Quiz 19–28: Python Sets and Dictionaries

Name: _____ Date: _____

Instructions

- Answer in the blanks.
- For “write code” questions, write **valid Python code** (no functions needed unless the question asks).
- 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.

Part A — Core ideas (sets vs dictionaries)

1) Pick the correct container (multiple choice)

Which line creates an **empty set**?

A. `x = {}`
B. `x = set()`
C. `x = dict()`
D. `x = []`

---

2) Fill in the blanks (unique rules)

Fill in the blanks.

1. Set items are _____. (Choose from: `unique` and `not unique`.)
2. Set items _____ a fixed order. (Choose from: `have` and `don't have`.)
3. A **dictionary** stores key → value pairs, and dictionary keys must be _____. (Choose from: `unique` and `not unique`.)

---

3) `in` behavior (short answer)

In Python, `x in some_dict` checks whether `x` is a _____ of the dict. (Choose from: `key` and `value`.)

4) Quick True/False

Write **True** or **False**.

1. You can do `s[0]` on a set `s`. _____
2. A dictionary can have the same key twice at the same time. _____
3. `len(d)` counts how many keys are in dictionary `d`. _____
4. A set can be used as a dictionary key. _____

Part B — Trace and predict

5) Set changes (what does it print?)

(order may vary)

```
s = set()
s.add("apple")
s.add("banana")
s.add("apple")

s.discard("pear")
s.remove("banana")

print(s)
print(len(s))
print("banana" in s)
```

Output:

```

```

6) Dictionary updates (what does it print?)

```
d = {"a": 1, "b": 2}
d["b"] = d["b"] + 5
d["c"] = d["a"] + d["b"]
d["a"] = 0

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

Output:

```

```

7) Keys vs values (what does it print?)

```
d = {"cat": 2, "dog": 1}

print("cat" in d)
print(2 in d)
print("dog" in d.keys())
print(1 in d.values())
```

Output:

```

```

8) Set math (what does it print?)

(order may vary)

```
a = {"red", "blue", "green"}
b = {"blue", "yellow"}

print(a | b)    # union
print(a & b)    # intersection
print(a - b)    # in a but not in b
print(a ^ b)    # in exactly one of them
```

Output:

```

```

Part C — Fill in the blanks (write the missing code)

9) Safe counting with `get` (fill in the blanks)

Complete the code so it counts correctly.

```
words = ["hi", "bye", "hi", "yes", "hi"]
freq = {}

for w in words:
    if _____________________________:
        freq[w] = _____________________
    else:
        freq[w] = _____________________

print(freq)
```

Expected output:

```
{'hi': 3, 'bye': 1, 'yes': 1}
```

10) Build a set from a string (fill in the blanks)

We want the set of **unique vowels** in the string.

```
text = "bananas"

vowels = {"a", "e", "i", "o", "u"}
found = ____________________   # empty set

for ch in text:
    if ch ___________________ vowels:
        found._____________________

print(found)
```

Output *(order may vary)*:

```
{'a'}
```

11) “Popular” words set (fill in the blanks)

From a frequency dict, build a set of words whose count is **at least 2**.

```
freq = {"sun": 3, "moon": 1, "star": 2, "sky": 1}

popular = _____________________  # empty set

for word, count in freq.items():
    if _____________________________:
        popular.______________________

print(popular)
```

Output *(order may vary)*:

```
{'sun', 'star'}
```

12) Filter a dictionary using a set of allowed keys (fill in the blanks)

Keep only items whose key is in `allowed`.

```
prices = {"apple": 3, "banana": 2, "cookie": 5, "donut": 4}
allowed = {"banana", "donut", "egg"}   # egg is not in prices

kept = {}

for item in allowed:
    if item _______________________________:
        kept[item] = ____________________

print(kept)
```

Output *(order may vary)*:

```
{'banana': 2, 'donut': 4}
```

Part D — Synergy problems (sets + dictionaries together)

13) Cart total with a “fast lookup” dict (write code)

You have:

```
prices = {"apple": 3, "banana": 2, "cookie": 5}
cart   = ["apple", "cookie", "cookie", "pear"]
```

Write code to compute the total cost of the items in `cart`.

Rules:

- If an item is not in `prices`, ignore it.
- Use a dictionary lookup for price.

Hint: membership check on dict keys is fast.

```
prices = {"apple": 3, "banana": 2, "cookie": 5}
cart   = ["apple", "cookie", "cookie", "pear"]

total = _____________________

for item in cart:
    if _____________________:
        total _________________

print(total)
```

---

14) Build “club → students” (dict of sets) (write code)

You are given a dictionary `student_to_club`.

```
student_to_club = {
    "Amy": "Robotics",
    "Ben": "Chess",
    "Chloe": "Robotics",
    "Dan": "Chess",
    "Eva": "Art",
}
```

Write code to build a new dictionary `club_to_students` where:

- each key is a club name
- each value is a **set** of students in that club

Then print `club_to_students`.

```
club_to_students = {}

for student, club in _____________________________:
    if _____________________________________________:
        club_to_students[club] = _____________
    club_to_students[club].add(_________________)

print(club_to_students)
```

Output *(order may vary)*:

```

```

15) Find clubs with more than 1 student (write code)

Using `club_to_students` from Question 14, build a set named `big_clubs` containing club names that have at least 2 students.

```
big_clubs = _______________

for club, students in ___________________________________:
    if _____________________________________:
        ___________________________________

print(big_clubs)
```

Output *(order may vary)*:

```

```

16) Two friends' favorite games (short answer + code)

We store each friend's favorite games in a dict.

```
favorites = {
    "Ava": {"chess", "tag", "puzzle"},
    "Bo":  {"tag", "soccer"},
}
```

1) What is the set of games they both like? _____

2) Write code to compute it and print it.

```

```

---

17) Reverse map: value → set of keys (write code)

Given:

```
pet_type = {"Mochi": "cat", "Boba": "dog", "Luna": "cat", "Nemo": "fish"}
```

Build a dictionary `type_to_names` like:

- `"cat"` maps to the set `{"Mochi", "Luna"}`
- `"dog"` maps to the set `{"Boba"}`
- `"fish"` maps to the set `{"Nemo"}`

Write code:

```
type_to_names = {}

for name, t in pet_type._____________________:
    if ___________________________________:
        type_to_names[t] = ___________________
    type_to_names[t].add(_____________________)

print(type_to_names)
```

Output *(order may vary)*:

```

```

---

18) Unique items per group (dict of sets) (fill in the blanks)

We want **unique** snacks per kid (no duplicates inside each kid's bucket).

```
snacks = [
    ("Amy", "chips"),
    ("Amy", "chips"),
    ("Ben", "apple"),
    ("Ben", "chips"),
    ("Ben", "apple"),
]

kid_snacks = {}

for kid, snack in snacks:
    if kid not in kid_snacks:
        kid_snacks[kid] = _____________________   # make an empty set here
    kid_snacks[kid].______________________________

print(kid_snacks)
```

Output *(order may vary)*:

```
{'Amy': {'chips'}, 'Ben': {'apple', 'chips'}}
```

19) “Same unique words?” (write code)

Two sentences might be written differently, but have the same **unique words**.

Write code that prints `True` if the two sentences have the same set of words (ignoring case).

```
s1 = "Cats and dogs"
s2 = "DOGS and CATS"

# your code here:
```

Expected output:

```
True
```

Part E — Debugging and design

20) Fix the bug: “set vs dict” confusion

This code tries to create an empty set, but crashes later.

```
seen = {}          # should be an empty set
seen.add("A")
```

Write the corrected code:

```

```

21) Choose the best tool (multiple choice)

Pick the best answer.

You need to:

- store **unique** student names,
- quickly check if a name is already there,
- and you do NOT care about order.

Which is best?

A. list
B. set
C. dict (keys only)
D. tuple

Your answer: _____

22) Mini challenge: tiny inverted index (dict of sets) (write code)

You are given short documents.

```
docs = {
    1: "red blue blue",
    2: "blue green",
    3: "red green green",
}
```

Build a dictionary `index` where:

- each key is a word
- each value is a set of document IDs that contain that word

Example:

- `"red"` should map to `{1, 3}`
- `"blue"` should map to `{1, 2}`
- `"green"` should map to `{2, 3}`

Write code (hint: `text.split()`):

```
docs = {
    1: "red blue blue",
    2: "blue green",
    3: "red green green",
}

index = {}

for doc_id, text in docs._____________:
    words = text.split()

    for w in set(words):  # set(...) avoids repeating the same doc_id
        if ____________________________:
            index[w] = set()
        index[w].add(___________________)

print(index)
```

Output *(order may vary)*:

```

```