Quiz 24-29: Python Dictionary

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 (line by line).
- If a question says “order may vary”, any correct order is acceptable.
- Assume Python 3.

Part A — Core dictionary skills (methods + reasoning)

1) Same or different? (`copy()` vs alias)

What does it print? (order may vary)

Hint:

- `d2 = d1` → same dictionary (alias). Any change via `d1` or `d2` shows up in both.
- `d3 = d1.copy()` → new dictionary (separate).
- So `d1["a"]=99` and `d2["c"]=3` affect both `d1` and `d2`.
- `d3["b"]=200` affects only `d3`.

```
d1 = {"a": 1, "b": 2}
d2 = d1
d3 = d1.copy()

d1["a"] = 99
d2["c"] = 3
d3["b"] = 200

print(d1)
print(d2)
print(d3)
```

Output:

```

```

2) Safe lookup without `get` (fill in the blank)

Fill in **ONE operator** to avoid `KeyError`.

```
pets = {"Mochi": "cat", "Boba": "dog"}

name = "Luna"

if name ________ pets:  # an operator
    animal = pets[name]
else:
    animal = "hamster"

print(animal)
```

3) Fix the bug (avoid KeyError)

This code crashes sometimes. Fix it so it prints `0` when the key is missing.

```
scores = {"Amy": 3, "Ben": 5}

name = "Chloe"
print(scores[name])
```

Fixed code:

```

```

4) Trace the updates — what is the final dictionary?

```
bag = {"apple": 2, "banana": 5, "cookie": 1}

bag["banana"] = bag["banana"] + 1

if "donut" in bag:
    bag["donut"] = bag["donut"] + 3
else:
    bag["donut"] = 3

x = bag.pop("cookie")
del bag["apple"]
bag["banana"] = x

print(bag)
```

Hint: `pop("cookie")` returns the value that was stored under `"cookie"` and removes that key from the dictionary.

Output:

```

```

5) Count items (fill in the blanks)

Complete the code to build a frequency dictionary.

```
items = ["pen", "pen", "pencil", "pen", "eraser"]

count = {}

for item in items:
    if ____________________________________:
        count[item] = _____________________
    else:
        count[item] = _____________________

print(count)
```

6) Filter + build (write code)

Make a dictionary `lengths` where:

- each key is a word from `words`
- each value is the word's length
- **only include** words with length **4 or more**

Write code:

```
words = ["cat", "tree", "bird", "sun", "plane"]

lengths = ________________

for w in words:
    if ___________________:
        __________________
```

7) Numbers to squares (write code)

Create a dictionary `sq` for numbers from **1 to n** (inclusive), mapping each number to its square.

Example: if `n = 4`, then `sq` should be `{1: 1, 2: 4, 3: 9, 4: 16}`.

Write code:

```
n = 6

sq = {}

for i in ______________________:
    sq[i] = ___________________
```

8) `keys()`, `values()`, `items()`

What does it print? (order may vary)

```
d = {"a": 1, "b": 2, "c": 3}

print(list(d.keys()))
print(list(d.values()))
print(list(d.items()))
```

Output:

```



```

Part B — Dictionaries with other structures

9) Nested dictionary

What does it print?

```
player = {"name": "Amy", "stats": {"hp": 10, "mp": 3}}

player["stats"]["hp"] = player["stats"]["hp"] + 5
player["stats"]["mp"] = 0

print(player["stats"])
```

Output:

```

```

10) Shopping cart total (fill in the blank)

`"ruler"` is **not** in `price`, so it should add **0**.

```
price = {"pencil": 0.5, "eraser": 1.0, "notebook": 2.5}
cart = ["pencil", "pencil", "notebook", "ruler"]

total = 0
for item in cart:
    if item ____ price:  # fill in one operator
        total = total + price[item]
    else:
        total = total + 0

print(total)
```

Output:

```

```

11) Bigram counting (fill in the blank)

A **bigram** is 2 letters next to each other.

For `s = "BANANA"`, the bigrams are: `BA`, `AN`, `NA`, `AN`, `NA`.

```
s = "BANANA"
bigrams = {}

for i in range(len(s) - 1):
    bg = s[i:i+2]
    if bg ____ bigrams:  # fill in one operator
        bigrams[bg] = bigrams[bg] + 1
    else:
        bigrams[bg] = 1

print(bigrams)
```

Output (order may vary):

```

```

---

12) Most common bigram (write code)

Using the `bigrams` dictionary from Question 11, write code to print the bigram with the **largest** count.

If there is a tie, print the **alphabetically smallest** bigram.

```
bigrams = {"BA": 1, "AN": 2, "NA": 2}
```

Write code:

```

```

13) Build a dictionary of tuples (write code)

Create a dictionary like this:

```
{"cat": ("c", "t"), "dog": ("d", "g"), "fish": ("f", "h")}
```

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

Write code:

Part C — Mini challenges (small programs)

14) Group to sets (write code)

Build a dictionary `groups` where:

- each key is a color
- each value is a **set** of numbers for that color (no duplicates)

Write code:

```
pairs = [("red", 1), ("blue", 2), ("red", 3), ("red", 1), ("blue", 2)]

groups = {}

for color, num in pairs:
    if ______________________________:
     ______________________________
    else:
```

```
    else:
        ____________________________

print(groups)
```

Expected output (order may vary):

```
{"red": {1, 3}, "blue": {2}}
```

15) Simple “translator” (fill in the blank)

Use `codebook` to translate each character. If a character is not in the dictionary, keep it the same.

```
codebook = {"A": "@", "E": "3", "I": "1", "O": "0"}
text = "I LOVE AI"

out = ""
for ch in text:
    if ch ____ codebook:  # one operator
        out = out + codebook[ch]
    else:
        out = out + ch

print(out)
```

Output:

```

```

16) Dice histogram (write code)

- You are given dice rolls (numbers 1 to 6).
- Build a dictionary `hist` that counts how many times each number appears.
- Then print counts in order from 1 to 6, one per line like:

Write code:

```
rolls = [1, 4, 4, 6, 3, 4, 1]

hist = {}

for r in rolls:
    if ______________________:
        ______________________
    else:
        ______________________

for face in range(1, 7):
    if ______________________:
        ______________________
    else:
        print(face, 0)
```

Expected output:

```
1 2
2 0
3 1
4 3
5 0
6 1
```