Solution 19: Python Set Basics

1) Create a set (write code)

One valid answer: `python s = {4, 7, 2} print(s)`

2) Set type (fill in the blank)

`type(s)` is `<class 'set'>`.

3) No duplicates (what does it print?)

```
{1, 2, 3}
3
```

Note: the first line may display as `{1, 3, 2}` or another order; the key idea is it contains **only** 1, 2, 3.

4) True / False (set properties)

1. False
2. False
3. True
4. True

5) Membership check (what does it print?)

```
True
False
```

6) Membership check in list vs set (fill in the blanks)

```
s = set(numbers)
```

7) Count hits (write code)

One valid answer:

```
food  = ["apple", "milk", "plum", "banana", "egg", "plum"]
fruit = {"apple", "banana", "cherry", "plum"}

count = 0
for x in food:
    if x in fruit:
        count = count + 1

print(count)
```

8) Empty set vs empty dictionary (what does it print?)

```
<class 'dict'>
<class 'set'>
```

9) Don’t use index

The code is wrong. It raises a **TypeError** because sets don’t support indexing (no fixed order).

10) Remove duplicates (what does it print?)

- `set(nums)` removes duplicates, so `len(s)` is `3` because the unique numbers are `{1, 2, 3}`.

```
{1, 2, 3}
3
```

Explain: sets don’t have a fixed order, so printing a set may show items in an order that looks “random”.

11) Remove duplicates but keep order (fill in the blanks)

The blank is:

- `not in`

Full code:

```
nums = [3, 1, 3, 2, 1]

seen = set()
result = []

for x in nums:
    if x not in seen:
        result.append(x)
        seen.add(x)

print(result)
```

Output: `text [3, 1, 2]`

12) Unique words (what does it print?)

Unique words are `{"cat", "dog", "bird"}`, so:

```
3
```

13) Filter names using a “blocked” set (write code)

One valid answer:

```
names = ["Alice", "Bob", "Chelsea", "David", "Bob"]
blocked = {"Bob", "David"}

result = set()
for name in names:
    if (name not in blocked) and (name not in result):  # keep unique in result
        result.add(name)

print(result)
```

Note: Another valid approach is to use a `seen` set for the result.

14) Any overlap? (write code)

One valid answer:

```
a = [2, 4, 6, 8]
b = [1, 3, 6, 9]

s = set(a)

found = False
for x in b:
    if x in s:
        found = True
        break

print(found)
```

15) Complexity recall (fill in the blanks)

- List membership: **O(n)** on average
- Set membership: **O(1)** on average

16) Many membership checks (short answer)

A. Use `target in students` for each target.

- Average time complexity: **O(m x n)**.
- Explain:

    - Each membership check (in list) has **O(n)** average time complexity.
    - Do membership check for **m** items.
    - Totally **O(m x n)** average time complexity.

B. Convert once: `s = set(numbers)`, then use `target in s` for each target.

- Average time complexity: **O(n + m)**.
- Explain:
    - Converting list (size **n**) to set has **O(n)** average time complexity.
    - Each membership check (in list) has **O(1)** average time complexity.
    - Do membership check for **m** items.
    - Membership check has totally **O(m)** time complexity.
    - Totally **O(m + n)** time complexity.

B is usually faster because **m x n** is usually greater than **m + n**.