Midterm #1 Review

CS 140 • Dr. Sam Vandervelde

On Tuesday, October 9 we will hold our first midterm exam in CS 140 during our normal class time, from 2:20 to 3:50. I will write the exam to take around 70 minutes, so that you will (hopefully) have time to review your work. The midterm will consist of ten "predict the output" questions worth 3 points each, then two "spot the bug" questions at 4 points each, followed by six programs worth 7 points each, for a total of 80 points.

You should review your quizzes and the programming tasks below thoroughly, since over half of the midterm will be based directly on these questions, meaning that I will make only a minor variation (if any) to them. The remaining questions will be new, although they will only involve material covered in this half of the semester, of course. I also recommend reviewing the code drills, homework programs, and class creations for further practice. *The best way to review past programming assignments is to try writing the program from scratch.* If necessary you can read the solution code for ideas, but then set the solution aside and return to your program to complete it without referring to the solution. Don't just read or memorize code; this will not help you overall!

Twelve Programming Tasks

1. *Number Antics* Write a program that asks the user to enter a two-digit number. Then use a `while` loop to check whether the input is actually from 10 to 99, and to repeatedly ask for a new number if not. Finally, print out the first twenty multiples of the user's number on a single line, as in `The first twenty multiples of 17 are 17 34 51 ...`

2. *Lucky Dice* Write a program that selects three random numbers, each from 1 to 6, and prints them out on a single line separated by tabs. It should repeatedly "roll three dice," printing each trio of numbers on a separate line, until all three numbers match.

3. *Wierd Countdown* Write a program that counts from 1000 down to 1 by threes but skips every number that is a multiple of 7 and every number whose final digit is 7. Print the numbers on a single line, so the output should begin `1000 991 988 985` and so on. Keep track of how many numbers are on the list, and print the total at the end on a new line.

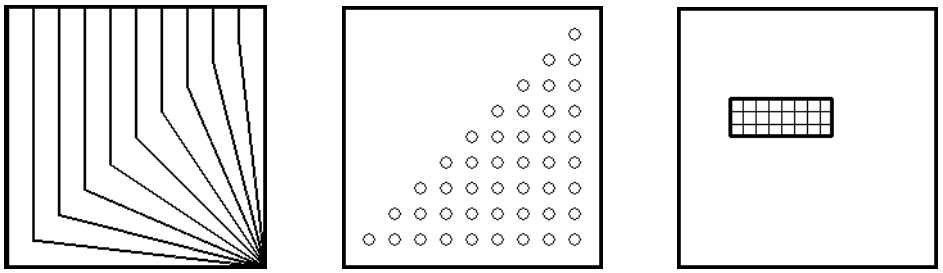
4. *Making Change* Write a program that asks the user to enter an amount (in cents) from 1 to 999. The program should then give directions for how to create this amount using quarters, dimes, nickels, and pennies. For instance, if the user enters `137` the program should print `To create 137 cents use 5 quarters, 1 dimes, 0 nickels, and 2 pennies.`

5. *Tab Sort* Write a program that generates one hundred random numbers, each from 1 to 8. Then print each number in its own column using tabs. So if the first two numbers happen to be 3 and 6, then the program should tab over three times and print `3`, and on the next line tab over six times and print `6`. (Don't use `if` statements; rather, employ `for` loops.)

6. *Find That Figure* Write a program that finds all three digit numbers with the following property. Let `S` be the sum of the cubes of the digits and let `P` be the product of the digits. Then if one adds twice `S` to three times `P`, one gets the original number back.

7. *Roy G. Biv* Write a program that asks the user to guess the computer's favorite color in seven tries or less. Define `fav = "hot pink"` at the start of your program, then use a while loop to repeatedly have the user guess until either they are correct or they have used up all seven guesses. The program should print an appropriate concluding remark in either case.

8. *Line Design* Write a program that will create the line design pictured below on the left, using two separate for loops. Use only a single `pygame.draw.line` within each loop. In this and the next two programs assume that the code necessary to create a 201×201 window named `screen` with white background has already been written.



9. *Circle Stack* Write a program that will create the circle pattern pictured above in the middle, using a single nested for loop. Use only a single `pygame.draw.circ` command in your program. (Each circle has radius five.)

10. *Ruled Rectangles* Write a program that asks the user to enter the x and y-coordinates of the left top corner of a rectangle. The program should then draw an 80×30 rectangle using a thickness of 3 and fill it in with horizontal and vertical lines of thickness 1 spaced ten pixels apart, beginning at the left top corner, as illustrated above right.

11. *Presidential Quiz* Write a program that asks the user to enter the last name of the current U.S. president, and then (in a separate question) his age. Using `if`, `elif` and `else` statements, the computer should then announce that the user either got both questions right, got one of them right, or missed both. Credit should be given for either `Obama` or `obama`.

12. *Computer Store* Write a program that asks the user for an item they wish to purchase, and then next for how many of that item they would like to buy. The computer should then choose a random number from 10 to 19 and declare how much the item costs and what the total will be. So the output will look something like this:

```
What would you like to buy today? bathtub
And how many bathtub do you need? 6
A bathtub costs 14 dollars, so your total is 84 dollars.
Thanks for shopping at Dewey, Cheatum and Howe.
```