Практическая работа 1

Цель

- Создать локальный и удаленный репозитории
- Связать их
- Добавлять и изменять файлы

Далее все примеры приведены с использованием bash команд. Если вы используете windows, то некоторые команды командной строки могут отличаться.

Задание 1. Регистрация на сайте github.com

1. Перейдите на сайт github.com
2. Пройдите регистрацию. Используйте почту, к которой у вас есть доступ. Все данные вводите *латиницей.*

Задание 2. Генерация токена

Данный токен нужен для того, чтобы вносить изменения в удаленном репозитории.

1. Справа вверху нажмите на свой профиль -> Settings
2. Слева внизу -> Developer Settings
3. Слева -> Personal access tokens -> Tokens (classic)
4. Generate new token (classic)
5. Далее введите:
    1. Имя токена (Token name)
    2. Срок действия токена (Expiration)
    3. Доступ (Repository Access) -> All repositories
    4. Generate token (обязательно сохраните его значение) Токен сгенерирован. Теперь во время подключения к удаленному репозиторию вместо пароля нужно будет записывать данный токен.

Задание 3. Первичная настройка git

1. В терминале проверьте, установлен ли git.

```
git --version
```

2. Если возникает ошибка, значит нужно скачать git с официальной сайта и установить
3. Настройки конфигурации git. Имя фамилия *латиницей.* Почта, которая указана в github

```
git config --global user.name "Имя Фамилия"
git config --global user.email почта@example.com
```

Задание 4. Создание локального репозитория

1. Создайте папку вашего проекта

```
mkdir my_project
cd my_project
```

2. Инициализируем локальный репозиторий

```
git init
```

Если все хорошо, то должна появиться надпись: `Initialized empty Git repository in путь к проекту/my_project/.git/` Теперь все изменения в папке `my_project` будут фиксироваться системой git.

Задание 5. Создание удаленного репозитория

1. Перейдите на сайт github.com
2. Справа сверху нажимаете на профиль -> Repositories
3. New
4. Далее указываете:
    1. Имя (Repository name) -> `my_project` (желательно, чтобы имена локального и глобального репозиториев *совпадали*, но не обязательно)
    2. Доступ. Choose visibility -> Public
    3. No template
    4. Off
    5. No .gitignore
    6. No license
    7. Create repository
5. После создания вы будете перенаправлены на страницу вашего удаленного репозитория. Сверху можете увидеть вкладки *Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, Settings.*

Задание 6. Связывание локального и удаленного репозитория

1. Перейдите по вкладке *Code*. Ниже будет написана полная инструкция, но продолжим по порядку
2. Ниже увидите блок с ссылкой на текущий репозиторий (*Quick setup — if you've done this kind of thing before*)
3. Выбираете *https* и копируете ссылку
4. Далее в терминале связываем локальный репозиторий с глобальным

```
git remote add origin ваша/ссылка/на/my_project.git
```

5. Если ничего не произошло, значит репозитории **связаны** без ошибок

Задание 7. Добавление файла

1. Создайте файл `main.py` со следующим содержимым

```
print('Hello, GitHub!')
```

2. Теперь проверим, что произошло в локальном репозитории

```
git status
```

Будет выведена следующая запись

```
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        main.py

nothing added to commit but untracked files present (use "git add" to track)
```

Что это значит?

- Мы находимся в основной ветке *master*
- Никаких коммитов не сделано
- Есть неотслеживаемые файлы. В нашем случае новый файл *main.py*

3. Сделаем новый файл *отслеживаемым* и снова проверим статус

```
git add .
git status
```

Будет выведена следующая запись:

```
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   main.py
```

Теперь git предлагает создать коммит с измененным *main.py* **Коммит в git** — это как сохранение в игре. Когда ты сделал какие-то изменения в проекте и сделал коммит, git запоминает текущее состояние файлов (что добавилось, изменилось, удалилось). 4. Совершим коммит с комментарием *Add main.py.* Комментарии необходимы для того, чтобы программисты могли по короткой записи понимать, что произошло на данном этапе.

```
git commit -m "Add main.py"
```

5. Самостоятельно проверьте, что изменилось `git status`

Задание 8. Отправка изменений на удаленный репозиторий

На текущий момент локально был добавлен файл *main.py* и информация об этом файле добавлена в git. Далее мы хотим синхронизировать наш проект с удаленным. для этого нужно отправить все изменения на сервер github, к которому мы ранее подключились.

1. Проверим, что сейчас находится в удаленном репозитории. Обновим страницу сайта. Ничего не изменилось, то есть удаленный репозиторий пустой.

2. Отправим на сервер все наши коммиты (сохранения)

```
git push --set-upstream origin master
```

Мы отправили изменения на удаленный сервер и связали с удаленной веткой *master*. Далее, если мы будем работать с одной веткой, то достаточно писать `git push` 3. Снова перезагрузите страницу github. Теперь можете заметить, что *main.py* загружен в удаленный.

Задания для самостоятельной работы

1. Добавьте в папку с проектом два файла *.gitignore* и *README.md*. В readme можете написать какую-нибудь информацию о вашем проекте. .gitignore пока пусть остается пустым. Добавьте их в git, закоммитьте и запушьте.
2. В файл *main.py* добавьте все функции, которые вы написали, выполняя Лабораторную работу №2. Отправьте это все на удаленный репозиторий. Теперь все выполненные задания хранятся в надежном месте и вы их не потеряете!