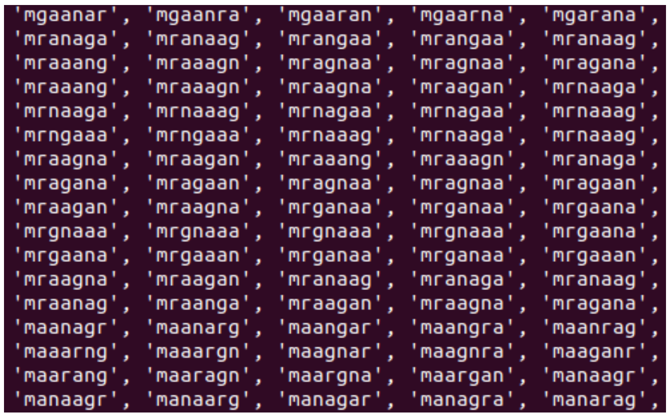
CMPSC 102 Discrete Structures Fall 2019

Practical 10: Duno Esom Bonker Ponyth Deco (*Undo Some Broken Python Code*)

Refer to your notes, slides and sample Python code from this week and other weeks. In particular, follow the python code that we created in class or check on line for interesting pieces of code to help you in your programming.



```
'mgaanar', 'mgaanra', 'mgaaran', 'mgaarna', 'mgarana',
'mranaga', 'mranaag', 'mrangaa', 'mrangaa', 'mranaag',
'mraaang', 'mraaagn', 'mragnaa', 'mragnaa', 'mragana',
'mraaang', 'mraaagn', 'mraagna', 'mraagan', 'mrnaaga',
'mrnaaga', 'mrnaaag', 'mrnagaa', 'mrnagaa', 'mrnaaag',
'mrngaaa', 'mrngaaa', 'mrnaaag', 'mrnaaga', 'mrnaaag',
'mraagna', 'mraagan', 'mraaang', 'mraaagn', 'mranaga',
'mragana', 'mragaan', 'mragnaa', 'mragnaa', 'mragaan',
'mraagan', 'mraagna', 'mrganaa', 'mrganaa', 'mrgaana',
'mrgnaaa', 'mrgnaaa', 'mrgnaaa', 'mrgnaaa', 'mrgaana',
'mrgaana', 'mrgaaan', 'mrganaa', 'mrganaa', 'mrgaaan',
'mraagna', 'mraagan', 'mranaag', 'mranaga', 'mranaag',
'mraanag', 'mraanga', 'mraagan', 'mraagna', 'mragana',
'maanagr', 'maanarg', 'maangar', 'maangra', 'maanrag',
'maaarng', 'maaargn', 'maagnar', 'maagnra', 'maaganr',
'maarang', 'maaragn', 'maargna', 'maargan', 'manaagr',
'manaagr', 'manaarg', 'managar', 'managra', 'manarag',
```

Figure 1: Anagrams are rearrangements of letters of a word to spell out new words from these same letters. Here, these are nonsense anagrams of the word, *anagram*.

Bithug Starter *Kiln* (GitHub Link)

https://classroom.github.com/a/QP0b54-Y

To use this link, please follow the steps below.

- Click on the link and accept the assignment.
- Once the importing task has completed, click on the created assignment link which will take you to your newly created GitHub repository for this lab.
- Clone this repository (bearing your name) and work on the practical locally.
- As you are working on your practical, you are to commit and push regularly. You can use the following commands to add a single file, you must be in the directory where the file is located (or add the path to the file in the command):
  - `git commit <nameOfFile> -m ''Your notes about commit here''`

– `git push`

Alternatively, you can use the following commands to add multiple files from your repository:

– `git add -A`

– `git commit -m ‘‘Your notes about commit here’’`

– `git push`

Yrasmum (Summary)

Shown in Figure 1, anagrams are created by taking the letters of a word, rearranging them and then creating a new word. The word, *tea* is an anagram of the word, *eat* since the number of each letters exists in each word.

In this practical, you will be fixing working Python code that should produces anagrams (once fixed). To fix the code, you will be adding tabs and white-spaces where they are necessary. Be careful, the indentations in code with classes may be different than the those in code without classes. Please study each line to make sure that the white-spacing is correctly done. As you edit, try running your code to ensure that the edits are appropriate.

Once you have finished, your anagrams-producing code, you will notice that there is a class doing the work of the anagram production. **Study this class to see how it works and to get you ready for the questions in blue below.**

Finish Editing and Run Your Code

Add your name to the edited code and try running the code at the terminal with commands such as the following:

```
 python3 anagram.py sister
```

returning the following input,

```
  Anagram(s) found :  {'resist', 'restis'}
```

Each time a word is returned as a solution, it is checked with a larger listing of words to determine whether they are truly words in the English Language.

• •

Be careful not to process words that have more than about ten letters as this may crash your computer and cause you to lose data.

• •

The Steps to *Comeeplt* (Complete)

1. Fix the code: replace missing whitespace to make the program run.
2. Spend some time to play with the program to see how it works.
3. Answer the below questions in blue.

Questions in Blue

Place your Markdown-written responses to these questions in the file: `writing/miniReflection.md`.

1. In your own words, how does the program work to find tested anagrams? Please describe each method or function to explain its role in the flow of the code.
2. What notable anagrams did you find?
3. It says above that you could crash your computer if you tried to find anagrams of words which are longer than about ten letters. Explain how this crash may happen.
4. How could the system crash be prevented in terms of writing better code?
5. What role does the `try` and `except` code blocks play? There are two such blocks in your code.

Devilersable (Deliverables)

1. Your completed reflection document (`writing/miniReflection.md`) where you respond to the questions in blue (above).
2. Your complete (and working) code saved as `src/anagramPractical10.py`.

```
.-.. --- ...- . / - .... .- - / .- -. .- --. .-. .- -- -.-.--
```