Worksheet 19: Python Set Basics

Name: ____________________ Date: ____________________

Instructions

- Answer in the blanks.
- For "write code" questions, write valid Python code (no functions needed).
- For "what does it print" questions, write the **exact** output (line by line), **unless** the question says "explain".

Part A — Set basics (create + properties)

1) Create a set (write code)

Write Python code to create a set named `s` that contains these items:

- `4`
- `7`
- `2`

Then print `s`.

```

```

2) Set type (fill in the blank)

What is the type of `s`?

```
s = {4, 7, 2}
print(type(s))
```

`type(s)` is `<class '__________'>`

3) No duplicates (what does it print?)

```
s = {1, 1, 2, 2, 3}
print(s)
print(len(s))
```

Output:

```

```

4) True / False (set properties)

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

1. A set can have duplicate items. ___________
2. A set has a fixed order (like a list). ___________
3. `in` on a set returns `True` or `False`. ___________
4. A set is created using braces `{ ... }` with items inside. ___________

Part B — Membership check (`in`)

5) Membership check (what does it print?)

```
members = {"Alice", "Bob", "Chelsea", "David"}

print("Chelsea" in members)
print("Cayden" in members)
```

Output:

```

```

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

Complete the code so it prints `True`.

```
numbers = [4, 7, 2, 9, 5]
target = 9

# Convert list to set (so membership check is fast)
s = ________________________

print(target in s)
```

7) Count hits (write code)

Goal: count how many items in `food` are fruits.

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

Write code to print the number of items in `food` that are also in `fruit`.

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

count = _____________

for x in ____________:
    if x in ____________:
        count = ____________

print(count)
```

Expected output:

```
4
```

Part C — Important details (empty set + no index)

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

```
a = {}
b = set()

print(type(a))
print(type(b))
```

Output:

```

```

9) Don’t use index

Here's a piece of Python code:

```
s = {"A", "B", "C"}
x = s[0]
```

Is the code right or wrong? ________________

Part D — Practical tasks with sets

10) Remove duplicates (what does it print?)

```
nums = [3, 1, 3, 2, 1, 2, 2]
s = set(nums)

print(s)
print(len(s))
```

Output:

```

```

Explain: Why might the `print(s)` show the numbers in a “weird order”?

Your answer: ____

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

Goal: remove duplicates from `nums` **but keep the first time each number appears**.

Example:

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

Fill in the blanks:

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

seen = set()
result = []

for x in nums:
    if x ___________ seen:  # in or not in?
        result.append(x)
        seen.add(x)

print(result)
```

12) Unique words (what does it print?)

```
words = ["cat", "dog", "cat", "bird", "dog", "cat"]
unique = set(words)

print(len(unique))
```

Output:

```

```

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

Goal: build a list of names that are **not** blocked. In other words, print unique names that are in `names` but not in `blocked`.

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

result = _______________  # create an empty set

for name in names:
    if (name _________ blocked) and (name _________ result):  # in or not in?
        result.______________(name)  # append or add?

print(result)
```

Expected output (order may vary):

```
['Alice', 'Chelsea']
```

14) Any overlap? (write code)

Goal: print `True` if the two lists share **at least one** common item; otherwise print `False`.

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

Hint: convert one list to a set.

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

s = ______________  # convert list a to set
found = __________  # True or False

for x in b:
    if x ______ s: # in or not in?
        found = __________  # True or False
        __________________  # stop the loop

print(found)
```

Expected output:

```
True
```

It is because both lists contain `6`.

Part E — Time complexity (conceptual)

15) Time complexity

Fill in the blanks:

- Membership check in a **list** is **O(____)** on average.
- Membership check in a **set** is **O(____)** on average.

---

16) Many membership checks (short answer)

- You have a list `students` with **n** items.
- You need to check membership for **m** different targets (many checks)

Example:

- `students = ["Alex", "Alice", ...., "Zellux"]`.
- `targets = ["Ada", "Adam", "Bobby", ...]`.
- Expected output: `[False, True, True, False, ....]`.

Two different algorithms:

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

- Average time complexity: **O(______)**.
- Explain:
    - ____
    - ____
    - ____
    - ____

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

- Average time complexity: **O(______)**.
- Explain:
    - ____
    - ____
    - ____
    - ____

Which approach is faster, and why?

Answer: ____