

Name: _____

Section: 1 2

1 = Mutchler, 2nd-3rd periods. 2 = Mutchler, 4th-5th periods.

Use this quiz to help make sure you understand the videos/reading. **Answer all questions.** Make additional notes as desired. **Not sure of an answer?** Ask your instructor to explain in class and revise as needed then. **Please print two-sided if practical.**

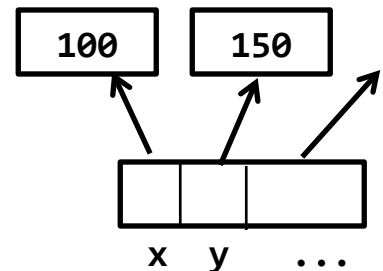
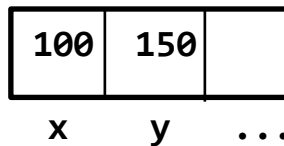
Throughout, where you are asked to “circle your choice”, you can circle or underline it (whichever you prefer).

Online reading: Box-and-Pointer Diagrams and Mutable Objects

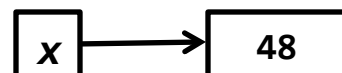
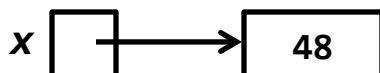
1. True or false: **Variables are REFERENCES to objects.** True False (circle your choice)
2. True or false: **Assignment** (e.g. `x = 100`) causes a variable to refer to an object. True False (circle your choice)
3. True or false: **Function calls** (e.g. `foo(54, x)`) also cause variables to refer to objects. True False (circle your choice)
4. Give one example of an object that is a **container** object:
5. Give one example of an object that is **NOT** a **container** object:
6. Which of the following demonstrates the correct way to depict a **NON-container** object in a box-and-pointer diagram? (Circle your choice.)



7. Which of the diagrams shown to the right demonstrates the correct way to depict a **container** object (here, a **z.g.Point**) in a box-and-pointer diagram? (Circle your choice.)

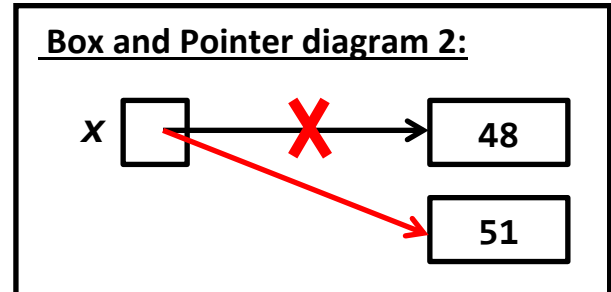
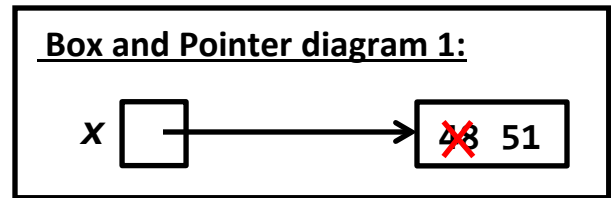


8. Which of the following demonstrates the correct way to depict an **assignment** statement in a box-and-pointer diagram? (Circle your choice.)



9. Which of the box-and-pointer diagrams shown to the right more accurately reflects the execution of the statements shown in the box below. (Circle your choice.)

```
x = 48
x = x + 3
```



10. True or false: When an object is mutated, it no longer refers to the same object to which it referred prior to the mutating. (circle your choice)

True False

11. Consider the following statements:

```
c1 = zg.Circle(zg.Point(200, 200), 25)
c2 = c1
```

At this point, how many *zg.Circle* objects have been constructed? (circle your choice)

1 2

12. Continuing the previous problem, consider an additional statement that follows the preceding two statements:

```
c1.radius = 77
```

After the above statement executes, the variable *c1* refers to the same object to which it referred prior to this statement. (circle your choice)

True False

13. Continuing the previous problems:

- What is the value of *c1*'s radius after the statement in the previous problem executes? 25 77 (circle your choice)
- What is the value of *c2*'s radius after the statement in the previous problem executes? 25 77 (circle your choice)

14. Which of the following two statements mutates an object? (Circle your choice.)

```
numbers1 = numbers2
```

```
numbers1[0] = numbers2[0]
```

15. Mutable objects are good because:

16. Explain briefly why mutable objects are dangerous.

17. Circle each of the following which is a **mutable** object:

- A ***zg.Rectangle*** object
- A ***Create*** object (as we use in our robotics exercises).
- **[3, 6, 22]**
- **(3, 6, 22)**
- **zg.Point(100, 30)**
- **(100, 30)**
- **100**
- **'Jello'**
- **'ice cream'**
- **"ice cream"**
- **True**
- **88.6**
- **zg.Point(88.6, 88.6)**
- A ***zg.Entry*** object
- **math.pi**
- **turtle.Turtle()**

Textbook RE-reading: Section 6.1.4 —List References (bottom of page 281)

You read this subsection for a previous session, but it is worth reading again in order to answer the following questions:

18. When the code snippet below is executed, what gets printed?

```
seq1 = ['John', 'Paul', 'George', 'Pete']
seq1[1] = 'Sir Paul'

seq2 = seq1
seq2[3] = 'Ringo'

print(seq1[1], seq2[1])

print(seq1[3], seq2[3])
```

Output (fill in the blanks):

19. In the above code snippet, there are 3 assignment statements:

```
seq1[1] = 'Sir Paul'
```

```
seq2 = seq1
```

```
seq2[3] = 'Ringo'
```

Which of the above assignment statements **mutates** a list? (Circle your choice(s).)

20. Continuing to refer to the code snippet in the box above, suppose that we wanted the code snippet to end in the following state:

- **seq2** is a **COPY** of **seq1** that contains the same information as **seq1** but with the index **3** item of **seq2** being **'Ringo'** instead of **'Pete'** (while the index **3** item of **seq1** remains **'Ringo'**).

How could we change the 3rd line above (that is, the statement: **seq2 = seq1**) to accomplish this effect?

Hint: Nothing in your reading tells you the answer to this question, so ask for the answer in class!