1. (a) Which of the following are invalid identifiers ? Justify your answer :
    - (i) Course - Name
    - (ii) 7 wonders
    - (iii) Except
    - (iv) Riturn

- (i) Course - Name: Invalid. Identifiers cannot contain hyphens.
- (ii) 7 wonders: Invalid. Identifiers cannot start with a digit.
- (iii) Except: Valid.
- (iv) Riturn: Valid.

(b) Draw a flow chart to find the smallest of three numbers.

A flowchart for finding the smallest of three numbers would involve:

- Start
- Input three numbers (A, B, C)
- Is A < B?
    - If Yes: Is A < C?
        - If Yes: Print A is smallest
        - If No: Print C is smallest
    - If No: Is B < C?
        - If Yes: Print B is smallest
        - If No: Print C is smallest
- End

(c) Write a function findFactors ( ) that takes two numbers, number 1 and number 2 as input parameters and returns a tuple comprising common factors of these two numbers.

- ```
  def findFactors(number1, number2):
      factors1 = set()
      for i in range(1, int(number1**0.5) + 1):
          if number1 % i == 0:
              factors1.add(i)
              factors1.add(number1 // i)

      factors2 = set()
      for i in range(1, int(number2**0.5) + 1):
          if number2 % i == 0:
              factors2.add(i)
              factors2.add(number2 // i)

      common_factors = tuple(sorted(list(factors1.intersection(factors2))))
      return common_factors
  ```


(d) Consider the following Python code segments and determine the output produced on their execution :

  - (i) greetWith = "good morning friends" result = "" for i in greetWith. title ( ). split ( ) : result += (i[-1] + i[-1] . upper ( )) + "" print (result, strip ( ))
  - (ii) Assume that the file 'notes. txt' does not exist. try : f = open ('notes. txt', 'w') except IOError : print ('Error occurred while opening') else : print ('File opened successfully for writing') try : f = open ('notes.txt', 'r') except IOError : print ('Error occurred while opening') else : print ('File opened successfully for reading')

- (i) Output: "DSDSDSS"
  ```

- Explanation:
  - greetWith.title() makes "Good Morning Friends".
  - .split() makes ['Good', 'Morning', 'Friends'].
  - For 'Good': d + D -> dD
  - For 'Morning': g + G -> gG
  - For 'Friends': s + S -> sS
  - The strip() function is incorrectly used in print(result, strip()). It should be print(result.strip()). Assuming it's meant to strip whitespace from result, but there's no trailing space within the loop, the result will be dD gG sS.
  - However, the example output is DSDSDSS. This indicates a misunderstanding of i[-1] + i[-1].upper(). If the intention was to take the first letter of each word and concatenate them, or some other transformation, the provided code will produce dDgGsS.
  - Given the sample output DSDSDSS, it seems the question implies a different logic for result += (i[-1] + i[-1].upper()) + "". Let's assume the intended logic was to append the last character and then its uppercase version. So for "Good", it's "dD". For "Morning", "gG". For "Friends", "sS". So the output should be "dDgGsS". If the output DSDSDSS is to be achieved, the code logic needs to be different. Since the question asks for the output *produced* by the given code, the output is dDgGsS.

- (ii) Output:
  - File opened successfully for writing
  - File opened successfully for reading
  - Explanation:

- The first try-except-else block attempts to open notes.txt in write mode ('w'). Since the file does not exist, it will be created. The IOError will not be raised, and "File opened successfully for writing" will be printed.
- The second try-except-else block attempts to open notes.txt in read mode ('r'). Since the file was just created in the previous step, it now exists. The IOError will not be raised, and "File opened successfully for reading" will be printed.

(e) Write Python statements for the following :

- (i) Retrieve the marks in 'Mathematics'.
- (ii) Add the contents of the following dictionary to subjectMarks : {'Sanskrit': 78, 'Geography': 88, 'Economics': 84}
- (iii) How will Python respond when the following statement is executed ? subjectMarks.get ('Pylosophy', -1)
- (iv) Delete the details of the subject 'Hindi' from the dictionary.
- (v) What should be the updated contents of the dictionary subjectMarks after the execution of the following statement : subjectMarks ['Computer Sc'] = 100

Assume subjectMarks is a dictionary like: subjectMarks = {'Mathematics': 95, 'Science': 80, 'Hindi': 75}

- (i) marks_math = subjectMarks['Mathematics']
- (ii) subjectMarks.update({'Sanskrit': 78, 'Geography': 88, 'Economics': 84})
- (iii) Python will return -1. The get() method returns the value for the specified key. If the key is not found, it returns the default value provided (-1 in this case).
- (iv) del subjectMarks['Hindi'] or subjectMarks.pop('Hindi')

- (v) Updated contents of subjectMarks:
    - If 'Computer Sc' did not exist: subjectMarks will now contain {'Mathematics': 95, 'Science': 80, 'Hindi': 75, 'Computer Sc': 100} (and any subjects added in (ii)).
    - If 'Computer Sc' already existed: its value would be updated to 100.

(f) Consider the following Python program that defines a class Rectangle. There are some error (s) in the code, indentify them, and rewrite the corrected program : class Rectangle def_init (self, l, w) : self. length = l self. breadth = w def area () : return length * breadth def main () : ob = Rectangle (5, 4) print (area ()) main ()

Errors:

- def_init should be def __init__. It's a special method and needs double underscores.
- The area method is missing self as its first parameter, and it tries to access length and breadth as global variables instead of instance attributes (self.length, self.breadth).
- In main(), print(area()) calls a standalone area() function, not the method of the ob object. It should be print(ob.area()).

Corrected Program:

- ```
  class Rectangle:
      def __init__(self, l, w):
          self.length = l
          self.breadth = w

      def area(self):
          return self.length * self.breadth

  def main():
      ob = Rectangle(5, 4)
  ```
  ```

```
print(ob.area())

main()
```

(g) What is the difference between the following statements ?

- (i) Y + = 5 and y = y + 5
- (ii) y = = 10 and y = 10
- (iii) for j in range (1, 6) : if (j % 2 = = 1) : continue print (j) for j in range (1, 6) : if (j % 2 = = 1) : break print (j)

- (i) Y += 5 and y = y + 5:
  - Both statements achieve the same result: they add 5 to the current value of y and assign the new value back to y.
  - y += 5 is a shorthand assignment operator, often preferred for its conciseness.
  - y = y + 5 is an explicit assignment.
  - In terms of performance for simple types, there is usually no significant difference.
- (ii) y == 10 and y = 10:
  - y == 10: This is a comparison operator (equality operator). It checks if the value of y is equal to 10 and returns a boolean value (True or False). It does not change the value of y.
  - y = 10: This is an assignment operator. It assigns the value 10 to the variable y. It changes the value of y.
- (iii) for j in range (1, 6) : if (j % 2 = = 1) : continue print (j) and for j in range (1, 6) : if (j % 2 = = 1) : break print (j):
  - First block (using continue):
    - This loop iterates j from 1 to 5.

- if (j % 2 == 1) checks if j is odd.
- If j is odd, continue skips the rest of the current iteration and moves to the next one.
- If j is even, print(j) is executed.
- Output:
  - 2
  - 4

- Second block (using break):
  - This loop iterates j from 1 to 5.
  - if (j % 2 == 1) checks if j is odd.
  - If j is odd, break immediately terminates the entire loop.
  - If j is even, print(j) is executed.
  - Output:
    - (No output, because when j is 1, it's odd, break is executed, and the loop terminates before any print(j) is reached.)

2. (a) Write a function named seriesSum ( ) that takes an integer n and x as input parameters and returns the sum of the first n terms of the following series : $x^1/1 ! - x^2/2 ! + x^3/3 ! - x^4/4 ! + ...x^n/n !$

- ```
  import math

  def seriesSum(n, x):
      total_sum = 0
      for i in range(1, n + 1):
          term = (x**i) / math.factorial(i)
          if i % 2 == 0:  # Even terms are subtracted
              total_sum -= term
  ```
  ```

```
        else:  # Odd terms are added
            total_sum += term
    return total_sum
```

(b) What will be the output produced on execution of the following Python statements ?

- (i) print (9 + 5 * 2** 3 ! = 15// 6 - 2)
- (ii) print (64>>2)
- (iii) print ('apple' > 'banana' and 'orange' < 'grape')
- (iv) print ('Hello' * 2)
- (v) print (2 ** 3** 2)

- (i) 9 + 5 * 2**3 != 15 // 6 - 2
  - 2**3 is 8.
  - 5 * 8 is 40.
  - 15 // 6 is 2 (integer division).
  - So, 9 + 40 != 2 - 2
  - 49 != 0
  - Output: True
- (ii) print (64>>2)
  - >> is the right bit shift operator. It shifts the bits of the number to the right by the specified number of places, equivalent to integer division by 2**n.
  - 64 >> 2 is 64 / (2**2) which is 64 / 4 = 16.
  - Output: 16
- (iii) print ('apple' > 'banana' and 'orange' < 'grape')

- String comparison is lexicographical.
- 'apple' > 'banana' is False (because 'a' is not greater than 'b').
- 'orange' < 'grape' is False (because 'o' is not less than 'g').
- False and False is False.
- Output: False

- (iv) print ('Hello' * 2)
  - String multiplication repeats the string.
  - Output: HelloHello
- (v) print (2 ** 3** 2)
  - Exponentiation (**) has right-to-left associativity.
  - First, 3**2 is calculated, which is 9.
  - Then, 2**9 is calculated, which is 512.
  - Output: 512

(c) Write a function named as oddWord ( ) that takes a string as an input parameter and returns a new string with every word of odd length replaced with the length of the corresponding word.

```
def oddWord(input_string):
    words = input_string.split()
    new_words = []
    for word in words:
        if len(word) % 2 != 0:  # Check if word length is odd
            new_words.append(str(len(word)))
        else:
            new_words.append(word)
    return ' '.join(new_words)
```

3. (a) Identify the line number where an exception may be raised on execution. Also specify the reason for the exception. marksLst = eval ('('Input (Enter list of marks in three subjects : ')) # Line 1 maxMarks = int (input ('Enter maximum marks per subject : ')) # Line 2 marksObtained = marksLst1 + marksLst8 + marksLst2 # Line 3 result = marksObtained/(3*maxMarks) # Line 4 print (result) # Line 5

- Line 1: marksLst = eval ('('Input (Enter list of marks in three subjects : '))
    - Potential Exception: SyntaxError or NameError.
    - Reason: The eval function is being used with a malformed string '('Input (Enter list of marks in three subjects : ')). It should likely be input(...) or eval(input(...)). If Input is intended as a function call, it's not defined by default, leading to NameError. If the user inputs something that eval cannot parse, it could lead to SyntaxError. The double opening parenthesis '((' is also syntactically incorrect. Assuming the intent was to take a list as input using eval(input(...)). Even then, if the input is not a valid Python literal (e.g., [1, 2, 3]), it will raise an error.
- Line 3: marksObtained = marksLst1 + marksLst8 + marksLst2
    - Potential Exception: TypeError or IndexError.
    - Reason:
        - If marksLst is a list, marksLst1, marksLst8, marksLst2 are incorrect ways to access elements (should be marksLst[0], marksLst[1], etc.). This would lead to a TypeError because you're trying to use integers as attributes of a list.
        - If it somehow parses to a tuple/list and the elements 1, 8, 2 are intended as indices, then IndexError would occur if the list/tuple does not have those indices (e.g., if it's [m1, m2, m3], it only has indices 0, 1, 2).

    - Given the common use case, marksLst[0], marksLst[1], marksLst[2] would be the correct way to access elements from marksLst.

- Line 4: result = marksObtained/(3*maxMarks)
    - Potential Exception: ZeroDivisionError.
    - Reason: If maxMarks is 0, then 3 * maxMarks will be 0, leading to division by zero.

(b) Write a function isComposite ( ) that takes an integer as an input parameter and returns True if the number is composite (i.e., not a prime number and greater than 1) and False otherwise.

```
def isComposite(number):
    if number <= 1:
        return False  # Composite numbers are greater than 1
    if number <= 3:
        return False  # 2 and 3 are prime, not composite
    if number % 2 == 0 or number % 3 == 0:
        return True  # Divisible by 2 or 3 (and > 3), so composite

    i = 5
    while i * i <= number:
        if number % i == 0 or number % (i + 2) == 0:
            return True
        i += 6
    return False # If no divisors found, it's prime, so not composite
```

(c) Write a program that takes a list of integers as input from the user and generates a corresponding cumulative list where each element in the resultant list at index 'i' is the sum of all integers at index J <= i.

```
# Take input from the user for the list of integers
input_str = input("Enter a list of integers (e.g., 1 2 3 4): ")
try:
    input_list = list(map(int, input_str.split()))
```

```
except ValueError:
    print("Invalid input. Please enter a list of integers separated by spaces.")
    exit()

cumulative_list = []
current_sum = 0
for number in input_list:
    current_sum += number
    cumulative_list.append(current_sum)

print("Original list:", input_list)
print("Cumulative list:", cumulative_list)
```

4. (a) Consider the following function calculateSpeed ( ) that calculates speed using the formula speed = distance/time : def calculateSpeed (distance, time) : try : speed = distance/time except ZeroDivissionError : print ('ZeroDivisionError') except TypeError : print ('TypeError') except ValueError : print ('ValueError') except : print ('An unexpected error occurred.') else : print ('Speed :', speed, 'm/s') finally : print ('Execution completed.') What will be the output produced on the execution of the following statements ?
    - (i) calculateSpeed (100, 0)
    - (ii) calculateSpeed (100, '20')

- (i) calculateSpeed (100, 0):
    - Output:
        - ZeroDivisionError
        - Execution completed.
    - Explanation: When time is 0, a ZeroDivisionError occurs in speed = distance/time. The except ZeroDivissionError (note the typo in the original code, assuming it's ZeroDivisionError) block

is executed, printing "ZeroDivisionError". The else block is skipped. The finally block is always executed, printing "Execution completed.".

- (ii) calculateSpeed (100, '20'):
    - Output:
        - TypeError
        - Execution completed.
    - Explanation: When time is a string '20', the operation distance/time (i.e., 100 / '20') attempts to divide an integer by a string, which raises a TypeError. The except TypeError block is executed, printing "TypeError". The else block is skipped. The finally block is always executed, printing "Execution completed.".

(b) Consider the following list representing product details : productList = [['Laptop', 800], ['Smartphone', 500], ['Tablet', 300]]

- (i) Write a Python code segment to make a shallow copy, named, copyProducts, of productList.
- (ii) What will be the output produced on execution of the following Python code segment ? copyProducts11 = 55012 copyProducts8 = ['Smartwatch', 250]12 print (productList)12 print (copyProducts)12

- (i) copyProducts = productList[:]
    - Or: copyProducts = list(productList)
- (ii) copyProducts11 = 550
    - copyProducts8 = ['Smartwatch', 250]
    - print (productList)
    - print (copyProducts)

- Output:
  - [['Laptop', 800], ['Smartphone', 500], ['Tablet', 300]]
  - [['Laptop', 800], ['Smartphone', 500], ['Tablet', 300]]
- Explanation: The lines copyProducts11 = 550 and copyProducts8 = ['Smartwatch', 250] seem to be typos or incorrect syntax attempting to access elements like copyProducts[1][1] or copyProducts[0].
  - copyProducts11 = 550 would attempt to create a variable named copyProducts11, not modify an element within copyProducts.
  - copyProducts8 = ['Smartwatch', 250] would attempt to create a variable named copyProducts8, not modify an element within copyProducts.
  - Since copyProducts itself is not modified (only new variables with similar names are created), both productList and copyProducts will retain their original contents from the shallow copy operation.
- Assuming the intent was to modify elements of the copyProducts list:
  - copyProducts[1][1] = 550 (modifies 'Smartphone' price)
  - copyProducts[0] = ['Smartwatch', 250] (replaces 'Laptop' entry)
  - If the original lines meant copyProducts[1][1] = 550 and copyProducts[0] = ['Smartwatch', 250], then:
    - productList would be [['Smartwatch', 250], ['Smartphone', 550], ['Tablet', 300]] (due to shallow copy, changes to mutable nested objects affect both lists, but reassignment of top-level elements only affects the copy).

- copyProducts would be [['Smartwatch', 250], ['Smartphone', 550], ['Tablet', 300]]

- Given the exact code as written: copyProducts11 = 550 creates a new variable. copyProducts8 = ['Smartwatch', 250] creates another new variable. Neither modifies the copyProducts list. Therefore, the original contents remain.

(c) Find all the errors (if any) in the following Python code segments :

- (i) f = open ('recordl', 'r') f.write (Weather is great today') f. close ( )
- (ii) name = 'Mohinder Amarnath' name [-5] = 'u' lastChar = name [len (name) -1]
- (iii) studentMarks = [1001, 'Rohan', 90, 85, 99, 50, 99]] studentMakrs8 = 95 print (max (studentMarks))

- (i) f = open ('recordl', 'r') f.write (Weather is great today') f. close ( )
  - Errors:
    - IOError / UnsupportedOperation: You opened the file in read mode ('r'). You cannot write to a file opened in read mode. To write, it should be opened in write mode ('w') or append mode ('a').
    - SyntaxError: 'Weather is great today' is a string literal, but it's not enclosed in parentheses if it's meant to be an argument to f.write(). It should be f.write('Weather is great today').
- (ii) name = 'Mohinder Amarnath' name [-5] = 'u' lastChar = name [len (name) -1]
  - Errors:
    - TypeError: Strings in Python are immutable. You cannot change a character at a specific index using assignment

like name[-5] = 'u'. If you want to change a character, you need to create a new string.

- No error in lastChar = name[len(name) - 1]. This correctly gets the last character.

- (iii) studentMarks = [1001, 'Rohan', 90, 85, 99, 50, 99]] studentMakrs8 = 95 print (max (studentMarks))
  - Errors:
    - SyntaxError: The list studentMarks has an extra closing square bracket]]. It should be [1001, 'Rohan', 90, 85, 99, 50, 99].
    - TypeError: studentMarks contains mixed data types (integers and strings). The max() function cannot compare an integer with a string (e.g., 99 vs 'Rohan'). This will raise a TypeError.
    - NameError: studentMakrs8 is a typo for studentMarks[8]. Even if it were studentMarks[8], it would be an IndexError as the list only has 7 elements (indices 0-6). The line studentMakrs8 = 95 creates a new variable, it does not modify the list.

5. (a) Write a function named as printPattern ( ) that accepts the number of rows n as an input parameter and prints the pattern comprising of nrows of the following for- mat (say, for n = 5) : 1 12 123 1234 12345

- ```
  def printPattern(n):
      for i in range(1, n + 1):
          for j in range(1, i + 1):
              print(j, end='')
          print()
  ```

(b) Write a program that takes a list of numbers as input from the user and creates a list of squares of all the positive even numbers using list comprehension method.
  ```

- ```
  # Take input from the user for the list of numbers
  input_str = input("Enter a list of numbers (e.g., 1 2 3 4 5 6): ")
  try:
      numbers = list(map(int, input_str.split()))
  except ValueError:
      print("Invalid input. Please enter a list of numbers separated by spaces.")
      exit()

  # Create a list of squares of positive even numbers using list comprehension
  squares_of_positive_evens = [num**2 for num in numbers if num > 0 and num % 2 == 0]

  print("Original list:", numbers)
  print("Squares of positive even numbers:", squares_of_positive_evens)
  ```


(c) Consider the following string : players = "kohli and rohit play great game " Write the output produced on execution of the following function calls :

  - (i) players.rfind ('l')
  - (ii) players.swapcase ( )
  - (iii) players.lstrip ( )
  - (iv) players.endswith ('!!')
  - (v) players.replace ('Great', 'outstanding')

- (i) players.rfind ('l')
  - rfind() returns the highest index in the string where substring 'l' is found.
  - Output: 21 (index of 'l' in 'play')
  ```

- (ii) players.swapcase ( )
    - swapcase() converts all uppercase characters to lowercase and vice versa.
    - Output: KOHLI AND ROHIT PLAY GREAT GAME
- (iii) players.lstrip ( )
    - lstrip() removes leading whitespace. The string has no leading whitespace.
    - Output: kohli and rohit play great game
- (iv) players.endswith ('!!')
    - endswith() checks if the string ends with the specified suffix.
    - Output: False
- (v) players.replace ('Great', 'outstanding')
    - replace() replaces occurrences of a substring. Case-sensitive. 'Great' (with a capital G) is not in the string.
    - Output: kohli and rohit play great game (The string remains unchanged as 'Great' with capital 'G' is not found.)

6. (a) Write a function that reads the file report. txt and copies even numbered lines to file evenfile.txt and odd numbered lines to file oddfile.txt.

- ```
  def split_lines_by_parity(input_filename="report.txt", even_filename="evenfile.txt", odd_filename="oddfile.txt"):
      try:
          with open(input_filename, 'r') as infile:
              with open(even_filename, 'w') as evenfile:
                  with open(odd_filename, 'w') as oddfile:
                      line_number = 1
                      for line in infile:
  ```
  ```

```
                if line_number % 2 == 0:
                    evenfile.write(line)
                else:
                    oddfile.write(line)
                line_number += 1
        print(f"Lines from '{input_filename}' successfully split into
'{even_filename}' and '{odd_filename}'.")
    except FileNotFoundError:
        print(f"Error: The file '{input_filename}' was not found.")
    except IOError as e:
        print(f"An I/O error occurred: {e}")
```

(b) Consider the following three sets : vehicles = {'Bicycle', 'Scooter', 'Car', 'Bike', 'Truck', 'Bus', 'Tempo Traveller', 'Rickshaw'} heavyVehicles = {'Truck', 'Bus', 'Tempo Traveller'} lightVehicles = {'Rickshaw', 'Scooter', 'Bike'} Write Python statements to perform the following operations on the given sets :

- (i) Add the transport 'Bicycle' to set lightVehicles.
- (ii) Remove the transport 'Tempo Traveller' from the set heavyVehicles.
- (iii) Determine the set of average weight vehicles which are neither heavy weight nor light weight.
- (iv) Determine the number of vehicles.

- (i) lightVehicles.add('Bicycle')
- (ii) heavyVehicles.remove('Tempo Traveller')
- (iii) averageVehicles = vehicles - heavyVehicles - lightVehicles
  - Or: averageVehicles = vehicles.difference(heavyVehicles).difference(lightVehicles)
- (iv) num_vehicles = len(vehicles)

7. (a) Consider the following function : def addition (num1, num2 = 5, num3 = 38) : return num1 + num2 + num3 What will be the output returned on execution of the following function calls :
    - (i) addition (num2 = 15, num 1 = 47)
    - (ii) addition (29)
    - (iii) addition ( )

- (i) addition (num2 = 15, num 1 = 47)
    - num1 becomes 47.
    - num2 becomes 15 (overriding the default).
    - num3 remains 38 (default value).
    - Result: $47 + 15 + 38 = 100$
    - Output: 100
- (ii) addition (29)
    - num1 becomes 29.
    - num2 remains 5 (default value).
    - num3 remains 38 (default value).
    - Result: $29 + 5 + 38 = 72$
    - Output: 72
- (iii) addition ( )
    - TypeError: This will raise a TypeError because num1 is a required positional argument and no value is provided for it.
    - Output: TypeError: addition() missing 1 required positional argument: 'num1'

(b) Define a class ComplexNumber that represents complex numbers and supports basic arithmetic operations using operator overloading. The class should contain the following data members :

- real – The real part of the complex number
- imag – The imaginary part of the complex number
- (i) The class should support the following methods :
  - **init**() to overload the + operator, allowing addition of two complex numbers.
  - **str**() that returns the complex number in the form a + bi or a – bi based on the values of real and imag.
- (ii) Also write Python statements for the following :
  - Create an object C1 of the ComplexNumber class with the values real = 3 and imag = 4 to represent the complex number 3+4i.
  - Create an object C2 of the ComplexNumber class with the values real = 4 and imag = 5 to represent the complex number 4 + 5i.
  - Add the complex numbers referred by C1 and C2 and assign the result to C3.
  - Display the complex number C3 using the **str**() method.

- (i) Class Definition:
  - class ComplexNumber:

```
def __init__(self, real, imag):
    self.real = real
    self.imag = imag

def __add__(self, other):
    # Overloads the + operator for addition
```

```
        new_real = self.real + other.real
        new_imag = self.imag + other.imag
        return ComplexNumber(new_real, new_imag)

    def __str__(self):
        # Returns the complex number in 'a + bi' or 'a - bi' format
        if self.imag >= 0:
            return f"{self.real} + {self.imag}i"
        else:
            return f"{self.real} - {abs(self.imag)}i"
```

- (ii) Python Statements:
    - Create an object C1:
        - C1 = ComplexNumber(3, 4)
    - Create an object C2:
        - C2 = ComplexNumber(4, 5)
    - Add C1 and C2, assign to C3:
        - C3 = C1 + C2
    - Display C3:
        - print(C3)