

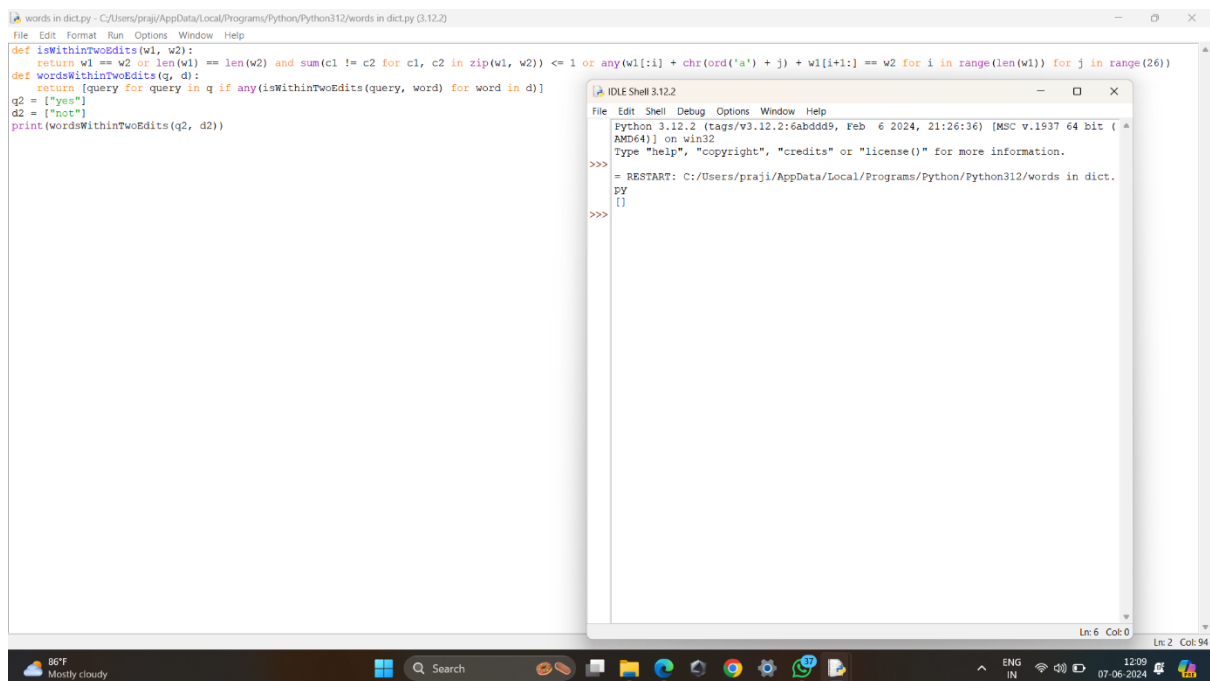
1. Odd String Difference You are given an array of equal-length strings words. Assume that the length of each string is n. Each string words[i] can be converted into a difference integer array difference[i] of length n-1 where  $\text{difference}[i][j] = \text{words}[i][j+1] - \text{words}[i][j]$  where  $0 \leq j \leq n-2$ . Note that the difference between two letters is the difference between their positions in the alphabet i.e. the position of 'a' is 0, 'b' is 1, and 'z' is 25. For example, for the string "acb", the difference integer array is  $[2-0, 1-2] = [2, -1]$ . All the strings in words have the same difference integer array, except one. You should find that string. Return the string in words that has different difference integer array

The image shows a screenshot of a Python IDE (IDLE 3.12.2) with a script titled 'odd string.py'. The script defines a function 'odd(words)' that iterates through a list of words, calculates their difference arrays, and identifies the one that is different. The words list is ['adc', 'vzy', 'abc']. The IDE window shows the script code, and the shell window shows the execution output, which is 'adc'.

```
odd string.py - C:/Users/praji/AppData/Local/Programs/Python/Python312/odd string.py (3.12.2)
File Edit Format Run Options Window Help
def odd(words):
    def con(s):
        return [ord(s[i+1]) - ord(s[i]) for i in range(len(s) - 1)]
    diff_count = {}
    for word in words:
        diff = tuple(con(word))
        diff_count[diff] = diff_count.get(diff, 0) + 1
    for key, value in diff_count.items():
        if value == 1:
            odd_diff = key
            break
    for i, word in enumerate(words):
        if tuple(con(word)) != odd_diff:
            return word
words = ["adc", "vzy", "abc"]
print(odd(words))

IDLE Shell 3.12.2
File Edit Shell Debug Options Window Help
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/praji/AppData/Local/Programs/Python/Python312/odd string.py
adc
>>>
```

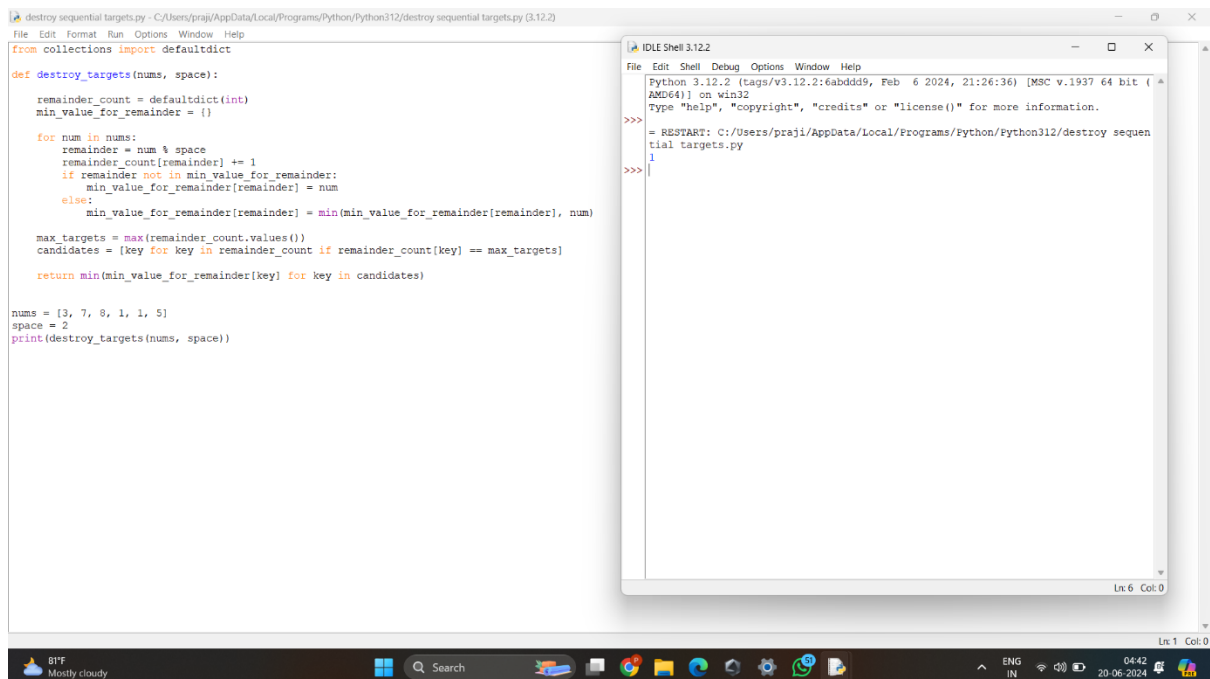
2. Words Within Two Edits of Dictionary You are given two string arrays, queries and dictionary. All words in each array comprise of lowercase English letters and have the same length. In one edit you can take a word from queries, and change any letter in it to any other letter. Find all words from queries that, after a maximum of two edits, equal some word from dictionary. Return a list of all words from queries, that match with some word from dictionary after a maximum of two edits. Return the words in the same order they appear in queries. Example 1: Input: queries = ["word", "note", "ants", "wood"], dictionary = ["wood", "joke", "moat"] Output: ["word", "note", "wood"]



```
words in dict.py - C:/Users/praji/AppData/Local/Programs/Python/Python312/words in dict.py (3.12.2)
File Edit Format Run Options Window Help
def isWithinTwoEdits(w1, w2):
    return w1 == w2 or len(w1) == len(w2) and sum(c1 != c2 for c1, c2 in zip(w1, w2)) <= 1 or any(w1[i:i] + chr(ord('a') + j) + w1[i+1:] == w2 for i in range(len(w1)) for j in range(26))
def wordsWithinTwoEdits(q, d):
    return [query for query in q if any(isWithinTwoEdits(query, word) for word in d)]
q2 = ["yes"]
d2 = ["not"]
print(wordsWithinTwoEdits(q2, d2))

IDLE Shell 3.12.2
File Edit Shell Debug Options Window Help
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> = RESTART: C:/Users/praji/AppData/Local/Programs/Python/Python312/words in dict.py
>>> []
>>>
```

3. Destroy Sequential Targets You are given a 0-indexed array `nums` consisting of positive integers, representing targets on a number line. You are also given an integer `space`. You have a machine which can destroy targets. Seeding the machine with some `nums[i]` allows it to destroy all targets with values that can be represented as `nums[i] + c * space`, where `c` is any non-negative integer. You want to destroy the maximum number of targets in `nums`. Return the minimum value of `nums[i]` you can seed the machine with to destroy the maximum number of targets



The screenshot shows a Python IDE with two windows. The main window displays a Python script for the 'Destroy Sequential Targets' problem. The script defines a function `destroy_targets` that takes a list of numbers `nums` and an integer `space`. It uses a dictionary to count the number of targets for each remainder when divided by `space`. The function then finds the remainder with the maximum count and returns the minimum value in `nums` that has that remainder. The script also includes a test case with `nums = [3, 7, 8, 1, 1, 5]` and `space = 2`.

```
def destroy_targets(nums, space):
    remainder_count = defaultdict(int)
    min_value_for_remainder = {}

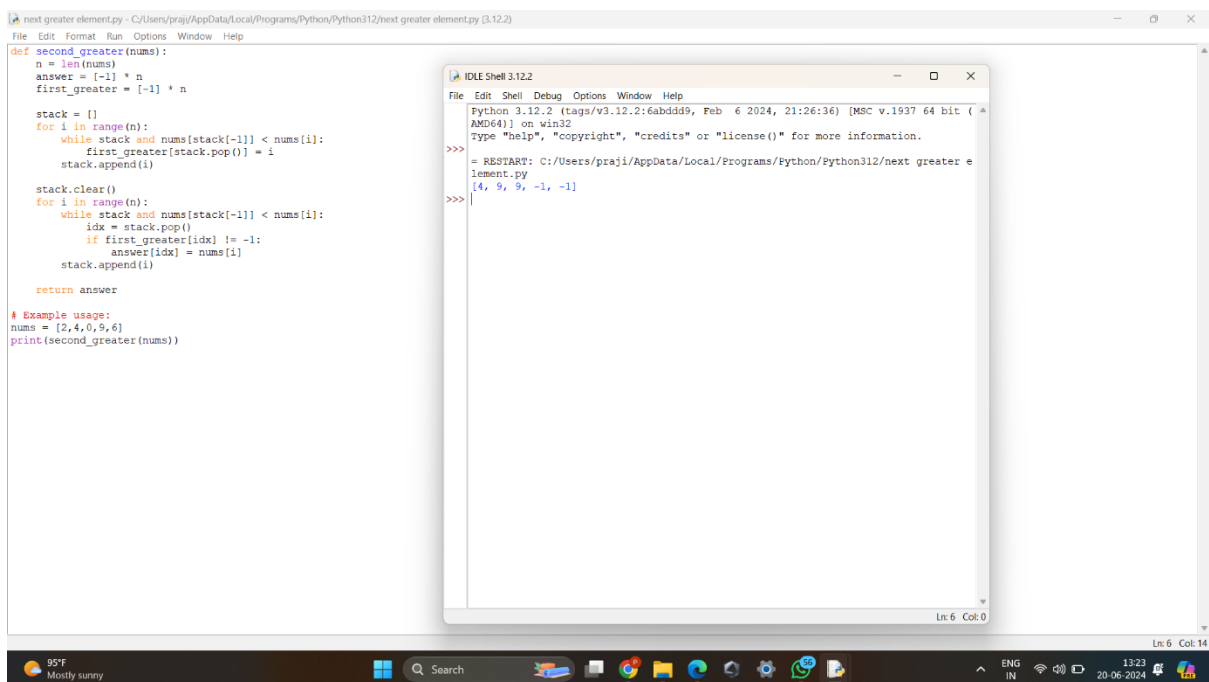
    for num in nums:
        remainder = num % space
        remainder_count[remainder] += 1
        if remainder not in min_value_for_remainder:
            min_value_for_remainder[remainder] = num
        else:
            min_value_for_remainder[remainder] = min(min_value_for_remainder[remainder], num)

    max_targets = max(remainder_count.values())
    candidates = [key for key in remainder_count if remainder_count[key] == max_targets]
    return min(min_value_for_remainder[key] for key in candidates)

nums = [3, 7, 8, 1, 1, 5]
space = 2
print(destroy_targets(nums, space))
```

The second window, titled 'IDLE Shell 3.12.2', shows the output of the script, which is the number 1, indicating the minimum value of `nums[i]` that can seed the machine to destroy the maximum number of targets.

4. Next Greater Element IV You are given a 0-indexed array of non-negative integers `nums`. For each integer in `nums`, you must find its respective second greater integer. The second greater integer of `nums[i]` is `nums[j]` such that:  $j > i$  • • •  $nums[j] > nums[i]$  There exists exactly one index  $k$  such that  $nums[k] > nums[i]$  and  $i < k < j$ . If there is no such `nums[j]`, the second greater integer is considered to be -1. • For example, in the array `[1, 2, 4, 3]`, the second greater integer of 1 is 4, 2 is 3, and that of 3 and 4 is -1. Return an integer array `answer`, where `answer[i]` is the second greater integer of `nums[i]`



```
next greater element.py - C:/Users/praji/AppData/Local/Programs/Python/Python312/next greater element.py (3.12.2)
File Edit Format Run Options Window Help

def second_greater(nums):
    n = len(nums)
    answer = [-1] * n
    first_greater = [-1] * n

    stack = []
    for i in range(n):
        while stack and nums[stack[-1]] < nums[i]:
            first_greater[stack.pop()] = i
            stack.append(i)

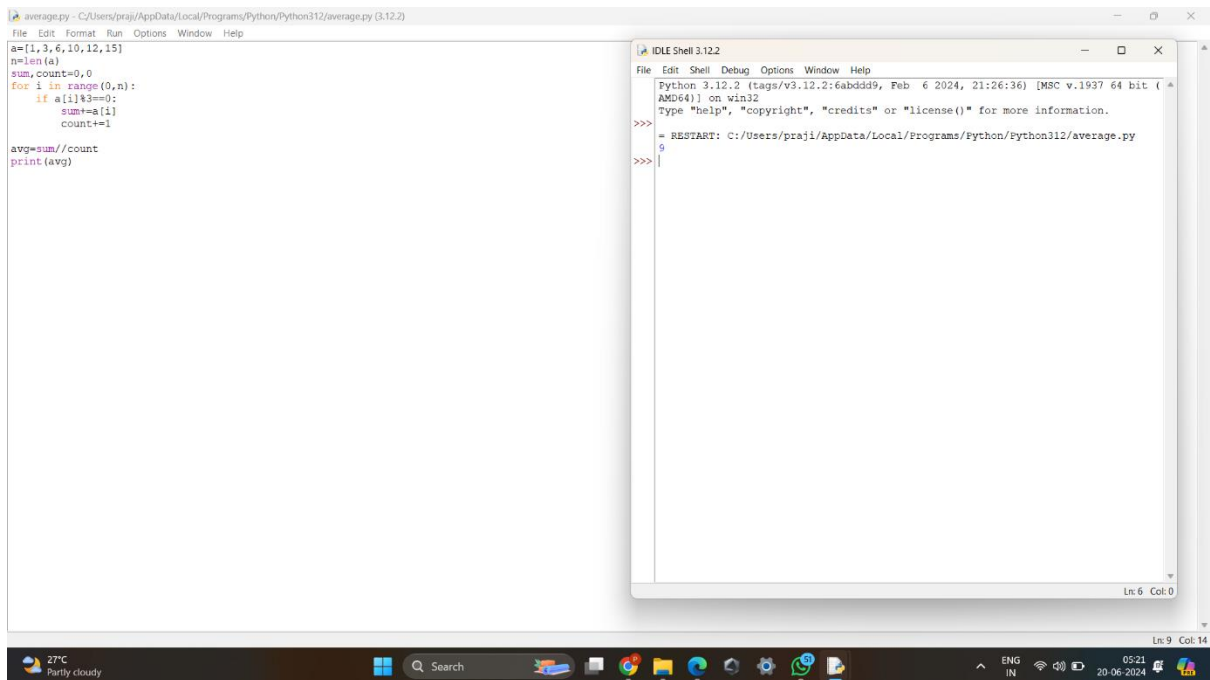
    stack.clear()
    for i in range(n):
        while stack and nums[stack[-1]] < nums[i]:
            idx = stack.pop()
            if first_greater[idx] != -1:
                answer[idx] = nums[i]
            stack.append(i)

    return answer

# Example usage:
nums = [2,4,0,9,6]
print(second_greater(nums))

IDLE Shell 3.12.2
File Edit Shell Debug Options Window Help
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/praji/AppData/Local/Programs/Python/Python312/next greater element.py
[4, 9, 9, -1, -1]
>>>
```

5. Average Value of Even Numbers That Are Divisible by Three Given an integer array nums of positive integers, return the average value of all even integers that are divisible by 3. Note that the average of n elements is the sum of the n elements divided by n and rounded down to the nearest integer.



The screenshot shows a Python IDE with a file named `average.py` and an interactive shell window.

**File Content:**

```
a=[1,3,6,10,12,15]
n=len(a)
sum,count=0,0
for i in range(0,n):
    if a[i]%3==0:
        sum+=a[i]
        count+=1

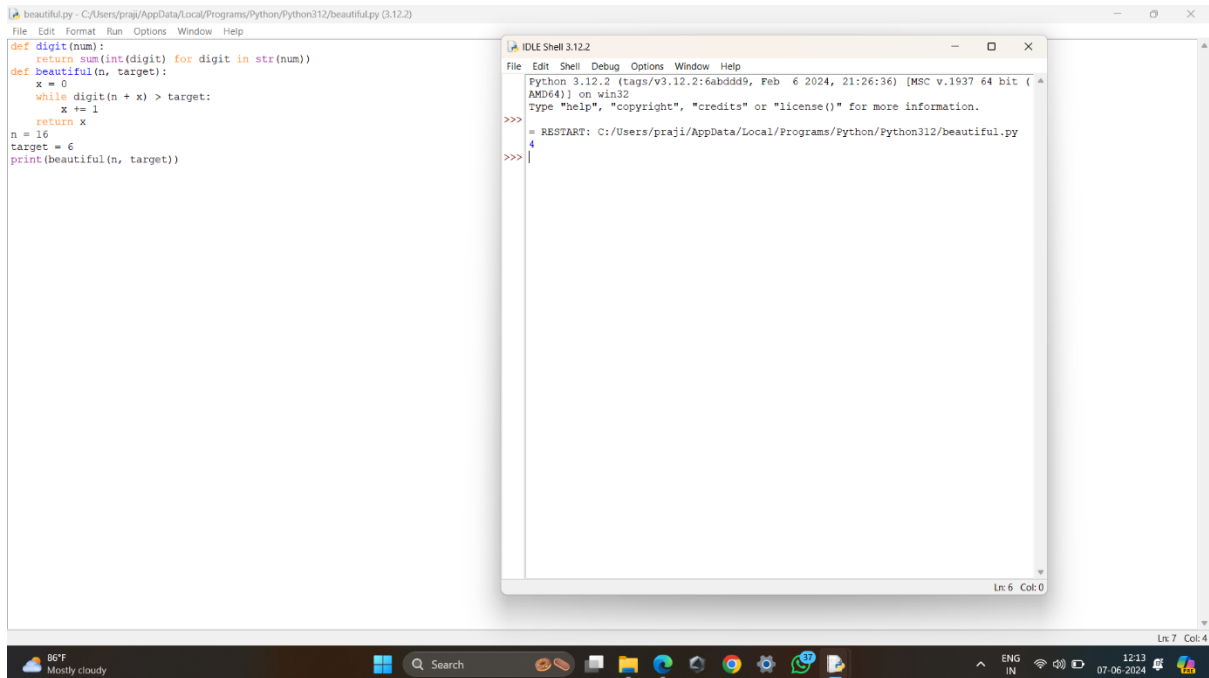
avg=sum//count
print(avg)
```

**Shell Output:**

```
Python 3.12.2 (tags/v3.12.2:6abdd99, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/praji/AppData/Local/Programs/Python/Python312/average.py
9
>>>
```

The task is to calculate the average of even numbers that are divisible by 3 from the array `a`. The code iterates through the array, checks if each element is even and divisible by 3, and calculates the average using integer division.

6. You are given two positive integers  $n$  and  $target$ . An integer is considered beautiful if the sum of its digits is less than or equal to  $target$ . Return the minimum non-negative integer  $x$  such that  $n + x$  is beautiful. The input will be generated such that it is always possible to make  $n$  beautiful



The image shows a screenshot of a Python IDE (likely IDLE) with a file named `beautiful.py` open. The code defines a function `digit` to calculate the sum of digits of a number, and a function `beautiful` to find the minimum non-negative integer  $x$  such that  $n + x$  is beautiful. The code sets  $n = 16$  and  $target = 6$ , and prints the result of `beautiful(n, target)`. A terminal window is also open, showing the Python 3.12.2 shell prompt and the file path.

```
beautiful.py - C:/Users/praji/AppData/Local/Programs/Python/Python312/beautiful.py (3.12.2)
File Edit Format Run Options Window Help
def digit(num):
    return sum(int(digit) for digit in str(num))
def beautiful(n, target):
    x = 0
    while digit(n + x) > target:
        x += 1
    return x
n = 16
target = 6
print(beautiful(n, target))

IDLE Shell 3.12.2
File Edit Shell Debug Options Window Help
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> = RESTART: C:/Users/praji/AppData/Local/Programs/Python/Python312/beautiful.py
>>>
```

7. Most Popular Video Creator You are given two string arrays creators and ids, and an integer array views, all of length n. The ith video on a platform was created by creator[i], has an id of ids[i], and has views[i] views. The popularity of a creator is the sum of the number of views on all of the creator's videos. Find the creator with the highest popularity and the id of their most viewed video.

- If multiple creators have the highest popularity, find all of them.
- If multiple videos have the highest view count for a creator, find the lexicographically smallest id.

Return a 2D array of strings answer where answer[i] = [creatori, idi] means that creatori has the highest popularity and idi is the id of their most popular video. The answer can be returned in any order

```

from collections import defaultdict

def most_popular_creator(creators, ids, views):
    creator_views = defaultdict(int)
    creator_top_video = {}

    for creator, video_id, view in zip(creators, ids, views):
        creator_views[creator] += view

        if creator not in creator_top_video:
            creator_top_video[creator] = (view, video_id)
        else:
            max_view, max_id = creator_top_video[creator]
            if view > max_view or (view == max_view and video_id < max_id):
                creator_top_video[creator] = (view, video_id)

    max_popularity = max(creator_views.values())

    result = []
    for creator, total_views in creator_views.items():
        if total_views == max_popularity:
            result.append([creator, creator_top_video[creator][1]])

    return result

creators1 = ["alice", "bob", "alice", "chris"]
ids1 = ["one", "two", "three", "four"]
views1 = [5, 10, 5, 4]
print(most_popular_creator(creators1, ids1, views1))

creators2 = ["alice", "alice", "alice"]
ids2 = ["a", "b", "c"]
views2 = [1, 2, 2]
print(most_popular_creator(creators2, ids2, views2))

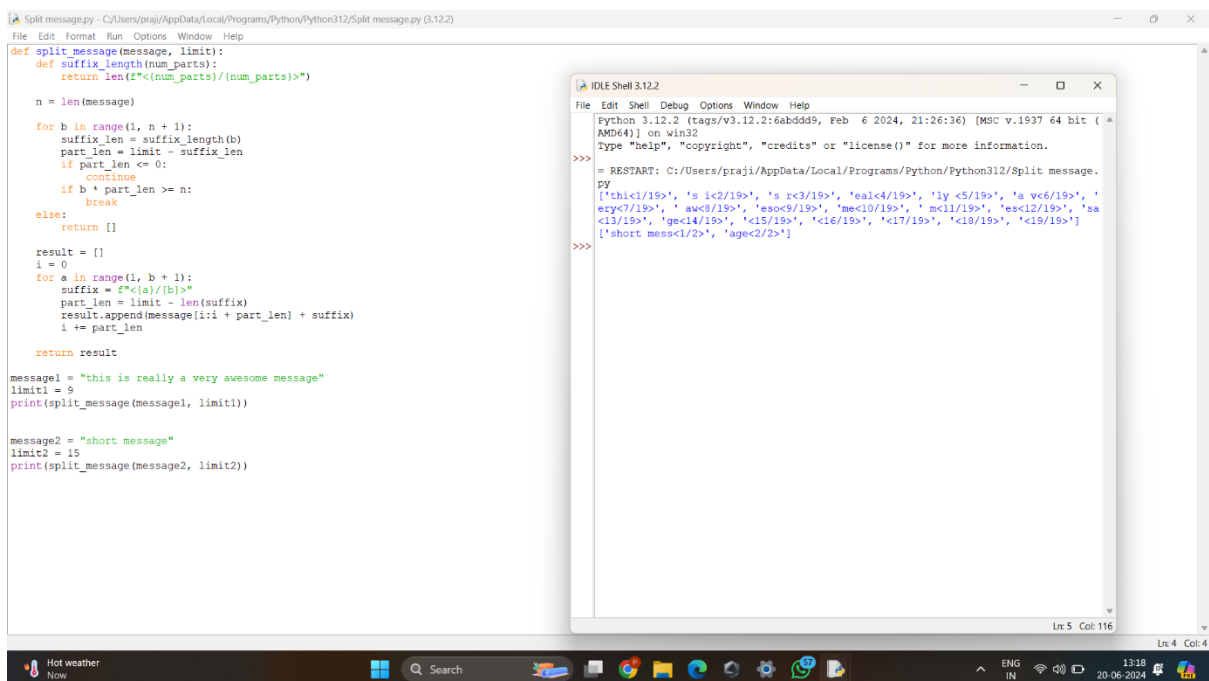
```

```

Python 3.12.2 (tags/v3.12.2:6abdd99, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/praji/AppData/Local/Programs/Python/Python312/video creator.py
>>> [['alice', 'one'], ['bob', 'two']]
>>> [['alice', 'b']]
>>>

```

8. Split Message Based on Limit You are given a string, message, and a positive integer, limit. You must split message into one or more parts based on limit. Each resulting part should have the suffix "", where "b" is to be replaced with the total number of parts and "a" is to be replaced with the index of the part, starting from 1 and going up to b. Additionally, the length of each resulting part (including its suffix) should be equal to limit, except for the last part whose length can be at most limit. The resulting parts should be formed such that when their suffixes are removed and they are all concatenated in order, they should be equal to message. Also, the result should contain as few parts as possible. Return the parts message would be split into as an array of strings. If it is impossible to split message as required, return an empty array



The screenshot shows an IDE window titled "Split message.py - C:/Users/praji/AppData/Local/Programs/Python/Python312/Split message.py (3.12.2)". The code defines a function `split_message(message, limit)` that splits a message into parts of a given limit. It includes a helper function `suffix_length(num_parts)` to calculate the length of the suffix. The main function calculates the number of parts, iterates to split the message, and returns the resulting array. Below the function, two test cases are provided: `message1 = "this is really a very awesome message"` with `limit1 = 9`, and `message2 = "short message"` with `limit2 = 15`. The output window shows the execution results, including the help text for Python 3.12.2 and the output of the function calls: `['thi<1/9>', 's i<2/9>', 's r<3/9>', 'eal<4/9>', 'ly <5/9>', 'a v<6/9>', 'ery<7/9>', 'aw<8/9>', 'esoc<9/9>', 'me<10/9>', 'm<11/9>', 'es<12/9>', 'sa<13/9>', 'ge<14/9>', 'c<15/9>', 'c<16/9>', 'c<17/9>', 'c<18/9>', 'c<19/9>']` and `['short mess<1/2>', 'age<2/2>']`.

```
def split_message(message, limit):
    def suffix_length(num_parts):
        return len(f"<(num_parts)>")

    n = len(message)
    for b in range(1, n + 1):
        suffix_len = suffix_length(b)
        part_len = limit - suffix_len
        if part_len <= 0:
            continue
        if b * part_len >= n:
            break
    else:
        return []

    result = []
    i = 0
    for a in range(1, b + 1):
        suffix = f"<(a)>(<b>)"
        part_len = limit - len(suffix)
        result.append(message[i:i + part_len] + suffix)
        i += part_len

    return result

message1 = "this is really a very awesome message"
limit1 = 9
print(split_message(message1, limit1))

message2 = "short message"
limit2 = 15
print(split_message(message2, limit2))
```

```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> = RESTART: C:/Users/praji/AppData/Local/Programs/Python/Python312/Split message.py
['thi<1/9>', 's i<2/9>', 's r<3/9>', 'eal<4/9>', 'ly <5/9>', 'a v<6/9>', 'ery<7/9>', 'aw<8/9>', 'esoc<9/9>', 'me<10/9>', 'm<11/9>', 'es<12/9>', 'sa<13/9>', 'ge<14/9>', 'c<15/9>', 'c<16/9>', 'c<17/9>', 'c<18/9>', 'c<19/9>']
>>> ['short mess<1/2>', 'age<2/2>']
```