**Lab Exercise 1- Working with Python collections**

**1. namedtuple**

* namedtuple is a factory function for creating tuple subclasses with named fields.

**Task**: Create a namedtuple called Employee with fields name, age, and department. Then, create a list of 5 employees and print their details in a formatted way.

from collections import namedtuple

# Define namedtuple Employee

Employee = namedtuple('Employee', ['name', 'age', 'department'])

# Create a list of employees

employees = [

Employee('John Doe', 28, 'HR'),

Employee('Jane Smith', 34, 'Finance'),

Employee('Mike Johnson', 40, 'IT'),

Employee('Emily Davis', 25, 'Marketing'),

Employee('Steve Brown', 30, 'Sales')

]

# Print employee details

for emp in employees:

print(f'Name: {emp.name}, Age: {emp.age}, Department: {emp.department}')

**2. defaultdict**

* defaultdict is a dictionary that provides default values for nonexistent keys.

**Task**: Create a defaultdict where the default value for nonexistent keys is an empty list. Add some key-value pairs representing students and their list of grades. Update grades and print the student data.

from collections import defaultdict

# Define defaultdict with list as default factory

student\_grades = defaultdict(list)

# Add grades to students

student\_grades['Alice'].extend([88, 92])

student\_grades['Bob'].extend([75, 85])

student\_grades['Charlie'].append(90)

# Print student grades

for student, grades in student\_grades.items():

print(f'{student}: {grades}')

**3. OrderedDict**

* OrderedDict remembers the insertion order of items.

**Task**: Create an OrderedDict to store product names and their prices. Then, print the products in the order they were added.

from collections import OrderedDict

# Define OrderedDict for products

products = OrderedDict()

# Add products and their prices

products['Laptop'] = 1000

products['Mouse'] = 25

products['Keyboard'] = 50

products['Monitor'] = 200

# Print products in insertion order

for product, price in products.items():

print(f'{product}: ${price}')

**4. deque**

* deque (double-ended queue) allows fast appends and pops from both ends of the list.

**Task**: Create a deque of numbers. Perform append, appendleft, pop, and popleft operations. Then, print the deque after each operation.

from collections import deque

# Create a deque of numbers

numbers = deque([1, 2, 3, 4, 5])

# Perform operations

numbers.append(6)

print('After append:', numbers)

numbers.appendleft(0)

print('After appendleft:', numbers)

numbers.pop()

print('After pop:', numbers)

numbers.popleft()

print('After popleft:', numbers)