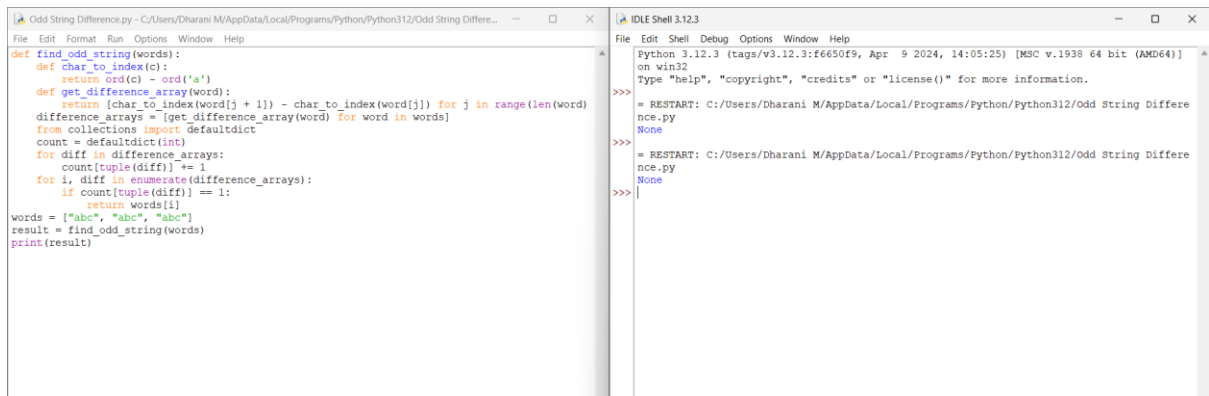


1. Odd String Difference



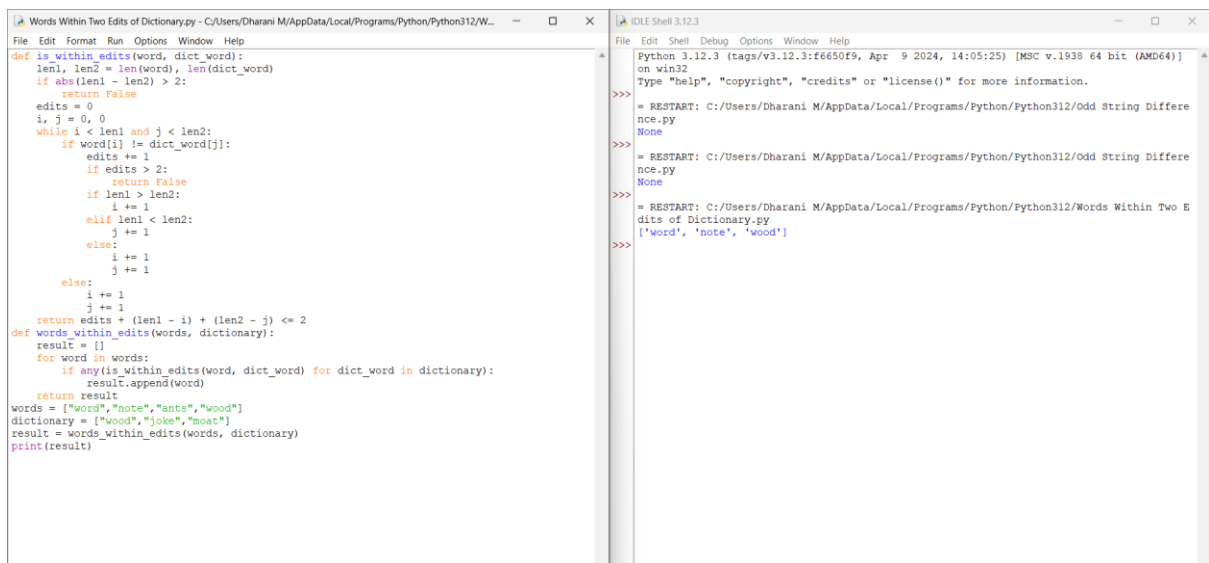
The screenshot shows a Python IDE with two windows. The left window, titled 'Odd String Difference.py', contains the following code:

```
def find_odd_string(words):
    def char_to_index(c):
        return ord(c) - ord('a')
    def get_difference_array(word):
        return [char_to_index(word[j] + 1) - char_to_index(word[j]) for j in range(len(word))]
    difference_arrays = [get_difference_array(word) for word in words]
    from collections import defaultdict
    count = defaultdict(int)
    for diff in difference_arrays:
        count[tuple(diff)] += 1
    for i, diff in enumerate(difference_arrays):
        if count[tuple(diff)] == 1:
            return words[i]
words = ["abc", "abc", "abc"]
result = find_odd_string(words)
print(result)
```

The right window, titled 'IDLE Shell 3.12.3', shows the execution output:

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/Odd String Differe
nce.py
None
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/Odd String Differe
nce.py
None
>>>
```

2. Words Within Two Edits of Dictionary



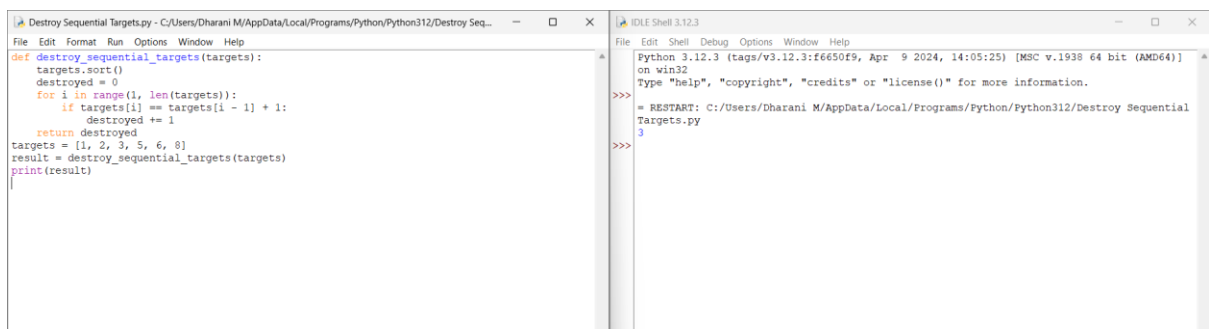
The screenshot shows a Python IDE with two windows. The left window, titled 'Words Within Two Edits of Dictionary.py', contains the following code:

```
def is_within_edits(word, dict_word):
    len1, len2 = len(word), len(dict_word)
    if abs(len1 - len2) > 2:
        return False
    edits = 0
    i, j = 0, 0
    while i < len1 and j < len2:
        if word[i] != dict_word[j]:
            edits += 1
            if edits > 2:
                return False
            if len1 > len2:
                i += 1
            elif len1 < len2:
                j += 1
            else:
                i += 1
                j += 1
        else:
            i += 1
            j += 1
    return edits + (len1 - i) + (len2 - j) <= 2
def words_within_edits(words, dictionary):
    result = []
    for word in words:
        if any(is_within_edits(word, dict_word) for dict_word in dictionary):
            result.append(word)
    return result
words = ["word", "note", "ants", "wood"]
dictionary = ["wood", "joke", "moat"]
result = words_within_edits(words, dictionary)
print(result)
```

The right window, titled 'IDLE Shell 3.12.3', shows the execution output:

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/Odd String Differe
nce.py
None
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/Odd String Differe
nce.py
None
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/Words Within Two E
dits of Dictionary.py
['word', 'note', 'wood']
>>>
```

3. Destroy Sequential Targets



The screenshot shows a Python IDE with two windows. The left window, titled 'Destroy Sequential Targets.py', contains the following code:

```
def destroy_sequential_targets(targets):
    targets.sort()
    destroyed = 0
    for i in range(1, len(targets)):
        if targets[i] == targets[i - 1] + 1:
            destroyed += 1
    return destroyed
targets = [1, 2, 3, 5, 6, 8]
result = destroy_sequential_targets(targets)
print(result)
```

The right window, titled 'IDLE Shell 3.12.3', shows the execution output:

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/Destroy Sequential
Targets.py
3
>>>
```

4. Next Greater Element

```
def next_gre_element_iv(nums):
    stack = []
    result = [-1] * len(nums)
    for i in range(len(nums)):
        while stack and nums[stack[-1]] < nums[i]:
            result[stack.pop()] = nums[i]
            stack.append(i)
    return result
nums = [3,3]
result = next_gre_element_iv(nums)
print(result)
```

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/Destroy Sequential
Targets.py
3
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/Next Greater Eleme
nt IV.py
[-1, -1]
>>>
```

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

```
def average_of_three(nums):
    valid_numbers = [num for num in nums if num % 6 == 0]
    return sum(valid_numbers) / len(valid_numbers) if valid_numbers else 0
nums = [1,3,6,10,12,15]
result = average_of_three(nums)
print(result)
```

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/Average Value of E
ven Numbers That Are Divisible by Three.py
9.0
>>>
```

6. Most Popular Video Creator

```
from collections import defaultdict
def most_popular_video_creator(creators, views):
    popularity = defaultdict(int)
    for creator, view in zip(creators, views):
        popularity[creator] += view
    max_views = max(popularity.values())
    most_popular_creators = [creator for creator, view in popularity.items() if view == max_v]
    return most_popular_creators
creators = ["dharu", "nandhu", "riya", "nilan", "ammubhuji"]
views = [100, 200, 300, 50, 150]
result = most_popular_video_creator(creators, views)
print(result)
```

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/Most Popular Video
Creator.py
['riya']
>>>
```

7. Minimum Addition to Make Integer Beautiful

```
def sum_of_digits(n):
    return sum(int(digit) for digit in str(n))
def min_addition_to_beautiful(n, k):
    current_sum = sum_of_digits(n)
    addition = 0
    while (current_sum + addition) % k != 0:
        addition += 1
    return addition
n = 1
k = 1
result = min_addition_to_beautiful(n, k)
print(result)
```

```
Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)]
on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/Minimum Addition t
o Make Integer Beautiful.py
0
>>>
```

8. Split Message Based on Limit

<div>Split Message Based on Limit.py - C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/Split Mess... File Edit Format Run Options Window Help def split_message(message, chunk_size): parts = [] for i in range(0, len(message), chunk_size): parts.append(message[i:i+chunk_size]) return parts message = "This is a sample message that needs to be split into parts." chunk_size = 10 result = split_message(message, chunk_size) print(result)</div>	<div>IDLE Shell 3.12.3 File Edit Shell Debug Options Window Help Python 3.12.3 (tags/v3.12.3:f6650f9, Apr 9 2024, 14:05:25) [MSC v.1938 64 bit (AMD64)] on win32 Type "help", "copyright", "credits" or "license()" for more information. >>> = RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/Minimum Addition t o Make Integer Beautiful.py 0 >>> = RESTART: C:/Users/Dharani M/AppData/Local/Programs/Python/Python312/Split Message Base d on Limit.py ['This is a', 'sample mes', 'sage that ', 'needs to b', 'e split in', 'to parts.'] >>></div>
---	---