Live Programming for Validation

```
def dominant_bigram(s):
    '''
    Return the most common bigram in string s.
    '''
    ## ---
    res = ''
    bigrams = {}
    for i in range(0, len(s) - 1):
        bigram = s[i] + s[i + 1]
        if bigram not in bigrams:
            bigrams[bigram] = 0
        bigrams[bigram] += 1
    max_value = max(bigrams.values())
    for bigram in bigrams:
        if bigrams[bigram] == max_value:
            res = bigram
    ## ---

    return res


dominant_bigram("agctagta")
```

```
# bigrams
0 {'ag': 0}
1 {'ag': 1, 'gc': 0}
2 {'ag': 1, 'gc': 1, 'ct': 0}
3 {'ag': 1, 'gc': 1, 'ct': 1, 'ta': 0}
4
5 {'ag': 2, 'gc': 1, 'ct': 1, 'ta': 1, 'gt': 0}
6
```

```
# bigrams
0 {'ag': 1}
1 {'ag': 1, 'gc': 1}
2 {'ag': 1, 'gc': 1, 'ct': 1}
3 {'ag': 1, 'gc': 1, 'ct': 1, 'ta': 1}
4 {'ag': 2, 'gc': 1, 'ct': 1, 'ta': 1}
5 {'ag': 2, 'gc': 1, 'ct': 1, 'ta': 1, 'gt': 1}
6 {'ag': 2, 'gc': 1, 'ct': 1, 'ta': 2, 'gt': 1}
```

```
max_value
2
```

```
#  res
0  'ag'
1
2
3  'ta'
4
```

Suggestion 2

Preview

```
res = {}
s = s.lower()
for i in range(0, len(s)):
    if s[i] in res:
        res[s[i]] += 1
    else:
        res[s[i]] = 1
print(res)
for i in res.keys():
    if res[i] < 2:
        res.pop(i)
print(res)
```

Suggestion 3

Preview

```
res = ''
bigrams = {}
for i in range(0, len(s) - 1):
```

Study: Effects of Live Programming on Validation