**BCA 4 PYTHON PROGRAMMING LAB**

**PRACTICAL QUESTIONS**

Practical 1  **– Simple Introductory Python Programs**

* 1. **WAP to Calculate Area and Perimeter of a Rectangle**

**Answer ~** # Function to calculate area of a rectangle

def calculate\_area(length, width):

return length \* width

# Function to calculate perimeter of a rectangle

def calculate\_perimeter(length, width):

return 2 \* (length + width)

# Main function

def main():

length = float(input("Enter length of the rectangle: "))

width = float(input("Enter width of the rectangle: "))

area = calculate\_area(length, width)

perimeter = calculate\_perimeter(length, width)

print("Area of the rectangle:", area)

print("Perimeter of the rectangle:", perimeter)

if \_\_name\_\_ == "\_\_main\_\_":

main()

* 1. **WAP to calculate Avg. marks of 3 subjects**

**Answer ~** # Function to calculate average marks

def calculate\_average(subject1, subject2, subject3):

return (subject1 + subject2 + subject3) / 3

# Main function

def main():

subject1 = float(input("Enter marks for subject 1: "))

subject2 = float(input("Enter marks for subject 2: "))

subject3 = float(input("Enter marks for subject 3: "))

average = calculate\_average(subject1, subject2, subject3)

print("Average marks:", average)

if \_\_name\_\_ == "\_\_main\_\_":

main()

* 1. **WAP to compute compound Interest**

**Answer ~** # Function to calculate compound interest

def calculate\_compound\_interest(principal, rate, time):

amount = principal \* (1 + rate/100) \*\* time

compound\_interest = amount - principal

return compound\_interest

# Main function

def main():

principal = float(input("Enter the principal amount: "))

rate = float(input("Enter the annual interest rate (in percentage): "))

time = float(input("Enter the time period (in years): "))

compound\_interest = calculate\_compound\_interest(principal, rate, time)

print("Compound interest:", compound\_interest)

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Practical 2 – Working with Strings – basic String Operations**

* 1. **WAP to demonstrate Slicing Operations in Strings**

**Answer ~** # Slicing Operations in Strings

string = "Hello, World!"

# Slicing from index 2 to 5 (exclusive)

print("Sliced string:", string[2:5])

# Slicing from the beginning to index 5 (exclusive)

print("Sliced string from beginning:", string[:5])

# Slicing from index 7 to the end

print("Sliced string till end:", string[7:])

# Slicing from index -5 to -2 (negative indexing)

print("Sliced string with negative indexing:", string[-5:-2])

# Slicing with a step of 2

print("Sliced string with step of 2:", string[::2])

* 1. **WAP to demonstrate built in functions of Strings**

**Answer** ~ # Built-in functions of Strings

string = "Hello, World!"

# Length of the string

print("Length of the string:", len(string))

# Convert string to uppercase

print("Uppercase:", string.upper())

# Convert string to lowercase

print("Lowercase:", string.lower())

# Capitalize the string (first character uppercase, rest lowercase)

print("Capitalized:", string.capitalize())

# Count occurrences of a substring

substring = "l"

print("Count of '", substring, "' in string:", string.count(substring))

# Find the index of a substring

substring = "World"

print("Index of '", substring, "' in string:", string.find(substring))

# Replace substring

old\_substring = "World"

new\_substring = "Universe"

print("String after replacing '", old\_substring, "' with '", new\_substring, "':", string.replace(old\_substring, new\_substring))

**Practical 3 – Conditionals in Python – Decision Control**

* 1. **WAP to check if one No. is divisible by the other or not**

**Answer ~** def check\_divisibility(num1, num2):

if num2 == 0:

return "Cannot divide by zero."

elif num1 % num2 == 0:

return f"{num1} is divisible by {num2}."

else:

return f"{num1} is not divisible by {num2}."

num1 = int(input("Enter the first number: "))

num2 = int(input("Enter the second number: "))

print(check\_divisibility(num1, num2))

* 1. **WAP to check if a Number is +ve, -ve or zero**
  2. **Answer ~**

def check\_number(num):

if num > 0:

return f"{num} is positive."

elif num < 0:

return f"{num} is negative."

else:

return "The number is zero."

num = float(input("Enter a number: "))

print(check\_number(num))

* 1. **WAP to check if a given year is Leap or not**

**Answer ~** def check\_leap\_year(year):

if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):

return f"{year} is a leap year."

else:

return f"{year} is not a leap year."

year = int(input("Enter a year: "))

print(check\_leap\_year(year))

* 1. **WAP to accept basic salary for the employee. Calculate DA as 30% of bs, HRA as 20% of bs if bs>=20000else compute DA as 20% and HRA as 10%. Display the result.**

**Answer** ~ def calculate\_salary(bs):

if bs >= 20000:

da = 0.3 \* bs

hra = 0.2 \* bs

else:

da = 0.2 \* bs

hra = 0.1 \* bs

total\_salary = bs + da + hra

return f"Basic Salary: {bs}, DA: {da}, HRA: {hra}, Total Salary: {total\_salary}"

basic\_salary = float(input("Enter the basic salary: "))

print(calculate\_salary(basic\_salary))

* 1. **WAP to accept sales amount for the salesman. Compute commission as 20% of sales amount if sales amount>=15000, comm as 15% if sales amount >=1000 else comm as 10 %. Display the result.**

**Answer ~** def calculate\_commission(sales\_amount):

if sales\_amount >= 15000:

commission = 0.2 \* sales\_amount

elif sales\_amount >= 1000:

commission = 0.15 \* sales\_amount

else:

commission = 0.1 \* sales\_amount

return f"Sales Amount: {sales\_amount}, Commission: {commission}"

sales\_amount = float(input("Enter the sales amount: "))

print(calculate\_commission(sales\_amount))

**Practical 4 – While and For loops – Repetition Control Statements**

* 1. **WAP to show working of a while Loop with a text**

**Answer ~** text = "Hello, World!"

index = 0

while index < len(text):

print(text[index])

index += 1

* 1. **WAP to show working of a while Loop with a Numbers**

**Answer ~** num = 1

while num <= 10:

print(num)

num += 1

* 1. **WAP to show the working of break and continue statement**

**Answer** ~num = 1

while num <= 10:

if num == 5:

break

print(num)

num += 1

num = 1

while num <= 10:

if num == 5:

num += 1

continue

print(num)

num += 1

* 1. **WAP to the use of else statement with while and break**

**Answer ~** num = 1

while num <= 10:

print(num)

num += 1

else:

print("Loop completed without breaking.")

num = 1

while num <= 10:

if num == 5:

break

print(num)

num += 1

else:

print("Loop completed without breaking.")

* 1. **WAP to compute the Sum of the Series 4 + 8 + 12 + … + 40**

**Answer ~** sum = 0

num = 4

while num <= 40:

sum += num

num += 4

print("Sum of the series:", sum)

* 1. **Write a program to display the Fibonacci sequences up**

**to nth term where n is provided by the user.**

**Answer ~** def fibonacci(n):

fib\_sequence = [0, 1]

for i in range(2, n):

next\_term = fib\_sequence[-1] + fib\_sequence[-2]

fib\_sequence.append(next\_term)

return fib\_sequence

n = int(input("Enter the value of n: "))

fib\_series = fibonacci(n)

print("Fibonacci sequence up to the nth term:", fib\_series)

* 1. **WAP that prints multiplication table of a umber using for loop.**

**Answer ~** num = int(input("Enter the number: "))

for i in range(1, 11):

print(f"{num} \* {i} = {num \* i}")

* 1. **WAP To print a Triangle Pattern**

**Answer ~** rows = int(input("Enter the number of rows: "))

for i in range(1, rows + 1):

print("\*" \* i)

* 1. **WAP to compute Sum of the series 3 + 6 + 9 + … + 30**

**Answer ~** sum = 0

for i in range(3, 31, 3):

sum += i

print("Sum of the series:", sum)

* 1. **WAP to print the product of the series m = 15 \* 13 \* 11 \* 9 \* 7**

**Answer** ~ product = 1

for i in range(15, 6, -2):

product \*= i

print("Product of the series:", product)

* 1. **WAP to compute factorial of a Number**

**Answer ~** num = int(input("Enter a number: "))

factorial = 1

for i in range(1, num + 1):

factorial \*= i

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

* 1. **WAP to display the Cube of first 10 even numbers**

**Answer ~** for i in range(2, 21, 2):

print("Cube of", i, ":", i \*\* 3)

* 1. **WAP to compute sum of first n natural numbers**

**Answer ~** n = int(input("Enter the value of n: "))

sum = 0

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

sum += i

print("Sum of the first", n, "natural numbers:", sum**)**

* 1. **WAP to display 1 to 10 numbers in reverse order**

**Answer ~** for i in range(10, 0, -1):

print(i)

* 1. **WAP to create a list of any specific size. Arrange all the**

**elements in ascending order. Display list before and after sorting**

**Answer ~** size = int(input("Enter the size of the list: "))

my\_list = []

for i in range(size):

my\_list.append(int(input("Enter element: ")))

print("List before sorting:", my\_list)

my\_list.sort()

print("List after sorting:", my\_list)

**Practical 5 Working with Lists in Python**

* 1. **WAP to show the creation and working of lists**

**Answer ~** # Creating a list

my\_list = [1, 2, 3, 4, 5]

# Accessing elements of a list

print("First element:", my\_list[0])

print("Last element:", my\_list[-1])

# Modifying elements of a list

my\_list[0] = 10

print("Modified list:", my\_list)

# Adding elements to a list

my\_list.append(6)

print("List after appending 6:", my\_list)

# Removing elements from a list

my\_list.remove(3)

print("List after removing 3:", my\_list)

# Slicing a list

print("Sliced list:", my\_list[1:4])

* 1. **WAP to print elements of a list[‘q’,’w’,’e’,’r’,’t’,’y’] in 3eparate**

**Answer ~** my\_list = ['q', 'w', 'e', 'r', 't', 'y']

for index, value in enumerate(my\_list):

print(f"Index: {index}, Negative Index: {index - len(my\_list)}, Value: {value}")

**lines along with element’s both indexes (Positive and Negative).**

* 1. **WAP to demonstrate the working of methods used with lists**

**Answer ~** my\_list = [1, 2, 3, 4, 5]

# append() method

my\_list.append(6)

print("List after appending 6:", my\_list)

# insert() method

my\_list.insert(2, 10)

print("List after inserting 10 at index 2:", my\_list)

# remove() method

my\_list.remove(3)

print("List after removing 3:", my\_list)

# pop() method

popped\_element = my\_list.pop()

print("Popped element:", popped\_element)

print("List after popping:", my\_list)

# sort() method

my\_list.sort()

print("Sorted list:", my\_list)

# reverse() method

my\_list.reverse()

print("Reversed list:", my\_list)

* 1. **WAP to create a 3\*3 Matrix and how to extract individual elements of the**

**Matrix**

**Answer** ~ matrix = [[1, 2, 3],

[4, 5, 6],

[7, 8, 9]]

# Extracting individual elements

print("Element at row 2, column 3:", matrix[1][2])

* 1. **WAP to demonstrate built-in functions in Lists**

**Answer ~** my\_list = [1, 2, 3, 4, 5]

# len() function

print("Length of the list:", len(my\_list))

# max() function

print("Maximum element of the list:", max(my\_list))

# min() function

print("Minimum element of the list:", min(my\_list))

# sum() function

print("Sum of the elements of the list:", sum(my\_list))

* 1. **WAP to calculate the mean, variance and std. deviation of given list of** numbers

**Answer ~** import statistics

numbers = [10, 20, 30, 40, 50]

# Mean calculation

mean = statistics.mean(numbers)

print("Mean:", mean)

# Variance calculation

variance = statistics.variance(numbers)

print("Variance:", variance)

# Standard deviation calculation

std\_dev = statistics.stdev(numbers)

print("Standard deviation:", std\_dev)