Worksheet 28: Grouping with Dict of Lists

Name: _____ Date: _____

Instructions

- Answer in the blanks.
- For “write code” questions, write valid Python code.
- For “what does it print” questions, write the exact output.

Part A — Warm-up: first and last letters

1) What does it print?

```
words = ["cat", "car", "dog", "door"]

for w in words:
    print(w[0])
```

Output:

```

```

2) Fill in the blanks (first + last letter)

```
w = "apple"

first = w[____]
last  = w[____]

print(first)
print(last)
```

Output:

```

```

3) What does it print? (last letters)

```
words = ["cat", "dog", "car"]

for w in words:
    print(w[-1])
```

Output:

```

```

Part B — Build buckets (group by first letter)

4) Fill in the blanks (basic grouping)

```
words = ["cat", "car", "dog"]

groups = {}

for w in words:
    key = w[0]
    if key not in groups:
        groups[key] = _______
    groups[key].______________

print(groups)
```

Expected output:

```
{'c': ['cat', 'car'], 'd': ['dog']}
```

5) What does it print? (trace the dictionary)

```
words = ["ant", "apple", "bat", "ball"]

groups = {}

for w in words:
    key = w[0]
    if key not in groups:
        groups[key] = []
    groups[key].append(w)

print(groups)
```

Output:

```

```

6) Fix the bug (missing bucket)

This code crashes. Fix it so it works.

```
words = ["cat", "car", "dog"]
groups = {}

for w in words:
    key = w[0]
    # BUG: missing something here
    groups[key].append(w)

print(groups)
```

Write the corrected code:

```

```

Part C — Change the key

7) Fill in the blanks (group by last letter)

```
words = ["cat", "mat", "car", "bar"]

groups = {}

for w in words:
    key = _____________
    if key not in groups:
        groups[key] = []
    groups[key].append(w)

print(groups)
```

Expected output:

```
{'t': ['cat', 'mat'], 'r': ['car', 'bar']}
```

8) Write code (group by word length)

Group the words by their length.

```
words = ["hi", "cat", "door", "a", "sun", "tree"]

# write your code here
```

Expected output:

```
{2: ['hi'], 3: ['cat', 'sun'], 4: ['door', 'tree'], 1: ['a']}
```

9) Write code (group numbers: even vs odd)

Group the numbers into two buckets: `"even"` and `"odd"`.

```
nums = [5, 2, 7, 4, 1, 6]

# write your code here
```

Expected output:

```
{'odd': [5, 7, 1], 'even': [2, 4, 6]}
```

Part D — Using the buckets

10) What does it print? (look inside one bucket)

```
words = ["cat", "car", "dog", "door", "apple"]

groups = {}

for w in words:
    key = w[0]
    if key not in groups:
        groups[key] = []
    groups[key].append(w)

print(groups["d"])
```

Output:

```

```

11) Fill in the blanks (print bucket sizes)

```
words = ["cat", "car", "dog", "door", "apple"]

groups = {}

for w in words:
    key = w[0]
    if key not in groups:
        groups[key] = []
    groups[key].append(w)

for key, bucket in groups._____():
    print(key, "has", _____, "words")
```

Expected output:

```
c has 2 words
d has 2 words
```

```
a has 1 words
```

12) Sort each bucket

Hint:

- list.sort() sorts a list in place, meaning it rearranges the items inside the same list and returns None.
- By default it sorts small → large (numbers) or A → Z (strings).
- Example:
  - `letters = ['c', 'd', 'a', 'b']`
  - `letters.sort()`
  - Now `letters` becomes `['a', 'b', 'c', 'd']`

What is the output?

```
words = ["car", "cat", "door", "dog"]

groups = {}

for w in words:
    key = w[0]
    if key not in groups:
        groups[key] = []
    groups[key].append(w)

for key in groups:
    groups[key].sort()

print(groups)
```

Output:

```

```

13) Fill in the blanks (turn groups into counts)

We already grouped the words. Now make a **count dictionary** like `{'c': 2, 'd': 1}`.

```
groups = {'c': ['cat', 'car'], 'd': ['dog']}

counts = {}

for key, bucket in groups.items():
    counts[key] = _____________

print(counts)
```

Output:

```

```

Part F — Mini challenge

14) Case-insensitive grouping (ignore upper/lowercase)

Hint:

- `lower()` makes a string all lowercase and returns a new string.
- It does not change the original.
- It's useful when you want to compare text without worrying about uppercase vs lowercase.
- Example:
  - `name = "ALICE"`
  - `print(name.lower())`
  - It shows `"alice"`

Group names by first letter, but treat `"A"` and `"a"` as the **same** key. Complete the code.

```
names = ["Ava", "adam", "Ben", "bobby", "ALICE"]

groups = {}

for name in names:
    key = name[0].___________
    if _________________________:
        groups[key] = []
    groups[key]._____________

print(groups)
```

Expected output:

```
{'a': ['Ava', 'adam', 'ALICE'], 'b': ['Ben', 'bobby']}
```