**Index**

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No.** | **Topic** | **Date** | **Teacher's Signature** |
| 1 | Q1 |  |  |
| 2 | Q2 |  |  |
| 3 | Q3 |  |  |
| 4 | Q4 |  |  |
| 5 | Q5 |  |  |
| 6 | Q6 |  |  |
| 7 | Q7 |  |  |
| 8 | Q8 |  |  |
| 9 | Q9 |  |  |

**Question 1**

|  |
| --- |
| **Source Code** |

|  |
| --- |
| # Write a program to demonstrate the use of different operators in python.  def operators\_demo(a, b):  print("Addition:", a + b)  print("Subtraction:", a - b)  print("Multiplication:", a \* b)  print("Division:", a / b if b != 0 else "Undefined")  print("Modulus:", a % b if b != 0 else "Undefined")  print("Floor Division:", a // b if b != 0 else "Undefined")  print("Exponentiation:", a \*\* b)  print("Equal:", a == b)  print("Not Equal:", a != b)  print("Greater:", a > b)  print("Smaller:", a < b)  print("Logical AND:", a > 0 and b > 0)  print("Logical OR:", a > 0 or b > 0)  print("Logical NOT:", not(a > 0))  a = int(input("Enter first number: ")) b = int(input("Enter second number: ")) operators\_demo(a, b) |

**Question 2**

|  |
| --- |
| **Source Code** |

|  |
| --- |
| # Write a program to print Fibonacci Series 0 1 1 2 3 5 ………..N  n = int(input("Enter the number of terms: ")) a, b = 0, 1 print("Fibonacci Series:") for \_ in range(n):  print(a)  a, b = b, a + b |

**Question 3**

|  |
| --- |
| **Source Code** |

|  |
| --- |
| # Write a program to print the sum of first n prime numbers.  def is\_prime(num):  if num < 2:  return False  for i in range(2, int(num \*\* 0.5) + 1):  if num % i == 0:  return False  return True  n = int(input("Enter Number of Prime Numbers to Sum: ")) count, num, total = 0, 2, 0 while count < n:  if is\_prime(num):  total += num  count += 1  num += 1 print("Sum of first", n, "prime numbers is:", total) |

**Question 4**

|  |
| --- |
| **Source Code** |

|  |
| --- |
| # Create a function Pall\_n to print all of the palindrome numbers between two ranges.  def is\_palindrome(num):  return str(num) == str(num)[::-1]  low = int(input("Enter lower range: ")) high = int(input("Enter upper range: "))  print("Palindrome numbers are:", end=" ") for i in range(low, high + 1):  if is\_palindrome(i):  print(i, end=" ") |

**Question 5**

|  |
| --- |
| **Source Code** |

|  |
| --- |
| # Write a Python program to perform the string slicing.  s = input("Enter a string: ")  while True:  print("\n--- String Slicing Menu ---")  print("1. First 5 characters")  print("2. Last 5 characters")  print("3. Characters from index 2 to 7")  print("4. Every second character")  print("5. Reversed string")  print("6. Exit")   choice = int(input("Enter your choice: "))   if choice == 1:  print("First 5 characters:", s[:5])  elif choice == 2:  print("Last 5 characters:", s[-5:])  elif choice == 3:  print("Characters from index 2 to 7:", s[2:8])  elif choice == 4:  print("Every second character:", s[::2])  elif choice == 5:  print("Reversed string:", s[::-1])  elif choice == 6:  print("Exiting program...")  break  else:  print("Invalid choice! Try again.") |

**Question 6**

|  |
| --- |
| **Source Code** |

|  |
| --- |
| # Write a Python program to demonstrate the use of List, Tuple, Dictionary.  my\_list = [10, 20, 30, 40] my\_tuple = (1, 2, 3, 4) my\_dict = {"a": 100, "b": 200, "c": 300}  while True:  print("\n--- Data Structure Menu ---")  print("1. Show List")  print("2. Show Tuple")  print("3. Show Dictionary")  print("4. Exit")   choice = int(input("Enter your choice: "))   if choice == 1:  print("List:", my\_list)  print("Access List element (index 2):", my\_list[2])  elif choice == 2:  print("Tuple:", my\_tuple)  print("Access Tuple element (index 1):", my\_tuple[1])  elif choice == 3:  print("Dictionary:", my\_dict)  print("Access Dictionary element (key 'b'):", my\_dict["b"])  elif choice == 4:  print("Exiting program...")  break  else:  print("Invalid choice! Try again.") |

**Question 7**

|  |
| --- |
| **Source Code** |

|  |
| --- |
| # Q1: Write a program to perform Linear Search  def linear\_search(arr, target):  for i in range(len(arr)):  if arr[i] == target:  return i  return -1  while True:  print("\n--- Linear Search Menu ---")  print("1. Perform Linear Search")  print("2. Exit")  choice = int(input("Enter your choice: "))   if choice == 1:  arr = list(map(int, input("Enter numbers separated by space: ").split()))  target = int(input("Enter the element to search: "))  result = linear\_search(arr, target)  if result != -1:  print(f"Element found at index {result}")  else:  print("Element not found")  elif choice == 2:  print("Exiting program...")  break  else:  print("Invalid choice! Try again.") |

**Question 8**

|  |
| --- |
| **Source Code** |

|  |
| --- |
| # Q2: Write a function to find all duplicates in two different lists  def find\_duplicates(list1, list2):  duplicates = set(list1).intersection(set(list2))  return list(duplicates)  while True:  print("\n--- Duplicate Finder Menu ---")  print("1. Find Duplicates")  print("2. Exit")  choice = int(input("Enter your choice: "))   if choice == 1:  list1 = list(map(int, input("Enter elements of first list separated by space: ").split()))  list2 = list(map(int, input("Enter elements of second list separated by space: ").split()))  duplicates = find\_duplicates(list1, list2)  if duplicates:  print("Duplicates found:", duplicates)  else:  print("No duplicates found")  elif choice == 2:  print("Exiting program...")  break  else:  print("Invalid choice! Try again.") |

**Question 9**

|  |
| --- |
| **Source Code** |

|  |
| --- |
| # Q3: Write a function to print the sum of numbers in list having 3 at their units place  def sum\_with\_units\_digit\_3(numbers):  return sum(num for num in numbers if num % 10 == 3)  while True:  print("\n--- Sum of Numbers Ending with 3 Menu ---")  print("1. Calculate Sum")  print("2. Exit")  choice = int(input("Enter your choice: "))   if choice == 1:  numbers = list(map(int, input("Enter numbers separated by space: ").split()))  total = sum\_with\_units\_digit\_3(numbers)  print(f"Sum of numbers ending with 3 is: {total}")  elif choice == 2:  print("Exiting program...")  break  else:  print("Invalid choice! Try again.") |