

AI ASSISTED CODING

NAME: G.DEEKSHITH RAJ

2403A51248

BATCH:11

SUB GROUP:C

C.1 — [S18C1] Debug de-duplication (case-insensitive)

Context:

Contact records in the real estate listings platform CRM contain duplicates differing only by case.

Your Task:

Return the first occurrence of each email (case-insensitive) while preserving original order.

Data & Edge Cases:

Input: list of emails. Normalize keys using lowercase; output retains original casing.

AI Assistance Expectation:

Use AI to spot bug (reinitializing `seen` in loop) and correct to stable algorithm.

Constraints & Notes:

Add unit tests covering mixed-case duplicates.

Sample Input

```
['A@x.com', 'a@x.com', 'B@y.com']
```

Sample Output

```
['A@x.com', 'B@y.com']
```

Acceptance Criteria: Preserves first occurrence order; case-insensitive matching

CODE:

```
index.html AL coding X
AL coding > ...
Windsurf: Refactor | Explain | Generate Docstring | X
1 def deduplicate_emails(emails):
2     seen = set()
3     result = []
4     for email in emails:
5         key = email.lower()
6         if key not in seen:
7             seen.add(key)
8             result.append(email)
9     return result
10
11 # Unit tests
12 Windsurf: Refactor | Explain | Generate Docstring | X
13 def test_deduplicate_emails():
14     assert deduplicate_emails(['A@x.com', 'a@x.com', 'B@y.com']) == ['A@x.com', 'B@y.com']
15     assert deduplicate_emails(['a@x.com', 'A@x.com', 'A@x.com']) == ['a@x.com']
16     assert deduplicate_emails(['A@x.com', 'B@y.com', 'C@z.com']) == ['A@x.com', 'B@y.com', 'C@z.com']
17     assert deduplicate_emails(['a@x.com', 'b@y.com', 'A@x.com', 'B@y.com']) == ['a@x.com', 'b@y.com']
18     assert deduplicate_emails([]) == []
19     print("All tests passed.")
20
21 if __name__ == "__main__":
22     test_deduplicate_emails()
23     test_deduplicate_emails()
```

OUTPUT:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS ... Filter
[Running] python -u "c:\Users\gandi\OneDrive\Desktop\lab (7)\AL coding"
All tests passed.

[Done] exited with code=0 in 0.195 seconds
```

QUESTION:2

C.2 — [S18C2] TDD: slugify titles

Context:

Titles in the real estate listings platform CMS must become SEO slugs.

Your Task:

TDD for slugify(text): lowercase, remove non-alnum except hyphen, spaces->hyphen, collapse and trim hyphens.

Data & Edge Cases:

Include punctuation and multiple spaces.

AI Assistance Expectation:

Have AI propose parametric tests then implement regex solution.

Constraints & Notes:

Return correct slugs.

Sample Input

['Hello World!', 'AI & You', 'Set18-C2']

Sample Output

['hello-world', 'ai-you', 'set18-C2']

Acceptance Criteria: All tests pass

CODE:

```
1 import re
2
3 def slugify(text):
4     # Lowercase
5     text = text.lower()
6     # Replace spaces with hyphens
7     text = re.sub(r'\s+', '-', text)
8     # Remove non-alphanumeric except hyphens
9     text = re.sub(r'[^\w-]', '', text)
10    # Collate (module) re
11    text =
12    # Trim
13    text =
14    return
15
16 # Parametric
17 def test_sl
18 cases =
19 ('H
20 ('A The special characters are:
21 ('S
22 ('
23 ('Punctuation!!!', 'punctuation'),
24 ('--Already--Slug--', 'already-slug'),
25 ('MiXeD CaSe', 'mixed-case'),
26 ('', ''),
27 (' ', ' '),
28 ('a_b_c', 'a-b-c'),
29 ('foo--bar', 'foo-bar'),
30 ('foo - bar', 'foo-bar'),
```

OUTPUT/INPUT:

```
PROBLEMS  OUTPUT  ...  Filter
Traceback (most recent call last):
AssertionError: Failed: a_b_c -> abc != a-b-c

[Done] exited with code=1 in 0.206 seconds
```