COMP110 QZ03 - Version A

Sara Huston

TOTAL POINTS

43.5 / 44

QUESTION 1

8 pts

1.1 **1 / 1**

✓ **+ 1 pts** *True*

+ 0 pts False

1.2 **1 / 1**

✓ **+ 1 pts** *True*

+ 0 pts False

+ 0 pts Blank

1.3 **1 / 1**

✓ **+ 1 pts** *True*

+ 0 pts False

1.4 **1 / 1**

✓ **+ 1 pts** *constructor*

+ 0 pts creator

+ 0 pts initializer

1.5 **1 / 1**

✓ **+ 1 pts** *attributes*

+ 0 pts vals

+ 0 pts constants

+ 0 pts vars

1.6 **1 / 1**

✓ **+ 1 pts** *self*

+ 0 pts item

+ 0 pts object

+ 0 pts elem

1.7 **1 / 1**

✓ **+ 1 pts** *the name of the class it's an instance of*

+ 0 pts dependent on the arguments in the constructor

+ 0 pts dict

+ 0 pts list

1.8 **1 / 1**

✓ **+ 1 pts** *True*

+ 0 pts False

QUESTION 2

5 pts

2.1 **2 / 2**

✓ **+ 2 pts** *Correct: `hat: Clothes = Clothes(48)`*

Partial Credit (select one)

+ 1 pts Incorrect, but correctly typed the variable (`hat: Clothes`)

+ 1 pts Incorrect, but correctly called the constructor (`Clothes(48)`)

+ 1.5 pts Mostly correct, but didn't label hat with correct type. (`hat = Clothes(48)`)

+ 0 pts Incorrect

2.2 **1.5 / 2**

+ 2 pts Correct: `hat.h += 3`
or `hat.h = 51`
or `hat.h = hat.h + 3`

Partial Credit (select one)

+ 1 pts Incorrect, but used a period to access the attribute (`hat.h`)

✓ **+ 1.5 pts** *Used "self" instead of "hat"*
`self.h += 3`
or `self.h = 51`
or `self.h = self.h + 3`

+ 0.5 pts Incorrect, but used a period to access the attribute (`self.h`)

+ 0 pts Incorrect

2.3 **1 / 1**

✓ **+ 1 pts** *Correct: `hat.foo()`*

+ 0 pts Incorrect

Partial Credit

+ 0.5 pts Called .foo()

QUESTION 3

9 pts

3.1 Output **1 / 1**

✓ **+ 1 pts** *Correct: `1.0`*

+ 0.5 pts Partial Credit: `1`

+ 0 pts Incorrect

3.2 Diagram **8 / 8**

Globals

✓ **+ 0.5 pts** *Defined `MyClass` in the stack as a reference to a class definition on the heap.*

✓ **+ 0.5 pts** *Defined `h` on the stack with the value `4`*

✓ **+ 0.5 pts** *Defined `i` on the stack with the value `8.0`*

✓ **+ 0.5 pts** *`j` is a reference to the same `MyClass` object on the heap as the `RV` for `MyClass#__init__`*

✓ **+ 0.5 pts** *`j`' has an `h` attribute with the value 4*

✓ **+ 0.5 pts** *`j`' has an `i` attribute with the initial value `8.0` and final value `1.0`*

`MyClass#__init__`

✓ **+ 0.5 pts** *Frame made and labeled `MyClass#__init__`*

✓ **+ 0.5 pts** *`RA` is `15`*

✓ **+ 0.5 pts** *`self` points to a `MyClass` object on the heap*

✓ **+ 0.5 pts** *`h` has the value `4`*

✓ **+ 0.5 pts** *`i` has the value `8.0`*

✓ **+ 0.5 pts** *`RV` is a reference to the same `MyClass` object on the heap as `self`*

`MyClass#foo`

✓ **+ 0.5 pts** *Frame made and labeled `MyClass#foo`*

✓ **+ 0.5 pts** *`RA` is `16`*

✓ **+ 0.5 pts** *`self` is a reference to a `MyClass` object on the heap*

✓ **+ 0.5 pts** *`i` is `4`*

- 0.5 pts Extra incorrect value

+ 0 pts Blank or Incorrect

QUESTION 4

12 pts

4.1 Output **2 / 2**

✓ **+ 2 pts** *Correct:*
`Hello`

`['Hello', 'World']`

(Quotes don't matter)

Partial Credit

+ 1 pts Incorrect, but printed out `Hello`

+ 1 pts Incorrect, but printed out `['Hello', 'World']`

+ 0 pts Incorrect

4.2 Diagram **10 / 10**

Globals

✓ **+ 0.5 pts** *Defined `Animals` in the stack as a reference to a class definition on the heap.*

✓ **+ 0.5 pts** *`hamster` is a reference to the same `Animals` object on the heap as the `RV` for `Animals#__init__`*

✓ **+ 1 pts** *`hamster` has an `h` attribute that points to a `list[str]` on the heap with elements `["Hello", "World"]`*

`Animals#__init__`

✓ **+ 0.5 pts** *Frame made and labeled `Animals#__init__`*

✓ **+ 0.5 pts** *`RA` is `15`*

✓ **+ 0.5 pts** *`self` points to an `Animals` object on the heap*

✓ **+ 0.5 pts** *`RV` is a reference to the same `Animals` object on the heap as `self`*

`Animals#insert` (first frame)

✓ **+ 0.5 pts** *Frame made and labeled `Animals#insert`*

✓ **+ 0.5 pts** *`RA` is `16`*

✓ **+ 0.5 pts** *`self` is a reference to same `Animals` object on the heap as global variable `hamster`*

✓ **+ 0.5 pts** *`j` is `'"Hello"`*

`Animals#insert` (second frame)

✓ **+ 0.5 pts** *Frame made and labeled `Animals#insert`*

✓ **+ 0.5 pts** *`RA` is `17`*

✓ **+ 0.5 pts** *`self` is a reference to same `Animals` object on the heap as global variable `hamster`*

✓ **+ 0.5 pts** *`j` is `'"World"`*

`Animals#__str__`

✓ **+ 0.5 pts** *`RA` is 18*

✓ **+ 0.5 pts** *`self` is a reference to same `Animals` object on the heap as global variable `hamster`*

✓ **+ 0.5 pts** *`result` is `'"Hello"`*

✓ **+ 0.5 pts** *`RV` is `'"Hello"`*

- 0.5 pts Extra incorrect value

+ 0 pts Blank or Incorrect

QUESTION 5

5 **10 / 10**

Class Fundamentals

✓ **+ 1 pts** *Defined class using `class Icecream:`*

✓ **+ 1 pts** *Listed attributes as:*

`count: int`

`info: str`

`__init__`

✓ **+ 1 pts** *Signature: `def __init__(self, count_val: int, info_val: str):`*

(It's ok if they put a different return type.)

✓ **+ 0.5 pts** *Sets `self.count = count_val`*

✓ **+ 0.5 pts** *Sets `self.info = info_val`*

`__mul__`

✓ **+ 1 pts** *Signature: `def __mul__(self, factor: int) ->*

Icecream:`

+ 0.5 pts (Partial Credit) Signature: `def __mul__(self, factor: int):`

✓ **+ 1 pts** *Creates a new `IceCream` object using `IceCream()` with a value equal to `factor*self.count` as the first argument and any string value as the second argument.*

✓ **+ 1 pts** *Returns new `IceCream` object*

`rename`

✓ **+ 1 pts** *Signature:*
`def rename(self, new_name: str) -> None:`
or
`def rename(self, new_name: str):`

+ 0.5 pts (Partial Credit) Signature: `def rename(self, new_name: str) -> str:`

✓ **+ 1 pts** *Updates `self.info = new_name`*

✓ **+ 1 pts** *Does NOT return anything. (`return None` is fine!)*

+ 0 pts Click here to replace this description.

+ 0 pts Incorrect or Blank

gradescope

Quiz 03 - A

COMP 110: Introduction to Programming and Data Science Fall 2023

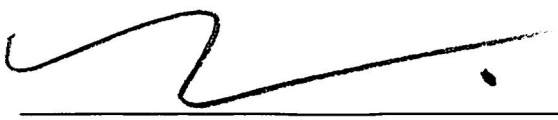
Nov 28, 2023

Name: Sara Huston

9-digit PID: 730459812

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 Answer the following questions about Object Oriented Programming.

1.1. True or False: In Python, a class is a blueprint for creating objects.

- (●) True
- ( ) False

1.2. True or False: A method in Python is a function that is called on an instance of a class

- (●) True
- ( ) False

1.3. True or False: Magic methods are called automatically when certain events occur, such as creating an object or performing arithmetic operations.

- (●) True
- ( ) False

1.4. A ____ is a magic method that gets called when an object is created.

- ( ) initializer
- ( ) creator
- (●) constructor

1.5. Each instance of a class will have its own variables with unique values. There variables are called:

- ( ) constants
- ( ) vals
- (●) attributes
- ( ) vars

1.6. The first parameter in the `__init__` method is named ____.

- ( ) `object`
- ( ) `item`
- (●) `self`
- ( ) `elem`

1.7. The **type** of an object of a class is ____.

- ( ) dependent on the arguments in the constructor
- ( ) `dict`
- ( ) `list`
- (●) the name of the class it's an instance of

1.8. True or False: In their definitions, magic methods have double underscores before and after their names.

- (●) True
- ( ) False

Question 2: Object Oriented Programming Short Answer Suppose you have the following class. Answer the related questions below.

```
1  class Clothes:
2
3    h : int
4
5    def __init__(self, h: int):
6      self.h = h
```

2.1. Write a line of code to create an instance of `Clothes` called `hat`, setting self.h equal to 48. (Remember to label `hat` with the correct *type*!)

```
hat : Clothes = Clothes(48)
```

2.2. Write a line of code to increase the value of hat's h attribute by 3.

```
self.h += 3
```

2.3. Write a line of code to call the method `foo` on hat with no arguments.

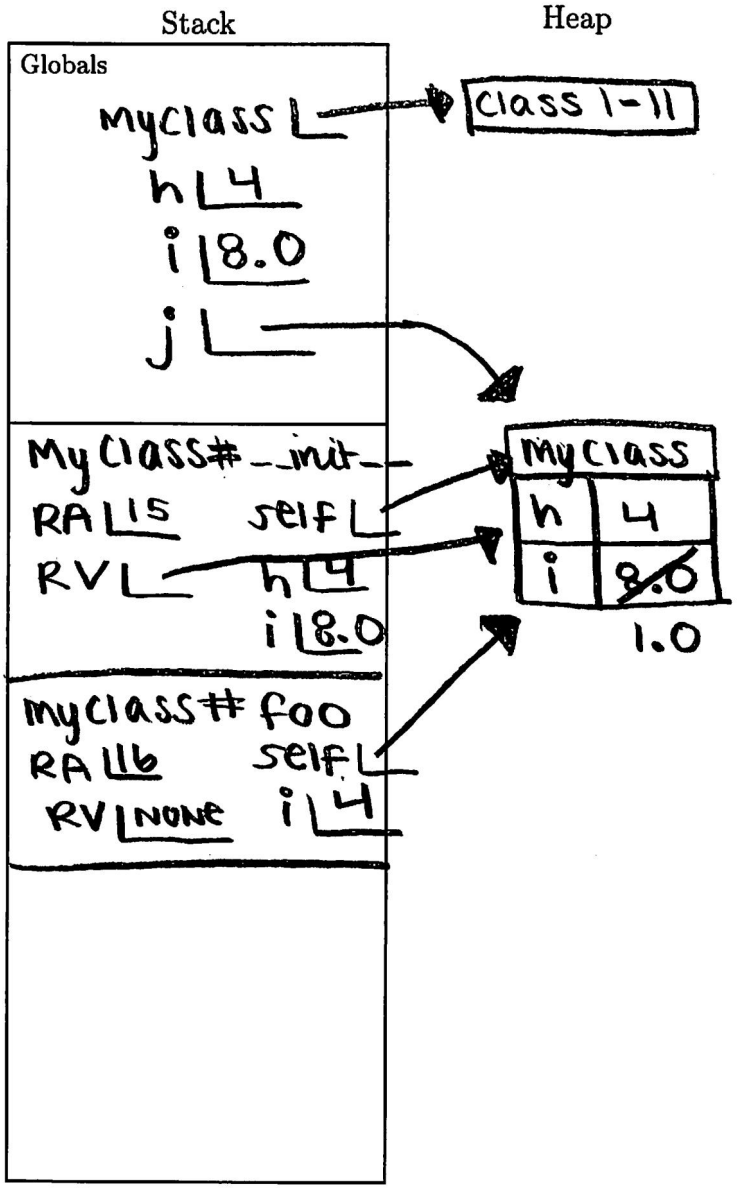
```
hat.foo()
```

Question 3: Trace a memory diagram of the following code listing.

```
1  class MyClass:
2
3    h: int
4    i: float
5
6    def __init__(self, h: int, i:
       float):
7      self.h = h
8      self.i = i
9
10   def foo(self, i: int):
11     self.i = self.h / i
12
13 h: int = 4
14 i: float = 8.0
15 j: MyClass = MyClass(h,i)
16 j.foo(h)
17 print(j.i)
```

Output

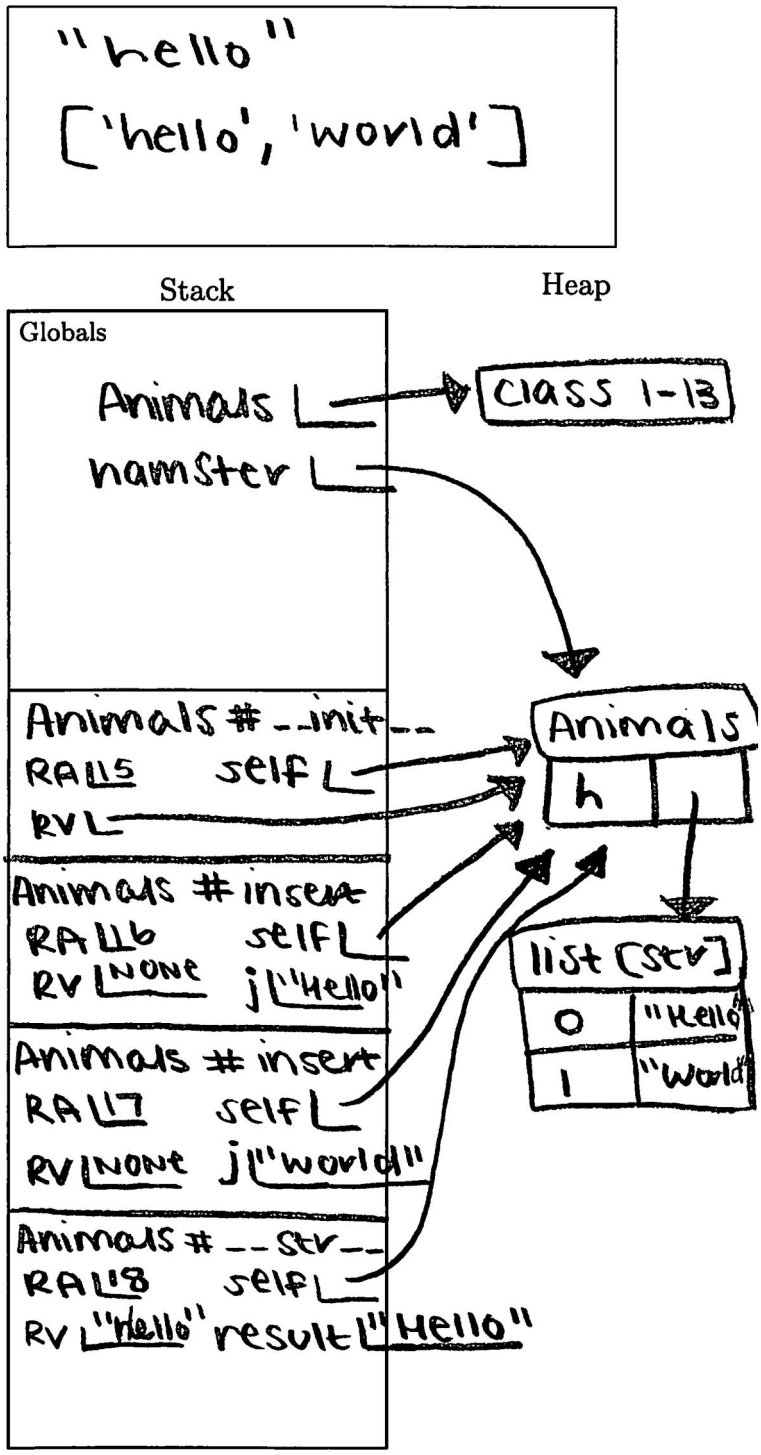
1.0



Question 4: Trace a memory diagram of the following code listing.

```
1  class Animals:
2
3    h: list[str]
4
5    def __init__(self):
6      self.h = []
7
8    def __str__(self)-> str:
9      result: str = self.h[0]
10     return result
11
12   def insert(self, j: str)-> None:
13     self.h.append(j)
14
15 hamster: Animals = Animals()
16 hamster.insert("Hello")
17 hamster.insert("World")
18 print(hamster)
19 print(hamster.h)
```

Output



Question 5: Class Writing Write a class definition with the following expectations:

- The Class is named `Icecream`.
- Icecream has two attributes: one named `count` with type `int` and one named `info` with type `str`.
- Write an `__init__` method that takes `count_val:int` and `info_val:str` as arguments and initializes `count` to equal `count_val` and `info` to equal `info_val`.
- Write a magic method `__mul__` that takes `self` and `factor:int` as arguments and returns a new Icecream object with `count` equal to `self`'s `count` attribute multiplied by `factor`.
- Write a method named `rename` that takes `self` and `new_name:str` as arguments and updates `self`'s `info` attribute to now equal `new_name`.
- Explicitly type variables, parameters, and return types.
- The following REPL examples demonstrate sample usage of the `Icecream` class:

```
>>> x: Icecream = Icecream(3,"praline")
>>> print(x.info)
"praline"
>>> x.rename("strawberry")
>>> print(x.info)
"strawberry"
>>> y: Icecream = x * 2
>>> print(y.count)
6
```

Write your class definition on the following blank page.

5.1. Write your class definition for Icecream here.

```
class Icecream:
    """creating class for icecream."""

    count: int
    info: str

    def __init__(self, count_val: int=0, info_val: str=""):
        """constructing an Icecream."""
        self.count = count_val
        self.info = info_val

    def __mul__(self, factor: int) -> Icecream:
        """magic multiplier method."""
        new_ice: Icecream = Icecream()
        new_ice.info = self.info
        new_ice.count = self.count * factor

        return new_ice

    def rename(self, new_name: str):
        """updates name."""
        self.info = new_name
```

This page intentionally left blank. Do not remove from exam packet.