CSc 110, Spring 2018
Programming Assignment #11: Artificial Composer (20 points)
Due Tuesday, April 24th, 7:00 PM

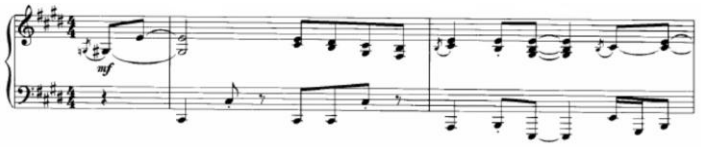
Thanks to Clark Penado, head SL.

This assignment focuses on 2D lists and dictionaries. Turn in a file named `composer.py`. You will also need the two input files `chords.txt` and `notes.txt` from the course web site. Save these files in the same folder as your program.

The assignment involves reading in two files, one containing the chords in each major key, and one containing all the notes in a pentatonic scale for the same major keys. Your program will then use this information to produce a chord progression and small solo piece. You do not need any prior musical knowledge to complete this assignment successfully.

Background Information:

Note: This section explains some information about guitar music that is related to this assignment.
It is for your information only; you do not need to fully understand it to complete the assignment.



Guitar music, like most other music, is written as notes on a staff, as pictured to the right. However, alongside this, the music is slightly modified with diagrams above each chord. These diagrams represent fret positions (finger positions) for specific notes so that it is clear where and how the music needs to be played. This is important because the same chord or note can be played in a variety of different ways and positions but will sound slightly different depending on how it is played. In order to make sure the appropriate sound is made it is important to specify the position. This format has led to a more informal formatting for most guitar music, known as *tabs*, where the formatting for a song such as John Mayer's "Slow Dancing in a Burning Room" can be seen below. To the left, there are chord assignments placed over lyrics, and to the right are fret positions placed visually on top of the corresponding strings of a guitar.

```
[Verse]

Am          Am          F          C
It's not a silly little moment
Am          Am          F          C
It's not the storm before the calm

Am          Am          F          C
This is the deep and dyin breath of
Am          Am          F          C
This love we've been workin on

Am          Am          F          C
Can't seem to hold you like I want to
Am          Am          F          C
So I can feel you in my arms

Am          Am          F          C
Nobody's gonna come and save you
Am          Am          F          C
We pulled too many false alarms
```

Chord assignments and lyrics

```
[Part 1]

e|------------x-----------------------------------
B|------------x-----------------------5----5--5--
G|--------9---x------9\8-----6----4----4h6--4--4--
D|------------x-----11-------6----4------------6--
A|-9\11-------x-----------------------------------
E|------------x-----------------------------------

e|------------x-------------------0-------0-------
B|--------9---x-------------------5-------5-------
G|--------9---x---9---8---6---4---4---4---4-------
D|--9\11------x--11---9---6---4---6---4---6-------
A|------------x-------------------7---7---7-------
E|------------x-----------------------------------
```

Fret positions and strings

Notice that both of these forms of notation leave out rhythm information. They assume that the player will already know the rhythm.

For this assignment, you will be outputting music in the format of the fret positions and strings diagram above. The letter at the beginning of each row represents which string the row represents. It is followed by a `|` character to denote that it is a label and not a note to play. Each number represents a fret position. An `x` represents a pause. Notes are read and played left to right with notes that are played at the same time written above and below one another. For your assignment, you may assume notes will be played one at a time.

Composing new songs for guitar can be challenging. Finding chords that work well with one another and writing solos are generally particularly challenging. That is what makes the program you are writing so useful. The program you write will be taught what chords work well with one another, as well as the notes within a specific scale, through input files. Your program will then take this information and write a chord progression, as well as a solo that works with this chord progression. By adding a rhythm component, what your program generates could easily be turned into a song.

Program Behavior:

Your program will first read in two files, and process the information into two different dictionaries, one for the chords associated with a specific key, and one for the fret positions of strings within a specific scale. The program will then generate an introduction, and then prompt the user for a key to play in, as well as the number of chords they would like to include in their chord progression. You should then select the user designated amount of **unique** chords, and select random fret positions to produce notes for your solo.

Log of execution (user input underlined):

```
<Custom message from favorite musician>
What key would you like to play in? Bb
How many unique chords? (up to 6) 4
Chord Progression: Cm Dm Bb Gm
e|--------------------------3--6--3--
B|---------3-----3--6-----------------
G|---5-------------------------------
D|--------------------3--3-----------
A|------3-----------------------------
E|3-----------3-----------------------
```

Implementation Guidelines, Hints, and Development Strategy:

The main purpose of this assignment is to demonstrate your understanding of dictionaries and lists of lists. Therefore, you should use these structures to store your data.

`chords.txt` contains data in the following format:

```
C# C# D#m E#m F# G# A#m
```

The first value on each line is the key and the rest are chords that go with the first key. Store each line in a dictionary so that the first value is a key and the rest of the values are that key's values.

`notes.txt` contains data in the following format:

```
C# E6 E9 A6 A8 D6 D8 G6 G8 B6 B9 e6 e9
```

The first value on each line is the key and the rest are fret positions and strings. Store each line in a dictionary so that the first value is a key and the rest of the values are that key's values.

After you have created the above dictionaries and prompted for user input, you should use the user input key to create a *chord progression*, which is in the same key the user input, and contains the correct number of unique chords. Create this progression by randomly selecting chords in the provided key. No chord should appear more than once.

You can find chords in a particular key by accessing the dictionary you created from `chords.txt` and looking up the values associated with the key the user typed in.

Once you have output the chords, you should then move on to the solo computation portion of this assignment. `notes.txt` contains the notes which you should use in your solo. Randomly select notes, allowing duplicates, that are in the key the user typed in. You can find out which notes are in the right key by looking up the key in the dictionary that you created from `notes.txt`.

Notes are represented by a letter denoting the string the note is played on, immediately followed by the fret they are played on. For example, the note *A*, played on the 5th fret on the low *E* string would be written as `E5`.

Since we have to print the whole first string before printing any of the second and we want to make sure that all notes are lined up correctly, create a list of lists to represent the text that will be output. The first row should represent the first row of output, the second the second, etc. There should be two dashes between each note and the next except in the case of two digit numbers. If the number is two digits there should be only one dash between it and the following note. For example, a solo of `E3 G5 A3 B11 E3` would be stored as follows:

```
notes = [[e,|,-,-,-,-,-,-,-,-,-,-,-,-,-,-],
         [B,|,-,-,-,-,-,-,-,-,-,1,1,-,-,-],
         [G,|,-,-,-,5,-,-,-,-,-,-,-,-,-,-],
         [D,|,-,-,-,-,-,-,-,-,-,-,-,-,-,-],
         [A,|,-,-,-,-,-,-,3,-,-,-,-,-,-,-],
         [E,|,3,-,-,-,-,-,-,-,-,-,-,3,-,-]]
```

Notice that since the 11 is two digits. To be printed evenly spaced it will need to have one less dash printed after it. Characters are shown without quotes in the above diagram to make it easier to read. Note, however, that if you print your list of lists all characters will be surrounded by quotes.

Once you have built up this representation of the solo your program is generating, you can output it by looping through your list of lists and outputting one character at a time.

You may assume that the input files exist, are readable, and contains valid input. You may also assume that the user inputs valid data.

Style Guidelines:

You are required to have a constant representing the names and order of the strings. This will make iterating over each row and outputting the contents of your list a bit easier. Set its value to `["e", "B", "G", "D", "A", "E"]`. This will make it easier to adjust your program to instruments with different strings.

You are required to have a constant to represent the number of notes to generate for the solo. Set this to 24 when you submit your program.

We will grade your function structure strictly on this assignment. Use at least **five nontrivial functions** besides `main`. These functions should use parameters and returns, including lists of lists and dictionaries, as appropriate. The functions should be well-structured and avoid redundancy. No one function should do too large a share of the overall task. You may not nest these functions inside each other or in main.

Your `main` function should be a concise summary of the overall program. It is okay for `main` to contain some code such as `print` statements. But `main` should not perform too large a share of the overall work itself, such as examining each line of input. Also avoid "chaining," when many functions call each other without ever returning to `main`.

We will also check strictly for redundancy on this assignment. If you have a very similar piece of code that is repeated several times in your program, eliminate the redundancy such as by creating a function, by using loops over the elements of lists or dictionaries, and/or by factoring `if/else` code.

Since dictionaries and lists of lists are a key component of this assignment, part of your grade comes from using them properly.

You are limited to features in lectures 1 - 35. Follow past style guidelines such as indentation, names, variables, line lengths, and comments (at the beginning of your program, on each function, and on complex sections of code). You may not have any global variables.