

# Python Programming (1323203) - Summer 2024 Solution

Milav Dabgar

June 18, 2024

## Question 1(a) [3 marks]

Lists the Importance of flowchart and algorithm

### Solution

**Table 1.** Importance of Flowchart and Algorithm

Importance of Flowchart	Importance of Algorithm
Visual representation of program logic	Step-by-step procedure to solve a problem
Easier to debug and identify errors	Language-independent solution approach
Helps in understanding complex processes	Serves as a foundation for programming
Improves communication among team members	Defines logic before coding begins

### Mnemonic

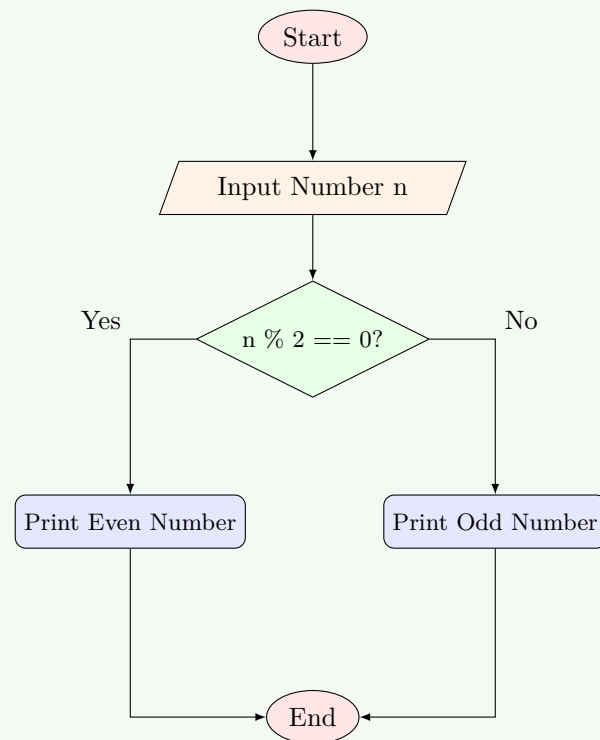
“VASE Decisions” - Visualize, Analyze, Sequence, Execute

## Question 1(b) [4 marks]

Draw a flowchart to find the entered number is even or odd.

### Solution

**Figure 1.** Flowchart to check Even or Odd number

**Key Steps:**

- **Input collection:** Get number from user
- **Modulo operation:** Divide by 2 and check remainder
- **Conditional output:** Display result based on remainder

**Mnemonic**

“MODE” - Modulo Operation Determines Evenness

**Question 1(c) [7 marks]**

List out all Logical operators and explain each by giving python code example.

**Solution****Table 2.** Logical Operators

Operator	Description	Example	Output
and	Returns True if both statements are true	x = 5; print(x > 3 and x < 10)	True
or	Returns True if one of the statements is true	x = 5; print(x > 10 or x == 5)	True
not	Reverse the result, returns False if result is true	x = 5; print(not(x > 3))	False

**Code Example:**

```

1 # Logical AND example
2 age = 25
3 income = 50000
4 print("Loan eligibility:", age > 18 and income > 30000) # True
5
6 # Logical OR example

```

```

7 has_credit_card = False
8 has_cash = True
9 print("Can purchase:", has_credit_card or has_cash) # True
10
11 # Logical NOT example
12 is_holiday = False
13 print("Should work today:", not is_holiday) # True

```

### Mnemonic

“AON Clarity” - And, Or, Not for logical clarity

OR

## Question 1(c) [7 marks]

Develop a Program that can calculate simple interest and compound interest on given data.

### Solution

```

1 # Program to calculate Simple and Compound Interest
2
3 # Input values
4 principal = float(input("Enter principal amount: "))
5 rate = float(input("Enter rate of interest (in %): "))
6 time = float(input("Enter time period (in years): "))
7
8 # Calculate Simple Interest
9 simple_interest = (principal * rate * time) / 100
10
11 # Calculate Compound Interest
12 compound_interest = principal * ((1 + rate/100) ** time - 1)
13
14 # Display results
15 print("Simple Interest:", round(simple_interest, 2))
16 print("Compound Interest:", round(compound_interest, 2))

```

### Key Formula

- **Simple Interest (SI):**  $\text{Principal} \times \text{Rate} \times \text{Time} / 100$
- **Compound Interest (CI):**  $\text{Principal} \times ((1 + \text{Rate}/100)^{\text{Time}} - 1)$

### Mnemonic

“PRT Money Grows” - Principal, Rate, Time make money grow

## Question 2(a) [3 marks]

Create a Program to find a minimum number among the given three numbers.

**Solution**

```

1  # Program to find minimum of three numbers
2
3  # Input three numbers
4  num1 = float(input("Enter first number: "))
5  num2 = float(input("Enter second number: "))
6  num3 = float(input("Enter third number: "))
7
8  # Find minimum using built-in min() function
9  minimum = min(num1, num2, num3)
10
11 # Display result
12 print("Minimum number is:", minimum)

```

**Mnemonic**

“MIN Finds Least” - Minimum Is Numerically Found with Least

**Question 2(b) [4 marks]**

Define pseudocode. Write pseudocode to find Largest of three numbers x, y and z.

**Solution**

**Table 3.** Pseudocode Definition

Definition
A detailed yet readable description of what a computer program must do, expressed in a formally-styled natural language rather than in a programming language.

**Pseudocode for finding largest of three numbers:**

```

1  BEGIN
2      INPUT x, y, z
3      SET largest = x
4
5      IF y > largest THEN
6          SET largest = y
7      END IF
8
9      IF z > largest THEN
10         SET largest = z
11     END IF
12
13     OUTPUT "Largest number is: ", largest
14 END

```

**Mnemonic**

“PIE Writing” - Program Ideas Expressed in simple writing

**Question 2(c) [7 marks]**

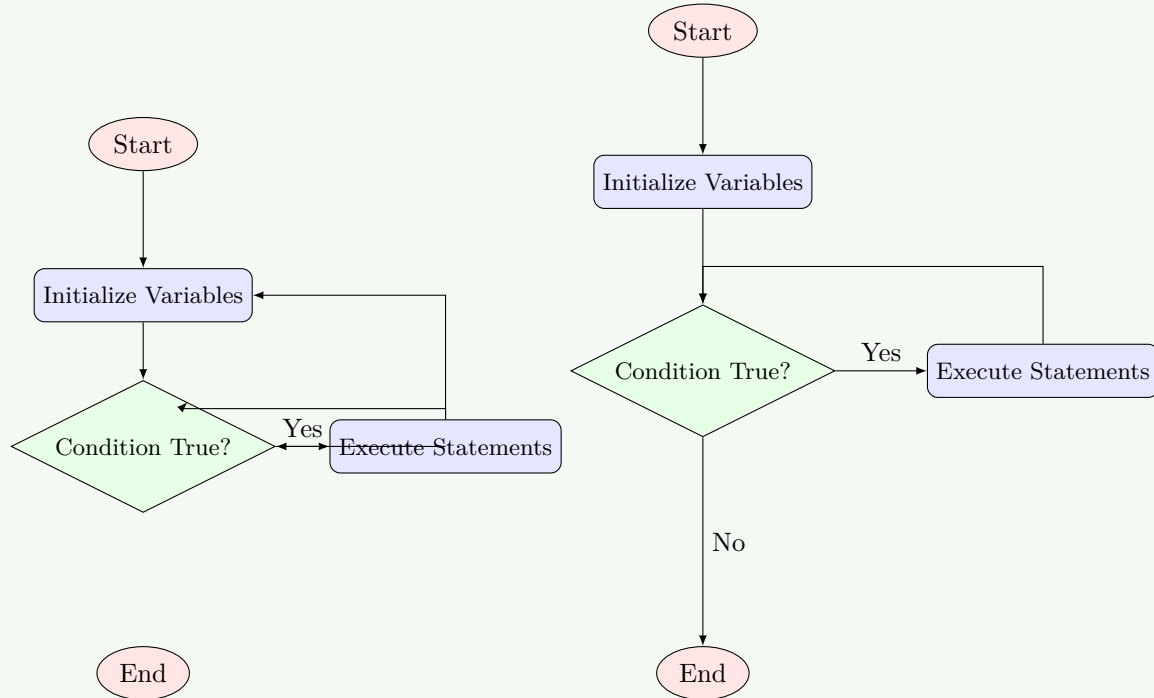
Explain While loop in python with its syntax, flowchart and with python code example.

## Solution

### Syntax:

```
1 while condition:
2     # code to be executed
```

**Figure 2.** Flowchart of While Loop



### Code Example:

```
1 # Print first 5 natural numbers using while loop
2 count = 1
3
4 while count <= 5:
5     print(count)
6     count += 1 # Increment counter
7
8 # Output:
9 # 1
10 # 2
11 # 3
12 # 4
13 # 5
```

### Key Characteristics:

- **Entry controlled:** Condition checked before loop execution
- **Initialization:** Variables set before the loop
- **Updation:** Variables updated inside the loop
- **Termination:** Loop exits when condition becomes False

### Mnemonic

“IUTE Loop” - Initialize, Update, Test for Exit

OR

## Question 2(a) [3 marks]

Describe continue statement in python in brief.

### Solution

**Table 4.** Continue Statement in Python

Description
The continue statement skips the current iteration of a loop and continues with the next iteration
When encountered, the code inside the loop following the continue statement is skipped
Useful for skipping specific conditions while keeping the loop running

### Code Example:

```

1 # Skip printing even numbers
2 for i in range(1, 6):
3     if i % 2 == 0:
4         continue
5     print(i) # Prints only 1, 3, 5

```

### Mnemonic

“SKIP Ahead” - Skip Keeping Iteration Process

OR

## Question 2(b) [4 marks]

What is the output of the following code:

### Solution

### Code:

```

1 x=8
2 y=2
3 print (x*y)
4 print (x ** y)
5 print (x % y)
6 print(x>y)

```

**Table 5.** Output Analysis

Operation	Result	Explanation
$x*y$	16	Multiplication: $8 \times 2 = 16$
$x**y$	64	Exponentiation: $8^2 = 64$
$x\%y$	0	Modulo (remainder): $8 \div 2 = 4$ with remainder 0
$x>y$	True	Comparison: $8 > 2$ is True

### Mnemonic

“MEMO” - Multiply, Exponent, Modulo, Operator comparison

OR

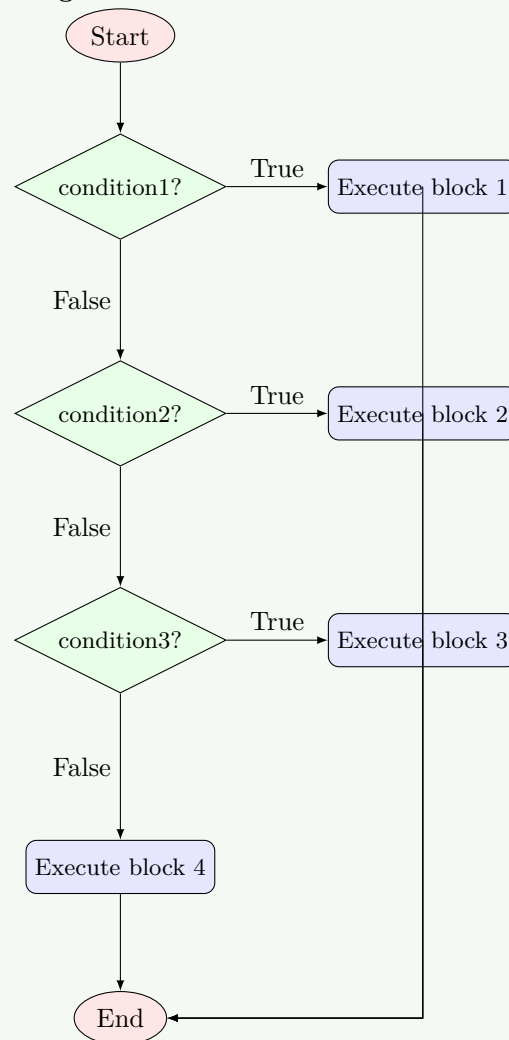
**Question 2(c) [7 marks]**

Explain if-elif-else Ladder in python with its syntax, flowchart and with python code example.

**Solution****Syntax:**

```
1 if condition1:
2     # code block 1
3 elif condition2:
4     # code block 2
5 elif condition3:
6     # code block 3
7 else:
8     # code block 4
```

**Figure 3.** Flowchart of If-Elif-Else Ladder

**Code Example:**

```

1 # Grade calculation based on marks
2 marks = 75
3
4 if marks >= 90:
5     grade = "A+"
6 elif marks >= 80:
7     grade = "A"
8 elif marks >= 70:
9     grade = "B"
10 elif marks >= 60:
11     grade = "C"
12 else:
13     grade = "D"
14
15 print("Grade:", grade) # Output: Grade: B

```

**Key Characteristics:**

- **Sequential evaluation:** Conditions checked from top to bottom
- **Exclusive execution:** Only one block executes
- **Default action:** Else block executes if no conditions are True

**Mnemonic**

“SEEP Logic” - Sequential Evaluation with Exclusive Path

**Question 3(a) [3 marks]**

Write a Python program to print odd numbers between 1 to 20 using loops.

**Solution**

```

1 # Program to print odd numbers between 1 to 20
2
3 # Using for loop with range and step
4 for number in range(1, 21, 2):
5     print(number, end=" ")
6
7 # Output: 1 3 5 7 9 11 13 15 17 19

```

**Alternate approach:**

```

1 # Using for loop with if condition
2 for number in range(1, 21):
3     if number % 2 != 0:
4         print(number, end=" ")

```

**Mnemonic**

“STEO” - Skip Two, Extract Odds

**Question 3(b) [4 marks]**

Explain Nested if statement in brief.



## Solution

Table 6. Nested if Statement

Description
An if statement inside another if statement
Allows for more complex conditional logic
Inner if only evaluated when outer if is True
Can have multiple levels of nesting

## Code Example:

```

1 age = 25
2 income = 50000
3
4 if age > 18:
5     print("Adult")
6     if income > 30000:
7         print("Eligible for credit card")
8     else:
9         print("Not eligible for credit card")
10 else:
11     print("Minor")

```

## Mnemonic

“LION” - Layered If-statements Operating Nested

## Question 3(c) [7 marks]

Using a user-defined function write a Program to check entered number is an 'Armstrong number' or a palindrome in which number is passed as argument in calling function.

## Solution

```

1 # Program to check Armstrong number or palindrome
2
3 def check_number(num):
4     # Check if Armstrong number
5     # An Armstrong number is one where sum of each digit raised to power of
6     # total digits equals the original number
7     temp = num
8     digits = len(str(num))
9     sum = 0
10
11     while temp > 0:
12         digit = temp % 10
13         sum += digit ** digits
14         temp //= 10
15
16     is_armstrong = (sum == num)
17
18     # Check if palindrome
19     # A palindrome reads the same backward as forward
20     is_palindrome = (str(num) == str(num)[::-1])
21
22     # Return results

```

```

23     return is_armstrong, is_palindrome
24
25 # Get input from user
26 number = int(input("Enter a number: "))
27
28 # Call function and display results
29 armstrong, palindrome = check_number(number)
30
31 if armstrong:
32     print(number, "is an Armstrong number")
33 else:
34     print(number, "is not an Armstrong number")
35
36 if palindrome:
37     print(number, "is a Palindrome")
38 else:
39     print(number, "is not a Palindrome")

```

**Examples:**

- **Armstrong:** 153 ( $1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153$ )
- **Palindrome:** 121 (Same forward and backward)

**Mnemonic**

“APTEST” - Armstrong Palindrome Test Equal Sum Test

OR

**Question 3(a) [3 marks]**

Write a python program to find sum of 1 to 100.

**Solution**

```

1  # Program to find sum of numbers from 1 to 100
2
3  # Method 1: Using loop
4  total = 0
5  for num in range(1, 101):
6      total += num
7  print("Sum using loop:", total)
8
9  # Method 2: Using formula n(n+1)/2
10 n = 100
11 sum_formula = n * (n + 1) // 2
12 print("Sum using formula:", sum_formula)
13
14 # Output:
15 # Sum using loop: 5050
16 # Sum using formula: 5050

```

**Mnemonic**

“SUM Formula” - Sum Using Mathematical Formula

OR

### Question 3(b) [4 marks]

Write a python program to print the following pattern.

#### Solution

##### Pattern:

```
1
2 3
4 5 6
7 8 9 10
```

##### Code:

```
1 # Program to print the number pattern
2
3 num = 1
4 for i in range(1, 5): # 4 rows
5     for j in range(i): # columns equal to row number
6         print(num, end=" ")
7         num += 1
8     print() # New line after each row
```

##### Pattern Logic:

- Row 1: 1 number (1)
- Row 2: 2 numbers (2, 3)
- Row 3: 3 numbers (4, 5, 6)
- Row 4: 4 numbers (7, 8, 9, 10)

#### Mnemonic

“CNIR” - Counter Number Increases with Rows

OR

### Question 3(c) [7 marks]

Write a Program using the function that reverses the entered value.

#### Solution

```
1 # Program to reverse entered value using functions
2
3 def reverse_number(num):
4     """Function to reverse an integer number"""
5     return int(str(num)[::-1])
6
7 def reverse_string(text):
8     """Function to reverse a string"""
9     return text[::-1]
10
11 # Main program
12 def main():
13     choice = input("What do you want to reverse? (n for number, s for string): ")
14
15     if choice.lower() == 'n':
16         num = int(input("Enter a number: "))
17         print("Reversed number:", reverse_number(num))
```

```

18 elif choice.lower() == 's':
19     text = input("Enter a string: ")
20     print("Reversed string:", reverse_string(text))
21 else:
22     print("Invalid choice!")
23
24 # Call the main function
25 main()

```

#### Alternate Method for Number Reversal:

```

1 def reverse_number_algorithm(num):
2     reversed_num = 0
3     while num > 0:
4         digit = num % 10
5         reversed_num = reversed_num * 10 + digit
6         num //= 10
7     return reversed_num

```

#### Mnemonic

“FLIP Digits” - Function Logic Inverts Position of Digits

## Question 4(a) [3 marks]

Describe python math module with proper python code example.

#### Solution

Table 7. Python Math Module

Features
Provides mathematical functions and constants
Includes trigonometric, logarithmic, and other functions
Contains mathematical constants like pi and e
Requires import before use

#### Code Example:

```

1 import math
2
3 # Constants
4 print("Value of pi:", math.pi) # 3.141592653589793
5 print("Value of e:", math.e) # 2.718281828459045
6
7 # Basic math functions
8 print("Square root of 16:", math.sqrt(16)) # 4.0
9 print("5 raised to power 3:", math.pow(5, 3)) # 125.0
10
11 # Trigonometric functions (radians)
12 print("Sine of 90°:", math.sin(math.pi/2)) # 1.0
13 print("Cosine of 0°:", math.cos(0)) # 1.0
14
15 # Logarithmic functions
16 print("Log base 10 of 100:", math.log10(100)) # 2.0
17 print("Natural log of e:", math.log(math.e)) # 1.0

```

**Mnemonic**

“CALM Operations” - Constants And Logarithmic Mathematical Operations

**Question 4(b) [4 marks]**

Write a python program that explains scope of variable.

**Solution**

```

1  # Program to demonstrate variable scope in Python
2
3  # Global variable
4  global_var = "I am global"
5
6  def demonstration():
7      # Local variable
8      local_var = "I am local"
9
10     # Accessing global variable
11     print("Inside function - Global variable:", global_var)
12
13     # Accessing local variable
14     print("Inside function - Local variable:", local_var)
15
16     # Creating a variable with same name as global
17     global_var = "I am local with global name"
18     print("Inside function - Shadowed global:", global_var)
19
20 # Function call
21 demonstration()
22
23 # Accessing global variable
24 print("Outside function - Global variable:", global_var)
25
26 # Trying to access local variable would cause error
27 # print("Outside function - Local variable:", local_var) # Error!

```

**Mnemonic**

“GLOVES” - Global Local Variable Encapsulation System

**Question 4(c) [7 marks]**

Explain List Methods and its built-in Functions

**Solution**

**Table 8.** List Methods and Functions

Method	Description	Example	Output
append()	Adds an element at the end	l=['a']; l.append('b')	['a', 'b']
insert()	Adds element at specified position	l=[1,3]; l.insert(1,2)	[1, 2, 3]
remove()	Removes specified item	l=['r','b']; l.remove('r')	['b']
pop()	Removes item at specified index	l=['a','b']; l.pop(1)	'b'
clear()	Removes all elements	l=[1,2]; l.clear()	[]
len()	Returns number of elements	len([1, 2, 3])	3
sorted()	Returns sorted list	sorted([3, 1, 2])	[1, 2, 3]
max()	Returns max value	max([5, 10, 3])	10

**Code Example:**

```

1 # Create a list
2 my_list = [3, 1, 4, 1, 5]
3 my_list.append(9)          # Add to end
4 my_list.insert(2, 7)       # Add at index 2
5 my_list.remove(1)          # Remove first occurrence of 1
6 popped = my_list.pop()     # Remove last element
7
8 print("Length:", len(my_list))
9 print("Sorted:", sorted(my_list))
10 print("Sum:", sum(my_list))
11 print("Count of 1:", my_list.count(1))

```

**Mnemonic**

“LISP Operations” - List Insert Sort Pop Operations

OR

**Question 4(a) [3 marks]**

List out Python standard library mathematical functions.

**Solution**

**Table 9.** Standard Mathematical Functions

Function	Description	Example
abs()	Returns absolute value	abs(-5) → 5
round()	Rounds to nearest integer	round(3.7) → 4
max()	Returns largest item	max(1, 5) → 5
min()	Returns smallest item	min(1, 5) → 1
sum()	Adds items of iterable	sum([1, 2]) → 3
pow()	Returns x to power y	pow(2, 3) → 8

**Additional from math module:**

- math.sqrt(): Square root
- math.floor(): Rounds down
- math.ceil(): Rounds up
- math.factorial(): Factorial of a number
- math.gcd(): Greatest common divisor

**Mnemonic**

“SMART Calculations” - Standard Mathematical Arithmetic Routines and Tools

OR

**Question 4(b) [4 marks]**

Explain built in function in python.

**Solution**

**Table 10.** Built-in Functions

Description
Pre-defined functions available in Python without importing any module
Called directly without any prefix
Designed to perform common operations
Examples include <code>print()</code> , <code>len()</code> , <code>type()</code> , <code>input()</code> , <code>range()</code>

**Categories with Examples:**

```

1 # Type conversion
2 print(int("10"))      # 10
3 print(str(10))        # "10"
4
5 # Math functions
6 print(abs(-7))        # 7
7 print(max(5, 10, 3))  # 10
8
9 # Collection processing
10 print(len("hello"))   # 5
11 print(sorted([3,1,2])) # [1, 2, 3]
```

**Mnemonic**

“EPIC Functions” - Embedded Python Integrated Core Functions

OR

**Question 4(c) [7 marks]**

Write a Python Program to count and display the number of vowels, consonants, uppercase, lowercase characters in a string.

**Solution**

```

1 # Program to count vowels, consonants, uppercase and lowercase characters
2
3 def analyze_string(text):
4     # Initialize counters
5     vowels = 0
6     consonants = 0
7     uppercase = 0
8     lowercase = 0
```

```

9
10 # Define vowels
11 vowel_set = {'a', 'e', 'i', 'o', 'u'}
12
13 # Analyze each character
14 for char in text:
15     # Check if alphabetic
16     if char.isalpha():
17         # Check case
18         if char.isupper():
19             uppercase += 1
20         else:
21             lowercase += 1
22
23     # Check if vowel (case-insensitive)
24     if char.lower() in vowel_set:
25         vowels += 1
26     else:
27         consonants += 1
28
29 # Return results
30 return vowels, consonants, uppercase, lowercase
31
32 # Get input
33 text = input("Enter a string: ")
34
35 # Get counts
36 vowels, consonants, uppercase, lowercase = analyze_string(text)
37
38 # Display results
39 print("Number of vowels:", vowels)
40 print("Number of consonants:", consonants)
41 print("Number of uppercase characters:", uppercase)
42 print("Number of lowercase characters:", lowercase)

```

### Mnemonic

“VOCAL Analysis” - Vowels Or Consonants And Letter case

## Question 5(a) [3 marks]

Write a python code to swap given two elements in a list.

### Solution

```

1 # Program to swap two elements in a list
2
3 def swap_elements(lst, pos1, pos2):
4     """Function to swap two elements in a list"""
5     lst[pos1], lst[pos2] = lst[pos2], lst[pos1]
6     return lst
7
8 # Example usage
9 my_list = [10, 20, 30, 40, 50]
10 print("Original list:", my_list)
11
12 # Swap elements at positions 1 and 3

```



```

13 result = swap_elements(my_list, 1, 3)
14 print("After swapping elements at positions 1 and 3:", result)
15
16 # Output:
17 # Original list: [10, 20, 30, 40, 50]
18 # After swapping elements at positions 1 and 3: [10, 40, 30, 20, 50]

```

### Mnemonic

“STEP Logic” - Swap Two Elements with Python Logic

## Question 5(b) [4 marks]

Write a python Program to check if a substring is present in a given string.

### Solution

```

1  # Program to check if a substring is present in a string
2
3  def check_substring(main_string, sub_string):
4      """Function to check if a substring exists in a string"""
5      if sub_string in main_string:
6          return True
7      else:
8          return False
9
10 # Get input from user
11 main_string = input("Enter the main string: ")
12 sub_string = input("Enter the substring to find: ")
13
14 # Check and display result
15 if check_substring(main_string, sub_string):
16     print(f"'{sub_string}' is present in '{main_string}'")
17 else:
18     print(f"'{sub_string}' is not present in '{main_string}'")

```

### Mnemonic

“FIND Method” - Find IN Directly with Methods

## Question 5(c) [7 marks]

Explain tuple Operations, Functions and Methods

### Solution

Table 11. Tuple Operations

Op/Func	Description	Example	Result
Creation	Create with parentheses	t=(1,2)	(1, 2)
Indexing	Access elements	t[1]	2
Slicing	Get subset	t[0:1]	(1,)
Concatenation	Join tuples	(1)+(2)	(1, 2)
Repetition	Repeat elements	(1)*2	(1, 1)
Membership	Check existence	1 in t	True
len()	Number of items	len(t)	2
count()	Count value	t.count(1)	1
index()	Find position	t.index(2)	1

**Code Example:**

```

1 my_tuple = (3, 1, 4, 1, 5, 9)
2 print("First:", my_tuple[0])
3 print("Slice:", my_tuple[1:4])
4 print("Count of 1:", my_tuple.count(1))
5 print("Index of 4:", my_tuple.index(4))
6 a, b, c, *rest = my_tuple # Unpacking

```

**Mnemonic**

“ICONS” - Immutable Collection Operations, Numbering, and Searching

OR

**Question 5(a) [3 marks]**

Write a python program find the sum of elements in a list.

**Solution**

```

1 # Program to find sum of elements in a list
2
3 def sum_of_list(numbers):
4     """Function to find sum of all elements in a list"""
5     total = 0
6     for num in numbers:
7         total += num
8     return total
9
10 # Example with user input
11 num_elements = int(input("Enter the number of elements: "))
12 my_list = []
13
14 # Get elements from user
15 for i in range(num_elements):
16     element = float(input(f"Enter element {i+1}: "))
17     my_list.append(element)
18
19 # Calculate sum using function
20 result1 = sum_of_list(my_list)
21 print("Sum using custom function:", result1)
22

```

```
23 # Calculate sum using built-in sum() function
24 result2 = sum(my_list)
25 print("Sum using built-in function:", result2)
```

### Mnemonic

“SALT” - Sum All List Together

OR

## Question 5(b) [4 marks]

Write a Program to demonstrate the set functions and operations.

### Solution

```
1 # Program to demonstrate set functions and operations
2
3 set1 = {1, 2, 3, 4, 5}
4 set2 = {4, 5, 6, 7, 8}
5
6 # Set operations
7 print("Union:", set1 | set2)
8 print("Intersection:", set1 & set2)
9 print("Difference:", set1 - set2)
10 print("Symmetric Difference:", set1 ^ set2)
11
12 # Set methods
13 set3 = set1.copy()
14 set3.add(6)
15 set3.remove(1)
16 set3.discard(10) # No error if not found
17 popped = set3.pop()
18 set3.clear()
```

### Mnemonic

“COSI Methods” - Create, Operate, Search, Investigate with Set Methods

OR

## Question 5(c) [7 marks]

Write a Program to demonstrate the dictionaries functions and operations.

### Solution

```
1 # Program to demonstrate dictionary functions and operations
2
3 # Creating a dictionary
4 student = {
5     'name': 'John',
6     'roll_no': 101,
```

```
7     'marks': 85
8 }
9
10 # Accessing elements
11 print("Name:", student['name'])
12 print("Roll No:", student.get('roll_no'))
13
14 # Modifying and Adding
15 student['marks'] = 90
16 student['address'] = 'New York'
17
18 # Removing items
19 removed = student.pop('address')
20 last_item = student.popitem()
21
22 # Dictionary methods
23 print("Keys:", list(student.keys()))
24 print("Values:", list(student.values()))
25 print("Items:", list(student.items()))
26
27 # Clearing
28 student.clear()
```

**Key Operations:**

- **Access:** Using key or get() method
- **Modify:** Assign new value to existing key
- **Add:** Assign value to new key
- **Remove:** Using pop(), popitem(), or del

**Mnemonic**

“ACME Dictionary” - Access, Create, Modify, Extract from Dictionary