Harsh Kasliwal

2003085

C21

Experiment 4

Q1. Write a menu driven program to demonstrate use of list in python

- Put even and odd elements in two different list
- Merge and sort two list
- Update the first element with a value X
- Print middle element of list

A list can be defined as a collection of values or items of different types.
In Python, a list has the following characteristics:

- The lists are ordered.
- The element of the list can access by index.
- The lists are the mutable type.
- The lists are mutable types.
- A list can store the number of various elements.

A list can be created by using square brackets. The elements can be put inside square brackets separated by a comma. If there are no elements, empty square brackets can be used.
For example,

```
a = [5, 3, 7]  # List with 3 elements
b = []  # Empty list
```

The index starts from 0 and goes to length - 1. The first element of the list is stored at the 0th index, the second element of the list is stored at the 1st index, and so on. Python also allows negative indexing for its sequences. The index of -1 refers to the last item, -2 to the second last item and so on.

For example, if a = [5, 3, 7] then a[0] = 5, a[1] = 3, a[-1] = 7, a[-2] = 3

There are many useful methods for working with a list in python :

- append() : Adds element to end of list
- extend() : Adds all element of another list to this list
- insert() : Adds element at specified index
- remove() : Removes item from list
- pop() : Returns and removes item from specified index
- clear() : Removes all elements from list
- index() : Returns index of first match with specified item
- count() : Returns count of number of items passed
- sort() : Sorts items in the list in ascending order
- reverse() : Reverses the order of items present in the list
- copy() : Returns a shallow copy of the items in the list
- len() : Returns number of elements (length) in the list

A list can also be sliced using : (colon) slicing operator. The syntax for slicing a list is :

lst_name[start:end:step]

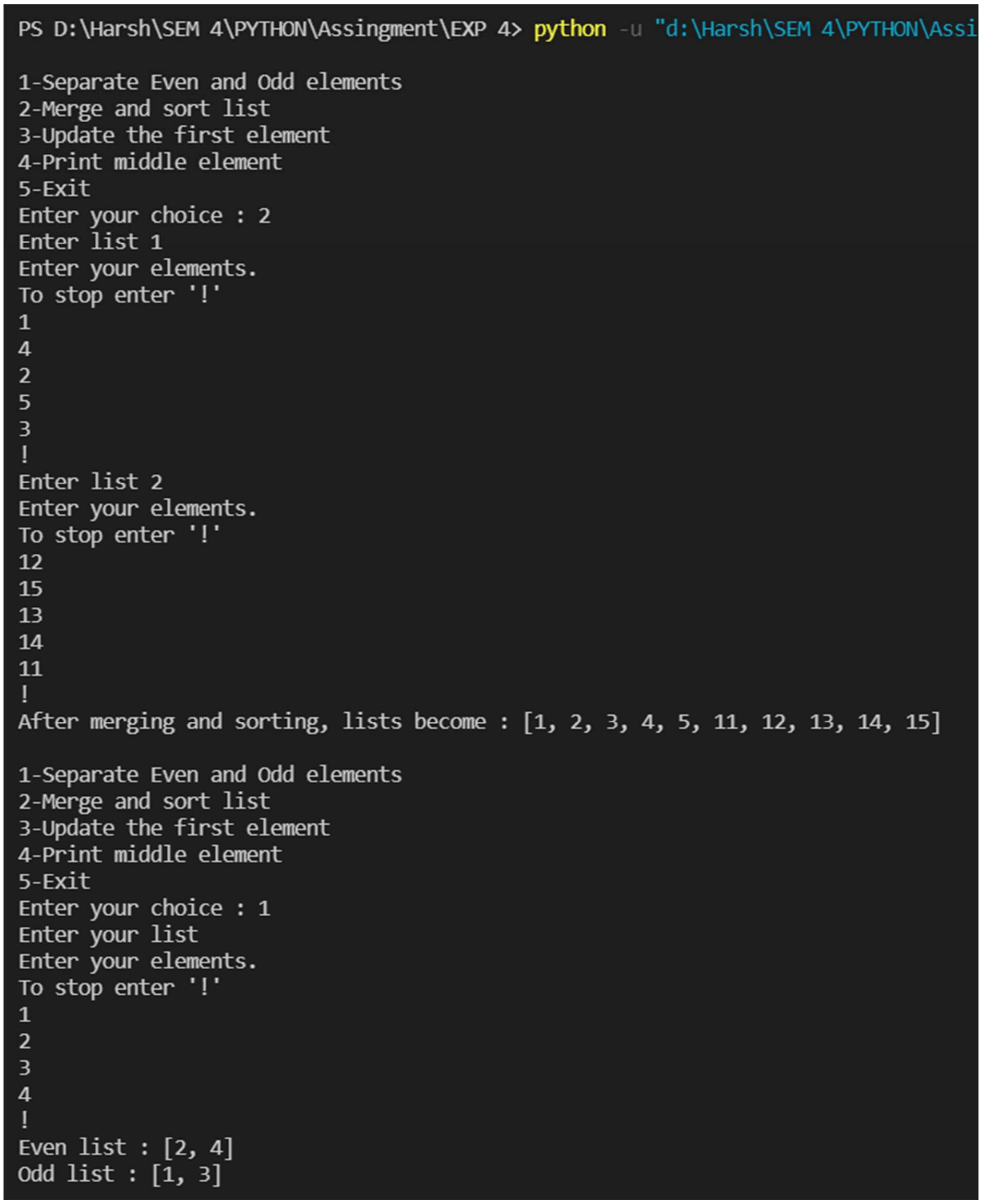
where start index is inclusive, end index is exclusive, and step tells us the amount of indices to skip while creating the slice. For example,

a[0:2] = [5, 3]
a[0:3:2] = [5, 7] 3 is skipped since step was specified as 2

CODE:

```
def accept():
    lst = []
    print("Enter your elements.\nTo stop enter '!'")
    while(True):
        n = input()
        if n == '!':
            break
        lst.append(int(n))
    return lst



lst = []
while(True):
    print("\n1-Separate Even and Odd elements\n2-Merge and sort list\n3-Update
the first element\n4-Print middle element\n5-Exit")
    ch = int(input("Enter your choice : "))
    if ch == 1:
        print("Enter your list")
        lst = accept()
        even_lst, odd_lst = [], []
        for n in lst:
            if n % 2 == 0:
                even_lst.append(n)
            else:
```

```
            odd_lst.append(n)

    print("Even list :", even_lst)
    print("Odd list :", odd_lst)
elif ch == 2:
    print("Enter list 1")
    lst1 = accept()
    print("Enter list 2")
    lst2 = accept()
    lst = lst1+lst2
    lst.sort()
    print("After merging and sorting, lists become :", lst)
elif ch == 3:
    print("Enter list")
    lst = accept()
    n = int(input("Enter element : "))
    lst[0] = n
    print("List with first element updated is :", lst)
elif ch == 4:
    print("Enter list")
    lst = accept()
    if len(lst) % 2 == 0:
        print("Middle elements of list are : ", lst[len(lst)//2-1],
              lst[len(lst)//2])
    else:
        print("Middle element of list is :", lst[len(lst)//2])
else:
    break
```



```
PS D:\Harsh\SEM 4\PYTHON\Assingment\EXP 4> python -u "d:\Harsh\SEM 4\PYTHON\Assi

1-Separate Even and Odd elements
2-Merge and sort list
3-Update the first element
4-Print middle element
5-Exit
Enter your choice : 2
Enter list 1
Enter your elements.
To stop enter '!'
1
4
2
5
3
!
Enter list 2
Enter your elements.
To stop enter '!'
12
15
13
14
11
!
After merging and sorting, lists become : [1, 2, 3, 4, 5, 11, 12, 13, 14, 15]

1-Separate Even and Odd elements
2-Merge and sort list
3-Update the first element
4-Print middle element
5-Exit
Enter your choice : 1
Enter your list
Enter your elements.
To stop enter '!'
1
2
3
4
!
Even list : [2, 4]
Odd list : [1, 3]
```

Q2. Write a menu driven program to demonstrate use of tuple in Python.

- Add and show details, i.e, roll no, name and marks of three subjects of N students in a list of tuple
- Display details of student whose name is X

A tuple in Python is similar to a list. The difference between the two is that we cannot change the elements of a tuple once it is assigned whereas we can change the elements of a list.

A tuple can be created just like a list, the only difference being that we use round brackets instead of square brackets. For example,

```
a = (5, 3, 7) # Tuple with 3 elements
b = () # Empty tuple
```

The round brackets are also optional, [We can just do a = 5, 3, 7] however, it is a good practice to use them. A tuple can have any number of items and they may be of different types (integer, float, list, string, etc.).

NOTE: To create a tuple with one element, we will need a trailing comma to indicate that it is a tuple and not an integer wrapped in brackets. For example,

```
a = (5,)
```

Without the comma, a would be assigned the integral value of 5.

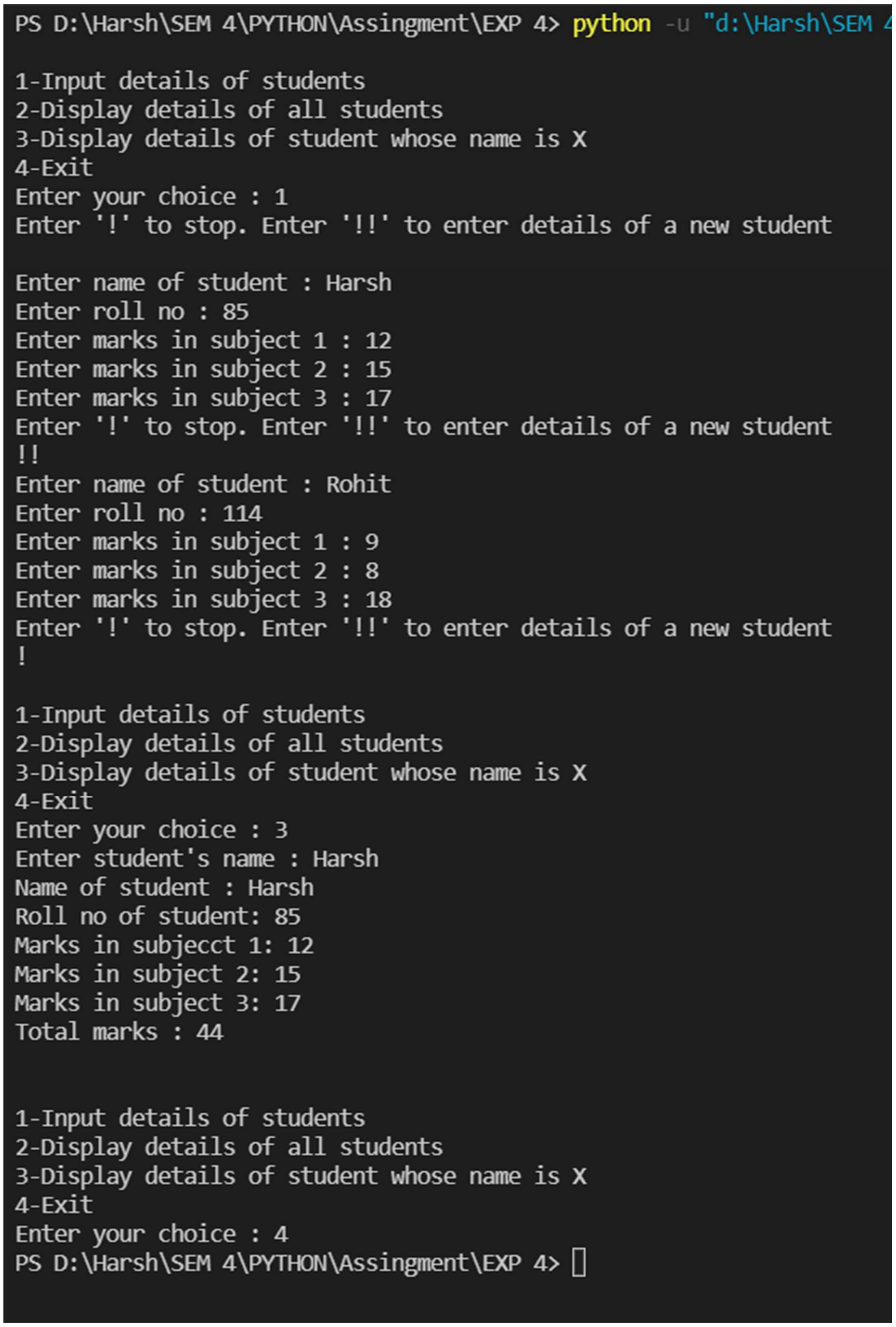
The indices for a tuple are the same as list. That is, 0 for the 1st element, 1 for 2nd etc. Just like a list, a tuple also allows for negative indexing and slicing.

Tuples also have the functions index(), count(), copy() and len().
However, since tuples are immutable, they do not have functions like sort(), reverse(), insert(), append(), etc

CODE:

```
def accept():
    lst = []
    while(True):
        print("Enter '!' to stop. Enter '!!' to enter details of a new student")
        n = input()
        if n == '!':
            break
        lst.append((input("Enter name of student : "),
                    input("Enter roll no : "),
                    int(input("Enter marks in subject 1 : ")),
                    int(input("Enter marks in subject 2 : ")),
                    int(input("Enter marks in subject 3 : "))))
    return lst


def display_student(student):
    print("Name of student :", student[0])
    print("Roll no of student:", student[1])
    print("Marks in subjecct 1:", student[2])
    print("Marks in subject 2:", student[3])
    print("Marks in subject 3:", student[4])
    print("Total marks :", student[2]+student[3]+student[4])
    print()
```

```
def display_all():
    global lst
    for student in lst:
        display_student(student)



def display(name):
    global lst
    for student in lst:
        if student[0] == name:
            display_student(student)



while(True):
    global lst
    print("\n1-Input details of students\n2-Display details of all students\n3-
Display details of student whose name is X\n4-Exit")
    ch = int(input("Enter your choice : "))
    if ch == 1:
        lst = accept()
    elif ch == 2:
        display_all()
    elif ch == 3:
        name = input("Enter student's name : ")
        display(name)
    else:
        break
```



```
PS D:\Harsh\SEM 4\PYTHON\Assingment\EXP 4> python -u "d:\Harsh\SEM 4

1-Input details of students
2-Display details of all students
3-Display details of student whose name is X
4-Exit
Enter your choice : 1
Enter '!' to stop. Enter '!!' to enter details of a new student

Enter name of student : Harsh
Enter roll no : 85
Enter marks in subject 1 : 12
Enter marks in subject 2 : 15
Enter marks in subject 3 : 17
Enter '!' to stop. Enter '!!' to enter details of a new student
!!
Enter name of student : Rohit
Enter roll no : 114
Enter marks in subject 1 : 9
Enter marks in subject 2 : 8
Enter marks in subject 3 : 18
Enter '!' to stop. Enter '!!' to enter details of a new student
!

1-Input details of students
2-Display details of all students
3-Display details of student whose name is X
4-Exit
Enter your choice : 3
Enter student's name : Harsh
Name of student : Harsh
Roll no of student: 85
Marks in subjecct 1: 12
Marks in subject 2: 15
Marks in subject 3: 17
Total marks : 44


1-Input details of students
2-Display details of all students
3-Display details of student whose name is X
4-Exit
Enter your choice : 4
PS D:\Harsh\SEM 4\PYTHON\Assingment\EXP 4> 
```

Q3. Write a menu driven program to demonstrate use of set in python

- Read two sets A and B from user and display set A and B
- Perform intersection A & B of the two sets
- Perform union A | B of the two sets
- Perform set difference A-B of the two sets
- Perform symmetric difference A^B of the two sets

A set is an unordered collection of items. Every set element is unique (no duplicates) and must be immutable (cannot be changed). However, a set itself is mutable. We can add or remove items from it. Sets can also be used to perform mathematical set operations like union, intersection, symmetric difference, etc.

In Python, to create a set, we can surround elements using curly brackets {} NOTE: Using empty curly brackets to create an empty set won't actually create one. It will create a dictionary. To create an empty set, use set() function without any arguments.

Sets also have useful functions that we can use:

- add() : Adds an element to the set
- clear() : Removes all elements from the set
- copy() : Returns a copy of the set
- difference() : Returns the difference of two or more sets as a new set
- difference_update() : Removes all elements of another set from this set
- discard() : Removes an element from the set if it is a member. (Do nothing if the element is not in set)
- intersection() : Returns the intersection of two sets as a new set
- intersection_update() : Updates the set with the intersection of itself and another
- isdisjoint() : Returns True if two sets have a null intersection
- issubset() : Returns True if another set contains this set
- issuperset() : Returns True if this set contains another set
- pop() : Removes and returns an arbitrary set element. Raises KeyError if the set is empty
- remove() : Removes an element from the set. If the element is not a member, raises a KeyError symmetric_difference() Returns the symmetric difference of

two sets as a new set symmetric_difference_update() Updates a set with the symmetric difference of itself and another union() Returns the union of sets in a new set update() Updates the set with the union of itself and others

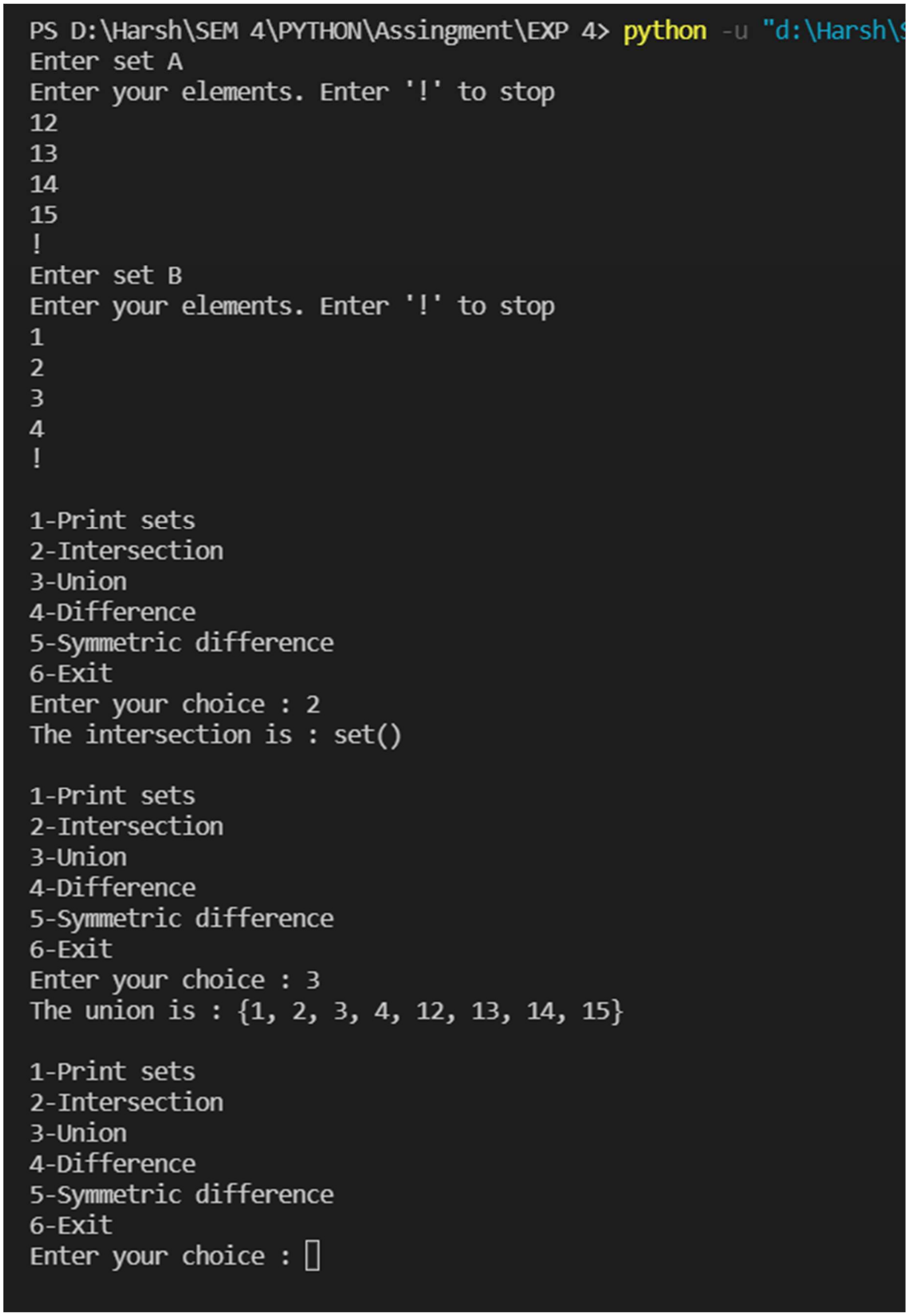
We can also use operators to perform operations on set. The operators we can use are:
& for intersection
| for union
- for difference
^ for symmetric difference

For example, if we have two sets A and B, then A & B will create a new set containing the intersection of sets A and B. The original sets A and B will remain unaltered, however. If we want to change them, we will have assigned them, for eg A = A & B to assign the intersection of A and B to A.

CODE:

```
def accept():
    st = set()
    print("Enter your elements. Enter '!' to stop")
    while(True):
        n = input()
        if n == '!':
            break
        st.add(int(n))
    return st



print("Enter set A")
A = accept()
```

```
print("Enter set B")

B = accept()

while(True):

   print("\n1-Print sets\n2-Intersection\n3-Union\n4-Difference\n5-Symmetric difference\n6-Exit")

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

   if(choice == 1):

      print("Set A: ", A)

      print("Set B: ", B)

   elif choice == 2:

      print("The intersection is :", A & B)

   elif choice == 3:

      print("The union is :", A | B)

   elif choice == 4:

      print("The difference is:", A-B)

   elif choice == 5:

      print("The symmetric difference is:", A ^ B)

   else:

      break
```



```
PS D:\Harsh\SEM 4\PYTHON\Assingment\EXP 4> python -u "d:\Harsh\
Enter set A
Enter your elements. Enter '!' to stop
12
13
14
15
!
Enter set B
Enter your elements. Enter '!' to stop
1
2
3
4
!

1-Print sets
2-Intersection
3-Union
4-Difference
5-Symmetric difference
6-Exit
Enter your choice : 2
The intersection is : set()

1-Print sets
2-Intersection
3-Union
4-Difference
5-Symmetric difference
6-Exit
Enter your choice : 3
The union is : {1, 2, 3, 4, 12, 13, 14, 15}

1-Print sets
2-Intersection
3-Union
4-Difference
5-Symmetric difference
6-Exit
Enter your choice : 
```

Q4. Write a program to demonstrate use of dictionary in Python:

- Read a dictionary from the user and display
- To sort a dictionary by key
- Concatenate two Python dictionaries into a new one

Python dictionary is an unordered collection of items. Each item of a dictionary has a key/value pair. Dictionaries are optimized to retrieve values when the key is known.

Creating a dictionary is as simple as placing items inside curly braces {} separated by commas. An item has a key and a corresponding value that is expressed as a pair (key: value). While the values can be of any data type and can repeat, keys must be of immutable type (string, number or tuple with immutable elements) and must be unique.

While indexing is used with other data types to access values, a dictionary uses keys. Keys can be used either inside square brackets [] or with the get() method. If we use the square brackets [], KeyError is raised in case a key is not found in the dictionary. On the other hand, the get() method returns None if the key is not found.

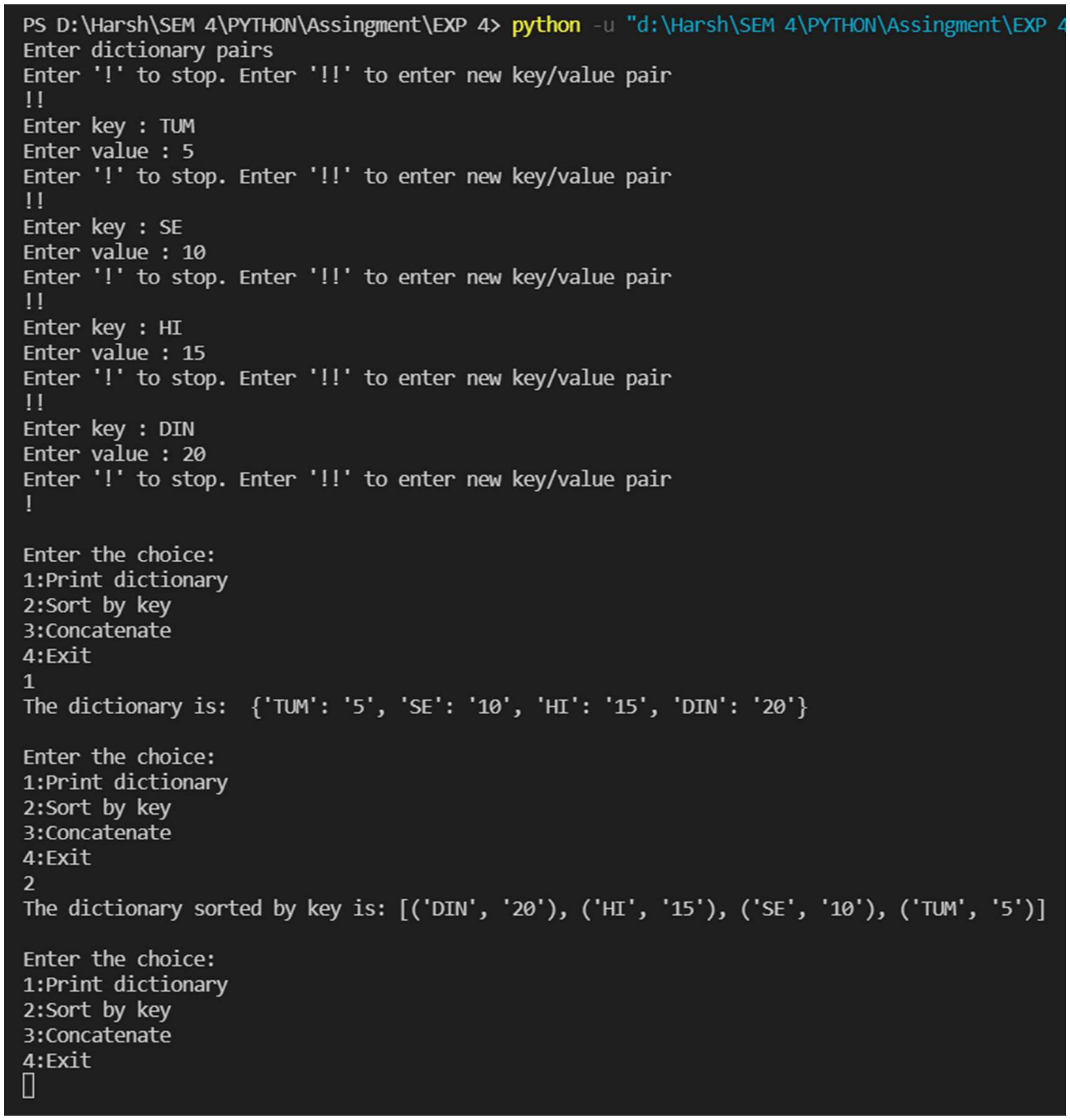
To add a key value pair to the dictionary, we can just do
dct[key] = value
If the key is already present, then the existing value gets updated. In case the key is not present, a new (key: value) pair is added to the dictionary.

We can remove a particular item in a dictionary by using the pop() method. This method removes an item with the provided key and returns the value. The popitem() method can be used to remove and return an arbitrary (key, value) item pair from the dictionary. All the items can be removed at once, using the clear() method. We can also use the del keyword to remove individual items or the entire dictionary itself.

To concatenate two dictionaries, we can use the .update() method on one of the dictionaries, which takes a second dictionary as input and updates the key/value pairs in the current one. If any key is present in the second dictionary that already exists in the first one, then that key is overwritten. Otherwise, that new key value pair is added to the first one.

CODE:

```
def accept():
    dct = {}
    while(True):
        print("Enter '!' to stop. Enter '!!' to enter new key/value pair")
        n = input()
        if n == '!':
            break
        key = input("Enter key : ")
        value = input("Enter value : ")
        dct[key] = value
    return dct



print("Enter dictionary pairs")
dct = accept()
while(True):
    print("\nEnter the choice: \n1:Print dictionary \n2:Sort by key 
\n3:Concatenate\n4:Exit")
    choice = int(input())
    if choice == 1:
        print("The dictionary is: ",  dct)
    elif choice == 2:
        print("The dictionary sorted by key is:", sorted(dct.items()))
    elif choice == 3:
        print("Enter second dictionary")
        dct2 = accept()
        dct_conc = dct.copy()
        dct_conc.update(dct2)
        print("Concatenated dictionary is:", dct_conc)
    else:
        break
```



```
PS D:\Harsh\SEM 4\PYTHON\Assingment\EXP 4> python -u "d:\Harsh\SEM 4\PYTHON\Assingment\EXP 4
Enter dictionary pairs
Enter '!' to stop. Enter '!!' to enter new key/value pair
!!
Enter key : TUM
Enter value : 5
Enter '!' to stop. Enter '!!' to enter new key/value pair
!!
Enter key : SE
Enter value : 10
Enter '!' to stop. Enter '!!' to enter new key/value pair
!!
Enter key : HI
Enter value : 15
Enter '!' to stop. Enter '!!' to enter new key/value pair
!!
Enter key : DIN
Enter value : 20
Enter '!' to stop. Enter '!!' to enter new key/value pair
!

Enter the choice:
1:Print dictionary
2:Sort by key
3:Concatenate
4:Exit
1
The dictionary is:  {'TUM': '5', 'SE': '10', 'HI': '15', 'DIN': '20'}

Enter the choice:
1:Print dictionary
2:Sort by key
3:Concatenate
4:Exit
2
The dictionary sorted by key is: [('DIN', '20'), ('HI', '15'), ('SE', '10'), ('TUM', '5')]

Enter the choice:
1:Print dictionary
2:Sort by key
3:Concatenate
4:Exit
```