Homework 1: Arithmetic and functions

This homework will be done using the hw1.py file available on the class website. The file contains a series of unwritten function definitions. Below are descriptions of what each of the functions should do. You should complete the function definitions so they do the expected thing. Remember to change the `return` statement in the function.

You should test your functions! Use `print()` commands in your program to make sure that you are getting the correct output from the functions. For this assignment, it is ok to leave these print commands in the program, but make sure there are no `print()` commands in the function bodies themselves – the functions should do nothing other than return the correct value. Each exercise includes examples of the sort of output the function should give. Those are good examples to use as tests, I recommend testing with other values as well.

It is always ok to use functions defined in one exercise when writing the functions of later exercises.

When you are done defining these functions, upload the file in a homework1 folder inside your submission folder. Also print the file and bring it to class.

Exercise 1: `feet_to_meters`

One foot is equal to approximately .3048 meters. The `feet_to_meters` function should take as input a length measured in feet and output the number of meters equal to the same length.

```
>>>feet_to_meters(10)
3.048
>>>feet_to_meters(6.5)
1.9812
```

Exercise 2: `tens_digit`

The `tens_digit` function should take as input an integer and output an integer equal to the tens digit of its input. We think of numbers less than 10 as having a zero in the tens digit.

```
>>>tens_digit(24)
2
>>>tens_digit(1847)
4
>>>tens_digit(6)
0
```

Exercise 3: `nth_digit`

The `nth_digit` function should take as input two integers `x` and `n`. It should then output the n^{th} digit of x. That is, if n=2, it should output the tens digit. If n=4 it should output the thousands digit, etc. Think of numbers as having invisible zeros to the left of the digits we write.

```
>>>nth_digit(8734, 3)
7
>>>nth_digit(829463, 1)
3
>>>nth_digit(123, 4)
0
```

Exercise 4: `reversed`

The `reversed` function should take as input a two-digit integer and output the integer that has the same two digits, but reversed. You should think of single-digit numbers as having an invisible zero in the tens digit.

```
>>>reversed(56)
65
>>>reversed(9)
90
>>>reversed(80)
8
```

Exercise 5: `coins`

The `coins` function should take as input an amount of money, measured in (an integer number of) cents, and output the minimum number of coins necessary to have that value. Assume that the coins available are quarters, dimes, nickels, and pennies.

```
>>>coins(25)
1
>>>coins(56)
4
>>>coins(192)
11
```

Exercise 6: `is_multiple`

The `is_multiple` function should take two positive integers, `a` and `b`, and return `True` if a is `a` multiple of `b`, and `False` otherwise.

```
>>>is_multiple(15, 3)
True
>>>is_multiple(20, 7)
False
>>>is_multiple(45, 1)
True
```

Exercise 7: `zap_buzz`

Zap-Buzz is a variant of a common children's game. The way the game works is that each person takes turns saying a number, counting upwards. However, if the number is a multiple of 7, you say "zap" instead of the number. If the number has a 3 in it, you say "buzz" instead of the number, and if both things are true you say "zap buzz". Your job is to implement a function that will decide what to say for a given number. It should take as input an integer and output the integer, the string "zap", the string "buzz", or the string "zap buzz", depending on which is correct. It is ok if your function only works for numbers below 1000.

```
>>>zap_buzz(8)
8
>>>zap_buzz(14)
'zap'
>>>zap_buzz(13)
'buzz'
>>>zap_buzz(35)
'zap buzz'
```