Quiz 06-26: Lists, Sets, and Dictionaries

Name: _____ Date: _____

Instructions

- Answer in the blanks.
- 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 — Lists (create, index, update)

1) Index + update (fill in the blanks)

You are given:

```
colors = ["red", "green", "blue", "yellow"]
```

Fill in the blanks so the code changes `"blue"` to `"purple"` and then prints the list.

```
colors[__________] = "purple"
print(___________)
```

Output:

```

```

2) List + loop (short answer)

You have:

```
nums = [3, 1, 4, 1, 5]
```

Without writing code, answer:

1. What is `nums[-1]`? _____
2. What is `len(nums)`? _____

3. After `nums.append(9)`, what is the new last item? _____

3) What does it print?

```
a = [10, 20, 30]
b = a
b.append(40)
print(a)
print(b)
```

Output:

```

```

Hint:

- `b = a` means `a` and `b` point to the same list, so both prints are the same.
- When you append a new item to `b`, you also changed `a` in the same way.

4) Fill in the blanks: build a new list

Fill in the blanks so `squares` becomes `[1, 4, 9, 16]`.

```
squares = []
for x in [1, 2, 3, 4]:
    squares.append(______)
print(squares)
```

Output:

```

```

Part B — Loops (nested for + while)

5) Nested loop: count prints (fill in the blanks)

How many times does `print("hi")` run?

```
for i in range(3):
    for j in range(4):
        print("hi")
```

It runs _____ times.

6) Nested loop: coordinate pairs (fill in the blanks)

Fill in the blanks to print:

`(0,0) (0,1) (1,0) (1,1)`

Python code:

```
for r in range(_______):
    for c in range(_______):
        print(f"({r},{c})", end=" ")
```

7) While loop: stop at a sentinel (fill in the blanks)

Complete the code so it keeps adding numbers from the list until it sees a `0`. (It should **not** add the `0`.)

```
nums = [5, 2, 7, 0, 9]
i = 0
total = 0

while nums[i] != ____:
    total = total + nums[i]
    i = i + ____

print(total)
```

Output:

```

```

8) Fix the bug (nested loop)

The goal is to make `pairs` become: `[(1, 'a'), (1, 'b'), (2, 'a'), (2, 'b')]`

But the code has a bug. Fix it.

```
pairs = []
for n in [1, 2]:
    for ch in ["a", "b"]:
        pairs.append((n, n))   # BUG is here
print(pairs)
```

Correct line:

```
```

9) What does it print? (while loop)

```
x = 1
while x < 20:
    x = x * 2
print(x)
```

Output:

```
```

Part C — Sets (membership + operations)

10) Duplicates disappear (what does it print?)

```
items = ["apple", "apple", "banana", "apple", "banana"]
s = set(items)
print(len(items))
print(len(s))
```

Output:

```
```

11) Membership: list vs set (short answer)

You want to check whether `"kiwi"` is in a big collection of fruit names.

Circle the faster choice (for large data):

- list
- set

Reason:

```
A set membership check is _________ on average.
A list membership check is _________ on average.
```

12) Set operations: friends (fill in the blanks)

```
alice = {"Tom", "Mia", "Zoe", "Kai"}
bob   = {"Mia", "Kai", "Leo"}
```

Write each answer as a set, order may vary.

1. Mutual friends (intersection): {_____}
2. All friends (union): {_____}
3. Friends Alice has but Bob doesn't (difference): {_____}

13) Fix the code: empty set

The goal is to create an empty set named `seen`. Fix the code.

```
seen = {}   # BUG
```

Correct code:

```

```

Part D — Dictionaries (create + traversal + membership)

14) What does it print? (keys only)

```
d = {1: "one", 2: "two"}
print(1 in d)
print("one" in d)
```

Output:

```

```

15) Fill in the blanks: safe counting pattern

Complete the code so it counts letters in the string `s`.

```
s = "BANANA"
count = {}

for ch in s:
    if ch not in count:
        count[ch] = ____
    count[ch] = count[ch] + ____

print(count)
```

What is the output? (order may vary)

```

```

16) Traverse `.items()` (fill in the blanks)

Fill in the blanks to print each key-value pair like: `dog -> 3`

```
pets = {"dog": 3, "cat": 2}

for k, v in pets._____:
    print(k, "->", v)
```

Hint: How to traverse key-value pairs of a dictionary?

17) Find the biggest value (write code, no functions)

Given:

```
scores = {"Amy": 7, "Ben": 9, "Cara": 8}
```

Write code to print the **name** of the person with the highest score.

```
scores = {"Amy": 7, "Ben": 9, "Cara": 8}

best_name = None
best_score = -1

for name, score in _________________:
    if ___________________________:
        best_score = _________________
        best_name = _________________

print(best_name)
```

18) Bug hunt: wrong key check (fix the bug)

The goal is to print `"Yes"` if the dictionary has the key `"age"`.

```
person = {"name": "Lily", "age": 8}

if "8" in person:
    print("Yes")
else:
    print("No")
```

Fix the `if` line:

```

```

Part E — Word frequency (dict + loops + sets)

19) Fill in the blanks: word frequency

Complete the code so it builds a word frequency dictionary.

```
words = ["a", "b", "a", "c", "b", "a"]
freq = {}

for w in words:
    if w not in freq:
        freq[w] = ____
    freq[w] = freq[w] + ____

print(freq)
```

Output (order may vary):

```

```

20) Unique words + frequency (short answer)

You are given:

```
text = "to be or not to be"
words = text.split()
```

1. How many **unique** words are there? _____

2. What data type is best for storing the unique words? _____

21) Combine list + set + dict (fill in the blanks)

Goal: Count how many **different** animals appear in the list, and also count **how many times** each appears.

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

unique_animals = ____________________  # convert the list to a set
freq = {}

for a in animals:
    if a not in freq:
        freq[a] = _________
    freq[a] = freq[a] + _________

print(len(unique_animals))
print(freq)
```

Part F — Time Complexity (choose: O(1), O(n), O(n²))

22) Complexity: single loop

```
def f(nums):
    total = 0
    for x in nums:
        total += x
    return total
```

Time complexity: _____

23) Complexity: nested loops

```
def g(nums):
    count = 0
    for i in range(len(nums)):
        for j in range(len(nums)):
            count += 1
    return count
```

Time complexity: _____

24) Complexity: membership in a list

Assume `nums` is a list with `n` items.

```
def h(nums, target):
    return target in nums
```

Time complexity: _____

---

25) Complexity: membership in a set

Assume `s` is a set with `n` items.

```
def k(s, target):
    return target in s
```

Time complexity (in this quiz): _____

---

26) Choosing the faster approach (short answer)

You have a list `words` with `n` strings, and you want to count how many strings are `"cat"`.

Two ideas:

Idea A

```
count = 0
for w in words:
    if w == "cat":
        count += 1
```

Idea B

```
freq = {}
for w in words:
    freq[w] = freq.get(w, 0) + 1
count = freq.get("cat", 0)
```

1. What is the time complexity of Idea A? _____
2. What is the time complexity of Idea B? _____
3. Which idea is better if you will ask **many** different queries like `"cat"`, `"dog"`, `"bird"` after building the result? _____

---

27) List vs set membership

Consider this function:

```
def count_bad(words, bad):
    count = 0
    for w in words:
        if w in bad:
            count += 1
    return count
```

Assume `len(words) = n` and `len(bad) = n`.

1. If `bad` is a **list**, time complexity is: _____
2. If `bad` is a **set**, time complexity is: _____

Hint: Choose from: O(1), O(n), O(n²).

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

Fill in the blanks so the code prints `[1, 2, 3, 4]`.

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

seen = ________
unique = []

for x in nums:
    if x not in ________:
        unique.append(x)
        ________.add(x)

print(unique)
```

29) Nested loops + dictionary (fill in the blanks)

Fill in the blank so the code stores a small “times table” in a dictionary.

```
table = {}

for r in range(1, 4):
    for c in range(1, 4):
        table[(r, c)] = ________

print(table[(2, 3)])
```

Output:

```
```