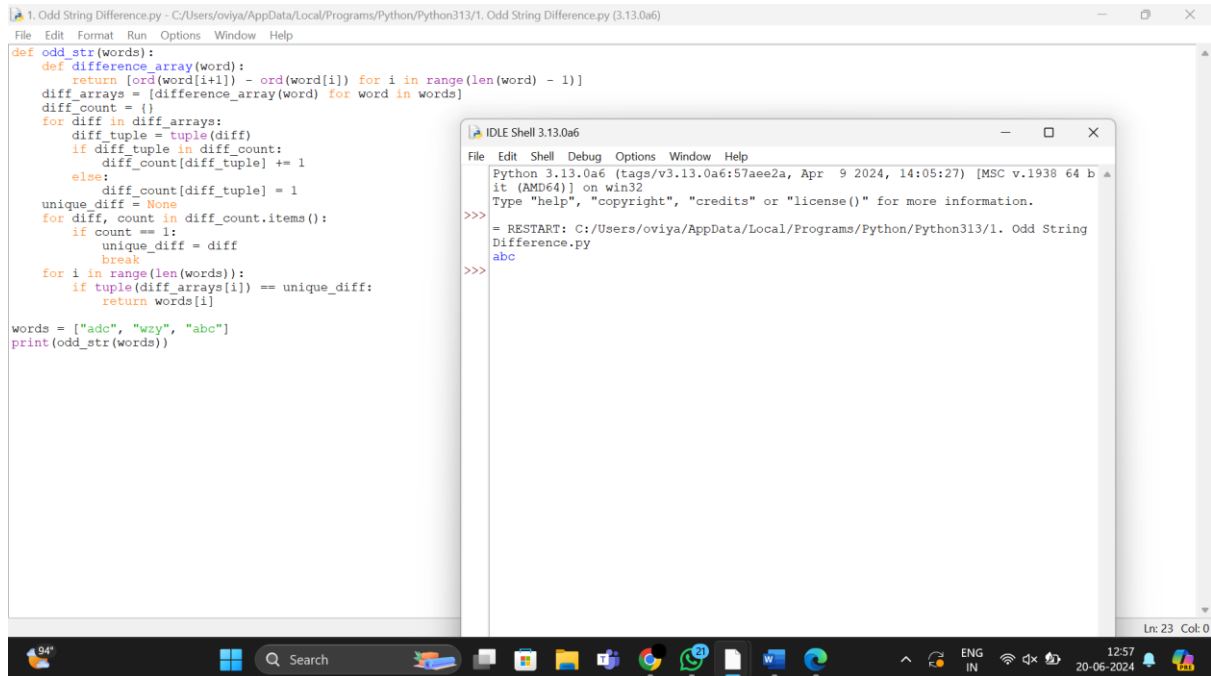


ASSIGNMENT (19.06.24)

1. Odd String Difference



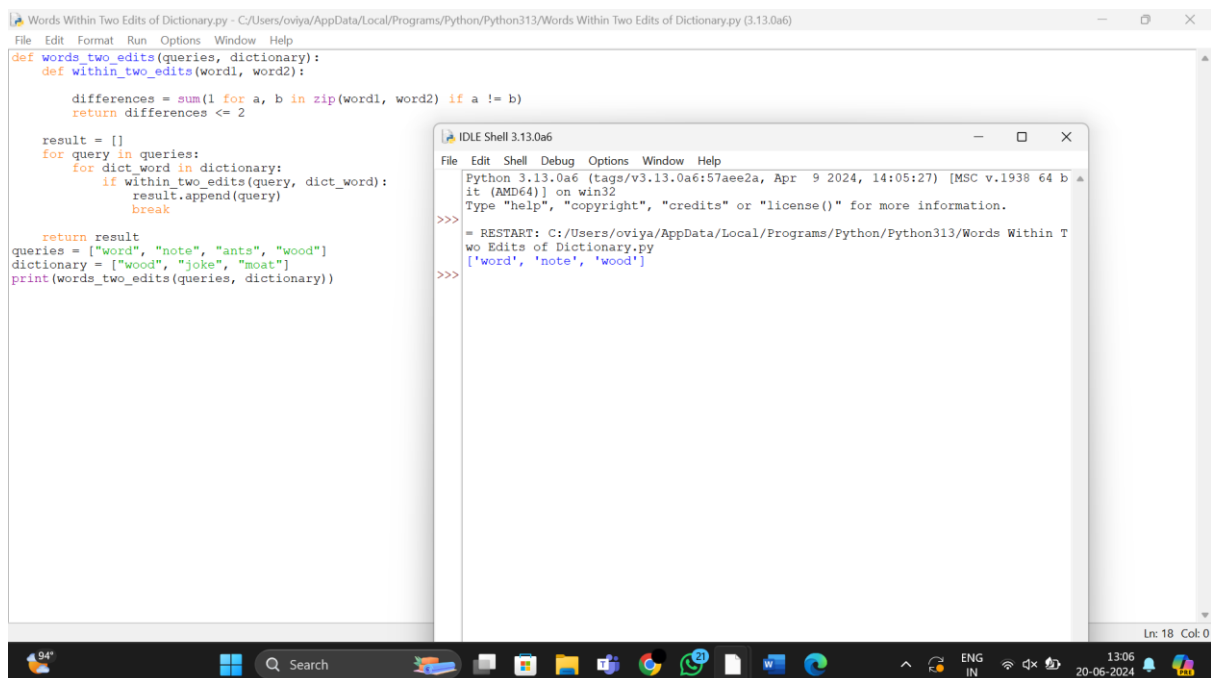
The screenshot shows a Python IDE with a file named "1. Odd String Difference.py". The code defines a function `odd_str(words)` that finds the word with the largest odd string difference. The difference is calculated as the absolute difference between the sum of odd-indexed characters and the sum of even-indexed characters. The words `["adc", "wzy", "abc"]` are used as an example, and the output is `adc`.

```
def odd_str(words):
    def difference_array(word):
        return [ord(word[i+1]) - ord(word[i]) for i in range(len(word) - 1)]
    diff_arrays = [difference_array(word) for word in words]
    diff_count = {}
    for diff in diff_arrays:
        diff_tuple = tuple(diff)
        if diff_tuple in diff_count:
            diff_count[diff_tuple] += 1
        else:
            diff_count[diff_tuple] = 1
    unique_diff = None
    for diff, count in diff_count.items():
        if count == 1:
            unique_diff = diff
            break
    for i in range(len(words)):
        if tuple(diff_arrays[i]) == unique_diff:
            return words[i]

words = ["adc", "wzy", "abc"]
print(odd_str(words))
```

The IDLE Shell shows the execution of the script, which restarts and outputs `adc`.

2. Words Within Two Edits of Dictionary



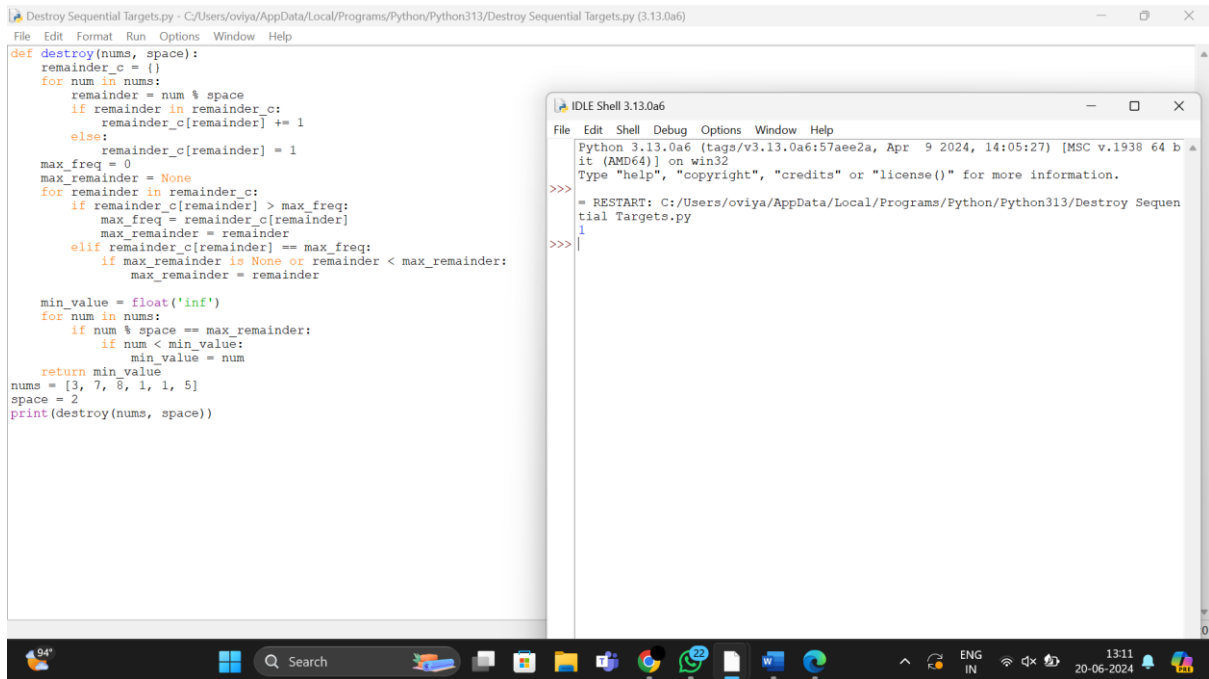
The screenshot shows a Python IDE with a file named "Words Within Two Edits of Dictionary.py". The code defines a function `words_within_two_edits(queries, dictionary)` that finds words in a dictionary that are within two edits of a given query. The edit distance is calculated as the number of characters that differ between two words. The queries `["word", "note", "ants", "wood"]` and the dictionary `["wood", "joke", "moat"]` are used as an example, and the output is `['word', 'note', 'wood']`.

```
def words_within_two_edits(queries, dictionary):
    def differences(word1, word2):
        return sum(1 for a, b in zip(word1, word2) if a != b)
    result = []
    for query in queries:
        for dict_word in dictionary:
            if differences(query, dict_word) <= 2:
                result.append(query)
                break
    return result

queries = ["word", "note", "ants", "wood"]
dictionary = ["wood", "joke", "moat"]
print(words_within_two_edits(queries, dictionary))
```

The IDLE Shell shows the execution of the script, which restarts and outputs `['word', 'note', 'wood']`.

3. Destroy Sequential Targets



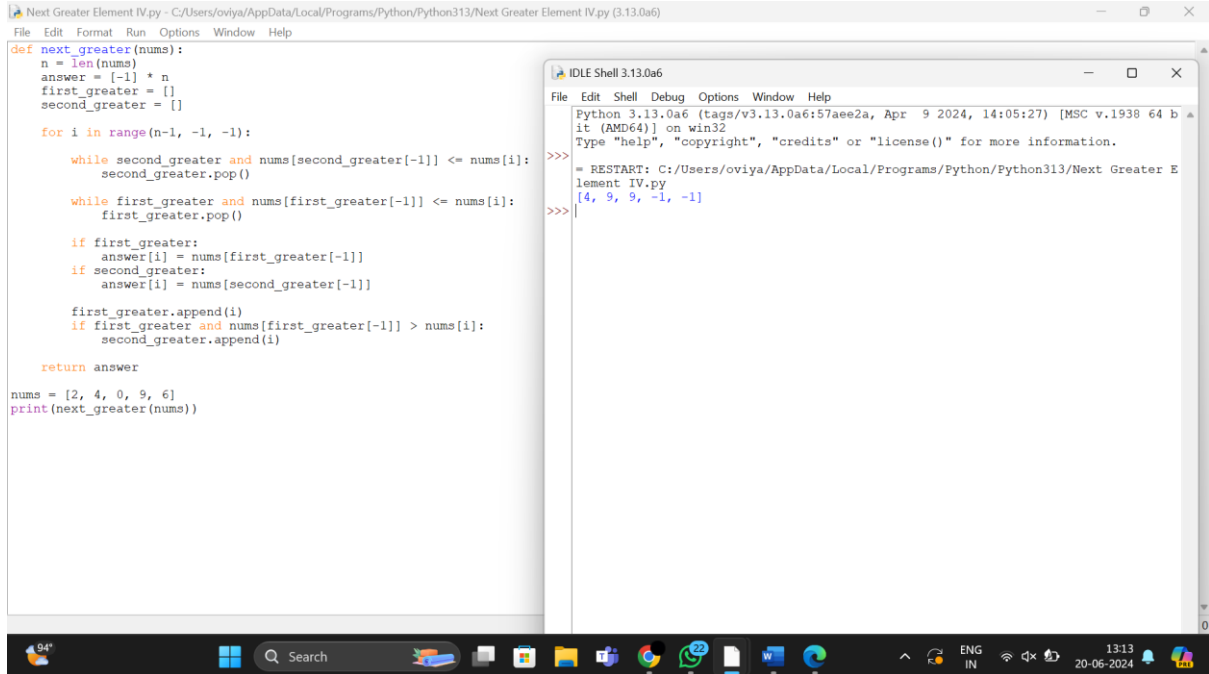
The screenshot shows a Python IDE with a file named 'Destroy Sequential Targets.py'. The code defines a function 'destroy' that takes a list of numbers and a space value. It iterates through the numbers, calculating remainders and finding the minimum value. The function is then called with the list [3, 7, 8, 1, 1, 5] and space 2. The output of the script is displayed in the IDLE Shell.

```
def destroy(nums, space):
    remainder_c = {}
    for num in nums:
        remainder = num % space
        if remainder in remainder_c:
            remainder_c[remainder] += 1
        else:
            remainder_c[remainder] = 1
    max_freq = 0
    max_remainder = None
    for remainder in remainder_c:
        if remainder_c[remainder] > max_freq:
            max_freq = remainder_c[remainder]
            max_remainder = remainder
        elif remainder_c[remainder] == max_freq:
            if max_remainder is None or remainder < max_remainder:
                max_remainder = remainder

    min_value = float('inf')
    for num in nums:
        if num % space == max_remainder:
            if num < min_value:
                min_value = num
    return min_value
nums = [3, 7, 8, 1, 1, 5]
space = 2
print(destroy(nums, space))
```

```
Python 3.13.0a6 (tags/v3.13.0a6:57aee2a, Apr 9 2024, 14:05:27) [MSC v.1938 64 b
it (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> = RESTART: C:/Users/oviya/AppData/Local/Programs/Python/Python313/Destroy Seque
tial Targets.py
1
>>>
```

4. Next Greater Element IV



The screenshot shows a Python IDE with a file named 'Next Greater Element IV.py'. The code defines a function 'next_greater' that takes a list of numbers and returns the next greater element. It uses two stacks, 'first_greater' and 'second_greater', to find the next greater element for each number in the list. The function is then called with the list [2, 4, 0, 9, 6]. The output of the script is displayed in the IDLE Shell.

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

    for i in range(n-1, -1, -1):
        while second_greater and nums[second_greater[-1]] <= nums[i]:
            second_greater.pop()

        while first_greater and nums[first_greater[-1]] <= nums[i]:
            first_greater.pop()

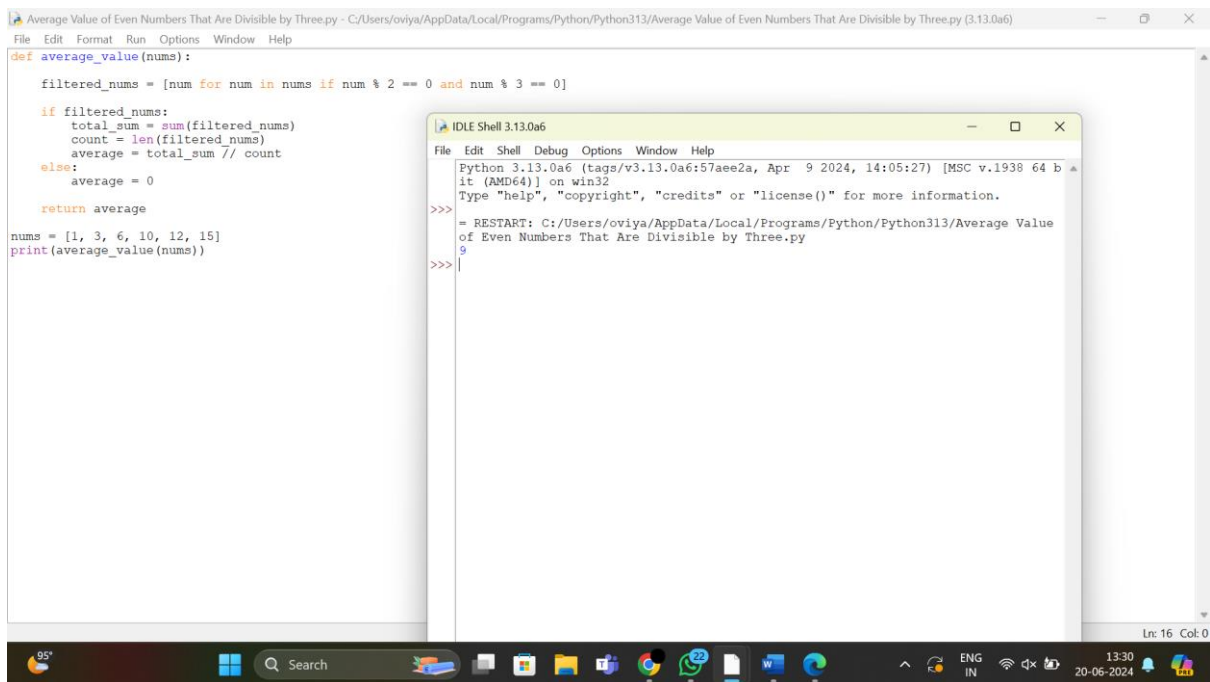
        if first_greater:
            answer[i] = nums[first_greater[-1]]
        if second_greater:
            answer[i] = nums[second_greater[-1]]

        first_greater.append(i)
        if first_greater and nums[first_greater[-1]] > nums[i]:
            second_greater.append(i)

    return answer
nums = [2, 4, 0, 9, 6]
print(next_greater(nums))
```

```
Python 3.13.0a6 (tags/v3.13.0a6:57aee2a, Apr 9 2024, 14:05:27) [MSC v.1938 64 b
it (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> = RESTART: C:/Users/oviya/AppData/Local/Programs/Python/Python313/Next Greater E
lement IV.py
[4, 9, 9, -1, -1]
>>>
```

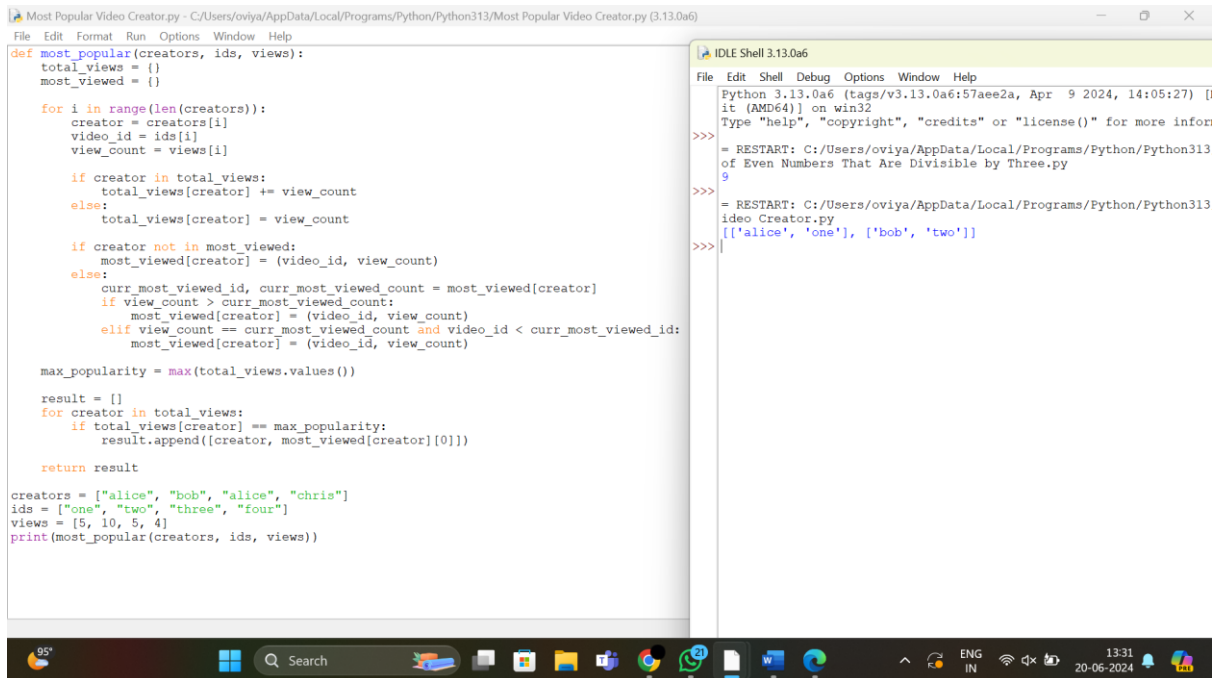
5. Average Value of Even Numbers That Are Divisible by Three



```
def average_value(nums):  
    filtered_nums = [num for num in nums if num % 2 == 0 and num % 3 == 0]  
    if filtered_nums:  
        total_sum = sum(filtered_nums)  
        count = len(filtered_nums)  
        average = total_sum // count  
    else:  
        average = 0  
    return average  
  
nums = [1, 3, 6, 10, 12, 15]  
print(average_value(nums))
```

```
Python 3.13.0a6 (tags/v3.13.0a6:57aee2a, Apr 9 2024, 14:05:27) [MSC v.1938 64 b  
it (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>>  
= RESTART: C:/Users/oviya/AppData/Local/Programs/Python/Python313/Average Value  
of Even Numbers That Are Divisible by Three.py  
9  
>>>|
```

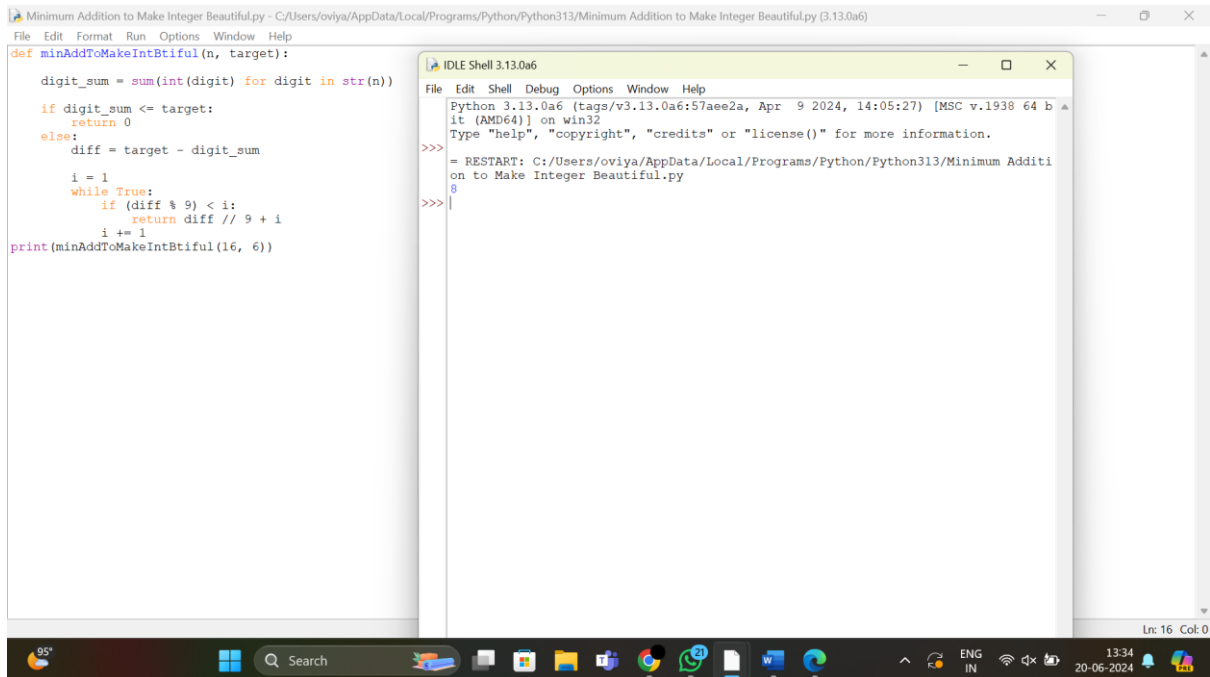
6. Most Popular Video Creator



```
def most_popular(creators, ids, views):  
    total_views = {}  
    most_viewed = {}  
  
    for i in range(len(creators)):  
        creator = creators[i]  
        video_id = ids[i]  
        view_count = views[i]  
  
        if creator in total_views:  
            total_views[creator] += view_count  
        else:  
            total_views[creator] = view_count  
  
        if creator not in most_viewed:  
            most_viewed[creator] = (video_id, view_count)  
        else:  
            curr_most_viewed_id, curr_most_viewed_count = most_viewed[creator]  
            if view_count > curr_most_viewed_count:  
                most_viewed[creator] = (video_id, view_count)  
            elif view_count == curr_most_viewed_count and video_id < curr_most_viewed_id:  
                most_viewed[creator] = (video_id, view_count)  
  
    max_popularity = max(total_views.values())  
    result = []  
    for creator in total_views:  
        if total_views[creator] == max_popularity:  
            result.append([creator, most_viewed[creator][0]])  
  
    return result  
  
creators = ["alice", "bob", "alice", "chris"]  
ids = ["one", "two", "three", "four"]  
views = [5, 10, 5, 4]  
print(most_popular(creators, ids, views))
```

```
Python 3.13.0a6 (tags/v3.13.0a6:57aee2a, Apr 9 2024, 14:05:27) [  
it (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more infor  
>>>  
= RESTART: C:/Users/oviya/AppData/Local/Programs/Python/Python313  
of Even Numbers That Are Divisible by Three.py  
9  
= RESTART: C:/Users/oviya/AppData/Local/Programs/Python/Python313  
ideo Creator.py  
>>> [['alice', 'one'], ['bob', 'two']]
```

7. Minimum Addition to Make Integer Beautiful

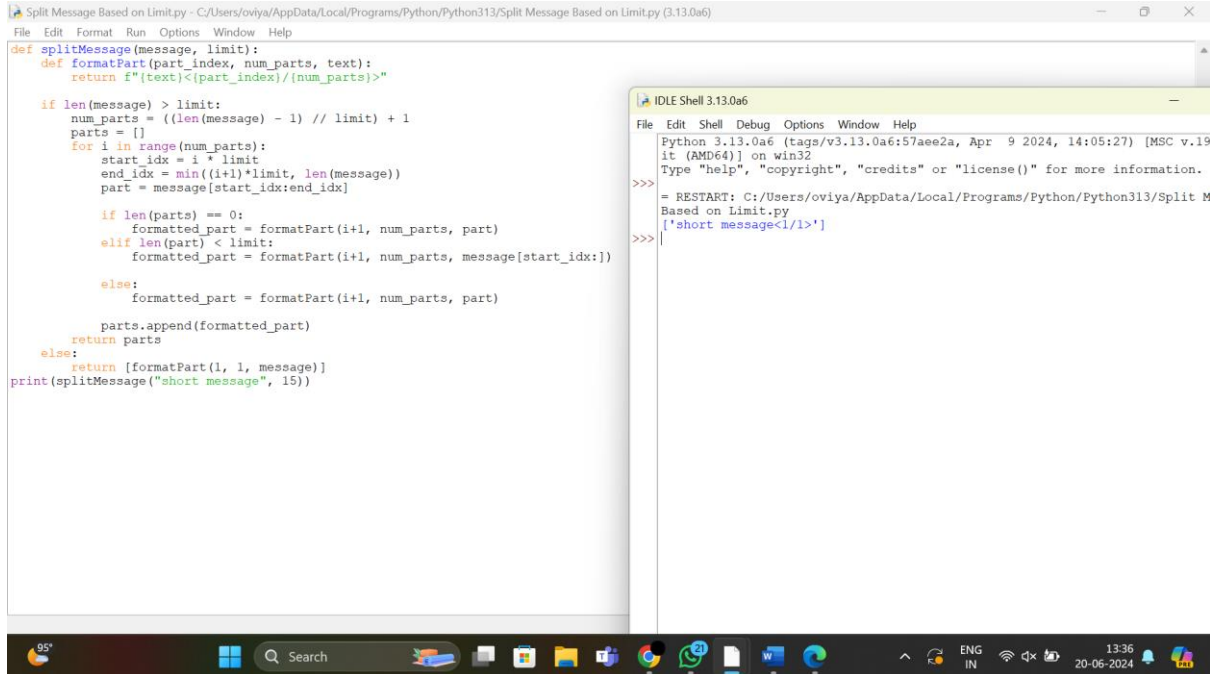


The screenshot shows a Python IDE with two windows. The main window displays the code for the 'Minimum Addition to Make Integer Beautiful' problem. The code defines a function `minAddToMakeIntegerBeautiful` that takes a number `n` and a target `target`. It calculates the sum of digits of `n`. If the sum is less than or equal to the target, it returns 0. Otherwise, it calculates the difference and repeatedly adds 9 to the number until the sum of digits is less than or equal to the target. The code is executed in the IDLE Shell, which shows the output 8.

```
def minAddToMakeIntegerBeautiful(n, target):
    digit_sum = sum(int(digit) for digit in str(n))
    if digit_sum <= target:
        return 0
    else:
        diff = target - digit_sum
        i = 1
        while True:
            if (diff % 9) < i:
                return diff // 9 + i
            i += 1
print(minAddToMakeIntegerBeautiful(16, 6))
```

```
Python 3.13.0a6 (tags/v3.13.0a6:57aee2a, Apr 9 2024, 14:05:27) [MSC v.1938 64 b
it (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/oviya/AppData/Local/Programs/Python/Python313/Minimum Additi
on to Make Integer Beautiful.py
>>>
8
```

8. Split Message Based on Limit



The screenshot shows a Python IDE with two windows. The main window displays the code for the 'Split Message Based on Limit' problem. The code defines a function `splitMessage` that takes a message and a limit. It splits the message into parts of length limit and returns a list of formatted parts. The code is executed in the IDLE Shell, which shows the output of the `splitMessage` function.

```
def splitMessage(message, limit):
    def formatPart(part_index, num_parts, text):
        return f"[{text}]<[{part_index}/{num_parts}]>"

    if len(message) > limit:
        num_parts = ((len(message) - 1) // limit) + 1
        parts = []
        for i in range(num_parts):
            start_idx = i * limit
            end_idx = min((i+1)*limit, len(message))
            part = message[start_idx:end_idx]

            if len(parts) == 0:
                formatted_part = formatPart(i+1, num_parts, part)
            elif len(part) < limit:
                formatted_part = formatPart(i+1, num_parts, message[start_idx:])
            else:
                formatted_part = formatPart(i+1, num_parts, part)

            parts.append(formatted_part)
        return parts
    else:
        return [formatPart(1, 1, message)]
print(splitMessage("short message", 15))
```

```
Python 3.13.0a6 (tags/v3.13.0a6:57aee2a, Apr 9 2024, 14:05:27) [MSC v.19
it (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/oviya/AppData/Local/Programs/Python/Python313/Split M
Based on Limit.py
>>>
["short message<[1/1]>"]
```