

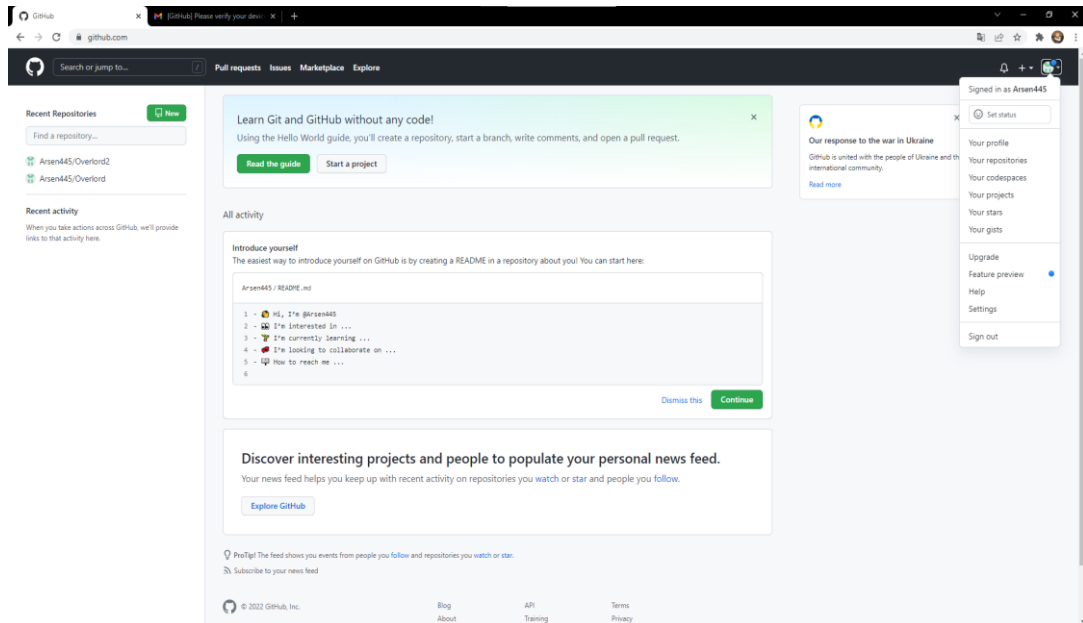
Лабораторная работа №1

Выполнил Эсеналиев Арсен

ИБТ-б-о-21-1

Цель: научиться пользоваться гитхабом.

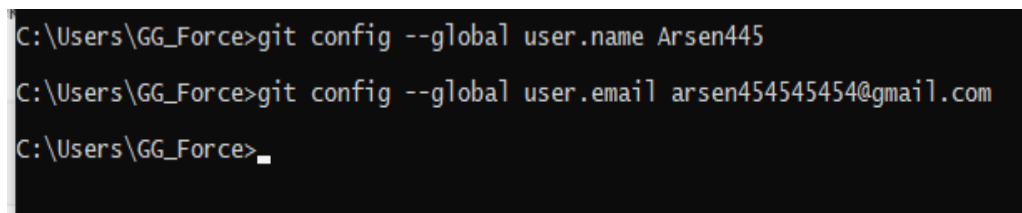
1) Авторизируемся на гитхаб.



2) Устанавливаем гитхаб на ПК и проверяем был ли он установлен успешно.




3) Добавляем в Гит имя и адрес эл.почты.



4) Создаем новый репозиторий


Owner * Repository name *


 Arsen445 ▾ / LB1.1 ✓

Great repository names are short and memorable. Need inspiration? How about [potential-octo-pancake?](#)

Description (optional)

LB1.1

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:
Skip this step if you're importing an existing repository.


☐ **Add a README file**
This is where you can write a long description for your project. [Learn more.](#)

☒ **Add .gitignore**
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: Python ▾

☒ **Choose a license**
A license tells others what they can and can't do with your code. [Learn more.](#)

License: MIT License ▾

This will set  main as the default branch. Change the default name in your [settings](#).

Create repository

5) Клонировем репозиторий

```
C:\Users\GG_Force>git clone https://github.com/Arsen445/LB1.1.git
Cloning into 'LB1.1'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.

C:\Users\GG_Force>
```

6) Дополняем файл гитигнор

```
<> Edit file    Preview changes    Spaces 2 No wrap

19 downloads/
20 eggs/
21 .eggs/
22 lib/
23 lib64/
24 parts/
25 sdist/
26 var/
27 wheels/
28 share/python-wheels/
29 *.egg-info/
30 .installed.cfg
31 *.egg
32 MANIFEST
33
34 # PyInstaller
35 # Usually these files are written by a python script from a template
36 # before PyInstaller builds the exe, so as to inject date/other infos into it.
37 *.manifest
38 *.spec
39
40 # Installer logs
41 pip-log.txt
42 pip-delete-this-directory.txt
43
44 # Unit test / coverage reports
45 htmlcov/
46 .tox/
47 .nox/
48 .coverage
49 .coverage.*
50 .cache
51 nosetests.xml
52 coverage.xml
53 *.cover
54 *.py.cover
```

7) Добавляем ридми

Arsen445 / LB1.1 Public

Pin Unwatch 1 Fork 0 Star 0

<> Code Issues Pull requests Actions Projects Wiki Security Insights

main

Go to file Add file Code

Arsen445 Create README.md now 3

.gitignore Update .gitignore 2 minutes ago

LICENSE Initial commit 41 minutes ago

README.md Create README.md now

README.md

LB1.1

LB1.1 ИБТ-6-о-21-1 Эсеналиев Арсен

About

LB1.1

Readme MIT License 0 stars 1 watching 0 forks

Releases

No releases published [Create a new release](#)

Packages

No packages published [Publish your first package](#)

8) Первый комит

The screenshot shows a code editor window titled 'ПРОГРАММА.py - D:\Ov...' with a menu bar (File, Edit, Format, Run) and a single line of Python code: `print('x y z w')`. To the right is a 'Git CMD' terminal window. It displays the command `git add .` being entered multiple times, followed by `git commit -m "Добавлен комит"`. The terminal output shows the commit hash `8acb394` and details: `1 file changed, 1 insertion(+)` and `create mode 160000 LB1.1`.

```
ПРОГРАММА.py - D:\Ov...
File Edit Format Run
print('x y z w')

Git CMD
The most similar command is
add
D:\Overlord>git add .
D:\Overlord>git add .
D:\Overlord>git add .
D:\Overlord>
D:\Overlord>
D:\Overlord>
D:\Overlord>
D:\Overlord>
D:\Overlord>
D:\Overlord>
D:\Overlord>
D:\Overlord>
D:\Overlord>
D:\Overlord>
D:\Overlord>
D:\Overlord>
D:\Overlord>
D:\Overlord>git commit -m "Добавлен комит"
[main (root-commit) 8acb394] Добавлен комит
1 file changed, 1 insertion(+)
create mode 160000 LB1.1
D:\Overlord>
```

9) 2 КОММИТ

The screenshot shows a code editor window titled 'asd.py - D:\LB1.1\asd.py (3.9.6)' with a menu bar (File, Edit, Format, Run, Options, Window, Help). The code contains a loop: `print('x y z w')` followed by `for y in 0, 1:`.

```
asd.py - D:\LB1.1\asd.py (3.9.6)
File Edit Format Run Options Window Help
print('x y z w')
for y in 0, 1:
```

10) 3 КОММИТ

The screenshot shows a code editor window titled 'asd.py - D:\LB1.1\asd.py' with a menu bar (File, Edit, Format, Run) and code: `print('x y z w')`, `for y in 0, 1:`, and `for x in 0, 1:`. The 'Git CMD' terminal shows the command `git commit -m "Добавлен комит 3"` being entered. The terminal output includes a warning that the branch is ahead of 'origin/main' and lists changes not staged for commit, specifically `modified: asd.py`. It then shows `git add .` and another `git commit -m "Добавлен комит 3"` command, resulting in a new commit hash `0566c64` with `1 file changed, 2 insertions(+)`.

```
asd.py - D:\LB1.1\asd.py
File Edit Format Run
print('x y z w')
for y in 0, 1:
for x in 0, 1:

Git CMD
D:\LB1.1>git commit -m "Добавлен комит 3"
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   asd.py

no changes added to commit (use "git add" and/or "git commit -a")
D:\LB1.1>git add .
D:\LB1.1>git commit -m "Добавлен комит 3"
[main 0566c64] Добавлен комит 3
1 file changed, 2 insertions(+)
D:\LB1.1>
```

11) 4 КОММИТ

The screenshot shows a code editor window titled 'asd.py - D:\LB1.1\asd.py (3.9.6)' with the following Python code:

```
print('x y z w')
for y in 0, 1:
    for x in 0, 1:
        for z in 0, 1:
```

To the right, a Git CMD terminal window shows the output of 'git status' and 'git log'.

```
D:\LB1.1>git status
On branch main
Your branch is ahead of 'origin/main' by 3 commits.
(use "git push" to publish your local commits)

nothing to commit, working tree clean

D:\LB1.1>git log
commit 855e8f0906866ca13162d5ecda8795fee0b6ba93 (HEAD -> main)
Author: Arsen445 <arsen454545454@gmail.com>
Date: Sat Mar 12 23:36:06 2022 +0300

Добавлен комит 4
```

12) 5 КОММИТ

The screenshot shows a code editor window titled 'asd.py - D:\LB1.1\asd.py (3.9.6)' with the following Python code:

```
print('x y z w')
for y in 0, 1:
    for x in 0, 1:
        for z in 0, 1:
            for w in 0, 1:
```

To the right, a Git CMD terminal window shows the output of 'git log'.

```
Git CMD - "C:\Program Files\Git\cmd\git.exe" log
Date: Sat Mar 12 22:10:48 2022 +0300

Create README.md

D:\LB1.1>git add .
D:\LB1.1>git commit -m "Добавлен комит 5"
[main c05b8a5] Добавлен комит 5
1 file changed, 1 insertion(+)

D:\LB1.1>git log
commit c05b8a5d450d705ac8a446ae950e1ef21b777a0b (HEAD -> main)
Author: Arsen445 <arsen454545454@gmail.com>
Date: Sat Mar 12 23:39:35 2022 +0300

Добавлен комит 5

commit 855e8f0906866ca13162d5ecda8795fee0b6ba93
Author: Arsen445 <arsen454545454@gmail.com>
Date: Sat Mar 12 23:36:06 2022 +0300

Добавлен комит 4

commit 0566c64f66ff59b0335b2ae2ba5f92b66f15b77b
```

13) 6 КОММИТ

The screenshot shows a code editor window titled 'asd.py - D:\LB1.1\asd.py (3.9.6)' with the following Python code:

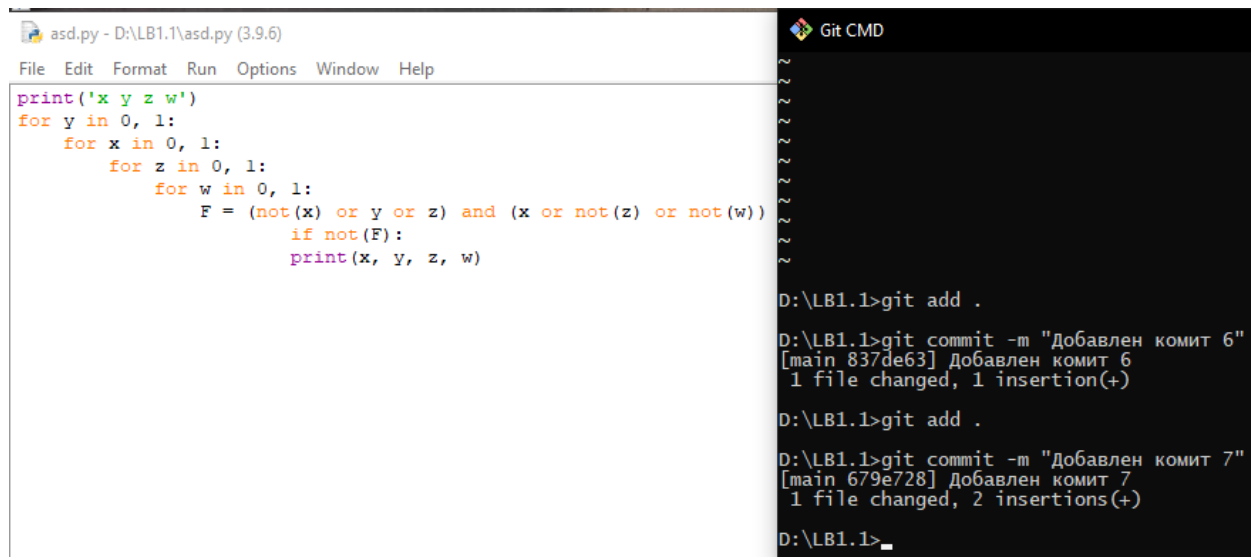
```
print('x y z w')
for y in 0, 1:
    for x in 0, 1:
        for z in 0, 1:
            for w in 0, 1:
                F = (not(x) or y or z) and (x or not(z) or not(w))
```

To the right, a Git CMD terminal window shows the output of 'git add' and 'git commit'.

```
Git CMD
D:\LB1.1>git add .
D:\LB1.1>git commit -m "Добавлен комит 6"
[main 837de63] Добавлен комит 6
1 file changed, 1 insertion(+)

D:\LB1.1>
```

14) 7 КОММИТ



The image shows a screenshot of a Python IDE window titled 'asd.py - D:\LB1.1\asd.py (3.9.6)' and a Git CMD terminal window. The IDE contains a Python script with nested loops and a logical expression. The Git CMD terminal shows a series of commands and their outputs, including adding files and committing them with messages in Russian.

```
print('x y z w')
for y in 0, 1:
    for x in 0, 1:
        for z in 0, 1:
            for w in 0, 1:
                F = (not(x) or y or z) and (x or not(z) or not(w))
                if not(F):
                    print(x, y, z, w)
```

```
D:\LB1.1>git add .
D:\LB1.1>git commit -m "Добавлен комит 6"
[main 837de63] Добавлен комит 6
1 file changed, 1 insertion(+)
D:\LB1.1>git add .
D:\LB1.1>git commit -m "Добавлен комит 7"
[main 679e728] Добавлен комит 7
1 file changed, 2 insertions(+)
D:\LB1.1>
```

1. Что такое СКВ и каково ее назначение?
2. В чем недостатки локальных и централизованных СКВ?
3. К какой СКВ относится Git?
4. В чем концептуальное отличие Git от других СКВ?
5. Как обеспечивается целостность хранимых данных в Git?
6. В каких состояниях могут находиться файлы в Git? Как связаны эти состояния?
7. Что такое профиль пользователя в GitHub?
8. Какие бывают репозитории в GitHub?
9. Укажите основные этапы модели работы с GitHub.
10. Как осуществляется первоначальная настройка Git после установки?
11. Опишите этапы создания репозитория в GitHub.
12. Какие типы лицензий поддерживаются GitHub при создании репозитория?

13. Как осуществляется клонирование репозитория GitHub? Зачем нужно клонировать

репозиторий?

14. Как проверить состояние локального репозитория Git?

15. Как изменяется состояние локального репозитория Git после выполнения следующих

операций: добавления/изменения файла в локальный репозиторий Git; добавления нового/

измененного файла под версионный контроль с помощью команды `git add` ; фиксации

(коммита) изменений с помощью команды `git commit` и отправки изменений на сервер с

помощью команды `git push` ?

16. У Вас имеется репозиторий на GitHub и два рабочих компьютера, с помощью которых Вы

можете осуществлять работу над некоторым проектом с использованием этого репозитория.

Опишите последовательность команд, с помощью которых оба локальных репозитория,

связанных с репозиторием GitHub будут находиться в синхронизированном состоянии.

Примечание: описание необходимо начать с команды `git clone` .

17. GitHub является не единственным сервисом, работающим с Git. Какие сервисы еще Вам

известны? Приведите сравнительный анализ одного из таких сервисов с GitHub.

18. Интерфейс командной строки является не единственным и далеко не самым удобным

способом работы с Git. Какие Вам известны программные средства с графическим

интерфейсом пользователя для работы с Git? Приведите как реализуются описанные в

лабораторной работе операции Git с помощью одного из таких программных средств.