**Program analysis**

‘diamonds.py’ is rather complex considering the previous examples and if you find it hard to follow there is something you can do that may help de-mystify it and allow you to understand what is going on. This is also useful if you ever encounter a peculiar bug or if the program does something you don’t understand, and may allow you to pinpoint the reason why.

Read through the program a few times, each line at a time. Then, with a pencil and paper, follow the program as though you were the interpreter, starting from 'main()'. This is also known as tracing the program. This can be a rather time consuming activity if the program in question is large.

Write down each variable as you encounter it and change them as the program advances. It may also be a good idea to note the line number you are currently on.

In 'diamonds.py' our analysis might look like the following:

1. main() is a function call, so let's enter that function

2. We ask the user for a number, so let's say they enter 2.45

2.1. Following the order of operations, we first force the value to be an integer so it becomes 2, then we multiply it by 2 to become 4.

loops = 4

3. We then ask the user for another number, so let's say they enter the number 7.

3.1. We then force this number to be an integer so it becomes 7.

size = 7

4. We then call a function called 'diamonds' and pass it 'loops' and 'size'.

5. We start at the top of 'diamonds'.

5.1. We create two variables to represent what our diamond and space will be drawn as, both are strings.

char\_block = "#"

char\_space = " "

5.3. We then create two variables to track what is to be drawn to the screen each loop, these start off as empty strings.

text = ""

space = ""

6. So far we have the following variables:

loops = 4

size = 7

char\_block = "#"

char\_space = " "

text = ""

space = ""

7. The next step is to fill out 'space' with 'char\_space' since the diamond should be surrounded with space.

while(len(space) < size):

translates to:

while(0 < 7):

The loop itself executes the following whenever the condition is

true:

space = space + char\_space

which translates to:

"" = "" + " "

which will result in the following operations:

len(space) = 0

"" = "" + " "

len(space) = 1

" " = " " + " "

len(space) = 2

" " = " " + " "

len(space) = 3

" " = " " + " "

len(space) = 4

" " = " " + " "

len(space) = 5

" " = " " + " "

len(space) = 6

" " = " " + " "

len(space) = 7

space = " "

8. Once this is done we print out space twice on the same line to precede the first diamond with empty space, for neatness' sake.

print(space \* 2)

which translates to:

print(" ")

9. Now we can start drawing our diamond. We will continue to do this so long as x is less than loops.

for x in range(loops):

translates to:

for each loop to a maximum of 4

10. Then we ask if the following statement is true or false.

if(x % 2 == 0):

which translates to:

if(the remainder of 0 / 2 is 0):

10.1. In this case this statement is true.

10.2. We ask this because there are two phases to drawing our diamond. First we must grow the diamond then we must shrink it.

10.2.1. When count is an even number, we grow the diamond.

10.2.2. When count is an odd number, we shrink the diamond.

11. We now do a while loop to start drawing our diamond.

while((len(text) / 2) < size):

translates to:

while((0 / 2) < 7):

11.1. If we don't divide 'len(text)' by 2 our diamond will be drawn at half the size.

11.2. This while loop does a few things.

text = text + (char\_block \* 2)

space = space.replace(char\_space, "", 1)

print(space + text + space)

which translates to:

"" = "" + "##"

replace 1 " " in " " with ""

print(" " + "##" + " ")

11.3. Our loop will therefore do the following operations.

((0 / 2) < 7):

"" = "" + "##"

replace 1 " " in " " with ""

print(" " + "##" + " ")

((2 / 2) < 7)

"##" = "##" + "##"

replace 1 " " in " " with ""

print(" " + "####" + " ")

((4 / 2) < 7)

"####" = "####" + "##"

replace 1 " " in " " with ""

print(" " + "######" + " ")

((6 / 2) < 7)

"######" = "######" + "##"

replace 1 " " in " " with ""

print(" " + "########" + " ")

((8 / 2) < 7)

"########" = "########" + "##"

replace 1 " " in " " with ""

print(" " + "##########" + " ")

((10 / 2) < 7)

"##########" = "##########" + "##"

replace 1 " " in " " with ""

print(" " + "############" + " ")

((12 / 2) < 7)

"############" = "############" + "##"

replace 1 " " in " " with ""

print("" + "##############" + "")

((14 / 2) < 7)

12. At this point 7 < 7 is false and the loop ends.

12.1. We now exit the 'if' statement at this point and move to the next loop in 'for x in range(loops)'.

12.2. At this point in our script our variables look like this.

loops = 4

size = 7

x = 1

char\_block = "#"

char\_space = " "

text = "##############"

space = ""

13. We are now on loop 1 of 4, since this is zero-indexed too there will be four loops, 0 to 3.

14. We move on to the 'if' statement again.

if(the remainder of 1 / 2 is 0):

14.1. A remainder of 1 is left over from this operation so this statement is false. As a result we move to the 'else' block.

15. Here we open a while loop to shrink our diamond.

while((len(text) / 2) > 0):

translates to:

while((14 / 2) > 0):

15.1. This loop is effectively the previous loop in reverse and executes the following operation.

"" = "" + " "

replace 2 " " in "##############" with ""

print("" + "##############" + "")

16. The loop will therefore carry out the following operations.

((14 / 2) > 0)

"" = "" + " "

replace 2 " " in "##############" with ""

print(" " + "############" + " ")

((12 / 2) > 0)

" " = " " + " "

replace 2 " " in "############" with ""

print(" " + "##########" + " ")

((10 / 2) > 0)

" " = " " + " "

replace 2 " " in "##########" with ""

print(" " + "########" + " ")

((8 / 2) > 0)

" " = " " + " "

replace 2 " " in "########" with ""

print(" " + "######" + " ")

((6 / 2) > 0)

" " = " " + " "

replace 2 " " in "######" with ""

print(" " + "####" + " ")

((4 / 2) > 0)

" " = " " + " "

replace 2 " " in "####" with ""

print(" " + "##" + " ")

((2 / 2) > 0)

" " = " " + " "

replace 2 " " in "##" with ""

print(" " + "" + " ")

((0 / 2) > 0)

17. At this point 0 > 0 is false and the loop ends.

17.1. At this point our variables look like this.

loops = 4

size = 7

x = 2

char\_block = "#"

char\_space = " "

text = ""

space = " "

18. We start the process over and over again from either 10. or 13.

until x == 4.

18.1. At this point our loop ends, along with our 'diamonds' function.

18.2. Once this function concludes so does our script.