CSCI 121: Computer Science Fundamentals I
Practice Midterm Exam

Here are a series of problems that are like what you'd see on the midterm exam.
The exam is Wednesday, October 26th, 2022.

---

1. Write a function `def percentage(amount,total)` that takes two integer quantities. The `total` is a positive integer. The `amount` is a value from 0 to `total`. The function should return a string giving a whole number percentage corresponding to the fractional amount of `amount` divided by `total`. For example:

```
>>> percentage(1,3)
'33%'
>>> percentage(2,3)
'66%'
>>> percentage(1,200)
'0%'
>>> percentage(7,7)
'100%'
```

   Note that it should round down the percentage value.

2. Beatrice drives a taxi. She normally charges riders a base cost of `$3` and then `$2` per mile driven. However, if the ride is more than 10 miles, she doesn't charge the base cost, but instead charges `$4` per mile (to cover the cost of driving back afterwards). Write a function `def taxi_cost(miles)` that takes an integer parameter giving the length of a trip in miles. It should return Beatrice's taxi charge for that trip. For example:

```
>>> taxi_cost(2)
7
>>> taxi_cost(6)
15
>>> taxi_cost(10)
23
>>> taxi_cost(12)
48
>>> taxi_cost(15)
60
```

3. Write the code for a Python script that requests a non-negative integer whose decimal representation has exactly three decimal digits (numbers like 23, 7, and -111 do not meet this criterion). It should then report its three digits. If they enter a value that doesn't meet this criterion then it should ask for another value to be input. It should continue to seek inputs until the criterion is met.

   It should mimic the interaction below, a result of running the script.

```
Enter a three-digit number: 23
That number has only two digits.
Please try again.
Enter a three-digit number: 7
That number has only one digit.
Please try again.
Enter a three-digit number: -111
That number is negative.
Please try again.
Enter a three-digit number: 537
Thank you.
Its hundreds digit is 5.
Its tens digit is 3.
Its ones digit is 7.
```

4. (a) Write a function `def squares_of(xs)` that takes a list of integers. It should return a new list that contains the squares of the list it is given, and in the same order. For example:

```
>>> squares_of([1,2,3,4])
[1,4,9,16]
>>> squares_of([-1,1,10,10,1])
[1,1,100,100,1]
>>> squares_of([])
[]
```

   (b) Now instead write a procedure `def square_all_of(xs)` that also takes a list of integers as a parameter. The code should have the effect of squaring all the values stored in that list. For example:

```
>>> list1 = [1,2,3,4]
>>> square_all_of(list1)
>>> list1
[1,4,9,16]
>>> list2 = [-1,1,10,10,1]
>>> square_all_of(list2)
>>> list2
[1,1,100,100,1]
>>> list3 = []
>>> squares_of(list3)
>>> list3
[]
```

5. Write the code for a function `def describe_list(xs)` that takes a list of integers and returns a string that describes its sequence. The string should mimic what we illustrate below:

```
>>> describe_list([1,2,3,4])
'That list holds the sequence 1, 2, 3, and 4.'
>>> describe_list([1])
'That list holds the value 1.'
>>> describe_list([10,1])
'That list holds the value 10 followed by 1.'
>>> describe_list([])
'That list is empty.'
```

6. Write the code for a procedure `def number_pyramid(height)` that takes a positive integer `height` and then prints a series of lines that form a pyramid of digits, like so:

```
>>> number_pyramid(5)
    0
   1 2
  3 4 5
 6 7 8 9
0 1 2 3 4
>>> number_pyramid(1)
0
>>> number_pyramid(3)
  0
 1 2
3 4 5
>>> number_pyramid(6)
     0
    1 2
   3 4 5
  6 7 8 9
 0 1 2 3 4
5 6 7 8 9 0
```

Each line has a series of spaces at its front followed by a series of digits separated by spaces. They form the pyramid shape shown. The digits that are output start at 0 and rotate through the decimal digits, from `0` through `9`, over and over.