

Worksheet 21: Python Set Remove Duplicates

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).
 - If a question says “order may vary”, any correct order is acceptable.
-

Part A — Remove duplicates (simple)

1) `list(set(...))`

What does it print?

```
letters = ["A", "B", "C", "B", "A"]  
  
unique = list(set(letters))  
  
print(unique)  
print(len(unique))
```

Output (order may vary):

2) Numbers with duplicates

What does it print?

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

unique = list(set(numbers))

print(unique)
print(1 in set(numbers))
```

Output (order may vary):

3) Remove duplicates (write code)

You are given a list `items` .

Write code that:

1. removes duplicates using a set (simple way)
2. prints the number of **unique** items

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

# write your code below:
```

Part B — Remove duplicates (stable order)

4) Fill in the blanks (stable order)

4) Fill in the blanks (Stable Order)

Complete the code so it keeps the **first** time we see each item.

```
numbers = ["A", "B", "B", "C", "A"]

unique = []
seen = _____ # create an empty set

for x in numbers:
    if x not in _____:
        unique._____
        seen._____

print(unique)
```

Expected output:

```
['A', 'B', 'C']
```

5) Stable order output (what does it print?)

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

unique = []
seen = set()

for x in numbers:
    if x not in seen:
        unique.append(x)
        seen.add(x)

print(unique)
print(len(seen))
```

Output:

6) Fix the bug (stable order)

This code is **supposed** to remove duplicates with stable order, but it has a bug (it does not update the set).

Fix it so it prints:

```
['cat', 'dog', 'bird']
```

Buggy code:

```
animals = ["cat", "dog", "cat", "bird", "dog"]

unique = []
seen = set()

for x in animals:
    if x not in seen:
        unique.append(x)
        # BUG: missing something here

print(unique)
```

What is the missing line of code?

////////////////////////////////////

7) Trace it (fill in the table)

We run this code:

```
nums = [5, 5, 2, 5, 2, 9]

unique = []
seen = set()

for x in nums:
    if x not in seen:
        unique.append(x)
```

```
unique.append(x)
seen.add(x)
```

Fill in the table (write sets like `{5, 2}` ; order inside a set does not matter):

Step	x	unique (list)	seen (set)
0 (start)	—	<code>[]</code>	<code>set()</code>
1	5	_____	_____
2	5	_____	_____
3	2	_____	_____
4	5	_____	_____
5	2	_____	_____
6	9	<code>[5, 2, 9]</code>	<code>{5, 2, 9}</code>

8) Stable order (write code)

Write code to remove duplicates with stable order.

Print the final `unique` list.

```
words = ["hi", "hi", "bye", "hi", "yes", "bye"]

# write your code below:
```

Expected output:

```
['hi', 'bye', 'yes']
```

Part C — Why use both `unique` and `seen` ?

9) Choose the best answer (multiple choice)

Pick **one** best answer.

Why do we keep both a list `unique` and a set `seen` ?

- A. Because sets can store duplicates but lists cannot.
- B. Because lists keep order, and sets make membership checks fast.
- C. Because sets automatically sort the items.
- D. Because lists are faster than sets for membership checks.

Your answer: _____

10) Time complexity (fill in the blanks)

Fill in the blanks using `n` or `1` .

- Membership check `x in some_list` takes $O(\text{____})$ time (average).
 - Membership check `x in some_set` takes $O(\text{____})$ time (average).
-

Part D — Mini challenges

11) Count unique items (what does it print?)

```
nums = [1, 2, 2, 2, 5, 5, 7]

s = set(nums)

print(len(s))
print(7 in s)
print(3 in s)
```

Output:

12) Find the duplicates (write code)

You are given a list `items`.

Write code that builds a set named `dupes` that contains items that appear **at least twice**.

Hint:

- Use a set `seen` for items you've already visited.
- Use a set `dupes` for duplicates you found.

```
items = ["A", "B", "A", "C", "B", "B"]

seen = _____ # create an empty set
dupes = _____ # create an empty set

for x in items:
    if x in seen:
        _____.add(x)
    else:
        _____.add(x)

print(dupes)
```

Expected output (order may vary):

```
{ 'A', 'B' }
```

Reason: 'A' and 'B' repeat in the input `items`. 'C' doesn't repeat, so the output does not contain 'C'.

13) Remove duplicates from two lists (write code)

You are given two lists of names. Create one list `unique_names` that:

- contains names from `a` first (left to right),
- then names from `b` (left to right),
- and keeps **stable order** with **no duplicates**.

```
a = ["Alice", "Bob", "Bob", "Chelsea"]
b = ["Bob", "David", "Alice", "Eva"]

unique_names = _____ # create an empty list
seen = _____ # create an empty set

for x in a:
    if x _____ seen: # in or not in?
        unique_names._____(x) # append or add?
        seen. _____(x) # append or add?

for x in b:
    if x _____ seen: # in or not in?
        unique_names._____(x) # append or add?
        seen. _____(x) # append or add?

print(unique_names)
```

Expected output:

```
['Alice', 'Bob', 'Chelsea', 'David', 'Eva']
```

14) Keep the *last* time we see each item (stable last)

This time, keep the **last** time we see each item.

Example:

- `nums = [1, 2, 1, 3, 2]`
- Output should be `[1, 3, 2]` (the last 1 is before 3; the last 2 is at the end)

Fill in the blanks:

```

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

unique_rev = []
seen = set()

for x in _____:    # go from right to left
    if x not in seen:
        unique_rev.append(x)
        seen.add(x)

unique = _____    # reverse unique_rev
print(unique)

```

Expected output:

```
[1, 3, 2]
```

15) Quick duplicate check (fill in the blanks)

Complete the code so it prints `"no duplicates"`.

```

l = ["x", "y", "z"]

s = _____

if len(s) == len(l):
    print("no duplicates")
else:
    print("duplicates")

```