

# Quiz 08–18: Loops and Time Complexity

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).
  - For “time complexity” questions, choose from:  **$O(1)$ ,  $O(n)$ ,  $O(n^2)$** .
- ////////////////////////////////////

## Part A — Core ideas (concepts)

---

### 1) While-loop facts (True/False)

Circle **True** or **False**.

1. A `while` loop checks its condition **before** each loop. True / False
  2. If a loop variable never changes, the loop might become infinite. True / False
  3. `break` stops the loop immediately. True / False
- ////////////////////////////////////

### 2) “In-place” (fill in the blanks)

Choose from `do` and `don't` .

In-place means we \_\_\_\_\_ change the same list.

We \_\_\_\_\_ create a new list.

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

### 3) Two pointers (fill in the blanks)

For a two-pointer palindrome checker:

- `left` starts at \_\_\_\_\_

- `right` starts at \_\_\_\_\_
  - We keep looping while \_\_\_\_\_
- 

#### 4) Digits with `%` and `//` (fill in the blanks)

- `n % 10` gives the \_\_\_\_\_ digit of `n`. (first / last)
  - `n // 10` \_\_\_\_\_ the last digit of `n`. (keeps / removes)
- 

#### 5) Negative indices (fill in the blanks)

Given:

```
letters = ["a", "b", "c", "d", "e"]
```

- `letters[-1]` is \_\_\_\_\_
  - `letters[-4]` is \_\_\_\_\_
- 

#### 6) Quick time complexity (choose one)

```
numbers = [9, 8, 7, 6]
print(numbers[2])
```

Time complexity: \_\_\_\_\_

---

## Part B — What does it print?

---

#### 7) Running total with a stop

```
numbers = [2, 3, 4, 1]

total = 0
for x in numbers:
    total = total + x
    print(total)
    if total >= 6:
        break

print("done")
```

Output:

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

## 8) Stop at the first multiple of 4

```
nums = [3, 6, 7, 8, 10]

for x in nums:
    if x % 4 == 0:
        break
    print(x)
print("stop")
```

Output:

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

## 9) Search with a Boolean `found`

```
numbers = [1, 3, 5, 7]
target = 4

found = False
for x in numbers:
    if x == target:
        found = True
        break

print(found)
```

Output:

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

## 10) While-loop doubling

```
i = 1
while i < 10:
    print(i)
    i = i * 2
```

Output:

//

## 11) Digits trace

```
n = 402

while n > 0:
    print(n % 10)
    n = n // 10
```

Output:

//

## 12) Nested loops: build short codes

```
letters = ["X", "Y"]
digits = [1, 2]

result = []
for a in letters:
    for b in digits:
        result.append(a + str(b))

print(result)
```

Output:

---

### 13) Move zeros (scan step only)

```
numbers = [0, 2, 0, 1]

read = 0
write = 0
n = len(numbers)

while read < n:
    if numbers[read] != 0:
        numbers[write] = numbers[read]
        write = write + 1
    read = read + 1

print(numbers)
print("write =", write)
```

Output:

---

## Part C — Fill / fix / write code

### 14) Running total (fill in the blanks)

Fill the blanks so the output is exactly `Sum = 10` .

```
numbers = [1, 2, 3, 4]

total = _____
for x in _____:
    total = _____

print("Sum =", total)
```

---

### 15) Filter (fill in the blanks)

Complete the code so it prints `[12, 15]` .

```
numbers = [5, 12, 9, 15, 10]

big = []
for x in numbers:
    if x _____:
        big.append(_____)

print(big)
```

Output:

```
[12, 15]
```

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

## 16) Fix the indentation

This code should print `hi` three times, but it has an indentation bug. Fix it.

```
for _ in range(3):
print("hi")
```

Write the corrected code:

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

## 17) Fix the infinite loop (add ONE line)

This loop should print `3, 2, 1, 0` and then stop.

```
n = 3

while n >= 0:
    print(n)
    # add one line here
```

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

## 18) In-place reverse (fill in the blanks)

Complete the in-place reverse.

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

left = 0
right = len(items) - 1

while left < right:
    temp = items[_____]
    items[_____] = items[_____]
    items[_____] = temp

    left = left + 1
    right = right - 1

print(items)
```

## 19) Debug move-zeros (2 bugs)

This code is **trying** to move zeros to the end, but it has **two bugs** (Line A and Line B).

```
numbers = [0, 1, 0, 3, 0, 2]

read = 0
write = 0
n = len(numbers)

while read <= n:          # Line A
    if numbers[read] != 0:
        numbers[write] = numbers[read]
        write = write + 1    # Line B
    read = read + 1
```

Correct the lines:

- Line A: `while _____ :`
- Line B: `_____`

## 20) Write code: first even index (use `break`)

Write code to find the **index** of the **first even number** in the list.

- If found, print the index.
- If there is no even number, print `-1` .
- Use `break` .

```
numbers = [7, 9, 5, 12, 3, 8]
```

```
# write your code here
```

---

## Part D — Digits and number problems

---

### 21) By hand: sum of digits

Compute the sum of digits of `n = 2026` .

Answer: \_\_\_\_\_

### 22) Write code: count digits (while-loop)

Write code to print how many digits `n` has.

Use: `n = 87531` .

### 23) Write code: print factors

Write code to print all factors of `20` in increasing order.

```
n = 20

# write your code here
```

---

## 24) Prime check (what does it print?)

```
n = 21
is_prime = True

for i in range(2, n):
    if n % i == 0:
        is_prime = False
        break

print(is_prime)
```

Output:

---

## Part E — Two pointers and swapping

### 25) Two pointers: print first mismatch (write code)

Write code that prints the **first mismatching pair** and stops.

Use:

```
s = "ABCDXDCBA"
```

Rules: - If you find a mismatch, print the two characters (example output: `C D`) and stop. - If there is **no mismatch**, print `No mismatch`.

```
s = "ABCDXDCBA"

left = 0
right = len(s) - 1
found = False

# write your code here
```

---

## 26) Swap first and last item only (write code)

Swap only the **first** and **last** item of the list (do NOT reverse the whole list).

```
letters = ["p", "y", "t", "h", "o", "n"]

# write your code here


print(letters)
```

Expected output:

```
["n", "y", "t", "h", "o", "p"]
```

---

## Part F — Time complexity (best/worst)

## 27) Search with `break`

Assume `numbers` has length `n`.

```
found = False
for x in numbers:
    if x == target:
        found = True
        break
```

- Best case time complexity: \_\_\_\_\_
- Worst case time complexity: \_\_\_\_\_

---

## 28) Nested loops with early stop

Assume `numbers` has length `n`.

```
found = False
for i in range(n - 1):
    for j in range(i + 1, n):
        if numbers[i] + numbers[j] == target:
            found = True
            break
    if found:
        break
```

- Best case time complexity: \_\_\_\_\_
- Worst case time complexity: \_\_\_\_\_