Git Hooks

1. Resources:

www.atlassian.com

https://www.atlassian.com/git/tutorials/git-hooks

2. Implementing Pre-commit Hooks

2.1 To detect whether the email set in the repository is correct or not.

2.2 To detect if there are any trailing whitespaces in the commit.

Following are the contents of the executable file: `.git/hooks/pre-commit`

```
#!/usr/bin/env python3

import re
import subprocess
import sys

def check_email():
    email_regex = r"^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$"
    git_author_email = subprocess.check_output(["git", "config",
"user.email"]).decode().strip()
    if not re.match(email_regex, git_author_email):
        print("Error: Incorrect email format. Please configure correct email in your Git
settings.")
        sys.exit(1)

def check_trailing_whitespaces():
    against = "HEAD" if subprocess.call(["git", "rev-parse", "--verify", "HEAD"]) == 0
else "4b825dc642cb6eb9a060e54bf8d69288fbee4904"
    if subprocess.call(["git", "diff-index", "--check", "--cached", against, "--"]) !=
0:
        print("Error: Trailing whitespaces found in the changes being committed.")
        sys.exit(1)

if __name__ == "__main__":
    check_email()
    check_trailing_whitespaces()
    sys.exit(0)
```

3. Using Hook Plugins

`pre-commit` package is used for generating pre-commit hooks. This package can be installed like a normal python package using pip package manager:

```
pip install pre-commit
```

Following are the contents of the yaml file: `.pre-commit-config.yaml`

```
repos:
-   repo: https://github.com/pre-commit/pre-commit-hooks
    rev: v3.2.0
    hooks:
    -   id: check-yaml
    -   id: end-of-file-fixer
    -   id: trailing-whitespace
    -   id: check-added-large-files
    -   id: debug-statements
        language_version: python3
-   repo: https://github.com/psf/black
    rev: 22.10.0
    hooks:
    -   id: black
        args: [--safe]
```

```
pre-commit install
```

4. Check Commit Messages for Formatting

Implementing a commit-msg hook to check if the commit message contains tags for commit types such as "feat", "fix", "refactor" etc.
Following are the contents of the executable file: `.git/hooks/commit-msg`

```
#!/bin/bash

# Commit-msg hook to check commit message format for commit types

# Function to check if the commit message contains valid tags
check_commit_message() {
    commit_msg_file="$1"
    commit_msg=$(cat "$commit_msg_file")
    # Regular expression to match valid commit types (feat, fix, refactor, etc.)
    commit_type_regex='^(feat|fix|refactor|chore|docs|style|test|perf|build|ci|revert)(\(.+\))?: .+'

    if [[ ! "$commit_msg" =~ $commit_type_regex ]]; then
        echo "Error: Invalid commit message format. Please include a valid commit type (feat, fix, refactor, etc.) in the commit message."
        echo "Aborting commit."
        exit 1
    fi
}

# Call the function to check the commit message
check_commit_message "$1"

# Exit with success status if the commit message format is valid
exit 0
```