3. WRITTEN RESPONSES

3 a.

3.a.i.

The purpose is to display custom arithmetic and geometric sequences by setting inputs such as where they start, how long they are, and how much they increment by, or also by displaying the constant Fibonacci sequence.

3.a.ii.

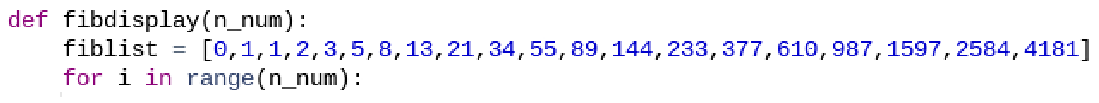
The program, given the initial value, length, and common ratio(multiplied by initial value for succeeding terms) of the chosen geometric sequence, not only listed all of the first eight terms of the sequence through graphic Text objects on the Python 3 Graphics Editor, but also the number of the term through adjacent Text objects. There is also a title to the game and some instructions about it. Where the common ratio is less than 1 and greater than -1, the terms approach zero over time, and “Eventually converges to zero” is also displayed on the canvas. Pressing the redo button asks those questions again, so I can display infinite sequences.

3.a.iii.

The input is "geometric", "8" terms, initial value: "9", common ratio: "0.8", and the x and y coordination for pressing the Redo button. The output is the title, the instructions, the rounded sequence, the term numbers, and the redo button on the canvas.

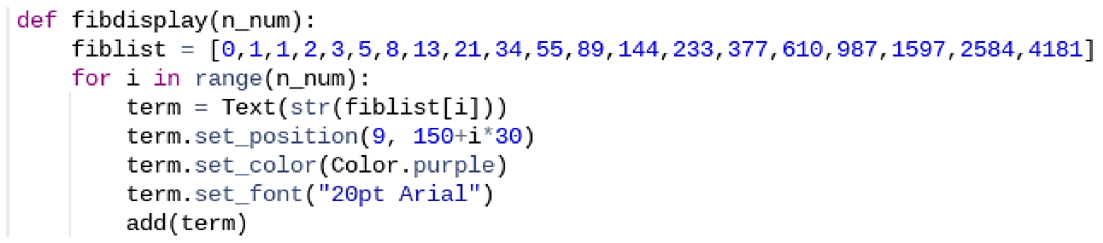
3 b.

3.b.i.



```
def fibdisplay(n_num):
    fiblist = [0,1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597,2584,4181]
    for i in range(n_num):
```

3.b.ii.



```
def fibdisplay(n_num):
    fiblist = [0,1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597,2584,4181]
    for i in range(n_num):
        term = Text(str(fiblist[i]))
        term.set_position(9, 150+i*30)
        term.set_color(Color.purple)
        term.set_font("20pt Arial")
        add(term)
```

3.b.iii.

fiblist

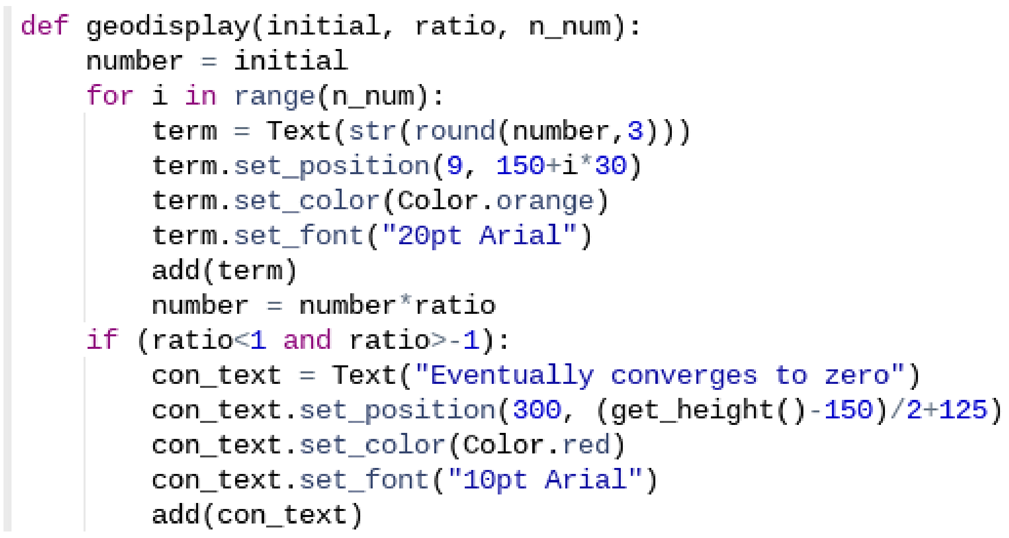
3.b.iv.

A user can display the Fibonacci sequence from 5 to 20 terms. A loop iterates a set number of times by going through the sequences left-to-right via the index and displaying the values through Text objects.

3.b.v.

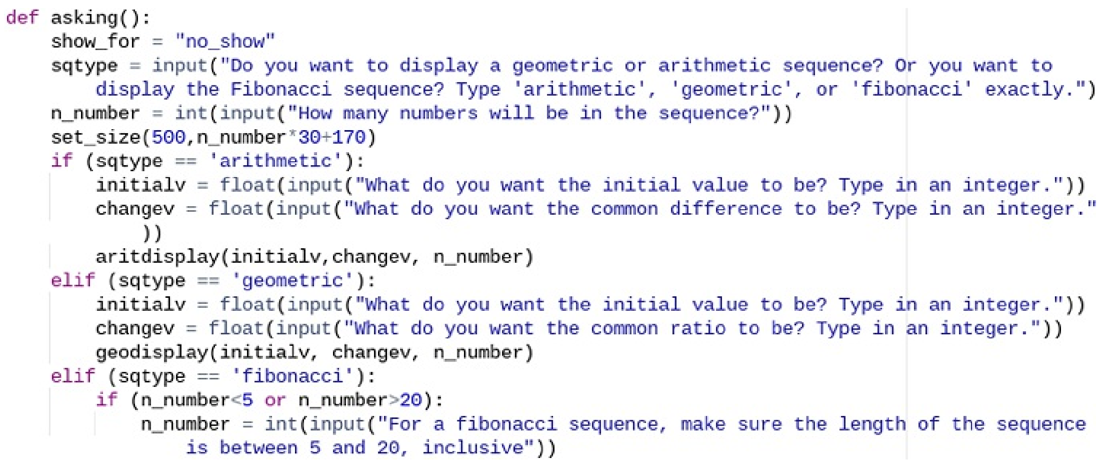
Calculating and incrementing a pattern(because each number is the addition of the two previous ones), would have been too complicated to code as two exchanging variables are needed to generate the Fibonacci sequence. So instead, I created a list of the 20 first internet-searched values of the sequence called
fiblist
.

3 c.
3.c.i.



```
def geodisplay(initial, ratio, n_num):
    number = initial
    for i in range(n_num):
        term = Text(str(round(number,3)))
        term.set_position(9, 150+i*30)
        term.set_color(Color.orange)
        term.set_font("20pt Arial")
        add(term)
        number = number*ratio
    if (ratio<1 and ratio>-1):
        con_text = Text("Eventually converges to zero")
        con_text.set_position(300, (get_height()-150)/2+125)
        con_text.set_color(Color.red)
        con_text.set_font("10pt Arial")
        add(con_text)
```

3.c.ii.



```
def asking():
    show_for = "no_show"
    sqtype = input("Do you want to display a geometric or arithmetic sequence? Or you want to
        display the Fibonacci sequence? Type 'arithmetic', 'geometric', or 'fibonacci' exactly.")
    n_number = int(input("How many numbers will be in the sequence?"))
    set_size(500,n_number*30+170)
    if (sqtype == 'arithmetic'):
        initialv = float(input("What do you want the initial value to be? Type in an integer."))
        changev = float(input("What do you want the common difference to be? Type in an integer."
            ))
        aritdisplay(initialv,changev, n_number)
    elif (sqtype == 'geometric'):
        initialv = float(input("What do you want the initial value to be? Type in an integer."))
        changev = float(input("What do you want the common ratio to be? Type in an integer."))
        geodisplay(initialv, changev, n_number)
    elif (sqtype == 'fibonacci'):
        if (n_number<5 or n_number>20):
            n_number = int(input("For a fibonacci sequence, make sure the length of the sequence
                is between 5 and 20, inclusive"))
```

3.c.iii.

It is modified through parameters(from user input) to iterate through and display any geometric sequence given the initial value, sequence length, and common ratio. It finds the pattern between each number in the sequence and displays each one(and rounds it to 3 decimal places if needed) as Text on the canvas individually through a loop. The function gives a user a wider range of options to learn in the program in addition to arithmetic sequences.

3.c.iv.

Geodisplay function takes three numerical parameters meaning the initial value, the number of terms, and the common ratio of the sequence. First, you set a variable called *number* *and set it equal to the initial value. Then you start a for loop that increments another variable i* *by 1 for each run of the loop, from 0 to the number of terms. For each iteration you display a different Text object that will display number* *rounded to 3 decimal places if needed. Each will be in 20-point font Arial and orange, and each Text's position should have an x of only 9, while the varying y position should be 150 PLUS (i TIMES 30). After the text is displayed, you change the number to get the next term of the sequence by setting number* *equal to the number* *times the common ratio(the definition of a geometric sequence). Then the loop iterates, i increases by one, and the changing numbers continue to get displayed until i reaches the number of terms. Then, geodisplay checks if the common ratio is less than 1 AND greater than -1. If so, another Text object displays "eventually converges to zero", which is in 10-point Arial font and red. Its position is 300 and its y-position is half the distance between the height of the canvas and the 150-y-position mark. In this way, the text is somewhat centered vertically between the edge of the canvas and the text on the top.*

3 d.

3.d.i.

First call:

In geodisplay(10, 2, 9), the user could instruct a geometric sequence with an initial value of 10, a common ratio of 2, and a sequence length of 9 to display.

Second call:

In geodisplay(9, 0.9, 5), the user could instruct a geometric sequence with an initial value of 9, a common ratio of 0.9, and a sequence length of 5 to display. Text also shows to inform the user that the sequence approaches 0 over the terms.

3 d.ii.

Condition(s) tested by first call:

Geodisplay(10, 2, 9) has a common ratio(the middle parameter) of 2, which is greater than -1 but also NOT less than 1, so the condition is not satisfied. The function only acts to display the sequence.

Condition(s) tested by second call:

Geodisplay(10, 2, 9) has a common ratio(the middle parameter) of 0.9. The code predicts that the sequence eventually converges to 0 because the ratio is less than 1 and greater than -1 (0.9), so it also displays the text "eventually converges to zero".

3.d.iii.

Results of the first call:

Each term, or number in the sequence, is displayed vertically going down through the Text object on the canvas. The numbers displayed are 10(the initial value), 20, 40, 80, 160, 320, 640, 1280, and finally 2560(9 terms).

Results of the second call:

Each term, or number in the sequence, is displayed vertically going down through the Text object on the canvas. Specifically, the numbers displayed are 9, 8.1, 7.29, 6.561, and finally 5.905(5 terms). Technically, the 5th term should be 5.9049, but the answer was rounded to three decimal places when displayed. Lastly, it displays the text "eventually converges to zero."