### 

**NAME:** Neev Shah

**USN:** 22BTRCL107

**CLASS:** AIML-C

**DATE:** 24th April 2024

# 

# **ASSIGNMENT on python**

**1. Create a Python script that takes a student's score (0-100) as input and prints their grade based**

**on the following criteria:**

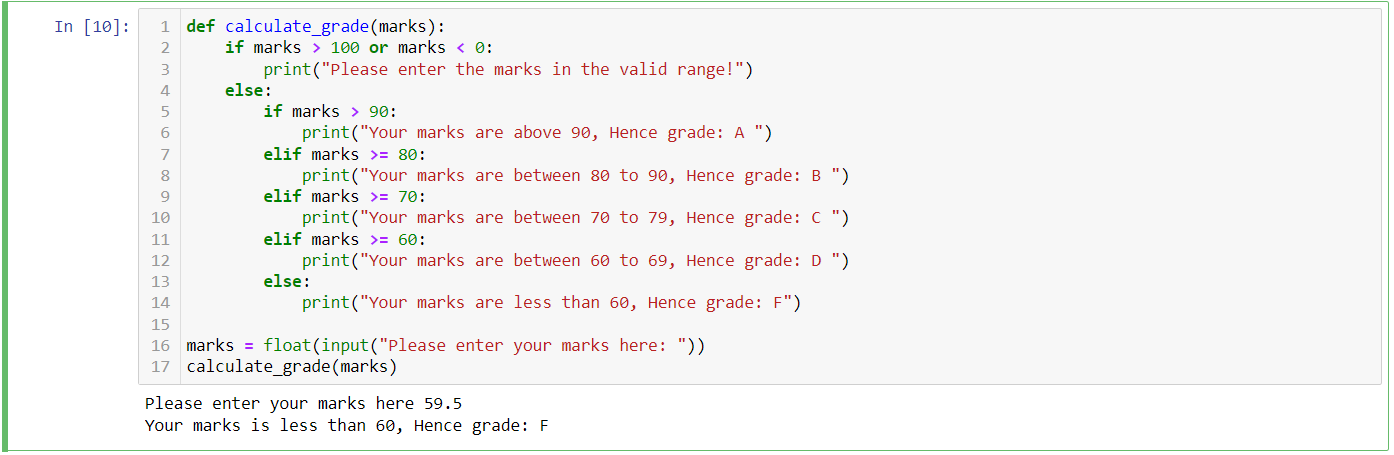
**Above 90: "Grade: A"**

**80 to 90: "Grade: B"**

**70 to 79: "Grade: C"**

**60 to 69: "Grade: D"**

**Below 60: "Grade: F"**



**2. Create a Python program that applies a discount to a purchase based on the amount spent. The**

**program asks for the total amount and applies the following discount rates:**

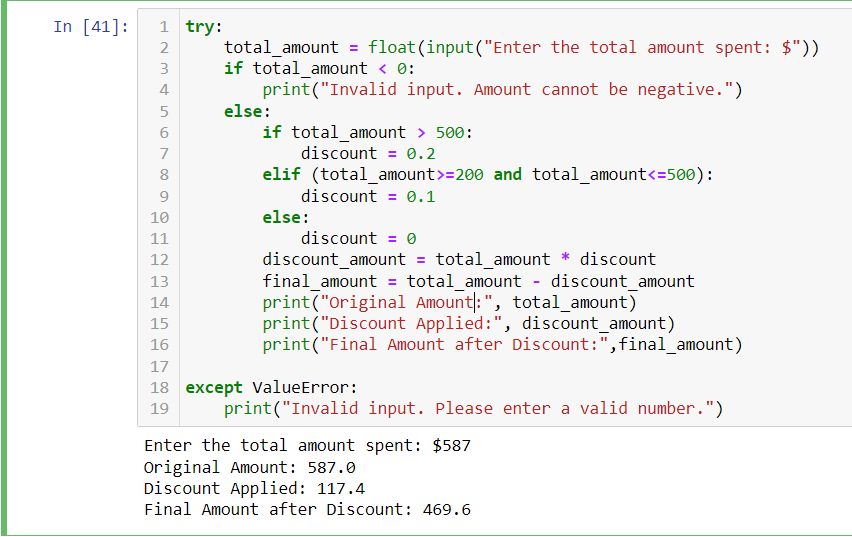
**Spend over $500: 20% discount**

**Spend $200 - $500: 10% discount**

**Spend below $200: No discount**

**The program should print the original amount, the discount applied, and the final amount after**

**the discount.**



**3. Create a program that asks for the user's birth month and day and then tells them their zodiac**

**sign. For simplicity, you can use the following date ranges:**

**Aries: March 21 - April 19**

**Taurus: April 20 - May 20**

**Gemini: May 21 - June 20**

**Cancer: June 21 - July 22**

**Leo: July 23 - August 22**

**Virgo: August 23 - September 22**

**Libra: September 23 - October 22**

**Scorpio: October 23 - November 21**

**Sagittarius: November 22 - December 21**

**Capricorn: December 22 - January 19**

**Aquarius: January 20 - February 18**

**Pisces: February 19 - March 20**

**Make sure to handle invalid inputs gracefully.**



**4. Write a Python program to check the validity of a password entered by the user. The password**

**is considered valid if it meets the following criteria:**

**At least 1 letter between [a-z] and 1 letter between [A-Z].**

**At least 1 number between [0-9].**

**At least 1 character from [$#@].**

**Minimum length of 6 characters.**

**Maximum length of 16 characters.**

**The program should print whether the password is valid or not based on these criteria.**

**A screenshot of a computer program

Description automatically generated**

**5. Implement a simple number guessing game. First, set a target number within a certain range**

**(e.g., 1 to 100). Then, using a while loop, ask the user to guess the number. Provide feedback for**

**each guess ("too high" or "too low"). The game ends when the user guesses the number**

**correctly. Use a break statement to exit the loop once the correct number is guessed.**

A screenshot of a computer program

Description automatically generated

**6. Write a Python program that asks the user to enter a range (start and end numbers). Use a for**

**loop to iterate through this range, and for each number, check if it is a prime number. If it is,**

**print the number. Use the continue statement to skip non-prime numbers efficiently.**

A screenshot of a computer

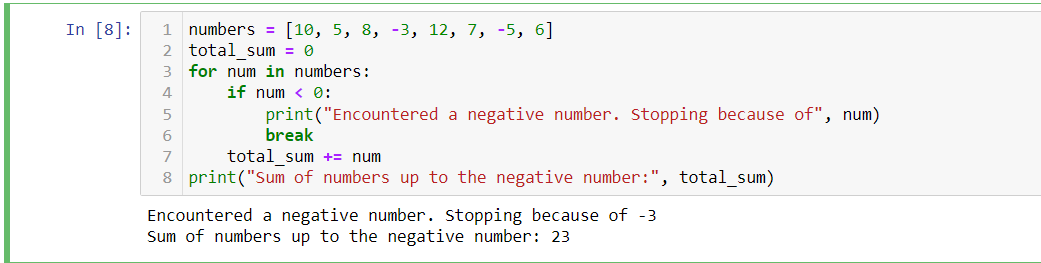
Description automatically generated

**7. Create a Python program that iterates through a list of numbers (you can define the list in the**

**code) and calculates the sum of the numbers. However, if the program encounters a number**

**that is negative, it should stop adding any further numbers (i.e., break out of the loop) and print**

**the current sum up to that point.**



**8. Write a Python program to print the following patterns**

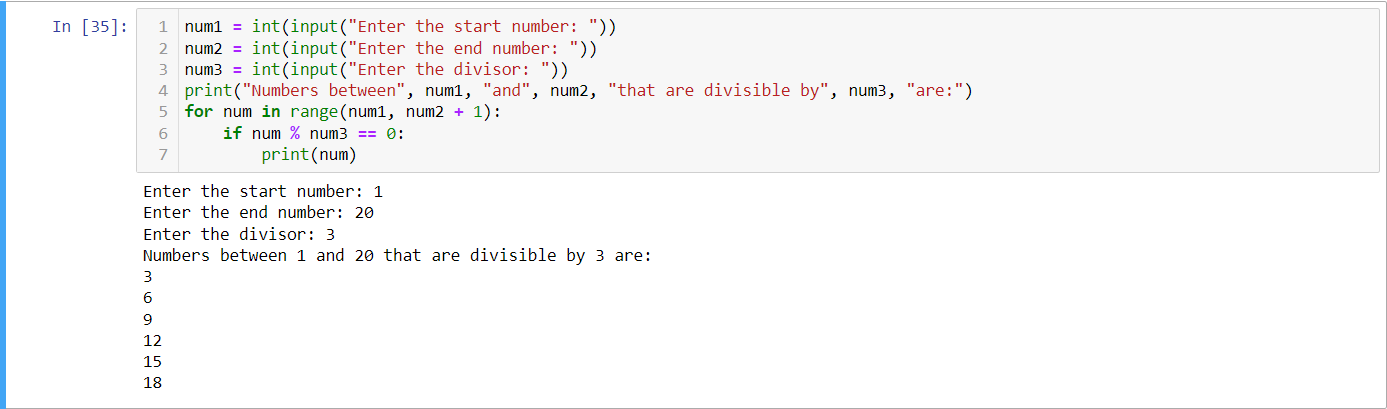
A white background with black text

Description automatically generatedA screenshot of a computer program

Description automatically generated

**9. Create a program that asks for two numbers and prints all the numbers between them that are**

**divisible by a third number asked from the user.**



**10. Write a recursive function named reverse\_string that takes a string as input and returns**

**its reverse. The function must use recursion to accomplish this task and should not use**

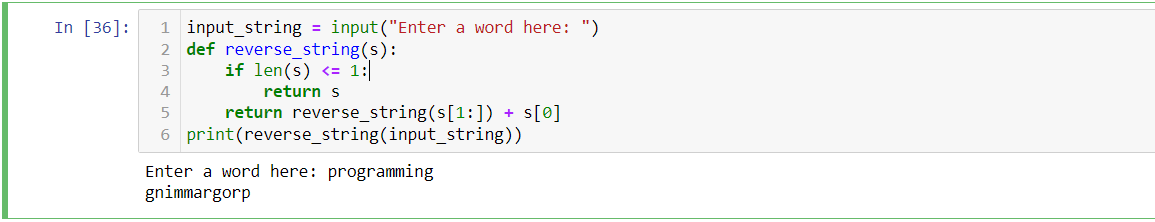
**any loops or slicing ([::-1]).**

**Example Usage:**

**print(reverse\_string("hello"))**

**Expected Output:**

**"olleh"**

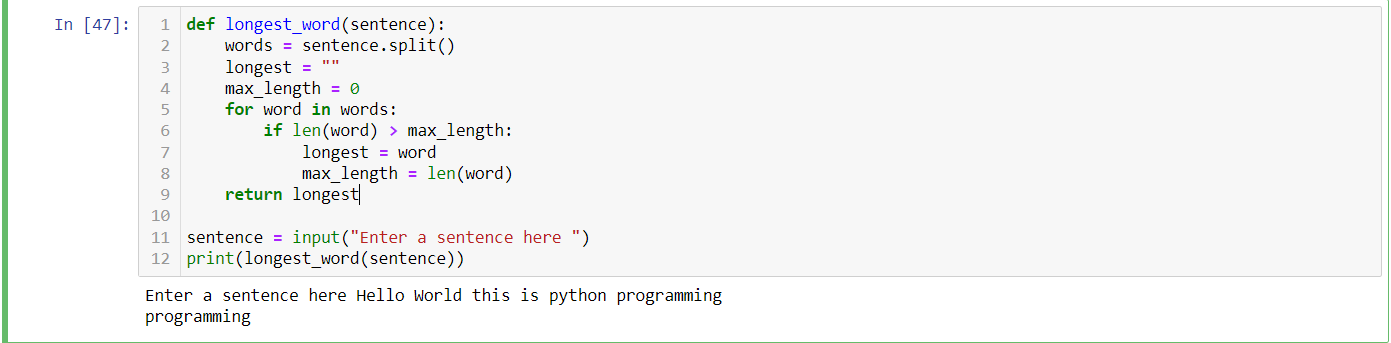


**11. Create a function longest\_word(sentence) that finds and returns the longest word in the**

**given string sentence. If there are multiple words of the same length, return the first one**

**encountered.**

**# Example: longest\_word("I love programming") should return "programming"**



**12. Create a Python function named custom\_sort that takes a list of tuples where each tuple**

**contains a name and a score. The function should return a new list sorted by scores in**

**descending order. If two tuples have the same score, they should be sorted**

**alphabetically by name in ascending order. Test your function with a predefined list of**

**tuples and print the sorted list.**

**Sample Input: [('Alice', 88), ('Bob', 95), ('Charlie', 88), ('Dave', 95)]**

**Sample Output: [('Bob', 95), ('Dave', 95), ('Alice', 88), ('Charlie', 88)]**

A screenshot of a computer

Description automatically generated

**13. Develop a Python function named transform\_string that takes a string and performs the**

**following transformations: it capitalizes every other letter starting with the first character**

**(ignoring non-letter characters for the alternation pattern), and it replaces spaces with**

**hyphens (-). For example, hello world becomes HeLlO-WoRlD. After defining the**

**function, ask the user for a string and print its transformation.**

**Sample Input: hello world**

**Sample Output: HeLlO-WoRlD**

A screenshot of a computer

Description automatically generated

**14. Create a function named simulate\_file\_renaming that takes two parameters: a list of**

**filenames (as strings) and a new name template (a string containing a placeholder for a**

**number, e.g., image\_##). The function should return a list of strings representing the**

**new filenames where the placeholder is replaced by an incremental number, starting**

**from 1 and formatted to have leading zeros if necessary, according to the placeholder's**

**length. For instance, renaming ['a.jpg', 'b.jpg', 'c.jpg'] with the template photo\_### would**

**result in ['photo\_001.jpg', 'photo\_002.jpg', 'photo\_003.jpg']. This exercise simulates the**

**renaming process, so you should only return the renamed list without actually renaming**

**any files.**

**Sample Input: ['a.jpg', 'b.jpg', 'c.jpg'], photo\_###**

**Sample Output: ['photo\_001.jpg', 'photo\_002.jpg', 'photo\_003.jpg']**

A screenshot of a computer code

Description automatically generated

**15. You are given a list of words. Write a Python function called group\_anagrams that groups all**

**anagrams together and returns them as a list of lists.**

**Two words are considered anagrams if they contain the same characters but in a different order.**

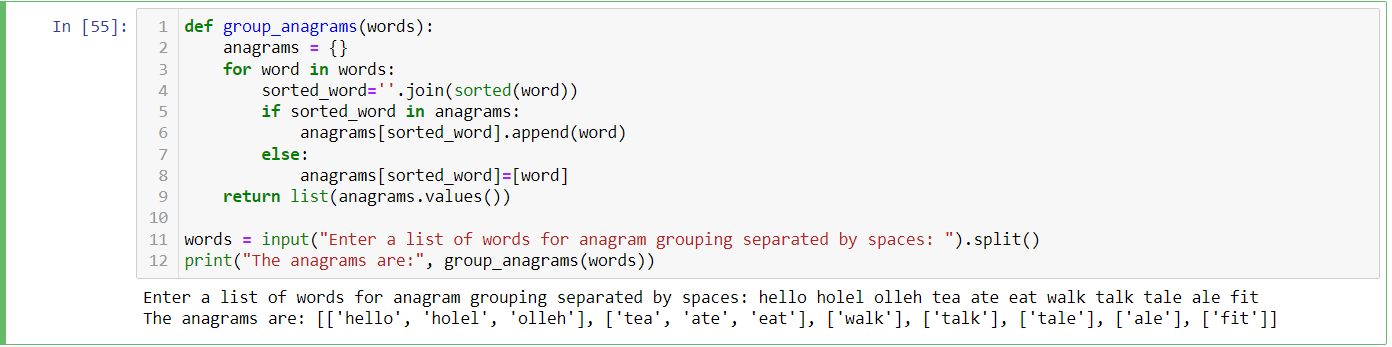
**Examples:**

**Input: ["eat", "tea", "tan", "ate", "nat", "bat"]**

**Output: [["eat", "tea", "ate"], ["tan", "nat"], ["bat"]]**

**Input: ["listen", "silent", "top", "pot", "hello", "world"]**

**Output: [["listen", "silent"], ["top", "pot"], ["hello"], ["world"]]**



**16. You are given a list of integers. Write a Python function called max\_subarray\_sum to find the**

**contiguous subarray within the list that has the largest sum and return that sum.**

**For example, given the list [−2, 1, −3, 4, −1, 2, 1, −5, 4], the contiguous subarray with the largest**

**sum is [4, −1, 2, 1], and the maximum sum is 6.**

**Examples:**

**Input: [-2, 1, -3, 4, -1, 2, 1, -5, 4]**

**Output: 6 (corresponding to the subarray [4, -1, 2, 1])**

**Input: [1, 2, 3, 4, 5]**

**Output: 15 (corresponding to the subarray [1, 2, 3, 4, 5])**

A close-up of a computer screen

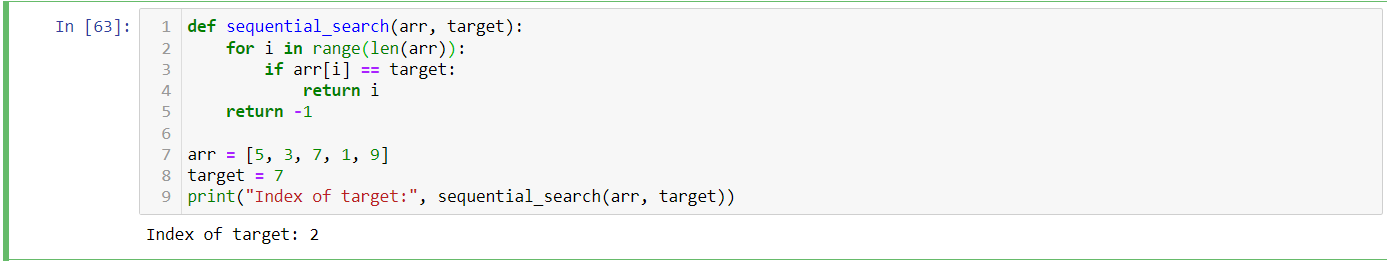
Description automatically generated

**17. Implement a function that performs a sequential search through a list for a specified target value.**

**The function should return the index of the target if found, and -1 if the target is not in the list.**

**Sample Input: ([5, 3, 7, 1, 9], 7)**

**Sample Output: 2**



**18. Design a method to encode a list of strings to a single string and another method to decode it**

**back to a list of strings.**

**The encoded string should be concise and easily decodable. Assume there are no character**

**restrictions for individual strings.**

**Examples:**

**Input: ["hello", "world"]**

**Encoded Output: "5#hello5#world" (or another unique format of your choice)**

**Decoded Output: ["hello", "world"]**

**Input: ["abc", "def", "ghi"]**

**Encoded Output: "3#abc3#def3#ghi"**

**Decoded Output: ["abc", "def", "ghi"]**

