Git Best Practices

Rationale: We are going to be using industry standard Git practices. This is good practice for writing consistent, well-formatted, and readable code. This consistency among our team also allows for more efficient workflow and less confusion about what code should look like.

- Commit early and commit often
    - Each feature should have one or more commit
- Commit messages
    - Every commit should have a summary (50 chars or less)
        - Commit summaries should be written in present tense, start with an uppercase letter, and complete the sentence "This commit will _____"
    - Every commit may also have a longer commit message
        - Any more extensive feature should have a longer commit message
        - Commit messages detail the feature being implemented/bug being fixed, plus a description of why the changes were being made
- Branching
    - Each new feature should get its own branch
    - Don't merge branches for features that have not been completed
    - Each feature branch will eventually be merged into the master
- Merge/Rebase
    - Prefer to rebase changes on master whenever possible
    - Every merge should get a pull request
    - Merges can only happen after review and approval from at least two other members of the team
- Public history
    - Don't change the shared public history of the git repo without all of the other members of the team's knowledge/presence

Python Best Practices

- Indentation
    - 4 spaces (not tabs)
    - Rationale: Python standard
- Variable / Function names
    - Snake case, unless global variables are uppercase, in which case use underscores
    - Rationale: Python standard
- Commenting
    - Function-level docstrings
        - Each function have a comment describing its purpose, inputs, and outputs
    - In-line comments for more complicated code
    - Rationale: Makes code readable and understandable for other team members
- Filenames
    - Snake case
    - Rationale: It's good to be consistent.

Coding Best Practices

- Indentation
    - 2 spaces (not tabs)
    - Rationale: Kip is using terminal code editor with 80 char-wide screens
- Variable / Function names
    - Camelcase, unless global variables are uppercase, in which case use underscores
    - Rationale: We like camelcase more. It's also good to be consistent.
- Commenting
    - Function-level comments
        - Each function have a comment describing its purpose, inputs, and outputs
    - In-line comments for more complicated code
    - Rationale: Makes code readable and understandable for other team members
- Filenames
    - Camelcase
    - Rationale: We like camelcase more. It's also good to be consistent.