1.1 Homework

Python Exercises.

Create a repository on github and put the solutions to these exercises there. Send me the link by the next weekend.

1. Write a function called tables. Should take 2 arguments (say m and n) and print the times tables for m from 1 to n

```
[2.7.9]>>> tables.tables(5, 10)
 1 x 5 = 5
 2 x 5 = 10
 3 x 5 = 15
 4 x 5 = 20
 5 x 5 = 25
 6 x 5 = 30
 7 x 5 = 35
 8 x 5 = 40
 9 x 5 = 45
10 x 5 = 50
```

1. Write a function called sq_tables. Should take one argument (say m). Should print a table that has m rows and m columns. Each cell will be the product of the row and column number.

```
[2.7.9]>>> tables.sq_tables(5)
   1 2 3 4 5
   2 4 6 8 10
   3 6 9 12 15
   4 8 12 16 20
   5 10 15 20 25
```

If you get this working, modify it to format things better like so

```
[2.7.9]>>> tables.sq_tables(5)
   1 | 2 | 3 | 4 | 5
----+-----+-----+-----+----
   2 | 4 | 6 | 8 | 10
```

```
  3 | 6 | 9 | 12 | 15
  4 | 8 | 12 | 16 | 20
  5 | 10 | 15 | 20 | 25
[2.7.9]>>>
```

1. Write a function called zzbizz. Should take one argument (say n). It should print numbers from 1 to n but should print zz for all multiples of 3, bizz for all multiples of 5 and zzbizz for all multiples of 15.

```
[2.7.9]>>> fizzbizz(16)
1
2
fizz
4
bizz
fizz
7
8
fizz
bizz
11
fizz
13
14
fizzbizz
16
```

1. Write a function called freq. Should take a string as input (say s) and return a dictionary that has the counts for all characters in s.

```
[2.7.9]>>> freq("the sixth sheiks sixth sheeps sick")
{' ': 5, 'c': 1, 'e': 4, 'i': 4, 'h': 5, 'k': 2, 'p': 1, 's': 7, 't': 3, 'x': 2}
```

1. Write a function called panagram. Should take a string as input (say s). Should return True if s is a panagram (i.e. contains all letters of the alphabet), False otherwise.

```
[2.7.9]>>> panagram("the quick brown fox jumps over the lazy dog") True
[2.7.9]>>> panagram("the quick brown fox jumped over the lazy dog") # s is missing False
```

```
[2.7.9]>>>
```

1. Write a function called breakdown which will take a currency amount and then break it down into denominations of 1000, 500, 100, 50, 20, 10, 5 and 1.

```
>>> exercises.breakdown(1784)
1000 x 1 = 1000 (1000)
 500 x 1 = 500 (1500)
 100 x 2 = 200 (1700)
  50 x 1 = 50 (1750)
  20 x 1 = 20 (1770)
  10 x 1 = 10 (1780)
   1 x 4 = 4 (1784)
```

1. Write a function called breakdown2 which does the same as above but takes two arguments. The amount to be broken down and a dictionary of available notes. It should break down using only available notes and the number of notes should reduce when the function is called again and again.

```
# This is what we have. As we use it, the available notes
# gets reduced.
[2.7.9]>>> d = {1000: 1, 500: 5, 100: 2, 20: 10, 10: 5, 1:20}
[2.7.9]>>> exercises.breakdown2(1784, d)
1000 x 1 = 1000 (1000)
 500 x 1 = 500 (1500)
 100 x 2 = 200 (1700)
  20 x 4 = 80 (1780)
   1 x 4 = 4 (1784)
[2.7.9]>>> exercises.breakdown2(1784, d)
 500 x 3 = 1500 (1500)
  20 x 6 = 120 (1620)
  10 x 5 = 50 (1670)
   1 x 16 = 16 (1686)
98 is left over
[2.7.9]>>> exercises.breakdown2(1784, d)
```

```
 500 x 1 = 500 (500)
1284 is left over
```

```
[2.7.9]>>> exercises.breakdown2(1784, d)
1784 is left over
[2.7.9]>>>
```