

# Quiz 01 - Practice

COMP 110: Introduction to Programming  
Spring 2025

February 7, 2024

Name: \_\_\_\_\_

9-digit PID: \_\_\_\_\_

Do not begin until given permission.

*Honor Code: I have neither given nor received any unauthorized aid on this quiz.*

Signed: \_\_\_\_\_

**Question 1: Multiple Choice** Completely fill in the bubble next to your answer using a pencil. Each question should have exactly one filled-in bubble.

1.1. The following string is an example of a formatted string literal (f-string):

1

- True
- False

1.2. What is the printed output of the following `print` function call?

1

- `fCOMP10010`
- `COMP110`
- `C'OM'P100 + 10`
- `Error: Invalid Syntax`

1.3. What is the *type* and *evaluation* of this expression in Python?

1

- `bool, True`
- `bool, False`

1.4. What is the primary difference between keyword arguments and positional arguments in Python?

- Keyword arguments must always be passed, while positional arguments are optional.
- Positional arguments are passed based on their position in the function call, while keyword arguments are explicitly named.
- Keyword arguments can only be used in built-in functions, while positional arguments can be used in both built-in and user-defined functions.
- Positional arguments must always come after keyword arguments in a function call.

1.5. Which operator has the highest precedence in an expression?

- `or`
- `>`
- `+`
- `and`
- `not`

1.6. Which of the following statements correctly describes the behavior of the `and`, `or`, and `not` operators in Python?

- The `and` operator returns `True` if at least one operand is `True`.
- The `or` operator returns `True` only if both operands are `True`.
- The `not` operator inverts the boolean value of an expression.
- The `and`, `or`, and `not` operators can only be used with boolean values.

1.7. What is the evaluation of the following expression:

1

- `False`
- `True`

1.8. What is the evaluation of the following expression:

1

- `False`
- `True`

1.9. What is the evaluation of the following Python expression?

```
1 not True or True
```

- False
- True
- Error

1.10. Which of the following are required in a recursive function that does not infinitely recur?

- A base case without a recursive function call
- Recursive case that progresses toward the base case
- Arguments changing in the recursive case
- All of the above

1.11. Which of the following is a valid function call to the following function signature?

```
1 def ex(x: int, y: int=0) -> int:  
2 ...
```

- A. `ex()`
- B. `ex(1)`
- C. `ex(1, 2)`
- B and C
- A, B, and C
- None of the above

1.12. What type of error occurs when a function keeps calling itself, indefinitely?

- `NameError`
- `IndexError`
- `RecursionError`
- `SyntaxError`
- `NeverendingError`

1.13. What will the following Python expression evaluate to?

```
1 1 + True
```

- True
- 2
- 1
- False

1.14. Consider the following function declaration:

```
1 def ex(x: int, y: int=0) -> int:  
2 ...
```

Which of the following are valid ways of calling the function?

- A. `ex(x=1, y=2)`
- B. `ex(x=1)`
- C. `ex(1, 2)`
- A and B
- A, B, and C
- None of the above

1.15. Consider the following code. What is the problem with it?

```
1 def charli(x: int) -> int:  
2     if x <= 0:  
3         return 1  
4     return x + charli(x)
```

- `RecursionError`; line 4 should be `return x * charli(x)`
- `RecursionError`; line 4 should be `return x + charli(x - 1)`
- `RecursionError`; line 4 should be `return x + charli(x + 1)`
- Nothing.

**Question 2: Respond** to the following questions. Write a function call, if any, to yield the correct return value.

Consider the following code listing:

```
1 def eight_ball(choice: int) -> str:
2     """Returns an 8-ball response."""
3     if choice <= 0:
4         return "Unlikely."
5     else:
6         if choice > 0:
7             return "It is certain."
8         else:
9             return "Ask again later."
```

2.1. Write a function call expression to the `eight_ball` function that evaluates to "It is certain."

2.2. Write a function call expression to the `eight_ball` function that evaluates to "Unlikely."

2.3. Write a function call expression to the `eight_ball` function that evaluates to "Ask again later."

2.4. Rewrite lines 3-9 of the code listing to eliminate any unreachable code and the nested if-else statement.

**Question 3: Respond** to the following questions.

3.1. What value and type does the following expression evaluate to: `3 + 4 == 6`

3.2. What value and type does the following expression evaluate to?

```
1 ((True and False) or (False or True)) != False
```

**Question 4: Memory Diagram** Trace a memory diagram of the following code listing and then answer the sub-questions. You do not need to diagram the sub-questions.

```
1 def fib(n: int) -> int:
2     """Compute the fibonacci of n"""
3     print(f"fib({n})")
4     if n == 0 or n == 1:
5         return n
6     else:
7         return fib(n - 1) + fib(n - 2)
8
9 print(fib(3))
```

Output

Stack

Heap

Globals

**Question 5: Memory Diagram** Trace a memory diagram of the following code listing and then answer the sub-questions. You do not need to diagram the sub-questions.

```
1 def rev(src: str, i: int, dest: str) -> str:
2     if i >= len(src):
3         return dest
4     else:
5         return rev(src=src, i=i + 1, dest=src[i] + dest)
6
7
8 print(rev(src="lwo", i=0, dest=""))
```

Output

Stack

Heap

Globals

**Question 6: Function Definition Writing** Write a function definition that returns a different string, depending on the value of a given `int`. Your function definition should meet the following expectations:

- The function should be named `fizzbuzz`, have one `int` parameter named `n`, and return a `str`.
- If `n` is divisible by 3 and not 5, the function should return `"fizz"`.
- If `n` is divisible by 5 and not 3, the function should return `"buzz"`.
- If `n` is divisible by 3 AND 5, the function should return `"fizzbuzz"`.
- If `n` is not divisible by 3 OR 5, the function should return `n` as a string.
- Explicitly type your parameter and return type.

The following REPL examples demonstrate the expected functionality of your `summit` function:

```
1 >>> print(fizzbuzz(-2))
2 -2
3 >>> print(fizzbuzz(1))
4 1
5 >>> print(fizzbuzz(2))
6 2
7 >>> print(fizzbuzz(3))
8 fizz
```

```
1 >>> print(fizzbuzz(5))
2 buzz
3 >>> print(fizzbuzz(12))
4 fizz
5 >>> print(fizzbuzz(15))
6 fizzbuzz
7 >>> print(fizzbuzz(20))
8 buzz
```

6.1. Write your function definition here:

**Question 7: CHALLENGE: Recursive Function Definition Writing** Write a recursive function definition that returns the sum of all positive, even integers less than or equal to a given `int`. Your function definition should meet the following expectations:

- The function should be named `summit`, have one `int` parameter named `n`, and return an `int`.
- If `n` is negative, the function should return `-1`.
- If `n` is positive, the function should return the sum of all positive, even integers less than or equal to `n`.
- Explicitly type your parameters and return types.
- Label your base case(s) and recursive case(s).

The following REPL examples demonstrate the expected functionality of your `summit` function:

```
1 >>> summit(-2)
2 -1
3 >>> summit(1)
4 0
5 >>> summit(2)
6 2
7 >>> summit(3)
8 2
```

```
1 >>> summit(4)
2 6
3 >>> summit(5)
4 6
5 >>> summit(6)
6 12
7 >>> summit(12)
8 42
```

7.1. Write your function definition here:



*This page intentionally left blank. Do not remove from quiz packet.*