Alan Pham

13/22

DCS 229: Quiz 1 **Scope, Classes & Inheritance** **Winter 2025**

Instructions: Complete this quiz without referring to any notes (other than your one-page sheet of notes), online resources, executing the code, or talking with anyone else. Just write your answers on separate paper, clearly indicating what question you are answering. Please hand in your note-sheet with your answers.

0/2 1. In an intro course, we often say that `int` is a function, used to convert a string to its integer equivalent (see below). Briefly discuss why thinking of `int` as a function is inaccurate, presenting a better description of what is taking place.

```
five = int("5")
```

2/3 2. Explain the difference between a class and an object. Also provide a specific example.

3. Consider the code implemented below. (a) Explain why "`[1,2,3,1]`" is printed. (b) Should `items` on lines 1 and 2 instead be called `data`? Why or why not?

2/3

```
1. def func(items: list[int]) -> None:
2.     items[-1] = items[0]
3.
4. def main() -> None:
5.     data = [1,2,3,4]
6.     func(data)
7.     print(data)
8.
9. main()
```

5. 0/2

6. 5/6

4. Consider the class implementations below.

```
class Upper:
    __slots__ = ('_a', '_b')
    def __init__(self, one: int, two: int) -> None:
        self._a = one
        self._b = two

    def setValues(self, one: int, two: int) -> None:
        self._a = one + one
        self._b = two + two

    def __str__(self) -> str: return f"{self._a}:{self._b}"

class Lower(Upper):
    __slots__ = ('_c', '_d')
    def __init__(self, one: int, two: int, thr: int) -> None:
        super().__init__(one, two)
        self._c = thr
        self._d = thr + thr

    def setValues(self, one: int, two: int, thr: int) -> None:
        self._a = one * one
        self._b = two * two
        self._c = thr * thr
        self._d = thr * thr * thr
```

4/6

(a) What instance variables and methods would an `Upper` object have?

(b) What instance variables and methods would a `Lower` object have?

(c) After executing the code below, what would the values of the instance variables inside `lower` be? Explain why.

(d) When exeucting the code below, what gets printed? Explain why.

```
lower = Lower(11, 22, 33)
lower.setValues(1, 2, 3)  # think carefully about what is being called
print(lower)              # think carefully about what is being called
```

Alan Pham

1) Describing Int as a function is inaccurate because Int is just pointing to a memory location with the value or string given. A function normally executes things but Int is just pointing somewhere in the memory

Int: a class
creating an int object (-2)

2) Everything in python is an object, which includes classes. objects are anything that is stored in the memory while classes ~~class~~ contain methods. ✓ but objects have methods
For example the registrar has a student class that takes in ~~object~~ info/parameters to create an object that has student data in it. ok here

class is blueprint of a specific type of data (+1)

3) a) line 5: stores list [1, 2, 3, 4] into variable data
line 7: print variable data

but talk mem addr or mutability (-1)

b) It doesnt matter what you name "items" in lines 1 & 2 so long as they are the same, the "func" function doesnt save anything that happens & so the list ends up printing the same. ok

4) a) Upper:
+1/1 methods: __init__, setValues, __str__
instance variables: _a, _b

b) Lower:
+1/1 Methods: __init__, setValues & inherited methods from upper: __str__
instance variables: _a, _b, _c, _d

c) _a = 11, _b = 22, _c = 33, _d = 66
+1/2 _a & _b gets stored normally from the parent class _c also just gets stored from input and _d is created by doubling up the 3rd value & storing.

[11, 22, 33, 66]
but lower.setValues(1,2,3) changes

d) [1, 4, a, 27]
lower = Lower[11, 22, 33] stores the values [11, 22, 33, 66] from __init__
+1/2 lower.setValues(1, 2, 3) changes the values in the object to specific multiplication giving [1, 4, 9, 27]

1:4 6/c --str-- inherited

5) Yes, because of polymorphism allows other class methods to be reused and repurposed, as long as the argument type corresponds.

0/2

no ... whether the object has getPerimeter regardless of type

6)

```
class schedule:
    __slots__ :('_course', '_list_classes')
    def __init__ (self, course_name: str,
        self._course = course_name

    def add (self, course_name: str)
        self._list_classes.append [course_name]
    def isTaking (self, course_name
```

```
class schedule:
    __slots__ = ('_course', '_catalogue')
    def __init__ (self, course: str)
        self._course = course
    def add (self, course: str)
        self._catalogue = append [course]
    def isTaking (self, course: str)
        for i in self._catalogue:
            if self._catalogue[i] == course
                return True
        return False
```

no, no arg here - see spec

you never defined the list

either i is an index (need range) or i is referring to a course in the list

you are mixing

schedule

-- slots = (course, - catalogue #) giving one variable & saving variable for list (draft paper

init # takes in course & stores

self.course

add - course # take given name & add to internal list

self. - catalogue. append [- course] ?

istaking - course,

for i in - catalogue

if _catalogue[i] == # _course

return True

return false