

# Final Practice Supplement

COMP 110: Introduction to Programming  
Spring 2024

Wednesday May 1, 2024

**Question 1: Loops** In this series of questions, you will trace code that modifies a boolean list `a`.

You will respond beneath each code listing by *completely shading in the squares of items whose value is assigned True*. If an error occurs during the evaluation of the loop, fill in the **Error** box and stop evaluating. If any item's value was assigned **True** prior to the error, keep its value shaded in.

You can assume `a` is initialized with *8 False elements*, as shown below, and that each question is independent of the next.

```
1 f: bool = False
2 a: list[bool] = [f, f, f, f, f, f, f, f]
```

1.1. Loop 1

```
1 i: int = 0
2 while i < len(a):
3     if i % 2 == 1 and i >= 3:
4         a[i] = True
5     i += 1
```

<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
0	1	2	3	4	5	6	7	Error

1.2. Loop 2

```
1 i: int = 1
2 while i < len(a):
3     a[i] = True
4     if i % 2 == 1:
5         i -= 1
6     else:
7         i += 2
```

<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
0	1	2	3	4	5	6	7	Error

1.3. Loop 3

```
1 i: int = len(a)
2 while i > 0:
3     a[i] = True
4     i -= 1
```

<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
0	1	2	3	4	5	6	7	Error

**Question 2: Method Writing** Complete the implementation of the `find` method. It should return the (row, column) of the `needle` parameter in the `data` attribute. If the `needle` cannot be found, return `(-1, -1)`.

```
1 class Table:
2     width: int
3     height: int
4     data: list[list[int]]
5
6     def __init__(self, width: int, height: int):
7         self.width = width
8         self.height = height
9         self.data = []
10        for _y in range(height):
11            row: list[int] = []
12            for _x in range(width):
13                row.append(0)
14            self.data.append(row)
15
16        def find(self, needle: int) -> tuple[int, int]:
17            # TODO
```

2.1. Write your function definition for `find` here.

**Question 3: Identifying Elements of a Python Class** Consider the following class definition.

```
1 class Pet:
2     name: str
3     age: int # in years
4
5     def __init__(self, name: str, age: int):
6         self.name = name
7         self.age = age
8
9     def greet(self) -> str:
10        return f"{self.name} says hello"
11
12    def ages(self, n: int) -> None:
13        """Increase the pet's age by n years."""
14        self.age += n
```

3.1. On what line(s) is a *return type* declared?  
Write *None* if none.

3.2. List the names of the *methods* defined in class *Pet*. Write *None* if none.

3.3. On what line(s) are *arguments* found?  
Write *None* if none.

3.4. On what line(s) are *docstrings* found?  
Write *None* if none.

3.5. On what line(s) are *comments* found?  
Write *None* if none.

3.6. What is another name for the definition of `__init__`?

**Question 4: Using a Class** Continuing from the code listing above, you will make use of the *Pet* class in the following questions.

4.1. Write one line of code to declare a variable named `pup`, *explicitly* of data type `Pet`, and assign it a newly constructed `Pet` object with an initialized `name` attribute value of "Ada" and `age` attribute value of 2.

4.2. Continuing from the previous sub-question, write one line of code that will cause the `pup` variable's `age` attribute to change to 3 using a *method call* on the `pup` object.

4.3. Continuing from the previous sub-question, write one line of code to declare an *explicitly typed* variable named `x`. Initialize `x` to the result of calling `greet` on `pup`.

**Question 5: Identifying Elements of a Python Program** Consider the following code listing:

```
1 def main() -> None:
2     """Entrypoint of program."""
3     start: int = int(input("Start: "))
4     end: int = int(input("End: "))
5     result: int = mystery(start, end)
6     print(f"Result: {result}")
7
8
9 def mystery(i: int, n: int, x: int = 0) -> int:
10    if i >= n:
11        return x + i
12    else:
13        return mystery(i + 1, n, x + i)
14
15 if __name__ == "__main__":
16    main()
```

5.1. On what line(s) is a *base case* declared?  
Write *None* if none.

5.2. On what line(s) is a *recursive case* declared? Write *None* if none.

5.3. Ignoring function calls to *built-in functions*, what 2 line(s) contain *function calls with arguments*?

5.4. On what line(s) are *default parameter(s)* found? Write *None* if none.

**Question 6: Evaluating Functions** These questions continue from the code listing above.

6.1. What value returns from `mystery(6, 6, 9)`? Write **Error** if an error occurs.

6.2. What value returns from `mystery(5, 6, 4)`? Write **Error** if an error occurs.

6.3. What value returns from `mystery(4, 6)`? Write **Error** if an error occurs.

6.4. What value returns from `mystery(1, 3)`? Write **Error** if an error occurs.

**Question 7: Memory Diagram** Trace a memory diagram of the following code listing. For the purposes of diagramming, you can ignore the imports, and use short-hand frames for *init*.

```
1 from typing import Self
2
3 class Vec2D:
4     x: float
5     y: float
6
7     def __init__(self, x: float, y: float):
8         self.x = x
9         self.y = y
10
11    def scale(self, factor: float) -> None:
12        self.x *= factor
13        self.y *= factor
14
15    def add(self, other: Self) -> Self:
16        return Vec2D(self.x + other.x, self.y + other.y)
17
18 a: Vec2D = Vec2D(1.0, 2.0)
19 b: Vec2D = a.add(a)
20 a.scale(3.0)
21 print(f"a:({a.x}, {a.y}) - b:({b.x}, {b.y})")
```



Stack

Heap

