Name: _____ **Section:** 1 2 3 4

1 = Mutchler, 1st-2nd periods. **2 =** Mutchler, 3rd-4th periods. **3 =** Anderson, 7th-8th periods. **4 =** Anderson, 9th-10th 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.

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

Video: ***The Debugger*** [13:04 minutes]

Note: Don't try to learn all the details of the Debugger – just breeze through this video to get the ideas, and return to it later as needed. These questions highlight the main points:

1. When the coder finds and fixes errors in her code, we call that: _____.

2. Technically, ***testing*** is the process one does ***after*** producing the code (or at least after producing its first version), while ***debugging*** is what the coder does to find and fix errors ***while*** producing the code.

   However, we will use the terms ***debugging*** and ***testing*** interchangeably in CSSE 120. **Yes** **No** (circle your choice)

3. True or false: A debugger lets you set a ***breakpoint*** and run the program to that point, pausing the execution at the breakpoint. **True** **False** (circle your choice)

4. True or false: A debugger lets you ***step*** through a program, line by line. **True** **False** (circle your choice)

5. True or false: A debugger lets you ***see the values of the variables*** in your program and how they change as you step through program. **True** **False** (circle your choice)

6. Examine the code snippet to the right.

   a. At what lines have breakpoints been set?

   b. At what line is the program currently paused?

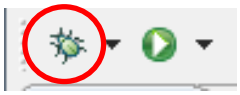
   c. At this point:

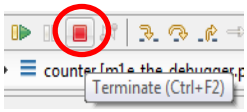
   - Has the program just executed line 37, ***or***
   - Is the program just about to execute line 37?

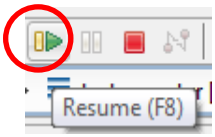
   (Circle your choice.)

```
21  def debugger_example(n, color):
22      """
26      window = zg.GraphWin('A window with
27      x = 100
28      y = 50
29      radius = 25
30      for k in range(n):
31          sink = math.sin(k)
32          k_to_kth = k ** k
33          print(k, sink, k_to_kth, x, y)
34
35          p = zg.Point(x, y)
36          c = zg.Circle(p, radius)
37          c.setFill(color)
38          c.draw(window)
39
40          x = x + (2 * radius)
41          y = y + radius
42
```

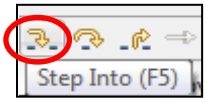
7. State, very briefly (just a few words for each is enough), what each of the following circled-in-red symbols do in the debugger:



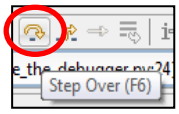




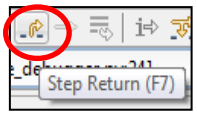
8. The ***Step Into*** button runs the next statement in the program. Explain, very briefly:



   a. How is the ***Step Over*** button different (in what it does) from Step Into?



   b. How is the ***Step Return*** button different (in what it does) from Step Into?



Video: ***The Accumulator Pattern – Summing*** [7:32 minutes]

9. Trace the snippet of code shown to the right ***by hand*** (no fair typing it into a program), and ***show what gets printed***:

```
total = 0
for k in range(5):
    total = total + (k + 10)
    print(k, total)

print('The sum 10 + 11 + 12 + 13 + 14 is')
print(total)
```

What gets printed?

```
The sum 10 + 11 + 12 + 13 + 14 is
```

10. Write a snippet of code that calculates:

    sine(3) + sine(4) + sine(5) + ... + sine(500)

    Assume that there is already an **import math** that executed previously in the code.

Textbook Reading: ***Section 5.1 – Functions as Black Boxes*** (pages 220 – 221)

11. Consider the function call **`round(3.14159, 2)`**, which rounds **`3.14159`** to **2** decimal places.[1]

    a. What are the ***arguments***: _____

    b. What is the ***return value***? _____

12. True or False: As a ***user*** of a function (that is, as someone who will ***call*** the function), you *don't need to know how the function is* ***implemented***; you just need to know the ***specification*** of the function. **True** **False** (circle your choice)

Textbook Reading: ***Section 5.2 – Implementing and Testing Functions*** (pages 222 – 225)

13. Consider the ***cubeVolume*** function defined to the right. What are the values of:[2]

```
def cubeVolume(sideLength):
    volume = sideLength ** 3
    return volume
```

    a. **`cubeVolume(3)`** _____

    b. **`cubeVolume(cubeVolume(2))`** _____

14. Continuing to use the ***cubeVolume*** function defined above, provide an ***alternate implementation*** of the body of the ***cubeVolume*** that does ***not*** use the exponent operator. (Write your answer in the box.)

```
def cubeVolume(sideLength):
```

15. Consider the ***mystery*** function defined to the right. What are the values of:[3]

```
def mystery(x, y):
    result = (x + y) / (y - x)
    return result
```

    a. **`mystery(2, 3)`** _____

    b. **`mystery(3, 2)`** _____

[1] This problem is taken from Self Check problem 1 in Chapter 5 of your textbook.
[2] This problem is taken from Self Check problems 5 and 6 in Chapter 5 of your textbook.
[3] This problem is taken from Self Check problem 9 in Chapter 5 of your textbook.

Handout: ***Functions with Parameters and Returned Values*** and
Textbook Reading: ***Section 5.3 – Parameter Passing*** (pages 226 – 228)

16. What gets printed when ***main*** is called in the program shown below? (Pay close attention to the order in which the statements are executed. ***Write the output in a column to the right of the program.***)

```
def main():
    hello()
    goodbye()
    hello_and_goodbye()
    goodbye()

def hello():
    print('Hello!')

def goodbye():
    print('Ciao!')

def hello_and_goodbye():
    print('Here is stuff!')
    goodbye()
    hello()
    hello()
    print('Here is more!')
    hello()
    goodbye()
```

Output

17. What gets printed when ***main*** is called in the program shown to the right?[4]

```
def main():
    a = 4
    answer = mystery(a + 1)
    print(answer)

def mystery(x):
    y = x * x
    return y
```

[4] This problem is taken from Self Check problem 11 in Chapter 5 of your textbook.

18. What gets printed when ***main*** is called in the program shown to the right? (Pay close attention to the order in which the statements are executed. ***Write the output in a column to the right of the program.***)

```
def main():
    big()
    bigger()
    biggest()
    big()

def big():
    print('basketball')

def bigger():
    print('truck')
    big()

def biggest():
    print('house')
    bigger()
    big()
```

Output

19. What gets printed when ***main*** is called in the program shown to the right?[5]

```
def main():
    a = 4
    print(mystery(a + 1))

def mystery(x):
    return x * x
```

20. Consider the ***totalCents*** function shown below and to the right. This function correctly calculates and returns the number of cents that is equivalent to a given number of dollars and cents. For example, **totalCents(3, 71)** correctly returns **371**.

```
def totalCents(dollars, cents):
    cents = (dollars * 100) + cents
    return cents
```

However, this function violates a style rule: ***Do Not Modify Parameter Values*** (in a function's body). This style rule is a good rule because modifying parameter values:

a. Yields ugly code.
b. Is an error-prone practice.
c. Causes the sky to fall.
d. Makes Pointy-Headed Managers unhappy.

```
def totalCents(dollars, cents):

    _____________________________________

    return _____________________________________
```

Circle your choice, and ALSO: Show in the box to the right how one could write **totalCents** ***without*** violating the *Do Not Modify Parameter Values* rule.

[5] This problem is taken from Self Check problem 11 in Chapter 5 of your textbook.

Textbook Reading: ***Section 5.5 – Functions without Return Values*** (pages 237 – 238)

21. As your textbook explains, the ***boxString*** function takes a string as its argument and displays that string "in a box". For your convenience, we show the function definition and a sample below.[6]

```
def boxString(contents):
    n = len(contents)
    print('-' * (n + 2))
    print('!' + contents + '!')
    print('-' * (n + 2))
```

Calling **boxString** with *'Hello Moon'* as its argument yields the following:

```
------------
!Hello Moon!
------------
```

Consider the following (silly!) statement:

```
print(boxString('Hello'))
```

a. What, exactly, does the above statement cause to appear on the Console?

b. How *should* the above statement been written, to be sensible?

c. Write statements that would use ***boxString*** to produce on the Console the output shown to the right.

```
-------
!Hello!
-------
------
!Moon!
------
```

[6] This problem is taken from Self Check problems 16 and 17 in Chapter 5 of your textbook.

Textbook Reading: ***Section 5.8 – Variable Scope*** (pages 251 – 253) [and also questions that summarize much of the previous videos/reading]

22. For each of the following boxes:

    - If the code is correct, state what gets printed when main runs.
    - If the code is wrong, explain why.

    For this and all subsequent problems, assume that ***no global variables have been defined***.

```
def main():
    x = foo()
    print(x)

def foo(m):
    return m ** 3
```

Correct? If so, prints _____

Wrong?

If so, explain why:

```
def main():
    x = foo(m)
    print(x)

def foo(m):
    return m ** 3
```

Correct? If so, prints _____

Wrong?

If so, explain why:

```
def main():
    x = foo('help')
    print(x)

def foo(m):
    return m ** 3
```

Correct? If so, prints _____

Wrong?

If so, explain why:

23. The code in the box to the right has syntax errors: it causes big red **X** error message(s). Circle the line(s) that will have red **X** error message(s) beside them and explain why those line(s) will have those **Xs**.

```
def main():
    foo()
    print(n)
    print(m)

def foo():
    n = 3
    m = 1
    return m
```

24. Suppose you want to write a function called ***foo*** that has two ***zg.Point*** objects sent to it and does something with them. Write the "header" line of ***foo***, that is the line that begins with **def**. Hint: Use good style!

```
def _____:
```

25. Suppose you want to write a function called ***blah*** that takes a ***zg.Point*** object and a ***zg.GraphWin*** object (in that order) and does something with them. Write the "header" line of ***blah***, that is the line that begins with **def**. Hint: Use good style!

```
def _____:
```

26. What gets printed when ***main*** is called in the program shown below? (Pay close attention to the order in which the statements are executed. ***Write the output in a column to the right of the program.***)

```
def main():
    a = 2
    b = 3

    m = do_it(a, b)
    print(m)

    m = do_it(b, a)
    print(m)

    m = do_it(a, a)
    print(m)

    m = do_it(b, b)
    print(m)

    c = do_it(b, a)
    m = do_it(c, a)
    print(m)

    b = do_it(b, a)
    m = do_it(b, a)
    print(m)


def do_it(x, y):
    return x ** y
```

Output

Note that **do_it** does exponentiation (there are 2 asterisks, not 1).

27. What gets printed when ***main*** is called in the program shown below? (Pay close attention to the order in which the statements are executed. ***Write the output in a column to the right of the program.***)

```
def main():
    a = 2
    b = 3

    foo1()
    print(a, b)

    foo2(a, b)
    print(a, b)

    foo3(a, b)
    print(a, b)

def foo1():
    a = 88
    b = 99

def foo2(a, b):
    a = 400
    b = 500

def foo3(x, y):
    x = 44
    y = 55
```

Output