Quiz 19–28: Python Sets and Dictionaries

Part A — Core ideas (sets vs dictionaries)

1) Pick the correct container (multiple choice)

Answer: B (`x = set()`)

---

2) Fill in the blanks (unique rules)

1. `unique`
2. `don't have`
3. `unique`

---

3) `in` behavior (short answer)

Answer: `key`

---

4) Quick True/False

1. **False**
2. **False**
3. **True**
4. **False** *(A normal set is not hashable, so it cannot be a dict key.* `frozenset` *can.)*

Part B — Trace and predict

5) Set changes (what does it print?)

One valid output is:

```
{'apple'}
1
False
```

(Order may vary for sets, but here the set has only one item.)

---

6) Dictionary updates (what does it print?)

```
{'a': 0, 'b': 7, 'c': 8}
8
```

---

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

```
True
False
True
True
```

---

8) Set math (what does it print?)

One valid output is:

```
{'red', 'blue', 'green', 'yellow'}
{'blue'}
{'red', 'green'}
{'red', 'green', 'yellow'}
```

(Order may vary.)

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

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

Fill-ins:

- `freq.get(w) is not None`
- `freq.get(w) + 1`
- `1`

Completed code:

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

for w in words:
    if freq.get(w) is not None:
        freq[w] = freq.get(w) + 1
    else:
        freq[w] = 1

print(freq)
```

Expected output:

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

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

Fill-ins:

- `set()`
- `in`
- `add(ch)`

Completed code:

```
text = "bananas"

vowels = {"a", "e", "i", "o", "u"}
found = set()

for ch in text:
    if ch in vowels:
        found.add(ch)

print(found)
```

Output:

```
{'a'}
```

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

Fill-ins:

- `set()`
- `count >= 2`
- `add(word)`

Completed code:

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

popular = set()

for word, count in freq.items():
    if count >= 2:
        popular.add(word)

print(popular)
```

Output:

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

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

Fill-ins:

- `in prices`
- `prices[item]`

Completed code:

```
prices = {"apple": 3, "banana": 2, "cookie": 5, "donut": 4}
allowed = {"banana", "donut", "egg"}

kept = {}

for item in allowed:
    if item in prices:
        kept[item] = prices[item]

print(kept)
```

Output:

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

Part D — Synergy problems (sets + dictionaries together)

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

Fill-ins:

- `0`
- `item in prices`
- `+= prices[item]`

Completed code:

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

total = 0

for item in cart:
    if item in prices:
        total += prices[item]

print(total)
```

Output:

```
13
```

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

Fill-ins:

- `student_to_club.items()`
- `club not in club_to_students`
- `set()`
- `student`

Completed code:

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

club_to_students = {}

for student, club in student_to_club.items():
    if club not in club_to_students:
        club_to_students[club] = set()
    club_to_students[club].add(student)

print(club_to_students)
```

One valid output is:

```
{'Robotics': {'Amy', 'Chloe'}, 'Chess': {'Ben', 'Dan'}, 'Art': {'Eva'}}
```

(Order may vary, especially inside sets.)

---

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

Fill-ins:

- `set()`
- `club_to_students.items()`
- `len(students) >= 2`
- `big_clubs.add(club)`

Completed code:

```
big_clubs = set()

for club, students in club_to_students.items():
    if len(students) >= 2:
        big_clubs.add(club)

print(big_clubs)
```

One valid output is:

```
{'Robotics', 'Chess'}
```

(Order may vary.)

---

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

1) **Answer:** `{'tag'}`

2) Code:

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

both = favorites["Ava"] & favorites["Bo"]
print(both)
```

Output:

```
{'tag'}
```

---

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

Fill-ins:

- `items()`
- `t not in type_to_names`
- `set()`
- `name`

Completed code:

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

type_to_names = {}

for name, t in pet_type.items():
    if t not in type_to_names:
        type_to_names[t] = set()
    type_to_names[t].add(name)

print(type_to_names)
```

One valid output is:

```
{'cat': {'Mochi', 'Luna'}, 'dog': {'Boba'}, 'fish': {'Nemo'}}
```

(Order may vary inside sets.)

---

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

Fill-ins:

- `set()`
- `add(snack)`

Completed code:

```
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] = set()
    kid_snacks[kid].add(snack)

print(kid_snacks)
```

Output:

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

---

19) “Same unique words?” (write code)

One solution:

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

set1 = {w.lower() for w in s1.split()}
set2 = {w.lower() for w in s2.split()}

print(set1 == set2)
```

Expected output:

```
True
```

Part E — Debugging and design

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

Corrected code:

```
seen = set()
seen.add("A")
```

---

21) Choose the best tool (multiple choice)

Answer: B (set)

---

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

Fill-ins:

- `items()`
- `w not in index`
- `doc_id`

Completed code:

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

index = {}

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

    for w in set(words):  # set(...) avoids repeating the same doc_id
        if w not in index:
            index[w] = set()
        index[w].add(doc_id)

print(index)
```

One valid output is:

```
{'red': {1, 3}, 'blue': {1, 2}, 'green': {2, 3}}
```

(Order may vary because we iterate over sets.)