting the number of students absent for the test

```
def absentcount(listofmarks):
    count=0
    for i in range(len(listofmarks)):
        if listofmarks[i]==-999:
            count+=1
    return(count)
```

```
#<.....>
```

Function for displaying marks with highest frequency

```
def maxFrequency(listofmarks):
    i=0
    Max=0
    print("Marks | Frequency")
    for j in listofmarks:
        if (listofmarks.index(j)==i):
            print(j," | ",listofmarks.count(j))
            if listofmarks.count(j)>Max:
                Max=listofmarks.count(j)
                mark=j
        i=i+1
    return(mark,Max)
```

```
#<.....>
```

Main function

```
marksinFDS=[]
numberofstudents=int(input("Enter total number of students : "))
for i in range(numberofstudents):
    marks=int(input("Enter marks of student "+str(i+1)+" : "))
    marksinFDS.append(marks)
```

```
flag=1
```

```
while flag==1:
    print("\n\n.....MENU.....\n")
    print("1. Total and Average Marks of the Class")
    print("2. Highest and Lowest Marks in the Class")
    print("3. Number of Students absent for the test")
    print("4. Marks with Highest Frequency")
    print("5. Exit\n")
    ch=int(input("Enter your Choice (from 1 to 5) :"))
```

```
if ch==1:
    average(marksinFDS)
    a = input("Do you want to continue (yes/no) :")
    if a == "yes":
        flag = 1
    else:
        flag = 0
        print("Thanks for using this program!")
```

```
elif ch==2:
    print("Highest Score in Class : ", Maximum(marksinFDS))
    print("Lowest Score in Class : ", Minimum(marksinFDS))
    a = input("Do you want to continue (yes/no) :")
```

```
if a == "yes":  
    flag = 1  
else:  
    flag = 0  
    print("Thanks for using this program!")
```

```
elif ch==3:  
    print("Number of Students absent in the test : ",absentcount(marksinFDS))  
    a = input("Do you want to continue (yes/no) :")  
    if a == "yes":  
        flag = 1  
    else:  
        flag = 0  
        print("Thanks for using this program!")
```

```
elif ch==4:  
    mark,fr = maxFrequency(marksinFDS)  
    print("Highest frequency is of marks {0} that is {1} ".format(mark,fr))  
    a = input("Do you want to continue (yes/no) :")  
    if a == "yes":  
        flag = 1  
    else:  
        flag = 0  
        print("Thanks for using this program!")
```

```
elif ch==5:  
    flag=0  
    print("Thanks for using this program!")
```

else:

```
print("!!Wrong Choice!! ")
```

```
a=input("Do you want to continue (yes/no) :")
```

```
if a=="yes":
```

```
    flag=1
```

else:

```
    flag=0
```

```
print("Thanks for using this program!")
```

```
#<.....-END OF PROGRAM.....->
```


OUTPUT:-

```
-----MENU-----
1. Total and Average Marks of the Class
2. Highest and Lowest Marks in the Class
3. Number of Students absent for the test
4. Marks with Highest Frequency
5. Exit

Enter your Choice (from 1 to 5) :4
Marks | Frequency
-999  | 1
98    | 2
69    | 1
52    | 1
Highest frequency is of marks 98 that is 2
Do you want to continue (yes/no) :yes

-----MENU-----
1. Total and Average Marks of the Class
2. Highest and Lowest Marks in the Class
3. Number of Students absent for the test
4. Marks with Highest Frequency
5. Exit

Enter your Choice (from 1 to 5) :5
Thanks for using this program!
(base) ubuntu@ubuntu-optiplex-5000:~$
```

```
(base) ubuntu@ubuntu-optiplex-5000:~$ python marks.py
```

```
Enter total number of students : 5
```

```
Enter marks of student 1 : -999
```

```
Enter marks of student 2 : 98
```

```
Enter marks of student 3 : 98
```

```
Enter marks of student 4 : 69
```

```
Enter marks of student 5 : 52
```

```
-----MENU-----
```

1. Total and Average Marks of the Class
2. Highest and Lowest Marks in the Class
3. Number of Students absent for the test
4. Marks with Highest Frequency
5. Exit

```
Enter your Choice (from 1 to 5) :1
```

```
Total Marks : 317
```

```
Average Marks : 79.25
```

```
Do you want to continue (yes/no) :yes
```

```
-----MENU-----
```

1. Total and Average Marks of the Class
2. Highest and Lowest Marks in the Class
3. Number of Students absent for the test
4. Marks with Highest Frequency
5. Exit

```
Enter your Choice (from 1 to 5) :2
```

```
Highest Score in Class : 98
```

```
Lowest Score in Class : -999
```

```
Do you want to continue (yes/no) :yes
```

```
-----MENU-----
```

1. Total and Average Marks of the Class
2. Highest and Lowest Marks in the Class
3. Number of Students absent for the test
4. Marks with Highest Frequency
5. Exit

```
Enter your Choice (from 1 to 5) :3
```

```
Number of Students absent in the test : 1
```

```
Do you want to continue (yes/no) :yes
```


FDS

Name- Saish Baviskar

Department-AI & DS

Section-SE A , Batch – A4

Roll No-23155

Experiment No.3:- Write a Python program that computes the net amount of a bank account based a transaction log from console input. The transaction log format is shown as following: D 100 W 200 (Withdrawal is not allowed if balance is going negative.

Write functions for withdraw and deposit) D means deposit while W means withdrawal.

Suppose the following input is supplied to the program:

D 300, D 300 , W 200, D 100 Then, the output should be: 500

```
def deposit(num):
```

```
    global balance
```

```
    balance += num
```

```
def withdrawal(num):
```

```
    global balance
```

```
    if balance >= num:
```

```
        balance -= num
```

```
    else:
```

```
        print("Withdrawal not possible because balance is insufficient.")
```

```
balance = 0
```

```
while True:
```

```
    data = input("Please enter the transaction details (or type 'Exit' to end): ")
```

```

if data.lower() == 'exit':
    break

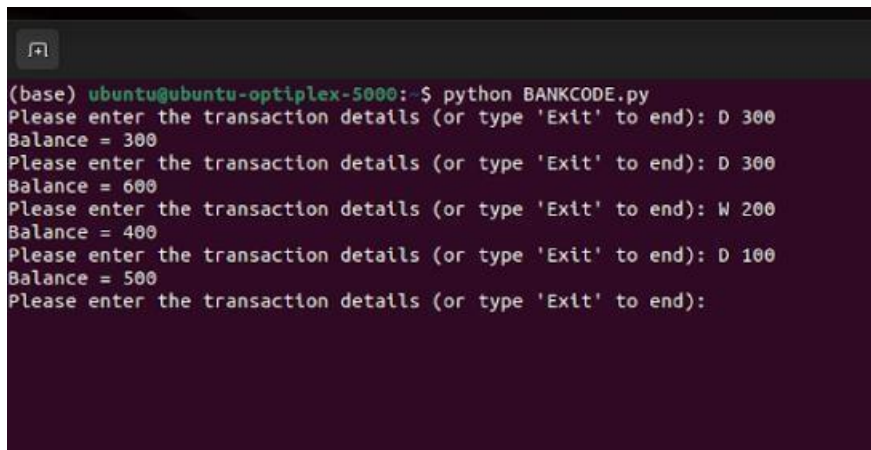
try:
    action, amount_str = data.split()
    amount = int(amount_str)

    if action.upper() == 'D':
        deposit(amount)
    elif action.upper() == 'W':
        withdrawal(amount)
    else:
        print("Invalid transaction type. Use 'D' for deposit and 'W' for withdrawal.")

    print("Balance =", balance)
except ValueError:
    print("Invalid input format. Please enter in the format 'D amount' or 'W amount'.")

```

OUTPUT:-



```

(base) ubuntu@ubuntu-optiplex-5000:~$ python BANKCODE.py
Please enter the transaction details (or type 'Exit' to end): D 300
Balance = 300
Please enter the transaction details (or type 'Exit' to end): D 300
Balance = 600
Please enter the transaction details (or type 'Exit' to end): W 200
Balance = 400
Please enter the transaction details (or type 'Exit' to end): D 100
Balance = 500
Please enter the transaction details (or type 'Exit' to end):

```

FDS

Name- Saish Baviskar

Department-AI & DS

Section-SE A , Batch – A4

Roll No-23155

Experiment No 4:

a) Write a Python program to store names and mobile numbers of your friends in sorted order on names. Search your friend from list using binary search (recursive and non- recursive). Insert friend if not present in phonebook

```
def binary_search_recursive(phonebook, name, left, right):  
    """Perform a recursive binary search for the given name."""  
    if left > right:  
        return None  
    mid = (left + right) // 2  
    if phonebook[mid][0] == name:  
        return phonebook[mid]  
    elif phonebook[mid][0] < name:  
        return binary_search_recursive(phonebook, name, mid + 1, right)  
    else:  
        return binary_search_recursive(phonebook, name, left, mid - 1)
```

```
def binary_search_nonrecursive(phonebook, name):  
    """Perform a non-recursive binary search for the given name."""  
    left, right = 0, len(phonebook) - 1  
    while left <= right:  
        mid = (left + right) // 2  
        if phonebook[mid][0] == name:
```

```
        return phonebook[mid]
    elif phonebook[mid][0] < name:
        left = mid + 1
    else:
        right = mid - 1
return None
```

```
def insert_friend(phonebook, name, number):
    """Insert a new friend into the phonebook maintaining the sorted order."""
    for i, entry in enumerate(phonebook):
        if entry[0] == name:
            print(f"{name} is already in the phonebook.")
            return
        elif entry[0] > name:
            phonebook.insert(i, (name, number))
            print(f"{name} has been added to the phonebook.")
            return
    phonebook.append((name, number))
    print(f"{name} has been added to the phonebook.")
```

```
def main():
    phonebook = [] # List of tuples (name, number)

    while True:
        print("\nPhonebook Options:")
        print("1. Search for a friend (binary search - recursive)")
        print("2. Search for a friend (binary search - non-recursive)")
        print("3. Insert a friend")
        print("4. Exit")
```

try:

choice = int(input("Enter your choice: "))

except ValueError:

print("Invalid input. Please enter a number.")

continue

if choice == 1:

name = input("Enter the name to search: ")

result = binary_search_recursive(phonebook, name, 0, len(phonebook) - 1)

if result:

print(f"Name: {result[0]}, Number: {result[1]}")

else:

print(f"{name} not found in the phonebook.")

elif choice == 2:

name = input("Enter the name to search: ")

result = binary_search_nonrecursive(phonebook, name)

if result:

print(f"Name: {result[0]}, Number: {result[1]}")

else:

print(f"{name} not found in the phonebook.")

elif choice == 3:

name = input("Enter the name to insert: ")

number = input("Enter the number: ")

insert_friend(phonebook, name, number)

elif choice == 4:


```
        print("Exiting the phonebook application.")
        break

    else:

        print("Invalid choice. Please enter a valid option.")

if __name__ == "__main__":
    main()
```

OUTPUT:-

```
ubuntu@ubuntu-optip... ×  ubuntu@ubuntu-optip... ×  ubuntu@ubuntu-optip... ×  
(base) ubuntu@ubuntu-optiplex-5000:~$ python BINARYSEARCHPHONEBOOK.py  
Phonebook Options:  
1. Search for a friend (binary search - recursive)  
2. Search for a friend (binary search - non-recursive)  
3. Insert a friend  
4. Exit  
Enter your choice: 3  
Enter the name to insert: Atharv  
Enter the number: 8767993565  
Atharv has been added to the phonebook.  
  
Phonebook Options:  
1. Search for a friend (binary search - recursive)  
2. Search for a friend (binary search - non-recursive)  
3. Insert a friend  
4. Exit  
Enter your choice: 1  
Enter the name to search: Atharv  
Name: Atharv, Number: 8767993565  
  
Phonebook Options:  
1. Search for a friend (binary search - recursive)  
2. Search for a friend (binary search - non-recursive)  
3. Insert a friend  
4. Exit  
Enter your choice: 2  
Enter the name to search: Atharv  
Name: Atharv, Number: 8767993565  
  
Phonebook Options:  
1. Search for a friend (binary search - recursive)  
2. Search for a friend (binary search - non-recursive)  
3. Insert a friend  
4. Exit  
Enter your choice: 4  
Exiting the phonebook application.  
(base) ubuntu@ubuntu-optiplex-5000:~$
```

b) Write a Python program to store names and mobile numbers of your friends in sorted order on names. Search your friend from list using Fibonacci search. Insert friend if not present in phonebook.

```
def fibonacci_search(phonebook, name):
```

```
    fib_m_minus_2 = 0
```

```
    fib_m_minus_1 = 1
```

```
    fib_m = fib_m_minus_1 + fib_m_minus_2
```

```
    while fib_m < len(phonebook):
```

```
        fib_m_minus_2 = fib_m_minus_1
```

```
        fib_m_minus_1 = fib_m
```

```
        fib_m = fib_m_minus_1 + fib_m_minus_2
```

```
    offset = -1
```

```
    while fib_m > 1:
```

```
        i = min(offset + fib_m_minus_2, len(phonebook) - 1)
```

```
        if phonebook[i][0] < name:
```

```
            fib_m, fib_m_minus_1, fib_m_minus_2 = fib_m_minus_1, fib_m_minus_2,
fib_m_minus_1
```

```
            offset = i
```

```
        elif phonebook[i][0] > name:
```

```
            fib_m, fib_m_minus_1, fib_m_minus_2 = fib_m_minus_2, fib_m_minus_1 -
fib_m_minus_2, fib_m_minus_2 - fib_m_minus_1
```

```
        else:
```

```
            return phonebook[i]
```

```
    if (fib_m_minus_1 and phonebook[offset + 1][0] == name):
```

```
    return phonebook[offset + 1]
```

```
    return None
```

```
def insert_friend(phonebook, name, number):
```

```
    for i, entry in enumerate(phonebook):
```

```
        if entry[0] == name:
```

```
            print(f"{name} is already in the phonebook.")
```

```
            return
```

```
        elif entry[0] > name:
```

```
            phonebook.insert(i, (name, number))
```

```
            print(f"{name} has been added to the phonebook.")
```

```
            return
```

```
    phonebook.append((name, number))
```

```
    print(f"{name} has been added to the phonebook.")
```

```
def main():
```

```
    phonebook = [] # List of tuples (name, number)
```

```
    while True:
```

```
        print("Phonebook Options:")
```

```
        print("1. Search for a friend (Fibonacci search)")
```

```
        print("2. Insert a friend")
```

```
        print("3. Exit")
```

```
    choice = int(input("Enter your choice: "))
```

```
    if choice == 1:
```

```
        name = input("Enter the name to search: ")
```

```
result = fibonacci_search(phonebook, name)

if result:

    print(f"Name: {result[0]}, Number: {result[1]}")

else:

    print(f"{name} not found in the phonebook.")
```

```
elif choice == 2:

    name = input("Enter the name to insert: ")

    number = input("Enter the number: ")

    insert_friend(phonebook, name, number)
```

```
elif choice == 3:

    break
```

```
else:

    print("Invalid choice. Please enter a valid option.")
```

```
if __name__ == "__main__":

    main()
```

OUTPUT:-

```
ubuntu@ubuntu-o... ×  ubuntu@ubuntu-o... ×  ubuntu@ubuntu-o... ×  ubuntu@ubur
(base) ubuntu@ubuntu-optiplex-5000:~$ python FIBONACCISEARCH.py
Phonebook Options:
1. Search for a friend (Fibonacci search)
2. Insert a friend
3. Exit
Enter your choice: 2
Enter the name to insert: Atharv
Enter the number: 8767993565
Atharv has been added to the phonebook.
Phonebook Options:
1. Search for a friend (Fibonacci search)
2. Insert a friend
3. Exit
Enter your choice: 1
Enter the name to search: Atharv
Name: Atharv, Number: 8767993565
Phonebook Options:
1. Search for a friend (Fibonacci search)
2. Insert a friend
3. Exit
Enter your choice: 3
(base) ubuntu@ubuntu-optiplex-5000:~$
```

FDS

Name- Saish Baviskar

Department-AI & DS

Section-SE A , Batch – A4

Roll No-23155

Experiment No.5: Write a Python program to store first year percentage of students in array.
Write function for sorting array of floating point numbers in ascending order using

a) Selection Sort

b) Bubble sort and display top five scores.

```
def selection_sort(arr):
```

```
    n = len(arr)
```

```
    for i in range(n):
```

```
        min_index = i
```

```
        for j in range(i + 1, n):
```

```
            if arr[j] < arr[min_index]:
```

```
                min_index = j
```

```
        arr[i], arr[min_index] = arr[min_index], arr[i]
```

```
def bubble_sort(arr):
```

```
    n = len(arr)
```

```
    for i in range(n):
```

```
        swapped = False
```

```
        for j in range(0, n - i - 1):
```

```
            if arr[j] > arr[j + 1]:
```

```
                arr[j], arr[j + 1] = arr[j + 1], arr[j]
```

```
                swapped = True
```

```
    if not swapped:
```

```
break
```

```
def display_top_scores(arr, top_n):
```

```
    if top_n > len(arr):
```

```
        top_n = len(arr)
```

```
    print(f"Top {top_n} Scores:")
```

```
    for i in range(top_n):
```

```
        print(f"{i+1}. {arr[i]}%")
```

```
def main():
```

```
    # Input the first-year percentage of students
```

```
    n = int(input("Enter the number of students: "))
```

```
    percentages = []
```

```
    for i in range(n):
```

```
        percentage = float(input(f"Enter the percentage for student {i + 1}: "))
```

```
        percentages.append(percentage)
```

```
    # Sorting using selection sort
```

```
    sorted_percentages_selection = percentages.copy() # Create a copy to sort
```

```
    selection_sort(sorted_percentages_selection)
```

```
    print("Sorted array using Selection Sort:")
```

```
    display_top_scores(sorted_percentages_selection, 5)
```

```
    # Sorting using bubble sort
```

```
    sorted_percentages_bubble = percentages.copy() # Create a copy to sort
```

```
    bubble_sort(sorted_percentages_bubble)
```

```
    print("\nSorted array using Bubble Sort:")
```

```
    display_top_scores(sorted_percentages_bubble, 5)
```



```
if __name__ == "__main__":  
    main()
```

OUTPUT:-

```
ubuntu@ubuntu-optiplex-5000: ~  
(base) ubuntu@ubuntu-optiplex-5000:~$ python SELECTIONSORTBUBBLESORT.py  
Enter the number of students: 7  
Enter the percentage for student 1: 60  
Enter the percentage for student 2: 70  
Enter the percentage for student 3: 40  
Enter the percentage for student 4: 50  
Enter the percentage for student 5: 80  
Enter the percentage for student 6: 90  
Enter the percentage for student 7: 100  
Sorted array using Selection Sort:  
Top 5 Scores:  
1. 40.0%  
2. 50.0%  
3. 60.0%  
4. 70.0%  
5. 80.0%  
  
Sorted array using Bubble Sort:  
Top 5 Scores:  
1. 40.0%  
2. 50.0%  
3. 60.0%  
4. 70.0%  
5. 80.0%  
(base) ubuntu@ubuntu-optiplex-5000:~$
```

FDS

Name- Saish Baviskar

Department-AI & DS

Section-SE A , Batch – A4

Roll No-23155

Expeiment No.6:- Write a Python program to store first year percentage of students in array. Write function for sorting array of floating point numbers in ascending order using quick sort and display top five scores.

```
import array as arr
```

```
# Accept the % marks of the students
```

```
def accept_perc():
```

```
    a = arr.array('f', [])
```

```
    no_stud = int(input("Enter the number of Students : "))
```

```
    for i in range(0, no_stud):
```

```
        a.append(float(input("Enter the First Year % of Student[{0}] : ".format(i))))
```

```
    return a
```

```
# Print the % marks of the Students
```

```
def print_perc(a):
```

```
    for i in range(0, len(a)):
```

```
        print("\t {0:.2f}".format(a[i]), end=" ")
```

```
    print()
```

```
# Quick Sort Partition function
```

```

def partition(a, start, end):
    pivot = a[start]
    low = start + 1
    high = end

    while True:
        # If the current value we're looking at is larger than the pivot
        # it's in the right place (right side of pivot) and we can move left,
        # to the next element.
        # We also need to make sure we haven't surpassed the low pointer, since that
        # indicates we have already moved all the elements to their correct side of the pivot
        while low <= high and a[high] >= pivot:
            high = high - 1

        # Opposite process of the one above
        while low <= high and a[low] <= pivot:
            low = low + 1

        # We either found a value for both high and low that is out of order
        # or low is higher than high, in which case we exit the loop
        if low > high:
            a[low], a[high] = a[high], a[low]
            # The loop continues
        else:
            # We exit out of the loop
            break

    a[start], a[high] = a[high], a[start]

```

```
    return high
```

```
# Quick Sort function
```

```
def quick_sort(a, start, end):
```

```
    if start >= end:
```

```
        return
```

```
    p = partition(a, start, end)
```

```
    quick_sort(a, start, p - 1)
```

```
    quick_sort(a, p + 1, end)
```

```
    return a
```

```
# Top 5 Score
```

```
def top_five(a):
```

```
    print("Top five score are : ")
```

```
    cnt = len(a)
```

```
    if cnt < 5:
```

```
        start, stop = cnt - 1, -1 # stop set to -1 as we want to print the 0th element
```

```
    else:
```

```
        start, stop = cnt - 1, cnt - 6
```

```
    for i in range(start, stop, -1):
```

```
        print("\t {0:.2f}".format(a[i]), end=" ")
```

```
# Driver program
```

```
if __name__ == "__main__":

    unsort_A = arr.array('f', [])
    quick_sort_A = arr.array('f', [])
    flag = 1

    while flag == 1:

        print("\n 1. Accept array elements \n 2. Display the Elements \n 3. Quick Sort \n 4.
exit")

        choice = int(input("Enter your choice : "))

        if choice == 1:

            unsort_A = accept_perc()

        elif choice == 2:

            print_perc(unsort_A)

        elif choice == 3:

            print("Elements after sorting using Sort :")
            quick_sort_A = quick_sort(unsort_A, 0, len(unsort_A) - 1)
            print_perc(quick_sort_A)
            top_five(quick_sort_A)

        else:

            print("Wrong choice")
            flag = 0
```

OUTPUT:-

```
ubuntu@ubuntu-optiplex-5000: ~  
(base) ubuntu@ubuntu-optiplex-5000:~$ python QUICKSORT.py  
  
1. Accept array elements  
2. Display the Elements  
3. Quick Sort  
4. Exit  
Enter your choice: 1  
Enter the number of Students: 7  
Enter the First Year % of Student[0]: 50  
Enter the First Year % of Student[1]: 60  
Enter the First Year % of Student[2]: 70  
Enter the First Year % of Student[3]: 80  
Enter the First Year % of Student[4]: 90  
Enter the First Year % of Student[5]: 100  
Enter the First Year % of Student[6]: 20  
  
1. Accept array elements  
2. Display the Elements  
3. Quick Sort  
4. Exit  
Enter your choice: 2  
Displaying the elements:  
50.00 60.00 70.00 80.00 90.00 100.00 20.00  
  
1. Accept array elements  
2. Display the Elements  
3. Quick Sort  
4. Exit  
Enter your choice: 3  
Elements after sorting using Quick Sort:  
20.00 50.00 60.00 70.00 80.00 90.00 100.00  
Top five scores are:  
100.00 90.00 80.00 70.00 60.00  
  
1. Accept array elements  
2. Display the Elements  
3. Quick Sort  
4. Exit  
Enter your choice: 4  
Exiting the program.  
(base) ubuntu@ubuntu-optiplex-5000:~$
```