FeedBuzz

A question scorer prototype, *not totally based from BuzzFeed.*

Contents

What is this?

In short, this is a program that scores the answers from questions. It takes a Feedbuzz File, and exports a barebones PDF file (if supported.) It adjusts the score of the weight of an answer a taker picks, and it describes the range of the score.

Setup

It is recommended to have Python 3.12 installed to run this program. Although the source code does not depend on the features from newer Python versions, the program *should* be compatible with Python 3.

Make sure the following source code files are in the same folder:

1. `main.py`
2. `reader.py`
3. `test.py`

Usage

`feedbuzz` is `python main.py` assuming if you run it in a directory or a folder containing `main.py`

```
feedbuzz [input file] [--output output_file] [--maximum] [--ranges] [--questions] [--help]
```

- `[input file]` Take an input file in Feedbuzz Lingo.
- `--output output_file` Output a file named *output_file*. If left unspecified, it will output a file named, *results.pdf* or *results.json*.
- `--maximum` Get the maximum possible points achieved in a test.
- `--ranges` List all the judgment scoring ranges in a test.
- `--questions` List all the questions in a test.
- `--help` Show brief help message.

Presets

Feedbuzz bundles with general-purpose presets and goodies.

Scoring

The scoring system allows you to configure if the deduction mechanic is punishing or sparing. If a score is almost to zero and the penalty of an answer is greater than the score, the scoring system will always push it to zero.

Punishing Deduction

For a single-choice question, a wrong answer will penalize your score.
For a multiple-choice question, the total score of all selected choices will penalize your score if the losses overweigh.

```
Question "What can be 10?":
    Choice Multiple
```

```
    Answer "8 + 2" Gain 1
    Answer "2 + 1" Loss 1 # selected
    Answer "9 + 8" Loss 1 # selected
    Answer "5 + 7" Loss 1; # selected

# this means the total penalty of 3
```

Sparing Deduction

For a progressive score system in your test, consider only having answer penalties in multiple-choice questions.

For a single-choice question, a wrong answer will still penalize your score.
For a multiple-choice question, the total score of all selected choices have no penalty if losses overweigh.

```
Question "What can be 10?":
    Choice Multiple
    Answer "8 + 2" Gain 1
    Answer "2 + 1" Loss 1 # selected
    Answer "9 + 8" Loss 1 # selected
    Answer "5 + 7" Loss 1; # selected

# this means no penalty because the test has a sparing deduction mechanic.
```

FeedBuzz Lingo

Feedbuzz uses a descriptive language to gather test metadata and to run questions. The language is simple to learn as the syntax is self-descriptive. This is the language all input files should be written in.

Parts

Sample File

```
Test:
    Name "Sample Test"
    TimeLimit 120
    ;

Scoring:
    At 2 "Good enough I guess."
    At 1 "Cringe"
    At 0 "No common sense you have"
    ;

Question "Which poison would you pick?":
    Choice Single
    Answer "Right" Gain 1
    Answer "Neutral"
    Answer "Wrong" Loss 99
    ;

Question "What is not the third color of the rainbow?":
    Choice Multiple
    Ordering Alphabetical
    Answer "red" Gain 1
    Answer "orange" Gain 1
    Answer "yellow" Loss 1
    Answer "green" Loss 1
    ;
```

Nested Pattern

The language has a pattern to describe data. Depending on the data, it goes something like this:

```
DataName [DataValues]:
    Attribute AttributeValue
    Attribute2 AttributeValue2
    ;
```

Indicate a colon after the values of data to begin listing attributes. Indicate a semicolon to stop the attribute list. Although use consistent indentation for clear lists when writing attributes, whitespace does not matter when read.

```
Test: Name "One Line Sample";
```

Quoted Strings

Quoted strings can be double-quoted or single-quoted. Use backticks to allow quotes, that would be confused with another string, in quoted strings.

```
Question "Sample":
        Answer 'Foo'
        Answer "Bar"
        Answer "The duck says, \"Quack!\""
        ;
```

Comments

Like writing code, the language supports one-line comments beginning with a hashtag.

```
# This is a weird question.
# Why do we have this in the test?

Question "Do you support [insert controversial policy here]?":
    Answer "Yes." Gain 99999
    Answer "No." Loss 0
    Answer "I prefer not to say." Loss 99999999;
```

File Structure

You have only one `Test` data entry. You are welcome to have as many questions with many answers, as you like. You can have as many score-ranged result descriptions as you like. This is all on the same indentation. *However,* the location of the question entries define the order of the questions. The ranged result descriptions are sorted by the starting point.

The typical file format of a FeedBuzz Lingo file ends with ".q" standing for questions.

```
Test:
    # test data here
    ;

Result 0,1:
    # result data here
    ;

Result 1,2:
    # result data here
    ;

Question "Question1":
    # question data here
    ;

Question "Question2":
    # question data here
    ;

Question "QuestionX":
    # question data here
    ;
```

Test Data

The basis of a FeedBuzz Lingo file. You can only declare it once.

```
# default values when unspecified

Test:
    Name "Test Name"
    Description "Test description"
    Deduction Sparing
```

```
    TimeLimit 0
    ;
```

- **Metadata**

  FeedBuzz displays the name and description before a test begins. [The Results JSON includes this metadata.]

  - `Name [name]` | Name the test with a proper name.
    - `[name]` | *quoted string*
  - `Description [description]` | Give the test a description.
    - `[description]` | *quoted string*
- `TimeLimit [seconds]` | Impose the time limit in seconds. Set to 0 to impose unlimited time.
  - `[seconds]` | *positive integer*
- `Deduction [gracefulness]` | Impose the time limit in seconds. Set to 0 to impose unlimited time.
  - `[gracefulness]` | *positive integer*

Judgment Data

The labeling of score ranges. You can only declare it once. If the scoring range positions are out of order, FeedBuzz will sort them.

```
Scoring:
    At 4 "A score at least more than 4 is displayed"
    At 2 "A score at least more than 2 is displayed"
    At 0 "A score below anything above is displayed"
    ;
```

- `Scoring:` | **Scoring data**
  - `At [point] [description?]` | Pinpoint a score range starting with `[point]`
    - `[point]` | *number* | A requirement if a score is at least the amount of it.

  You can always look how FeedBuzz is treating the judgment data by running `feedbuzz [input file] --ranges`

Question Entry

The basis of a question.

```
# default values when unspecified

Question "[question name placeholder]":
    Choice Single
    ;
```

Note: The question entry must have at least one answer, or else FeedBuzz will not accept it.

- `Answer "[answer contents]" [consequence?] [score?]` | Add an answer to the question.
  - `[answer contents]` | *quoted string* | Describe the contents of an answer
  - Scoring (Optional) (If left unspecified, this has no reward or penalty)
    - `[consequence]` | `Gain` *or* `Loss` | Choose the consequence of an answer when selected
    - `[score]` | *positive integer* | Reward the consequence by this amount
- `Choice [type]` | Make the question single or multiple choice.
  - `[type]` | `Single` *or* `Multiple`

Samples

```
Question "What is 1+1?":
    Choice Single

    Answer "0"
    Answer "1"
    Answer "2" Gain 1
    Answer "3"
    ;
```

JSON Result Data

FeedBuzz provides JSON for easily processing data into other services you wish.

- `metadata`

- `name` | string | The name of the test taken.

Visions

This is the only beginning, this is a prototype that gets the idea done. We have visions of shaping this into a polished web application. Perhaps a service platform, despite being despicably similar to a

To make it user-friendly enough, I have plans to rebuild this with different interfaces to have the same system. For now, this is a passion project made for a family member and for a school project.

Cramming Roadmap

Said in days:

1. Documentation Planning
2. Language Implementation
3. Question Asking
4. Wizard Program
5. Goodies