CS303E Week 7 Worksheet: Objects and Classes

Name: _____ EID: _____

Read the questions carefully, and answer each question in the space provided. Use scratch paper to do your work and then copy your answers neatly and legibly onto the test paper. Only answers recorded on the test paper will be graded.

1. (11 points: 1 point each) The following are true/false questions. **Write either T or F in the boxes at the bottom of page 1.** If there's any counterexample, it's false.

   (a) Python allows you to create a class without any attributes or methods.

   (b) Variables defined in a class's `__init__` require the `"self"` prefix to be recognized as an instance attribute.

   (c) All user-defined classes in Python are mutable.

   (d) In Python, if a class's `__str__` method is not defined, you can directly print an object of that class.

   (e) In Python, it is possible to create an instance of a user-defined class without explicitly defining a constructor (`__init__`) method.

   (f) All attributes of a class in Python can be accessed and modified from outside the class.

   (g) Modifying an attribute of one instance of a class affects the same attribute in all other instances of that class.

   (h) When creating a class object, it is necessary to explicitly provide values for all instance attributes.

   (i) In Python class method definitions, `"self"` is a reference to a class instance, and using 'self' allows the method to access the attributes of that particular instance.

   (j) You can change the class of an object after it has been created in Python.

   (k) In Python, if a class defines attributes in its `__init__` method, all instances of that class will have those attributes.

a	b	c	d	e	f	g	h	i	j	k
										.

Page total: _____/11

Questions 2-7 are multiple choice. Each counts 2 points. **Write the letter of the BEST answer in the box on the next page. Please write your answer in UPPERCASE. Each problem has a single answer.**

2. Why does the `__str__` method in Python need to return strings instead of using `print()` directly?

   A. Using `print()` within `__str__` would cause an infinite loop, because print implicitly calls `__str__`.

   B. The `__str__` method is used for the creation of the string to print, not for the printing itself.

   C. `print()` can only be used within the constructor, not in methods.

   D. Using `print()` in `__str__` is not erroneous but it makes the code less efficient and slower to execute.

3. Why are getter or setter methods useful?

   A. Getter and setter methods are required to define a class.

   B. For an attribute to be changed after it is initialized, a setter method is required.

   C. Getter and setter methods help us understand the structure of a class.

   D. Getter and setter methods provide a way to access and modify private attributes of a class.

4. Which of the following is true about classes?

   A. `"self"` can be used outside of a class to refer to specific instances.

   B. Defining only the `__gt__` (greater than) and `__ge__` (greater than or equal to) comparisons methods in a class handles all comparisons.

   C. If the `__eq__` method for a class is not explicitly defined, then comparing two instances of this class by doing `X == Y` will cause an error.

   D. Using `type()` on any user-defined class will always return back 'Object'.

5. When might the result of `X is Y` be different from `X == Y`?

   A. When comparing two lists that have the same elements.

   B. When comparing two strings with the same content stored in different variables.

   C. When comparing two integers having the same value.

   D. All of the above.

   E. Never; they are equivalent expressions.

   F. B and C

6. What is the difference between functions and methods?

   A. Functions and methods are interchangeable terms in Python.

   B. Functions can only take one parameter, while methods can take multiple parameters.

   C. Functions can be defined independently, while methods are always associated with a class or object.

   D. Functions return values, while methods perform actions without returning any value.

7. Which of the following scenarios benefit from creating a class in Python?

   A. Building a representation of a real-world object with various measurements and behaviors.

   B. When you want to override or customize "magic methods" in Python, such as `__str__`.

   C. Grouping together specific data and and multiple functions that act on that data.

   D. To create blocks of code that perform a specific task.

   E. All of the above

   F. Both A and B

   G. A, B, and C

2	3	4	5	6	7

Page total: _____/12

The following 8 questions require you to trace the behavior of some Python code and identify the output of that code. For each question, write the output for the code segment on the provided line.

```
class Cow:
    def __init__(self, name, milkPerHour):
        self.__name = name
        self.__milkPerHour = milkPerHour

    def makeMilk(self, hoursSpent):
        gallonsMilk = self.__milkPerHour * hoursSpent
        print(gallonsMilk)

    def getSpeed(self):
        return self.__milkPerHour

    def __lt__(self, other):
        return not other.getSpeed() < self.__milkPerHour

    def __eq__(self, other):
        return other.getSpeed() == self.__milkPerHour
```

8. (3 points)

```
moomoo = Cow("MooMoo", 20)
print(str(moomoo.getSpeed()) + moomoo.__name)
```

9. (3 points)

```
moomoo = Cow("MooMoo", 20)
elsie = Cow("Elsie", 25)
print(moomoo == elsie, moomoo < elsie, moomoo > elsie)
```

Page total: _____/6

10. (3 points)

```
moomoo = Cow("MooMoo", 20)
milks = moomoo.makeMilk(10)

if milks < 10:
    print("Not much milk...")
else:
    print("Tons of milk!")
```

```
class Doll:
    def __init__(self, name):
        self.name = name

    def __str__(self):
        return "Hi, " + self.name + "!"

    def dreamhouse():
        print("Come on, Barbie, let's go party!")

    def __add__(self, other):
        return Doll(self.name + " (and " + other.name  + ")")
```

11. (3 points)

```
barbie = Doll("Barbie")
barbie2 = Doll("Barbie")
print(str(barbie) == str(barbie2), barbie == barbie2)
```

12. (3 points)

```
barbie = Doll("Barbie")
ken = Doll("Ken")

ship = barbie + ken
print(ship)
```

Page total: _____/9

13. (3 points)

```
barbie = Doll("Barbie") # this question and the next are
barbie.dreamhouse()     # only meant to get you thinking!
```

14. (3 points)

```
Doll.dreamhouse()
```

```
class smileyFace:
    def __str__(self):
        return "=)"
```

15. (3 points)

```
cheery = smileyFace()
print(cheery)
```

Page total: _____/9