How to Use the Provided Python Files in “Holt\_Working Code” Folder

For every file in bold below…

1. Right-click on the file and go to “Open With -> IDLE.”
   1. If you do not see “IDLE” as an option, you must install Python at <https://www.python.org/downloads/>
2. Once the .py file is open, press **“fn”** followed by the **F5 key** to run the program
3. Then proceed as instructed below depending on the individual file

**norm\_plus\_breakings.py (FINDING A VECTOR’S NORM VALUE AND BREAKINGS THAT YIELD IT)**

1. You will be prompted to enter “a sequence of numbers (without []'s and with

each number separated by a single space without commas),” so do so in the following manner…

* To see what the norm is for the vector (1,1,.5,.5,.5,.5,.5,.5), you would type **1 1 .5 .5 .5 .5 .5 .5** and then press **ENTER**

2. After hitting ENTER, you will be prompted for a norm level that you want to see, so simply enter a number and press **ENTER** to get the result

3. To better understand how the breakings are represented, see “How breakings are represented in output of ‘norm\_plus\_breakings.py’” on the next page of this document.

**extreme\_pts.py (FINDING WHICH VECTORS WITH SLIGHTLY BUMPED COORDINATES DO NOT CHANGE THE NORM VALUE)**

1. Follow the same instructions as above for “norm\_plus\_breakings.py”
2. You will be given output in the form of all the vectors that are simply your input vector with one or more of its coordinates changed by an epsilon that maintain the same norm value.

**just\_norm\_value.py (FINDING A VECTOR’S NORM VALUE)**

1. Follow the same instructions as above for “norm\_plus\_breakings.py”
2. You will be given just the norm value, without the breakings.

**breaking\_possibilities.py (RETURN ALL POSSIBLE BREAKINGS FOR A GIVEN SIZE >= 7)**

1. You will be prompted for a vector length, so input a length greater than or equal to 7.
2. You will be given a list of all possible breakings for vectors of your input length.

How breakings are represented in output of “norm\_plus\_breakings.py”:

1. If you see JUST ONE SET OF PARENTHESES, there are two cases…
2. If there is ONLY ONE NUMBER in parentheses, then we are just taking the zero norm (with max at an index of that number). NO PENALTY OF ½ PAID in this case.
3. If the parentheses contain more than one number, then we are taking the first Schreier norm with indices of the given numbers. PAY A PENALTY OF ½ in this case.
   1. Example: If x = (0,1,1) and output breaking is (2,3), we have a norm value of ½(1+1).
4. If you see MORE THAN ONE SET OF PARENTHESES, this represents a breaking of our sequence into subsequences. PAY A PENALTY OF ½ for this initial breaking. Then treat each inner set using rules A.1 and A.2 from above.
   * 1. Example: If x = (1,1,.5,.5,.5,.5,.5,.5) and one output breaking is ((2,), (4,5,6,7)) like in “Test 1” below, the two inner sets of parentheses mean we are dealing with an interval breaking. In particular, our Schreier set (2,4) did this breaking, and we are taking just the item at index 2 from the first subsequence from the breaking (since (2,) is just ONE NUMBER) and we are adding up (and then dividing by 2) the fourth, fifth, sixth, and seventh term from the second subsequence from the breaking (since (4,5,6,7) has more than one number).
     2. Example: If x = (1,1,.5,.5,.5,.5,.5,.5) and two output breakings are ((4,), (5,), (6,), (7,)) and (4, 5, 6, 7) like in “Test 1” below, here is what is happening… ((4,), (5,), (6,), (7,)) has more than one set of parentheses, so we are dealing with an interval break. In particular, we broke the sequence into four subsequences using (4,5,6,7), and then we just took the max of each of these subsequences. On the contrary, (4, 5, 6, 7) refers back to rule A.2, so we would have a norm value of ½(.5 + .5 + .5 + .5). Therefore, ((4,), (5,), (6,), (7,)) and (4, 5, 6, 7) yield the same output, but they come from different level norms.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Below are a couple tests done when running “norm\_plus\_breakings.py.” If you were to run tests by running my code in the Python shell, you would see all below text that is black (your input values) and blue (computer’s work). Any text you see in red is just comments I added in after the fact to add clarity.

Test 1.

Give me a sequence of numbers (without []'s and with

each number separated by a single space without commas): 1 1 .5 .5 .5 .5 .5 .5

Give me a non-negative integer to indicate which level

norm you want me to return: 2

The level 2 norm is 1.00

due to the following sequences resulting in this value:

(1,)

(2,)

(4, 5, 6, 7) #This gives a value of ½ (0.5 + 0.5 + 0.5 + 0.5), or 1.

(4, 5, 6, 8)

(4, 5, 7, 8)

(4, 6, 7, 8)

(5, 6, 7, 8)

((2,), (4, 5, 6, 7))

((2,), (4, 5, 6, 8))

((2,), (4, 5, 7, 8))

((2,), (4, 6, 7, 8))

((2,), (5, 6, 7, 8))

((3,), (4,), (5, 6, 7, 8))

((3,), (4, 5, 6, 7), (8,)) #This gives a value of ½ (0.5 + ½ (0.5 + 0.5 + 0.5 + 0.5) + 0.5), or 1.

((4,), (5,), (6,), (7,))

((4,), (5,), (6,), (8,))

((4,), (5,), (6,), (7, 8)) #This gives a value of ½ (0.5 + 0.5 + 0.5 + ½ (0.5 + 0.5)), or 1.

((4,), (5,), (7,), (8,))

((4,), (5,), (6, 7), (8,))

((4,), (6,), (7,), (8,))

((4,), (5, 6), (7,), (8,))

((5,), (6,), (7,), (8,))

((4, 5), (6,), (7,), (8,))

Test 2.

Give me a sequence of numbers (without []'s and with

each number separated by a single space without commas): 0 0 0 1 1 1 0 .25 .25 .25 .25 .25 .25 .25 .25

Give me a non-negative integer to indicate which level

norm you want me to return: 2

The level 2 norm is 2.00

due to the following sequences resulting in this value:

((4,), (5,), (6,), (8, 9, 10, 11, 12, 13, 14, 15))