CS 111 - Winter 2020
Quiz 4 Practice Questions

The quiz will be worth 30 points, and each point should correspond roughly to one minute of your time. This list of practice questions is not necessarily representative of the length of the quiz – it is hopefully longer to give you more practice, but there might be more questions in a given category on the actual quiz.

True/False questions (approximately 2 points)

1) Functions that live in objects are called methods.

2) The first parameter of a Python method definition should be called `this`.

3) If an object has instance variables, it cannot use local variables.

4) In a Python class, the constructor is called `__init__`.

5) Instance variables go away once a method terminates.

6) It is considered bad style to directly access an instance variable outside of a class definition.

7) A tuple is similar to an immutable list.

8) All elements of a list must have the same type.

9) A list must contain at least one item.

10) Object-oriented design is the process of finding and defining a useful set of functions for solving a problem.

11) Polymorphism is when one class derives its behavors from another.

12) Hiding the details of an object in a class definition is called instantiation.

13) A parent class inherits behaviors from its child class(es).

14) A greedy algorithm is guaranteed to find the best solution.

15) A greedy algorithm is useful particularly when the brute-force approach requires an exponential number of steps to complete.

Multiple-choice questions (approximately 3 points)

1) What Python keyword starts a class definition?
(a) `def` **(b)** `class` **(c)** `object` **(d)** `__init__`

2) A method definition with two formal parameters is generally called with how many actual parameters?
(a) zero **(b)** one **(c)** two **(d)** three

3) A method definition is most similar to a(n)
(a) loop **(b)** module **(c)** `import` statement **(d)** function definition

4) A Python convention for defining methods that are "private" to a class is to begin the method name with
(a) "private" **(b)** a pound sign (#) **(c)** an underscore (_) **(d)** a hyphen (-)

5) The term applied to hiding details inside class definitions is
(a) obscuring **(b)** subclassing **(c)** documentation **(d)** encapsulation

6) Which of the following is not a characteristic of a Python list?
(a) It is an object. **(b)** It is a sequence. **(c)** It can hold objects. **(d)** It is immutable.

7) What keyword parameter is used to send a key-function to the `sort` method?
(a) `get` **(b)** `key` **(c)** `reverse` **(d)** `cmp`

Code prediction (approximately 4 points)

1) Show the output that would result from the following nonsense program:

```
class Bozo:
    def __init__(self, value):
        print("Creating a Bozo from:", value)
        self.value = 2 * value

    def clown(self, x):
        print("Clowning:", x)
        print(x * self.value)
        return x + self.value

def main():
    print("Clowning around now.")
    c1 = Bozo(3)
    c2 = Bozo(4)
    print(c1.clown(3))
    print(c2.clown(c1.clown(2)))

if __name__ == "__main__":
    main()
```

2) # The mini-quiz questions from Lesson 20 on list methods.

3) # The mini-quiz question from Lesson 22 on inheritance.

Short answer (approximately 3 points)

1) List one similarity and one difference between instance variables and local (“regular”) variables.

2) List one similarity and one difference between instance variables and method parameters.

3) What is one example of encapsulation that we’ve seen in class?

4) What is one example of polymorphism that we’ve seen in class?

5) What is one example of inheritance that we’ve seen in class?

Algorithm prediction (approximately 3 points)

1) Partition the following tasks into the three bins, each of size 5.

A	1
B	2
C	4
D	3
E	2
F	1
G	2

2) Solve the Knapsack Problem with a greedy choice function that chooses based on highest value first, breaking ties by choosing lowest cost, assuming your maximum cost is 8. Give your chosen letters in order. What is the value you obtained?

Name	Cost	Value
A	5	6
B	2	10
C	4	5
D	3	6
E	2	4
F	1	5
G	2	3

3) Solve the previous problem again, but assuming you choose based on the highest value-to-cost ratio, breaking ties by lowest cost. What is your chosen set of letters, and your final value?

Rewriting code (approximately 3 points)

1) # The mini-quiz question from Lesson 21 on the `key` parameter to `<list>.sort()`.

2) Rewrite the following program, which is all in `main`, to use a `House` class instead. The `House` class should have instance variables for the body and the roof (what is "has") and methods to draw and move the house (what is "does").

Original code:

```
from graphics import *

def main():
    # Make the window
    win = GraphWin("A house!", 800, 600)

    # Choose the initial position
    p1x = 100 # left
    p1y = 100 # top
    p2x = 250 # right
    p2y = 150 # bottom

    # Make the body of the house
    body = Rectangle(Point(p1x, p1y), Point(p2x, p2y))
    body.setFill("tan")

    # Make the roof of the house
    roof = Polygon([Point(p1x, p1y),
                    Point(p2x, p1y),
                    Point((p1x+p2x)/2.0, p1y - (p2y-p1y)*0.4)])
    roof.setFill("darkGray")

    # Draw the house
    body.draw(win)
    roof.draw(win)

    # Wait for the user to click, and move the house with each click
    for i in range(5):
        p = win.getMouse()

        # Figure out where the center of the house currently is
        prevX = (body.getP1().getX() + body.getP2().getX()) / 2.0
        prevY = (body.getP1().getY() + body.getP2().getY()) / 2.0

        # Move the house
        dx = p.getX() - prevX
        dy = p.getY() - prevY
        body.move(dx, dy)
        roof.move(dx, dy)

    win.close()

if __name__ == "__main__":
    main()
```

After your changes, your new program should look like this:

```
class House:
    # TODO

def main():
    # Make the window
    win = GraphWin("A house!", 800, 600)

    # Choose the initial position
    p1x = 100 # left
    p1y = 100 # top
    p2x = 250 # right
    p2y = 150 # bottom

    # Make the house
    house = House(Point(p1x, p1y), Point(p2x, p2y))

    # Draw the house
    house.draw(win)

    for i in range(5):
        p = win.getMouse()

        # Move the house
        house.moveTo(p)

    win.close()

if __name__ == "__main__":
    main()
```

Fixing bugs (approximately 4 points)

1) The following program has bugs in five lines. Identify them and fix them.

```
 1   class Student:
 2       def init(self, name, year, major):
 3           self.name = name
 4           self.year = year
 5           self.major = major
 6
 7       def getName(self):
 8           return name
 9
10       def getYear(self):
11           return self.year
12
13       def getMajor(self):
14           return self.major
15
16       def __str__():
17           print(self.name + \
18                 " (year: " + str(self.year) + ", major: " + str(self.major) + ")")
19
20   def main():
21       lulu = Student("Lulu", 3, "Climbing")
22       therese = Student("Therese", 5, "Sleeping")
23       colin = Student("Colin", 1, "Climbing")
24
25       students = [lulu, therese, colin]
26
27       # Sort by major, then by name
28       students.sort(key = lambda s: (s.getMajor, s.getName))
29
30       # Print the students
31       for student in students:
32           print(student)
33
34   if __name__ == "__main__":
35       main()
```

Writing code (approximately 8 points)

1) Write a class to represent spheres. Your class should implement the following methods:

- `__init__(self, radius)` - Creates a sphere having the given `radius`.
- `getRadius(self)` - Returns the radius of this sphere.
- `getSurfaceArea(self)` - Returns the surface area of the sphere.
- `getVolume(self)` - Returns the volume of the sphere.
- `changeRadius(self, newR)` - Updates the radius of the sphere.
- `__str__(self)` - Returns a string of the form "sphere (radius=4.0)".

Note that the surface area of a sphere with radius r is given by $4\pi r^2$, and the volume is given by $4/3\pi r^3$.

2) The Python list method `index` results in a ValueError if the element is not present in the list. Write a function `find(myList, x)` that takes in a list and an element, and returns the index of the first occurence of that element in the list if it is present, or -1 if it is not. You should try to use the list method `index` as part of your implementation. (Hint: The keyword `in` might be helpful to you.)

3) Complete the previous problem, but do it *without* using the list method `index`.