**PROGRAMMING IN PYTHON**

**ASSIGNMENT**

**NAME :- SAVI**

**ROLL NO.:- 23CY-14**

**SUBJECT :-COMPUTER SCIENCE**

**COURSE:- BSC CHEMISTRY**

**Q1.** Write a program to determine the type of input using match statement.

Sol. Def print\_type(value):

match value:

case int():

print(“It’s a integer.”)

case str():

print(“It’s a string”)

case list():

print(“It’s a list.”)

case dict() as d:

print (f “It’s a dictionary with {len(d)} keys.”)

case \_:

print (“ \_\_\_\_”)

**Q2.** Find the factorial of a number using function.

Sol. def factorial(n):

     if n < 0:

         return 0

     elif n == 0 or n == 1:

         return 1

      else:

         fact = 1

         while(n > 1):

             fact \*= n

             n -= 1

         return fact

num = 5

print("Factorial of",num,"is", factorial(num))

**Q3.** Write a program to print even numbers using continue statement.

Sol. L = []

for i in range :

if i%2==0;

L.append(i)

print(L)

**Q4.** Write a program to demonstrate list, set, tuple and dictionary**.**

Sol. # List

fruits\_list = ["apple", "banana", "orange", "apple", "grape"]

print("List:")

print(fruits\_list)

# Set

fruits\_set = {"apple", "banana", "orange", "apple", "grape"}

print("\nSet:")

print(fruits\_set)

# Tuple

fruits\_tuple = ("apple", "banana", "orange", "apple", "grape")

print("\nTuple:")

print(fruits\_tuple)

# Dictionary

fruits\_dict = {"apple": 2, "banana": 3,"orange": 1, "grape": 4}

print("\nDictionary:")

print(fruits\_dict)

**Q5**. Write a program to count number of vowels in a string.

Sol. def count\_vowels(string):

vowels = "aeiouAEIOU"

count = 0

for char in string:

if char in vowels:

count += 1

return count

**Q6.** Write a program to find maximum and minimum of n numbers.

Sol. def find\_max\_min(numbers):

if not numbers:

return None,

max\_num = min\_num = numbers[0]

for num in numbers:

if num > max\_num:

max\_num = num

elif num < min\_num:

min\_num = num

return max\_num, min\_num

**Q7.** Write a program to check number is prime or not.

Sol. def is\_prime(number):

if number <= 1:

return False

elif number <= 3:

return True

elif number % 2 == 0 or number % 3 == 0:

return False

i = 5

while i \* i <= number:

if number % i == 0 or number % (i + 2) == 0:

return False

i += 6

return True

**Q8.** Write a program to count occurance of each element in the list.

Sol. def count\_occurrences(lst):

occurrence\_count = {}

for item in lst:

if item in occurrence\_count:

occurrence\_count[item] += 1

else:

occurrence\_count[item] = 1

return occurrence\_count

**Q9.** Write a program to display the first occurrence of number divisible by k in the list.

Sol. def first\_divisible\_index(lst, k):

for index, num in enumerate(lst):

if num % k == 0:

return index

return -1

**Q10.** Check a string is pallidrome or not.

Sol. def is\_palindrome(s):

return s == s[::-1]

input\_string = input("Enter a string: ")

if is\_palindrome(input\_string):

print("The string is a palindrome.")

else:

print("The string is not a palindrome.")

**Q11.** Generate Fibonacci sequence upto n terms using while loop.

Sol. def fibonacci\_sequence(n):

fibonacci\_seq = []

if n <= 0:

return fibonacci\_seq

elif n == 1:

fibonacci\_seq.append(0)

elif n == 2:

fibonacci\_seq.extend([0, 1])

else:

fibonacci\_seq.extend([0, 1])

i = 2

while i < n:

next\_num = fibonacci\_seq[-1] + fibonacci\_seq[-2]

fibonacci\_seq.append(next\_num)

i += 1

return fibonacci\_seq

**Q.12** Write a program to find all prime numbers from 1 to n numbers.

Sol. def sieve\_of\_eratosthenes(n):

primes = [True] \* (n + 1)

primes[0] = primes[1] = False

p = 2

while p\*\*2 <= n:

if primes[p]:

for i in range(p\*\*2, n + 1, p):

primes[i] = False

p += 1

prime\_numbers = []

for i in range(2, n + 1):

if primes[i]:

prime\_numbers.append(i)

return prime\_numbers

**Q13.** Write a program to print all the names from list whose length greater than 6.

Sol. def print\_names\_greater\_than\_six(names):

for name in names:

if len(name) > 6:

print(name)

**Q14.** Find the sum of digit of a number.

Sol. def sum\_of\_digits(number):

sum\_digits = 0

for digit in str(number):

sum\_digits += int(digit)

return sum\_digits

**Q15.** Write a program to check a number is binary.

Sol. def is\_binary(number):

for digit in str(number):

if digit not in ['0', '1']:

return False

return True

**Q16**. Write a program to remove vowels from string.

Sol. def remove\_vowels(string):

vowels = "aeiouAEIOU"

result = ""

for char in string:

if char not in vowels:

result += char

return result

**Q17.** Write a program to display nth Fibonacci number.

Sol. def fibonacci(n):

if n <= 0:

return "Invalid input. Please enter a positive integer."

elif n == 1:

return 0

elif n == 2:

return 1

else:

fib = [0, 1]

for i in range(2, n):

fib.append(fib[-1] + fib[-2])

return fib[-1]

**Q18.** Check a number is Armstrong.

Sol. def is\_armstrong(number):

num\_digits = len(str(number))

sum\_of\_powers = sum(int(digit)\*\*num\_digits for digit in str(number))

if sum\_of\_powers == number:

return True

else:

return False

**Q19.** Write a program to print ASCII value of all the characters of string with character.

Sol. def print\_ascii\_values(string):

for char in string:

print(f"Character: {char}, ASCII value: {ord(char)}")

**Q20**. Write a program to check weather a list is monotonic or not.

Sol. def is\_monotonic(arr):

non\_decreasing = all(arr[i] <= arr[i + 1] for i in range(len(arr) - 1))

non\_increasing = all(arr[i] >= arr[i + 1] for i in range(len(arr) - 1))

return non\_decreasing or non\_increasing

**Q21.** Write a program to check a particular element is present in the array or not.

Sol. Using a loop:

def is\_element\_present(arr, element):

for item in arr:

if item == element:

return True

return False

Using the **in** operator:

def is\_element\_present(arr, element):

return element in arr

arr = [1, 2, 3, 4, 5]

element = int(input("Enter the element to check: "))

if is\_element\_present(arr, element):

print("Element is present in the array.")

else:

print("Element is not present in the array.")

**Q22.** Write a program to find nth largest and nth smallest element from list.

Sol. def find\_nth\_largest\_and\_smallest(input\_list, n):

sorted\_list = sorted(input\_list)

nth\_smallest = sorted\_list[n - 1]

nth\_largest = sorted\_list[-n]

return nth\_smallest, nth\_largest

**Q23.** Write a program to remove kth character from string.

Sol. def remove\_kth\_character(input\_string, k):

if k < 0 or k >= len(input\_string):

print("Invalid index.")

return input\_string

string\_list = list(input\_string)

del string\_list[k]

result\_string = ''.join(string\_list)

return result\_string

**Q24.** Check a particular substring present in the string.

Sol. def check\_substring\_presence(string, substring):

if substring in string:

print(f"The substring '{substring}' is present in the string.")

else:

print(f"The substring '{substring}' is not present in the string.")

index = string.find(substring)

if index != -1:

print(f"The substring '{substring}' is present in the string at index {index}.")

else:

print(f"The substring '{substring}' is not present in the string.")

**Q25.** Write a program to print all the characters whose length is even.

Sol. def print\_even\_length\_characters(strings):

for string in strings:

if len(string) % 2 == 0:

for char in string:

print(char, end=" ")

print()

**Q26.** Remove duplicate elements from list.

Sol. def remove\_duplicates(input\_list):

unique\_set = set(input\_list)

unique\_list = list(unique\_set)

return unique\_list

**Q27.** Write a program to display a character whose is more than other characters.

Sol. def most\_frequent\_char(input\_string):

char\_frequency = {}

for char in input\_string:

if char in char\_frequency:

char\_frequency[char] += 1

else:

char\_frequency[char] = 1

max\_occurrence = max(char\_frequency.values())

max\_chars = [char for char, freq in char\_frequency.items() if freq == max\_occurrence]

return max\_chars, max\_occurrence

**Q28.** Find words whose length is greater than j and less than k.

Sol. def find\_words\_with\_length\_between(input\_string, j, k):

words = input\_string.split()

filtered\_words = [word for word in words if len(word) > j and len(word) < k]

return filtered\_words

**Q29.** Check a string is binary or not.

Sol. def is\_binary\_string(input\_string):

for char in input\_string:

if char != '0' and char != '1':

return False

return True

**Q30.** Find uncommon words from two string.

Sol. def find\_uncommon\_words(str1, str2):

words1 = set(str1.split())

words2 = set(str2.split())

uncommon\_words = words1.symmetric\_difference(words2)

return uncommon\_words

**Q31.** Check a mail is valid or not.

Sol. import re

def is\_valid\_email(email):

pattern = r'^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'

if re.match(pattern, email):

return True

else:

return False

**Q32.** Check a mobile number is valid or not.

Sol. import re

def is\_valid\_mobile\_number(number):

pattern = r'^\d{10}$'

if re.match(pattern, number):

return True

else:

return False

**Q33.** Display only digits from tuple.

Sol. def display\_digits\_from\_tuple(input\_tuple):

digits = []

for item in input\_tuple:

if isinstance(item, int) or isinstance(item, float):

item\_str = str(item)

for char in item\_str:

if char.is digit():

digits.append(char)

return digits

**Q34.** Write a program to display a character having higher occurance.

Sol. def highest\_occurrence (input\_string):

char\_frequency = {}

for char in input\_string:

if char in char\_frequency:

char\_frequency[char] += 1

else:

char\_frequency[char] = 1

max\_occurrence = max(char\_frequency.values())

max\_chars = [char for char, freq in char\_frequency.items() if freq == max\_occurrence]

return max\_chars, max\_occurrence