Вопросы на 3.

1. Создайте репозиторий на GitHub / GitLab по имени GitTraining
2. Создайте локальную версию репозитория в некоторой папке через git init
   git init
3. Создайте файл readme.md, с содержимым: “My first Repo”, добавьте его в git
   vim readme.md
   git add --all
   git commit -m ‘first’
4. Присоедините remote с именем origin к адресу репозитория
   git remote add origin https://github.com/KusokMIPT/GitTraining
5. Выполните команду “git push -u origin master” (-u необходимо, чтобы задать оригинальный - upstream repo)
   git push -u origin master
6. Создайте папку GitTraining.git вне папки с проектом
   cd ..
   mkdir GitTraining.git
   cd GitTraining.git
7. Создайте удаленную (remote) версию репозитория через “git init --bare”
   git init --bare
8. Сделайте “git remote add local <path to GitTraining.git>
   git remote add local ../GitTraining.git
9. Сделайте git push local master
   git push local master
10. Теперь мы умеем поднимать свой инстанс git

Вопросы на 4.

1. Создайте новую ветку my-new-branch
   git checkout -b my-new-branch
2. Сделайте изменения в новой ветке - напишите hello world.
   vim readme.md
3. Закоммитьте изменения и залейте на удаленный сервер (GitHub, GitLab)
   git add readme.md
   git commit -m 'readme change'
   git push origin my-new-branch
4. После этого создайте новую ветку из ветки master на удаленном сервере (GitHub, GitLab) - my-remote-branch, создайте в нем файл readmes/readme.md.
5. После этого выкачайте ветку my-remote-branch.
   git fetch origin
   git checkout my-remote-branch
6. Слейте ветку my-remote-branch в ветку my-new-branch.
   git checkout my-new-branch
   git merge my-remote-branch
7. Какой граф изменений получился?
   git log --graph --oneline

```
*   028bb3d (HEAD -> my-new-branch) Merge branch 'my-remote-branch' into my-new-
branch task
|\
| * 4959a5c (origin/my-remote-branch, my-remote-branch) Create readme.md
* | 1e77a17 (origin/my-new-branch) readme change
|/
* 512a28f (origin/master, master) first
```

Вопросы на 5

1. Начните изменять свой код в репозитории (создайте в master A.cpp, закоммитьте его, переключитесь в ветку my-new-branch, создайте A.cpp, не коммитьте)
   git checkout master
   vim A.cpp
   комит
   git checkout my-new-branch
   vim A.cpp
   git checkout master
2. Попытайтесь переключиться на другую ветку. Какое сообщение вы получите?

```
error: The following untracked working tree files would be overwritten by checkout:
        A.cpp
Please move or remove them before you switch branches.
Aborting
```

3. Как можно решать эту проблему? Найдите хотя бы два решения этой проблемы
   - переименовать файл (rename)
   - приберечь наработки (git stash - спрячет изменения в специально выделенное для этого хранилище)
4. Откройте папку в среде разработки (Pycharm, IDEA). Какие файлы появились при этом?

```
new file:   .idea/.gitignore
new file:   .idea/inspectionProfiles/profiles_settings.xml
new file:   .idea/misc.xml
new file:   .idea/modules.xml
new file:   .idea/vcs.iml
new file:   .idea/vcs.xml
```

5. Добавьте эти файлы в git при помощи git add.
6. Как можно удалить эти файлы из git, но не из файловой системы?
   git rm --cached -r .idea/
7. Как сделать так, чтобы эти файлы не добавлялись в git автоматически при команде git add --all?
   - использовать gitignore

   .gitignore:

> .idea/

Вопросы на 6.

1. Сделайте в ветке my-new-branch следующие пять файлов: a.py, b.py, c.py, d.py, e.py - лучше через веб интерфейс
2. Скачайте изменения локально
   git pull origin my-new-branch
3. Выполните команду git rebase -i HEAD~5

```
pick 5f2e0b9 Create a.py
pick bee1709 Create b.py
pick 2419733 Create c.py
pick 1dbd711 Create d.py
pick a1cfc5c Create e.py

# Rebase 1e77a17..a1cfc5c onto 1dbd711 (5 commands)
#
# Commands:
# p, pick <commit> = use commit
# r, reword <commit> = use commit, but edit the commit message
# e, edit <commit> = use commit, but stop for amending
# s, squash <commit> = use commit, but meld into previous commit
# f, fixup <commit> = like "squash", but discard this commit's log message
# x, exec <command> = run command (the rest of the line) using shell
# b, break = stop here (continue rebase later with 'git rebase --continue')
# d, drop <commit> = remove commit
# l, label <label> = label current HEAD with a name
# t, reset <label> = reset HEAD to a label
# m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]
# .       create a merge commit using the original merge commit's
# .       message (or the oneline, if no original merge commit was
# .       specified). Use -c <commit> to reword the commit message.
"~/Desktop/1/GitTraining/.git/rebase-merge/git-rebase-todo" 30L, 1217C
```

4. Изменим названия коммитов “Create b.py и d.py”
   - Напротив тех коммитов, которые хотим исправляем, нужно поменять pick на нужную команду. В нашем случае это reward.
   - Потом в файлах, которые появляются исправить Create b.py на новый комментарий. Для d.py - так же.
5. Выполняем git rebase --continue.
6. Повторяем действие со вторым коммитом.
7. Выполнится ли git push origin my-new-branch?
   - не выполняется, ошибка(

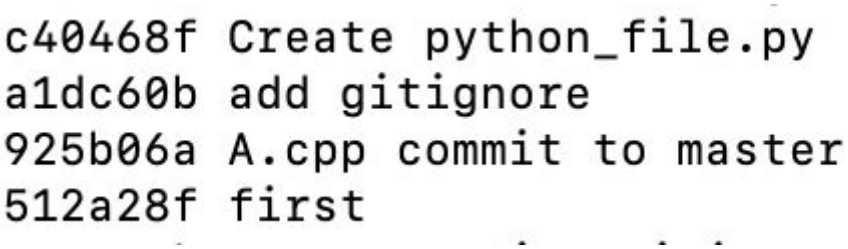
```
To https://github.com/AnnNesterenko/GitTraining.git
 ! [rejected]        my-new-branch -> my-new-branch (non-fast-forward)
error: failed to push some refs to 'https://github.com/AnnNesterenko/GitTraining
.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Integrate the remote changes (e.g.
hint: 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

8. Разберитесь с опцией squash - как слить все коммиты в один?
   Аналогично изменению одного/нескольких комментариев, мы вызываем git rebase -i HEAD~<нужное число> и у всех коммитов, кроме первого меняем pick на squash. Когда мы сохраним это изменение и выйдем, то откроется новый документ, в котором можно написать общее сообщение для объединенных коммитов.
   *делала как тут описано
   https://git-scm.com/book/ru/v2/Инструменты-Git-Исправление-истории

Вопрос на 7.

1. Создайте коммит с созданным python-файлом, закоммитьте в ветку master. Как запушить только этот коммит в local remote? Продемонстрируйте это.

   Посмотрим какой токен у сделанного коммита через git log --pretty=format:"%h %s".

   Потом наверное можно написать git push local <токен коммита>:master, но у меня не заработало