Final Practice Supplement

COMP 110: Introduction to Programming Spring 2024

Wednesday May 1, 2024

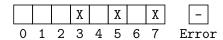
Question 1: Loops In this series of questions, you will trace code that modifies a boolean list a. You will respond beneath each code listing by completely shading in the squares of items whose value is assigned True. If an error occurs during the evaluation of the loop, fill in the Error box and stop 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 & 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]
```

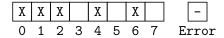
1.1. Loop 1

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



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</pre>
```



1.3. Loop 3

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



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
     data: list[list[int]]
4
5
     def __init__(self, width: int, height: int):
6
7
       self.width = width
       self.height = height
8
9
       self.data = []
       for _y in range(height):
10
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.

```
Solution: One possible solution, of many possible valid solutions:

def find(self, needle: int) -> tuple[int, int]:
    """Returns the (row, column) of the needle in data.

If the needle cannot be found, returns (-1, -1)."""

for row in range(self.height):
    for col in range(self.width):
        if self.data[row][col] == needle:
            return (row, col)
    return (-1, -1)
```

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:
       return f"{self.name} says hello"
10
11
12
     def ages(self, n: int) -> None:
       """Increase the pet's age by n years."""
13
14
       self.age += n
```

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

Solution: 9, 12

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

Solution: __init__, greet, ages

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

Solution: None

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

Solution: 13

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

Solution: 3

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

Solution: Constructor

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.

Solution: pup: Pet = Pet("Ada", 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.

Solution: pup.ages(1)

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.

Solution: x: str = pup.greet()

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

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

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

Solution: 10, 11

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

Solution: 12, 13

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

Solution: 5, 13

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

Solution: 9

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.

Solution: 15

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

Solution: 15

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

Solution: 15

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

Solution: 6

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
   a: Vec2D = Vec2D(1.0, 2.0)
18
19
   b: Vec2D = a.add(a)
20
   a.scale(3.0)
  print(f"a:({a.x}, {a.y}) - b:({b.x}, {b.y})")
```

Solution: a(3.0, 6.0) - b(2.0, 4.0)

	Globals	Vec 2D Listo	
		a lis:1	
		6 212:2	
	Vec 20# in	1 RV[1):1	
Ì	IL.		
	Vec2D#0		
	RA-[19	Self [i]:1	
	RV [i]:2 °	stry id:1	
	KV (2,2		
^	Vec 20# in(+.		
	RA (16	pv1i2:2	
	KA (15		
	Vec 2D#	= Scale	
	00120	self list	
	1 12/A (20 4	SELV [15.11]	
	RV fac	to [3.0	
	Durk		

X 4.0	Heap id: 6 [class 3-16] id: 1 Vec 2D x 183.0 y 286.0 id: 2 Vec 2D	Objects constructed initialized in short-hand init frames
	X 18 3.0 Y 28 6.0	Short-hand