# Practical - 4

**Aim:**

Understand the lists and tuples data structures and operations.

**Q1:** Explain difference between insert, append and extend operations on list. Write a program to create and initialize a list with your name, enrollment number, age, branch and result. Perform insert, remove, update, append and extend operation on list.

**Code:**

```
student = ["Vatsal", "24012011142", 21, "CE", 9.11]
print(student)
student.insert(1,"Patel")
print(student)
student.remove(21)
print(student)
student[4] = 8.49
print(student)
```

**Output:**

```
['Vatsal', '24012011142', 21, 'CE', 9.11]
['Vatsal', 'Patel', '24012011142', 21, 'CE', 9.11]
['Vatsal', 'Patel', '24012011142', 'CE', 9.11]
['Vatsal', 'Patel', '24012011142', 'CE', 8.49]
```

**Q2:** Write a program to search an element, find maximum & minimum value from the list.

- Using inbuilt function
- Using for loop

**Code:**

```
l = [1,6,8,3,8,2,9,2,6]
minimum = min(l)
maximun = max(l)

print(f"""
Using In-Built Methods:
Minimum: {minimum}
Maximum: {maximun}
""")
```

```
l = [1,6,8,3,8,2,9,2,6]
minimum = float('inf')
maximun = float('-inf')

for i in l:
  if i > maximun:
    maximun = i
  if i < minimum:
    minimum = i

print(f"""
Using Loop:
Minimum: {minimum}
Maximum: {maximun}
""")
```

**Output:**

```
Using In-Built Methods:
Minimum: 1
Maximum: 9

Using Loop:
Minimum: 1
Maximum: 9
```

**Q3:** Create a program that asks the user for a number and then prints out a list of all the divisors of that number.

**Code:**

```
num = int(input("Enter Number You want Divisor Of: "))
divisor_list = []
for i in range(1,num + 1):
  if num % i == 0:
    divisor_list.append(i)


print(divisor_list)
```

**Output:**

```
Enter Number You want Divisor Of: 15
[1, 3, 5, 15]
```

**Q4:** Write a program to sort element in list

- In same list
- Create sorted copy of original list & print both.
- Sort without any built-in function

**Code:**

```
# in same List
l = [1,6,8,3,8,2,9,2,6]
print("1. in same List")
l.sort()
print("list :", l)

# Make Sorted Copy
l = [1,6,8,3,8,2,9,2,6]
print("2. Sorted Copy")
print("list :", l)
l1 = sorted(l)
print("list1 :", l1)

# without Any Builtin Function
for i in range(len(l)):
  for j in range(len(l)):
    if l[j] > l[i]:
      temp = l[i]
      l[i] = l[j]
      l[j] =  temp

print("3. without Any Builtin Function")
print("list :", l)
```

**Output:**

```
1. in same List
list : [1, 2, 2, 3, 6, 6, 8, 8, 9]
2. Sorted Copy
list : [1, 6, 8, 3, 8, 2, 9, 2, 6]
```

```
list1 : [1, 2, 2, 3, 6, 6, 8, 8, 9]
3. without Any Builtin Function
list : [1, 2, 2, 3, 6, 6, 8, 8, 9]
```

**Q5:** Take two lists, say for example these two:

- a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
- b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]

and write a program that returns a list that contains only the elements that are common between the lists (without duplicates). Make sure your program works on two lists of different sizes.

**Code:**

```
a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
common_elements = set()

for i in a:
  if i in b:
    common_elements.add(i)

print(f"""
Common Elements: {common_elements}
""")
```

**Output:**

```
Common Elements: {1, 2, 3, 5, 8, 13}
```

**Q6:** Write a Python program which takes a list and returns a list with the elements "Shifted left by one position" so [1, 2, 3] yields [2, 3, 1]. Example: [11, 12, 13] → [12, 13, 11]

**Code:**

```
l = [1,2,3,4,5]
l = l[1:] + l[:1]
print("Method1:",l)

# Another Method
l = [1,2,3,4,5]
temp = l.pop(0)
l.append(temp)
print("Method2:",l)
```

**Output:**

```
Method1: [2, 3, 4, 5, 1]
Method2: [2, 3, 4, 5, 1]
```

**Q7:** Write a program which takes a comma separated string from user & store each string which separated by comma in list & display list.

**Code:**

```
# String: Hii,My,Name,Is,Vatsal
st = input("Enter String: ")
l = st.split(",")
print(l)
```

**Output:**

```
Enter String: Hii,My,Name,Is,Vatsal
['Hii', 'My', 'Name', 'Is', 'Vatsal']
```

**Q8:** Write a program to create and initialize the tuple. Also remove 3rd element from tuple.

**Code:**

```
user = ("Vatsal", "Patel", 21, "Himmatnagar")
user1 = list(user)
user1.pop(2)
user1 = tuple(user1)
print(f"""
Original Tuple: {user}
Modified Tuple: {user1}
""")
```

**Output:**

```
Original Tuple: ('Vatsal', 'Patel', 21, 'Himmatnagar')
Modified Tuple: ('Vatsal', 'Patel', 'Himmatnagar')
```

**Q9:** Create a tuple with name courses and initialize it with JAVA, PHP, C#, Android. Insert two items HTML and Python at the 3rd position in tuple.

**Code:**

```
l = ('JAVA', 'PHP', 'C#', 'Android')
l = list(l)
l[3:3] = ['HTML', 'Python']
l = tuple(l)
print(l)
```

**Output:**

```
('JAVA', 'PHP', 'C#', 'HTML', 'Python', 'Android')
```

**Q10:** Write a python program which shows the effect of mutability.

**Code:**

```
my_list = [1, 2, 3]
original_id = id(my_list)
print(f"Original List: {my_list} | ID: {original_id}")

# Modifying the list in place
my_list.append(4)
new_id = id(my_list)
print(f"Modified List: {my_list} | ID: {new_id}")
print(f"IDs are the same: {original_id == new_id} (Same object in
memory)")
```

**Output:**

```
Original List: [1, 2, 3] | ID: 138533525644672

Modified List: [1, 2, 3, 4] | ID: 138533525644672

IDs are the same: True (Same object in memory)
```

**Q11:** Write a program to create a regular expression which verifies whether given mobile number is valid or not.

**Code:**

```
import re
phoneNumberRegex = re.compile(r"^[6-9]\d{9}$")

phoneNumber = "9409361272"

if phoneNumberRegex.search(phoneNumber):
  print("Valid Number")
else:
  print("Invalid Number")
```

**Output:**

```
Valid Number
```

## Practice Exercise:

**1.** Write a program that takes a list of student scores [85, 92, 78, 65, 90, 76, 88, 79] and performs the following operations:

- Create a new list with only scores above 80
- Find average score using for loop (without built-in functions)
- Sort scores in descending order without using built-in functions

Display all results.

**Code:**

```
l = [85, 92, 78, 65, 90, 76, 88, 79]
score_above_80 = []

for i in l:
  if i > 80:
    score_above_80.append(i)

sum = 0
n = 0
for i in l:
  sum += i
  n += 1
avarage = sum / n

for i in range(n):
  for j in range(n):
    if l[i] > l[j]:
      temp = l[i]
      l[i] = l[j]
      l[j] = temp

print(f"""
Score Above 80: {score_above_80}
Avarge: {avarage}
Sorted List: {l}
""")
```

**Output:**

```
Score Above 80: [85, 92, 90, 88]

Avarge: 81.625

Sorted List: [92, 90, 88, 85, 79, 78, 76, 65]
```

**2.** Create a program that maintains two lists:
- products = ['Laptop', 'Mouse', 'Keyboard', 'Monitor']
- prices = [45000, 890, 920, 15600]
- Perform following operations:
- Insert new product with price at position 2
- Remove product 'Mouse' and its corresponding price
- Update price of 'Monitor'
- Display products with prices > 1000

**Code:**

```
products = ['Laptop', 'Mouse', 'Keyboard', 'Monitor']
prices = [45000, 890, 920, 15600]

# Task 1
products.insert(2, "Printer")
prices.insert(2, 5200)
print(f"""
Lists After Task 1:
Products: {products}
Prices: {prices}
""")

# Task 2
idx_mouse = products.index("Mouse")
products.pop(idx_mouse)
prices.pop(idx_mouse)

print(f"""
Lists After Task 2:
Products: {products}
Prices: {prices}
""")



# Task 3
prices[products.index("Monitor")] = 20600
```

```
print(f"""
Lists After Task 3:
Products: {products}
Prices: {prices}
""")

# Task 4
prices_above_1000 = []
products_above_1000 = []
for i in range(len(prices)):
  if prices[i] > 1000:
    prices_above_1000.append(prices[i])
    products_above_1000.append(products[i])

print(f"""
Lists After Task 4:
Products Above 1000: {products_above_1000}
Prices Above 1000: {prices_above_1000}
""")
```

**Output:**

```
Lists After Task 1:
Products: ['Laptop', 'Mouse', 'Printer', 'Keyboard', 'Monitor']
Prices: [45000, 890, 5200, 920, 15600]


Lists After Task 2:
Products: ['Laptop', 'Printer', 'Keyboard', 'Monitor']
Prices: [45000, 5200, 920, 15600]


Lists After Task 3:
Products: ['Laptop', 'Printer', 'Keyboard', 'Monitor']
Prices: [45000, 5200, 920, 20600]


Lists After Task 4:
Products Above 1000: ['Laptop', 'Printer', 'Monitor']
Prices Above 1000: [45000, 5200, 20600]
```

**3.** Write a program that takes a tuple of subject marks (Python, Java, SQL, HTML) and performs:

- Convert tuple to list
- Add two new subjects with marks
- Remove lowest mark subject
- Convert back to tuple

Display original and final tuple

**Code:**

```
sub = ('Python', 'Java', 'SQL', 'HTML')
marks = (100, 90, 95, 98)

# Convert Into List
sub = list(sub)
marks = list(marks)

# Add 2 Subject with marks
sub.append("C")
marks.append(94)
sub.append("OS")
marks.append(92)

# Remove Lowest Marks
sub.pop(marks.index(min(marks)))
marks.pop(marks.index(min(marks)))

# Convert Back in Tuple

sub = tuple(sub)
marks = tuple(marks)

print(f"""
Subjects: {sub}
Marks: {marks}
""")
```

**Output:**

```
Subjects: ('Python', 'SQL', 'HTML', 'C', 'OS')

Marks: (100, 95, 98, 94, 92)
```

**4.** Create a program that takes a list of words and generates:

- A list containing length of each word
- A list of words with odd number of characters
- A list of words starting with vowels
- All operations should be done without built-in functions (except len())

**Code:**

```
# st = input("Enter List with Comma Separated: ")
st = 'Hii,My,Name,is,Vatsal,And,I,am,doing,Computer,Engineering'
st = st.split(",")
length = []
odd_words = []
words_starting_with_vowels = []
vowels = ['a', 'e', 'i', 'o', 'u']

for i in st:
  length.append(len(i))
  if len(i) % 2 != 0:
    odd_words.append(i)
  if i[0] in vowels:
    words_starting_with_vowels.append(i)

print(f"""
Length: {length}
Odd Len Words: {odd_words}
Word Starting With Vowels: {words_starting_with_vowels}
""")
```

**Output:**

```
Length: [3, 2, 4, 2, 6, 3, 1, 2, 5, 8, 11]

Odd Len Words: ['Hii', 'And', 'I', 'doing', 'Engineering']

Word Starting With Vowels: ['is', 'am']
```

**5.** Write a program that works with two lists: cities = ['Ahmedabad', 'Mumbai', 'Delhi', 'Bangalore'] codes = ['AMD', 'BOM', 'DEL', 'BLR']

- Create a comma-separated string from cities list
- Split it back into a new list
- Compare original and new list
- Create a tuple combining city and its code

Display all intermediate results

**Code:**

```python
cities = ['Ahmedabad', 'Mumbai', 'Delhi', 'Bangalore']
codes = ['AMD', 'BOM', 'DEL', 'BLR']

comma_sparated_string_cities = ",".join(cities)
comma_sparated_string_codes = ",".join(codes)

print(f"""
String Cities: {comma_sparated_string_cities}
String Codes: {comma_sparated_string_codes}
""")

new_cities = comma_sparated_string_cities.split(",")
new_codes = comma_sparated_string_codes.split(",")

print(f"""
Orignial Cities List: {cities}
Modified Cities List: {new_cities}
Orignial Codes List: {codes}
Modified Codes List: {new_codes}
""")

tuple_list_cities = []

for i in range(len(codes)):
  tuple_list_cities.append((cities[i], codes[i]))

tuple_list_cities = tuple(tuple_list_cities)

print(f"Tuple Of Cities And Codes: {tuple_list_cities}")
```

**Output:**

```
String Cities: Ahmedabad,Mumbai,Delhi,Bangalore
String Codes: AMD,BOM,DEL,BLR


Orignial Cities List: ['Ahmedabad', 'Mumbai', 'Delhi', 'Bangalore']
Modified Cities List: ['Ahmedabad', 'Mumbai', 'Delhi', 'Bangalore']
```

```
Orignial Codes List: ['AMD', 'BOM', 'DEL', 'BLR']
Modified Codes List: ['AMD', 'BOM', 'DEL', 'BLR']

Tuple Of Cities And Codes: (('Ahmedabad', 'AMD'), ('Mumbai', 'BOM'),
('Delhi', 'DEL'), ('Bangalore', 'BLR'))
```