ans 1

In [ ]:

```python
fruits = ["apple", "banana", "orange", "pineapple", "strawberry"]
n = len(fruits)
print(fruits[0])
print(fruits[n-1])

print(fruits[1:n-1]) #slicing
```

```
apple
strawberry
['banana', 'orange', 'pineapple']
```

ans 2

In [ ]:

```python
students = {
    "Divij" : 21,
    "Avinash" : 20,
    "Supra" : 20,
    "David": 23,
    "Eve": 19
}

student_name = "Supra"
print(f"The age of {student_name} is : {students[student_name]}") #f is for

students["Aniket"] = 16

print(f"Updated Dict: {students}")
```

```
The age of Supra is : 20
Updated Dict: {'Divij': 21, 'Avinash': 20, 'Supra': 20, 'David': 23, 'Eve':
19, 'Aniket': 16}
```

Ans 3

In [ ]:

```python
def duplicate(numbers):
  seen = set();
  duplicates = set()
  for number in numbers:
    if(number in seen):
      duplicates.add(number)
    else:
      seen.add(number)
  return list(duplicates)

my_list = [1,2,3,4,5,6,3,4,5,2]
print(f"Original List: {my_list}")
print(f"Duplicates: {duplicate(my_list)}")
```

```
Original List: [1, 2, 3, 4, 5, 6, 3, 4, 5, 2]
Duplicates: [2, 3, 4, 5]
```

Ans 4

In [1]:
```python
def group(input, size):
    # to split a list into smaller lists of given size
    return [input[i:i+size] for i in range(0, len(input), size)]

my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9]
size = 3
print(f"Original List: {my_list}")
print(f"Grouped List: {group(my_list, size)}")
```

```
Original List: [1, 2, 3, 4, 5, 6, 7, 8, 9]
Grouped List: [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

Ans 5

In [2]:
```python
def lensort(string_list):
  return sorted(string_list, key = len)

def extsort(file_list):
  return sorted(file_list, key=lambda file: file.split('.')[-1])

strings = ["apple", "banana", "kiwi", "orange", "grape"]
print(f"Original List: {strings}")
print(f"Sorted List: {lensort(strings)}")

files = ["a.txt", "b.py", "c.cpp", "d.java", "e.txt"]
print(f"\nOriginal List: {files}")
print(f"Sorted List: {extsort(files)}")
```

```
Original List: ['apple', 'banana', 'kiwi', 'orange', 'grape']
Sorted List: ['kiwi', 'apple', 'grape', 'banana', 'orange']

Original List: ['a.txt', 'b.py', 'c.cpp', 'd.java', 'e.txt']
Sorted List: ['c.cpp', 'd.java', 'b.py', 'a.txt', 'e.txt']
```

Ans 6

In [3]:
```python
#creating and writing on file
with open("sample.txt", "w") as f:
  f.write("This is the first line. \n")
  f.writelines(["This is the second line.\n", "This is the third line.\n"])

#Read the entire file
with open("sample.txt", "r") as f:
  content = f.read()
  print("--- Reading the File ---")
  print(content)
```

```python
#read the file line by line
with open("sample.txt", "r") as f:
  print("--- Reading the File Line by Line ---")
  line1 = f.readline()
  print(f"Line 1: {line1.strip()}")
  line2 = f.readline()
  print(f"Line 2: {line2.strip()}")
```

```
--- Reading the File ---
This is the first line.
This is the second line.
This is the third line.

--- Reading the File Line by Line ---
Line 1: This is the first line.
Line 2: This is the second line.
```

Ans 7

In [5]:
```python
def file_stats(filename):
    try:
        with open(filename, "r") as f:
            lines = f.readlines()
            num_lines = len(lines)
            num_words = sum(len(line.split()) for line in lines)
            num_chars = sum(len(line) for line in lines)
            return num_lines, num_words, num_chars
    except FileNotFoundError:
        return 0, 0, 0

# Using sample.txt
lines, words, chars = file_stats("sample.txt")
print(f"Lines: {lines}, Words: {words}, Characters: {chars}")
```

```
Lines: 3, Words: 15, Characters: 74
```

Ans 8

In [9]:
```python
def reverse_file(filename):
    try:
        with open(filename, "r") as f:
            lines = f.readlines()
        for line in reversed(lines):
            print(line.strip())
    except FileNotFoundError:
        print(f"Error: File '{filename}' not found.")

reverse_file("sample.txt")
```

```
This is the third line.
This is the second line.
This is the first line.
```

Ans 9

```
In [13]: def reverse_each(filename):
           try:
             with open(filename, 'r') as f:
               for line in f:
                 print(line.strip()[::-1])
           except FileNotFoundError:
             print(f"Error: File '{filename}' not found.")

         reverse_each("sample.txt")
```

```
.enil tsrif eht si sihT
.enil dnoces eht si sihT
.enil driht eht si sihT
```

Ans 10

```
In [14]: import sys
         import textwrap

         def wrap_file(filename, width):
           try:
             with open(filename, 'r') as f:
               for line in f:
                 print(textwrap.fill(line, width=width))
           except FileNotFoundError:
             print(f"Error: File '{filename}' not found.")

         with open("long_line.txt", "w") as f:
           f.write("This is a very ling line of text that needs to be wrapped to fit

         wrap_file("long_line.txt", 20)
```

```
This is a very ling
line of text that
needs to be wrapped
to fit within a
certain width.
```

Ans 11

```
In [15]: def map_comprehension(func, iterable):
           return [func(item) for item in iterable]

         numbers = [1, 2, 3, 4, 5]
         squared_numbers = map_comprehension(lambda x: x**2, numbers)
         print(f"Original List: {numbers}")
         print(f"Squared List: {squared_numbers}")
```

```
Original List: [1, 2, 3, 4, 5]
Squared List: [1, 4, 9, 16, 25]
```

Ans 12

In [16]:
```python
def filter_comprehension(func, iterable):
  return [item for item in iterable if func(item)]

numbers = [1,2,3,4,5,6,7,8,9,10]
even_numbers = filter_comprehension(lambda x: x % 2 == 0, numbers)
print(f"Original List: {numbers}")
print(f"Even Numbers: {even_numbers}")
```

```
Original List: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Even Numbers: [2, 4, 6, 8, 10]
```

Ans 13

In [17]:
```python
def triplets(n):
  triplets = []
  for c in range(1, n):
    for a in range(1, c):
      b = c-a
      if a < b:
        triplets.append((a, b, c))
  return triplets

print(triplets(10))
```

```
[(1, 2, 3), (1, 3, 4), (1, 4, 5), (2, 3, 5), (1, 5, 6), (2, 4, 6), (1, 6,
7), (2, 5, 7), (3, 4, 7), (1, 7, 8), (2, 6, 8), (3, 5, 8), (1, 8, 9), (2, 7,
9), (3, 6, 9), (4, 5, 9)]
```

Ans 14

In [19]:
```python
import csv

def parse_csv(filename):
  data = []
  try:
    with open(filename, 'r', newline='') as f:
      reader = csv.reader(f)
      for row in reader:
        data.append(row)
  except FileNotFoundError:
    print(f"Error: File '{filename}' not found.")
  return data

with open("data.csv", "w", newline="") as f:
  writer = csv.writer(f)
  writer.writerow(["Name", "Age", "City"])
  writer.writerow(["Divij", 21, "Delhi"])
  writer.writerow(["Sputnik", 20, "Varanasi"])

parsed_data = parse_csv("data.csv")
print(f"Parsed Data:\n {parsed_data}")
```

```
Parsed Data:
 [['Name', 'Age', 'City'], ['Divij', '21', 'Delhi'], ['Sputnik', '20', 'Vara
nasi']]
```

MUTATE STRING

In [20]:
```
def mutate(word):
    letters = 'abcdefghijklmnopqrstuvwxyz'
    splits = [(word[:i], word[i:]) for i in range(len(word) + 1)]

    deletes = [L + R[1:] for L, R in splits if R]
    transposes = [L + R[1] + R[0] + R[2:] for L, R in splits if len(R)>1]
    replaces = [L + c + R[1:] for L, R in splits if R for c in letters]
    inserts = [L + c + R for L, R in splits for c in letters]

    return set(deletes + transposes + replaces + inserts)

mutations = mutate("hello")
print(f"Number of mutations for 'hello': {len(mutations)}")
```

```
Number of mutations for 'hello': 284
```

Ans 15

In [22]:
```
def nearly_equal(a,b):
  return a in mutate(b)

print(f"'Hello' and 'Hallo' are nearly : {nearly_equal('Hello', 'Hallo')}")
print(f"'python' and 'pythno' are nearly equal: {nearly_equal('python', 'pyt
print(f"'java' and 'jaxa' are nearly equal: {nearly_equal('java', 'jaxa')}")
print(f"'apple' and 'apply' are nearly equal: {nearly_equal('apple', 'apply'
```

```
'Hello' and 'Hallo' are nearly : True
'python' and 'pythno' are nearly equal: True
'java' and 'jaxa' are nearly equal: True
'apple' and 'apply' are nearly equal: True
```

Ans 16

In [24]:
```
from collections import Counter

def char_frequency(filename):
  try:
    with open(filename, 'r') as f:
      return Counter(f.read())
  except FileNotFoundError:
    return Counter()


with open("test.py", "w") as f:
  f.write("def my_func(x):\n  return x+1")

freq = char_frequency("test.py")
print(f"Character Frequencies: {freq}")
```

```
Character Frequencies: Counter({' ': 4, 'e': 2, 'f': 2, 'u': 2, 'n': 2, 'x':
2, 'r': 2, 'd': 1, 'm': 1, 'y': 1, '_': 1, 'c': 1, '(': 1, ')': 1, ':': 1,
'\n': 1, 't': 1, '+': 1, '1': 1})
```

Ans 17

In [25]:
```
from collections import defaultdict

def find_anagrams(words):
  """
  Finds and groups anagrams in a given list of words.
  """
  anagram_map = defaultdict(list)
  for word in words:
    sorted_word = "".join(sorted(word))
    anagram_map[sorted_word].append(word)

  return [group for group in anagram_map.values() if len(group) > 1]

# Example usage:
word_list = ['eat', 'ate', 'done', 'tea', 'soup', 'node', 'tan', 'nat']
anagram_groups = find_anagrams(word_list)
print(f"Anagrams in {word_list}:")
print(anagram_groups)
```

```
Anagrams in ['eat', 'ate', 'done', 'tea', 'soup', 'node', 'tan', 'nat']:
[['eat', 'ate', 'tea'], ['done', 'node'], ['tan', 'nat']]
```