Python Programming

2025-26

Manav Rachna International Institute of Research and Studies

School of Computer Applications

Department Of Computer Applications

Submitted By	
Student Name	**AYUSH DAS**
Roll No	**24/SCA/BCA(AI&ML)/012**
Programme	**BCA (AI&ML)**
Semester	**3rd**
Section/Group	**c**
Department	**Computer Applications**
Session / Batch	**2025-26**
Submitted To	
Faculty Name	**Dr. Sakshi Gupta**

Q1. Write a Python program to calculate number of days between two dates. Sample dates: (2014, 7, 2), (2014, 7,11)

INPUT-

```
from datetime import datetime

date1 = input("Enter first date (dd-mm-yyyy): ")

date2 = input("Enter second date (dd-mm-yyyy): ")

d1 = datetime.strptime(date1, "%d-%m-%Y")

d2 = datetime.strptime(date2, "%d-%m-%Y")

diff = abs((d2 - d1).days)

print(f"Difference between the two dates is {diff} days.")
```

Output-

```
Enter first date (dd-mm-yyyy): 2-7-2014
Enter second date (dd-mm-yyyy): 11-7-2014
Difference between the two dates is 9 days.
```

Q2. Write a Python program that accepts an integer (n) and computes the value of n+nn+nnn

INPUT-

```
n = input("Enter an integer: ")

result = int(n) + int(n*2) + int(n*3)

print(f"The value of n+nn+nnn for n={n} is {result}")
```

Output-

```
Enter an integer: 20
The value of n+nn+nnn for n=20 is 204060
```

Q3. Ask the user for a number. Depending on whether the number is even or odd, print out an appropriate message to the user.

INPUT-

```
a = int(input("Enter the Number: "))
if a % 2 == 0:
    print("This is an Even Number")
else:
    print("This is an Odd Number")
```

Output-

```
Enter the Number: 21
This is an Odd Number
```

Q4. Write a Python program which accepts a sequence of comma-separated numbers from user and generate a list and a tuple with those numbers.

INPUT-

```
numbers = input("Enter comma-separated numbers: ")
list_numbers = numbers.split(",")
tuple_numbers = tuple(list_numbers)
print("List :", list_numbers)
print("Tuple:", tuple_numbers)
```

Output-

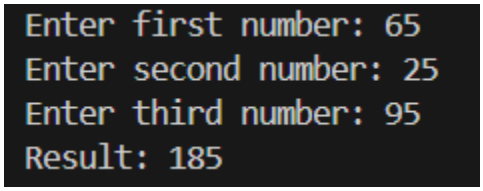
```
Enter comma-separated numbers: 15,25,35,45,55
List : ['15', '25', '35', '45', '55']
Tuple: ('15', '25', '35', '45', '55')
```

Q5. Write a Python program to calculate the sum of three given numbers, if the values are equal then return thrice of their sum.

INPUT-

```
def calculate_sum(a, b, c):
    total = a + b + c
    if a == b == c:
        return 3 * total
    else:
        return total
x = int(input("Enter first number: "))
y = int(input("Enter second number: "))
z = int(input("Enter third number: "))
result = calculate_sum(x, y, z)
print("Result:", result)
```

Output-

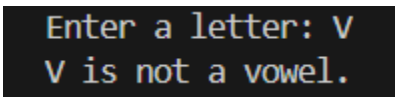


Q6. Write a Python program to test whether a passed letter is a vowel or not.

INPUT-

```
def is_vowel(letter):
    vowels = "aeiouAEIOU"
    if letter in vowels:
        return True
    else:
        return False
ch = input("Enter a letter: ")

if len(ch) == 1 and ch.isalpha():
    if is_vowel(ch):
        print(f"{ch} is a vowel.")
    else:
        print(f"{ch} is not a vowel.")
else:
    print("Please enter a single alphabet only.")
```

Output-



Q7. Take a list, say for example this one:

a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]

and write a program that prints out all the elements of the list that are less than 5.

Extras:Program-

a. Instead of printing the elements one by one, make a new list that has all the elements less than 5 from this list in it and print out this new list.

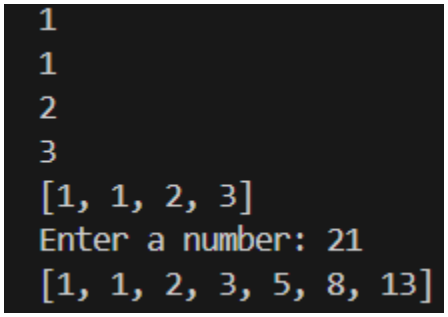
b.Write this in one line of Python.

c.Ask the user for a number and return a list that contains only elements from the original list a that are smaller than that number given by the user.

INPUT-

```
a = [1,1,2,3,5,8,13,21,34,55,89]

for i in a:

   if i < 5:

      print(i)


print([i for i in a if i<5])


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

result = [x for x in a if x < num]

print(result)
```

OUTPUT-



Q8. Create a program that asks the user for a number and then prints out a list of all the divisors of that number.

INPUT-

```
num = int(input("Enter a number: "))

divisors = [i for i in range(1, num + 1) if num % i == 0]

print(f"Divisors of {num} are: {', '.join(map(str, divisors))}")
```

OUTPUT-

```
Enter a number: 25
Divisors of 25 are: 1, 5, 25
```

Q9. Take two lists, say for example these two:

a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]

b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]

and write a program that returns a list that contains only the elements that are common between the lists (without duplicates). Make sure your program works on two lists of different sizes.

INPUT-

```
a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]

b = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]

common = list(set(a) & set(b))

print("Common elements:", common)
```

OUTPUT-

```
Common elements: [1, 2, 3, 5, 8, 13]
```

Q10. Ask the user for a string and print out whether this string is a palindrome or not. (A palindrome is a string that reads the same forwards and backwards.)

INPUT-

```
string = input("Enter a string: ")

string = string.lower()

string = string.replace(" ", "")

if string == string[::-1]:

    print("The string is a palindrome")

else:

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

Output-

```
Enter a string: 21
The string is not a palindrome
```

Q11. Let's say I give you a list saved in a variable: a = [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]. Write one line of Python that takes this list a and makes a new list that has only the even elements of this list in it.

INPUT-

```
a = [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

even_elements = [element for element in a if element % 2 == 0]

print("Even Elements are:",even_elements)
```

Output-

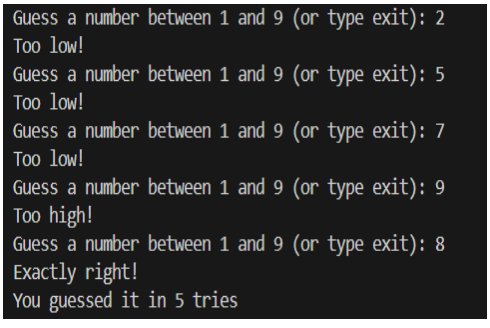
```
Even Elements are: [4, 16, 36, 64, 100]
```

Q12. Generate a random number between 1 and 9 (including 1 and 9). Ask the user to guess the number, then tell them whether they guessed too low, too high, or exactly right. (Hint: remember to use the user input lessons from the very first exercise)

INPUT-

```
import random

number = random.randint(1, 9)

guess = 0

count = 0

while guess != number and guess != "exit":
    guess = input("Guess a number between 1 and 9 (or type exit): ")
    if guess == "exit":
        break
    guess = int(guess)
    count += 1
    if guess < number:
        print("Too low!")
    elif guess > number:
        print("Too high!")
    else:
        print("Exactly right!")
        print("You guessed it in", count, "tries")
```

Output-



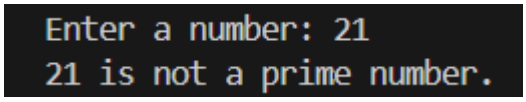
Q13)Ask the user for a number and determine whether the

number is prime or not. (For those who have forgotten, a prime number is a number that has no divisors.).

INPUT-

```
n = int(input("Enter a number: "))

if n > 1:
    for i in range(2, n):
        if n % i == 0:
            print(n, "is not a prime number")
            break
    else:
        print(n, "is a prime number")
else:
    print(n, "is not a prime number")
```

OUTPUT-



Q14)Write a program (function!) that takes a list and returns a

new list that contains all the elements of the first list minus all the duplicates.

INPUT-

```
def remove_duplicates(lst):
    return list(set(lst))
a = [1, 2, 2, 3, 4, 4, 5]
print("Original:", a)
print("Without duplicates:", remove_duplicates(a))
```

OUTPUT-

```
Original: [1, 2, 2, 3, 4, 4, 5]
Without duplicates: [1, 2, 3, 4, 5]
```

Q15)Write a function that takes an ordered list of numbers (a list where the elements are in order from smallest to largest) and another number. The function decides whether or not the given number is inside the list and returns (then prints) an appropriate boolean.

INPUT-

```
def check_number(lst, num):
    if num in lst:
        print(True)
    else:
        print(False)
lst = [1, 3, 5, 7, 9, 11]
num = int(input("Enter number to search: "))
check_number(lst, num)
```

OUTPUT-

```
Enter number to search: 21
False
```

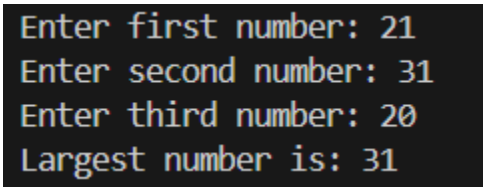
Q16)Implement a function that takes as input three variables,

and returns the largest of the three. Do this without using the Python max() function!

INPUT-

```
a = int(input("Enter first number: "))
b = int(input("Enter second number: "))
c = int(input("Enter third number: "))
if a >= b and a >= c:
    largest = a
elif b >= a and b >= c:
    largest = b
else:
    largest = c
print("Largest number is:", largest)
```

OUTPUT-

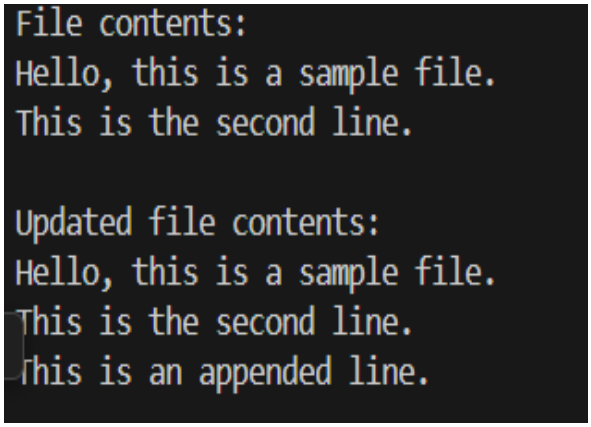


Q17 Python program to perform read and write operations on a file.

INPUT-

```
from pathlib import Path

filename = "sample.txt"

Path(filename).write_text("Hello, this is a sample file.\nThis is the second line.\n")

print("File contents:")
print(Path(filename).read_text())

Path(filename).write_text(Path(filename).read_text() + "This is an appended line.\n")

print("Updated file contents:")
print(Path(filename).read_text())
```

OUTPUT-



Q18 Python program to copy the contents of a file to another file.

INPUT-

```
import os

source_file = 'source.txt'
destination_file = 'destination.txt'

if not os.path.exists(source_file):
    open(source_file, 'w').close()

with open(source_file, 'r') as src, open(destination_file, 'w') as dst:
    for line in src:
        dst.write(line)
```

OUTPUT-

```
FileNotFoundError: source.txt does not exist.
```

Q19 Python program to count frequency of characters in a given file.

INPUT-

```
def count_char_frequency(filename):
    freq = {}
    try:
        with open(filename, 'r', encoding='utf-8') as file:
            for line in file:
                for char in line:
                    freq[char] = freq.get(char, 0) + 1
        return freq
    except FileNotFoundError:
        print(f"File '{filename}' not found.")
        return {}

if __name__ == "__main__":
    filename = input("Enter the filename: ")
    frequencies = count_char_frequency(filename)
    for char, count in frequencies.items():
        print(f"'{char}': {count}")
```

OUTPUT-

```
Enter the filename: .PDF
File '.PDF' not found.
```

Q20 Python program to print each line of a file in reverse order.

INPUT-

```
def reverse_file_lines(filename):
    try:
        with open(filename, 'r') as file:
            for line in file:
                print(line.strip()[::-1])
    except FileNotFoundError:
        print("File not found. Please check the filename.")

if __name__ == "__main__":
    filename = input("Enter the file path: ")
    reverse_file_lines(filename)
```

OUTPUT-

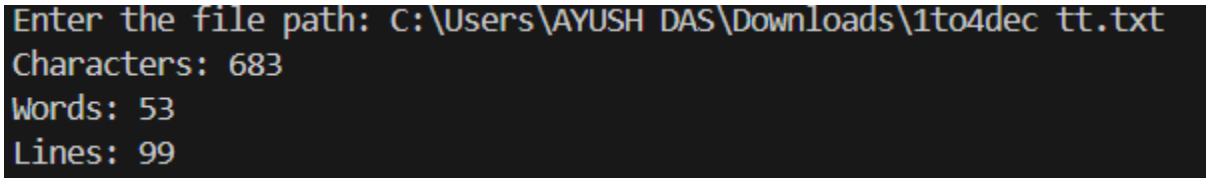
```
Enter the file path: C:\Users\AYUSH DAS\Downloads\1to4dec tt.txt
retsemeS drIII ACB
8       7       6       5       4       3       2       1

00:9-01:8
05:9-00:9
04:01-05:9
03:11-04:01
02:21-03:11
01:1-02:21
00:2-01:1
05:2-00:20
```

Q21 Python program to compute the number of characters, words and lines in a file.

INPUT-

```
def file_statistics(filename):
    with open(filename, 'r', encoding='utf-8') 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_chars, num_words, num_lines

if __name__ == "__main__":
    file_path = input("Enter the file path: ")
    chars, words, lines = file_statistics(file_path)
    print(f"Characters: {chars}")
    print(f"Words: {words}")
    print(f"Lines: {lines}")
```

OUTPUT-

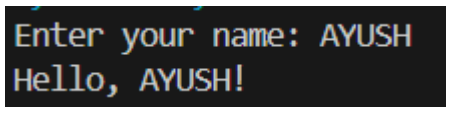


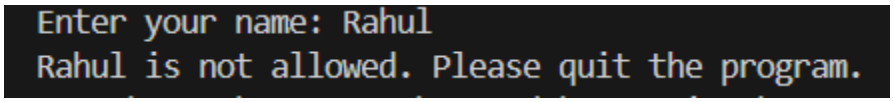
Q22 Write a program that prompts the user to enter his name. The program then greets the person with his name. But if the person's name is 'Rahul' and exception is thrown and he is asked to quit the program.

INPUT-

```
class RahulException(Exception):
    pass

def main():
    name = input("Enter your name: ")
    try:
        if name == "Rahul":
            raise RahulException("Rahul is not allowed. Please quit the program.")
        print(f"Hello, {name}!")
    except RahulException as e:
        print(e)

if __name__ == "__main__":
    main()
```

OUTPUT-



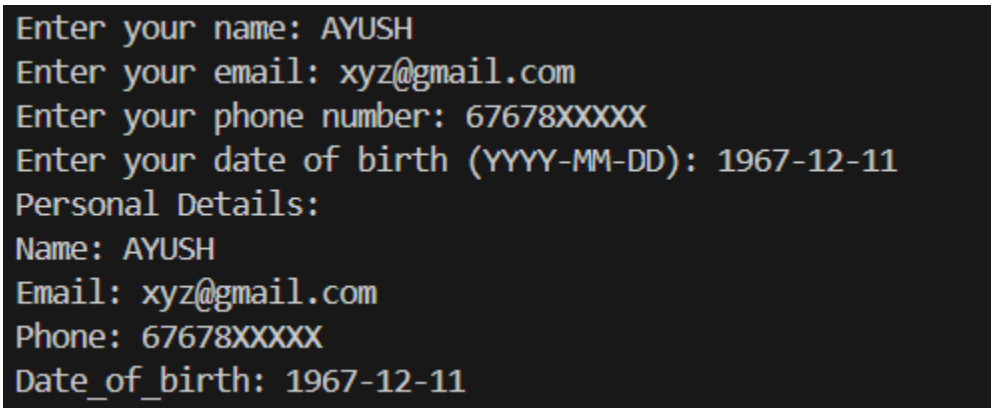


Q23 Write a program that accepts date of birth along with the other personal details of a person. Throw an exception if an invalid date is entered.

INPUT-

```
from datetime import datetime

class InvalidDateError(Exception):
    pass

def get_personal_details():
    name = input("Enter your name: ")
    email = input("Enter your email: ")
    phone = input("Enter your phone number: ")
    dob_str = input("Enter your date of birth (YYYY-MM-DD): ")
    try:
        dob = datetime.strptime(dob_str, "%Y-%m-%d")
    except ValueError:
        raise InvalidDateError("Invalid date format or value entered.")
    return {
        "name": name,
        "email": email,
        "phone": phone,
        "date_of_birth": dob.strftime("%Y-%m-%d")
    }

if __name__ == "__main__":
    try:
        details = get_personal_details()
        print("Personal Details:")
        for key, value in details.items():
            print(f"{key.capitalize()}: {value}")
    except InvalidDateError as e:
        print(f"Error: {e}")
```

OUTPUT-

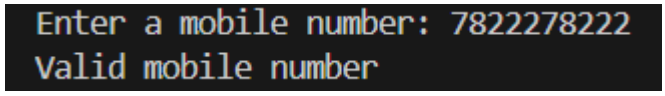


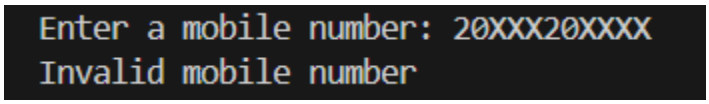
Q24 Write a Regular Expression to represent all 10 digit mobile numbers. Rules: 1. Every number should contains exactly 10 digits. 2. The first digit should be 7 or 8 or 9 Write a
Python Program to check whether the given number is valid mobile number or not?

INPUT-

```
import re

def is_valid_mobile(number):
    pattern = r'^[789]\d{9}$'
    return bool(re.match(pattern, number))

# Example usage
number = input("Enter a mobile number: ")
if is_valid_mobile(number):
    print("Valid mobile number")
else:
    print("Invalid mobile number")
```

OUTPUT-





Q25 A spell checker can be a helpful tool for people who struggle to spell words correctly. In this exercise, you will write a program that reads a file and displays all of the words in it that are misspelled. Misspelled words will be identified by checking each word in the file against a list of known words. Any words in the user's file that do not appear in the list of known words will be reported as spelling mistakes. The user will provide the name of the file to check for spelling mistakes as a command line parameter. Your program should display an appropriate error message if the command line parameter is missing. An error message should also be displayed if your program is unable to open the user's file. Words followed by a comma, period or other punctuation mark are not reported as spelling mistakes. Ignore the capitalization of the words when checking their spelling.
INPUT-

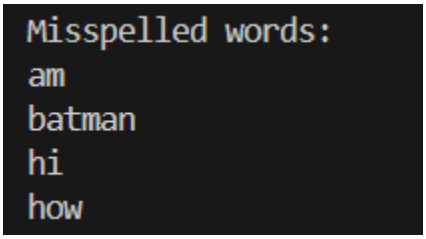
```
import sys
import string
import os

def load_known_words(filename="known_words.txt"):
    if not os.path.exists(filename):
        print(f"Error: Known words file '{filename}' not found.")
        sys.exit(1)

    with open(filename, "r", encoding="utf-8") as f:
        return set(word.strip().lower() for word in f if word.strip())


def clean_word(word):
    return word.strip(string.punctuation).lower()


def main():
    # Check if input file is provided
    if len(sys.argv) < 2:
        print("Error: Please provide the filename to check as a command line parameter.")
        sys.exit(1)

    filename = sys.argv[1]

    # Load known words
    known_words = load_known_words()

    # Try reading input file
    if not os.path.exists(filename):
        print(f"Error: File '{filename}' not found.")
        sys.exit(1)
```

```
    with open(filename, "r", encoding="utf-8") as f:
        text = f.read()

    misspelled = set()

    for word in text.split():
        cleaned = clean_word(word)
        if cleaned and cleaned not in known_words:
            misspelled.add(cleaned)

    # Output results
    if misspelled:
        print("\nMisspelled words:")
        for word in sorted(misspelled):
            print(word)
    else:
        print("\nNo misspelled words found.")


if __name__ == "__main__":
    main()
```

JSON INPUT -

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Run Spell Checker",
            "type": "python",
            "request": "launch",
            "program": "${file}",
            "console": "integratedTerminal",
            "cwd": "${workspaceFolder}",
            "args": [
                "C:\\Users\\AYUSH
DAS\\.vscode\\extensions\\sourcery.sourcery-1.37.0-win32-x64\\__pycache__\\new file.txt"
            ]
        }
    ]
}
```

OUTPUT-



Q26 A fitness coach wants to analyze the BMI of a group of people to assess their health status. The data includes the name, age, weight (kg), and height (m) of each person. The coach wants: To calculate BMI for each person.

To categorize each person based on their BMI.

To get a summary of how many people

fall into each category.
INPUT-

```
def bmi_category(bmi):
    if bmi < 18.5:
        return "Underweight"
    elif bmi < 25:
        return "Normal weight"
    elif bmi < 30:
        return "Overweight"
    else:
        return "Obese"

people = []

num = int(input("Number of entries you want? "))

for i in range(num):
    print(f"\n--- Person {i+1} ---")
    name = input("Name: ")
    age = int(input("Age: "))
    weight = float(input("Weight (kg): "))
    height_cm = float(input("Height (cm): "))

    height_m = height_cm / 100
    bmi = weight / (height_m ** 2)
    category = bmi_category(bmi)

    people.append({
        "name": name,
        "age": age,
        "weight": weight,
        "height_cm": height_cm,
        "BMI": round(bmi, 2),
        "Category": category
    })

# Summary
```

```
summary = {}
for p in people:
    cat = p["Category"]
    summary[cat] = summary.get(cat, 0) + 1

print("\n===== BMI Results =====")
for p in people:
    print(f"{p['name']} → BMI: {p['BMI']} → {p['Category']}")

print("\n===== Summary =====")
for cat, count in summary.items():
    print(f"{cat}: {count}")
```

OUTPUT-