**Лабораторна робота №1**

**Тема: Основи роботи з Git.**

**Мета роботи:** Ознайомитися з основами **Git** і його командами. Навчитися працювати з віддаленим репозиторієм **GitHub**. навчитися працювати і налаштовувати **VS Сode.**

**Теорія**

**VS code –** кросплатформенний редактор містить вбудований дебагер, інструменти для роботи з Git і засоби рефакторингу, навігації по коду, автодоповнення типових конструкцій і контекстної підказки.

**Git (Гіт)** — система контролю й управління версіями файлів.

**GitHub (Гітхаб)** —Web-сервіс для розміщення репозиторіїв для спільної розробки. Працює за принципами Git і є хостингом IT-проектів. Надає весь необхідний функціонал для роботи Git. Є подібні сервіси: GitLab і BitBucket.

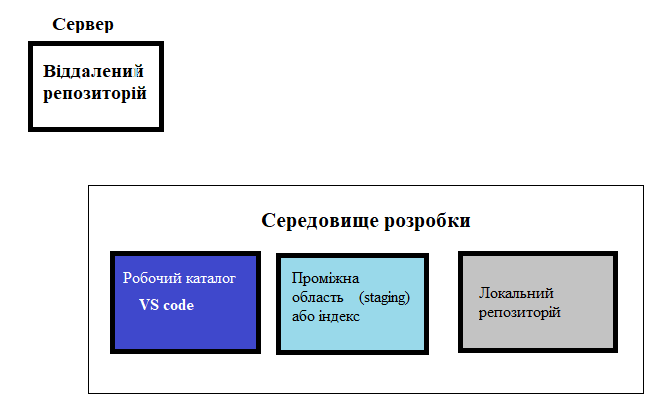
****

Рис.1 Схема розподілу системи контроля версій

**Репозиторій (Repository)** - каталог файлової системи, в якій містяться файли конфігурації, файли журналів операцій, що виконуються в репозиторії, індекс розташування файлів і сховище, що зберігає найбільш контрольовані версії файлів.

**Основні команди git:**

$ git init

Команда ініціалізує новий підкаталог .git, який містить всі необхідні файли вашого репозиторія — скелет Git-репозиторія. На цей момент, у вашому проєкті ще нічого не відстежується. (Див Git зсередини для отримання додаткової інформації про файли, що містяться в каталозі .git, котрий ви щойно створили).

$ git clone gitHubNameRepository

Команда клонує папку с проєктом з глобального репозиторію. Наприклад з віддаленого репозиторію: <https://github.com/beyretb/AnimalAI-Olympics>

$ git clone projectName2 <https://github.com/beyretb/AnimalAI-Olympics>

Якщо ви бажаєте зробити клон репозиторія в директорію з іншою назвою, ви можете передати її як другий параметр.

$ git add .

Команда додає всі існуючі файли під версійний контроль (на відміну від порожнього каталогу), індексує ці файли. Ви додаєте файли в індекс, у такий спосіб їх зліпки додаються в область підготовлених файлів.

$ git commit -m ‘Коментар’

Коли Ви робите коміт, використовуються файли з індексу, даний зліпок зберігається у Вашу Git-директорію.

$ git status

Команда відображає стан робочого каталогу і розділу проіндексованих файлів. З її допомогою можна перевірити індексацію змін і побачити файли, які не відслідковуються Git. Інформація про історію коммітів проєкту не відображається при виведенні даних про стан.

$ git log

Команда відображає інформацію про історію коммітів проєкту.

$ git branch **<ім’я гілки>**

Команда створює нову гілку.

$ git branch

Команда перевіряє актуальний стан гілок.

$ git checkout <hash> або <treehash>

Переключає на вказану гілку, або комміт. Команда git checkout дозволяє переміщатися між гілками, створеними командою git branch.

$ git checkout –b newBranch

Команда створює нову гілку і відразу на неї переходить.

$ git cat-file –p < hash >

Команда показує зміст git об’єктів.

$ find .git/objects –type f

Команда показує вміст папки об’єктів.

git remote add origin <адреса вашого репозиторію>

Командає git remote add origin підключає віддалений репозиторій до локального.

git push -u origin master

git push 'remote\_name' 'branch\_name'

Команда git push, яка публікує проєкт на віддалений репозиторій в вказану гілку master за замовчанням.

(так само, як назва гілки “master” не має якогось особливого значення для Git, так само й “origin”. Просто “master” дається за-замовчуванням для початкової гілки, коли ви запускаєте git init — ось чому воно так часто зустрічається, а “origin” — це ім’я за-замовчуванням для віддалених посилань команди git clone.)

git merge

Команда git merge пов'язує низку коммітів в одне ціле. У свою чергу git створює коміт злиття, де й поєднуються зміни обох послідовностей.

git pull

Команда git pull відповідає за скачування даних із сервера. При виконані команди завантажуються не всі комміти, а лише нові.

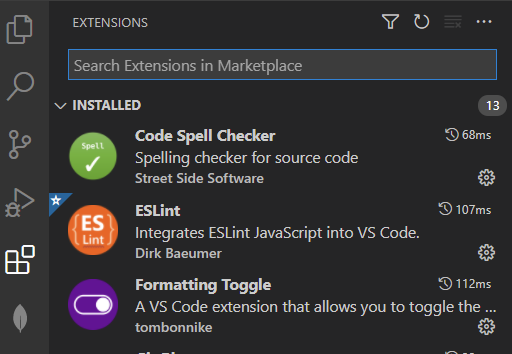
**Файл . gitignore**

Файл .gitignore в своєму репозиторії, Git використовує, щоб визначити, які файли і каталоги (або цілі шаблони) ігнорувати, перш ніж здійснювати commit. Зазвичай він використовується, щоб уникнути передачі тимчасових файлів з вашого робочого каталогу, які не є корисними для інших співавторів, таких як продукти компіляції, файли налаштувань проекту, створені середовищем розробки і тощо.

**!.gitignore це не розширення файлу, а його ім'я.**

**Хід роботи**

**Завдання 1. Налаштування VS Code**

1. Відкрити **VS Code**
2. Вивчіть інтерфейс
3. Установіть додаткові розширення (Extention), які вбудовані в VS Code

***-Prettier*** – форматує код;

***-*** Git Graph

1. Знайдіть вкладку системи контроля версій
2. Створіть на диску :d/Foldername/taskName\_Surname

***Додатково: Ви можете створити папку через термінал VS Code використовуючи команду:***

$ mkdir Foldername\taskName\_Surname

1. Використовуючи вкладку VS code перейдіть до папки вашого проєкту.

File/Open Folder/відкрийте папку з проєктом

***Додатково: Ви можете перейти до папки через термінал VS Code використовуючи команду:***

*для Linux:*

**$ cd** Foldernam**\**taskName\_Surname

*для Mac:*

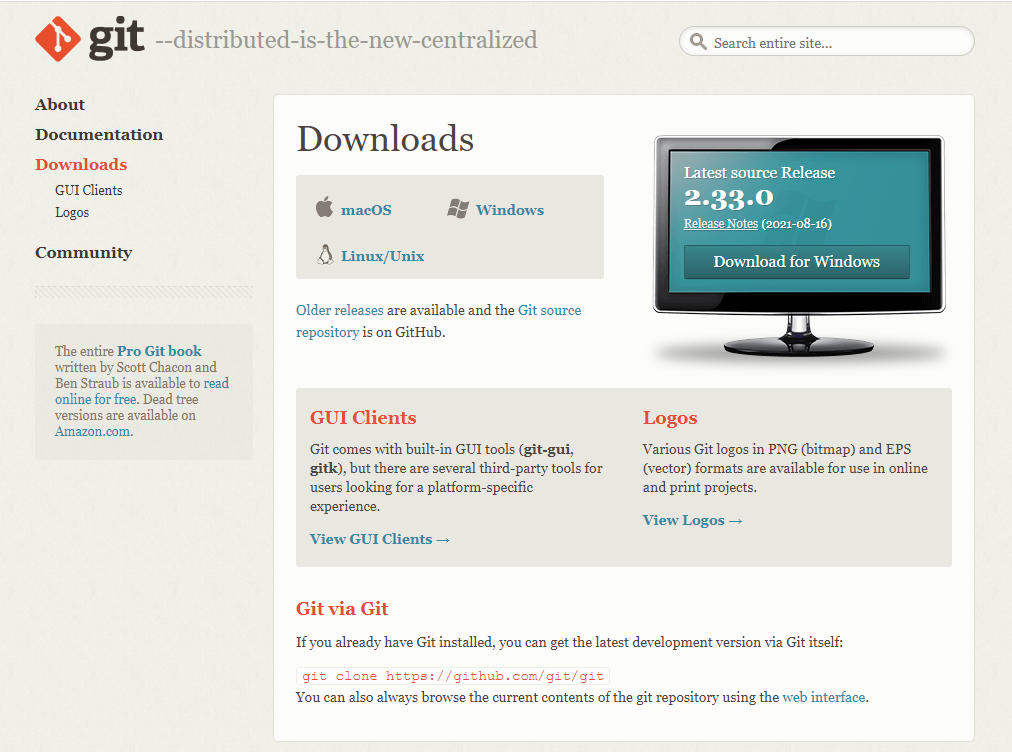
**$ cd /Users/**taskName\_Surname

*для Windows:*

**$ cd .\**Foldername\taskName\_Surname

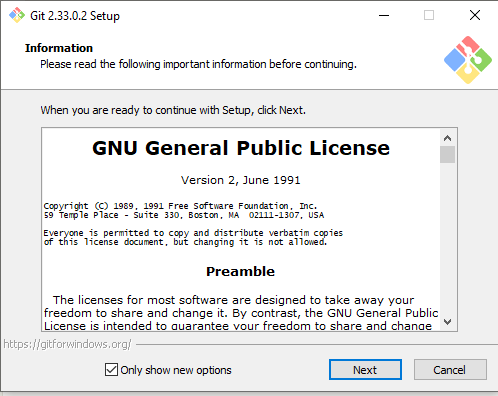
**Завдання 2. Установка Git**

1. Cкачати git з сайту: <https://git-scm.com/downloads>

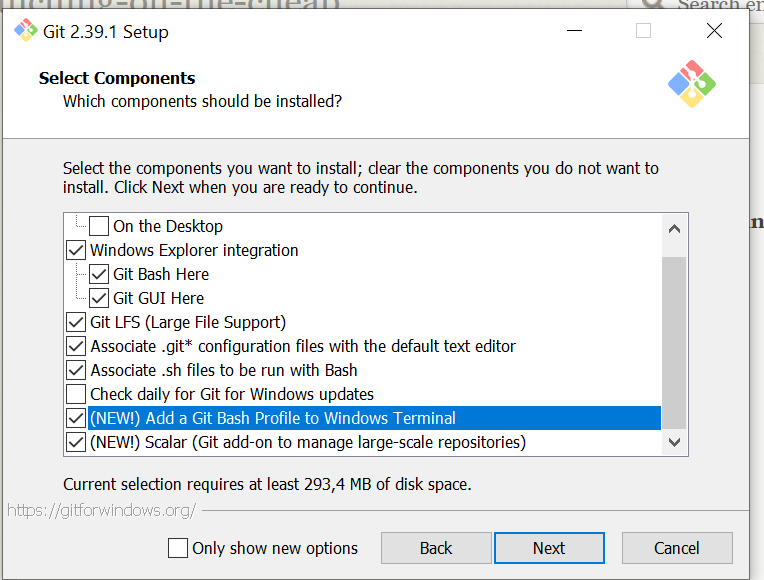


\*Якщо у вас Mac-Book –тоді для Download for Mac (пропонує автоматично)

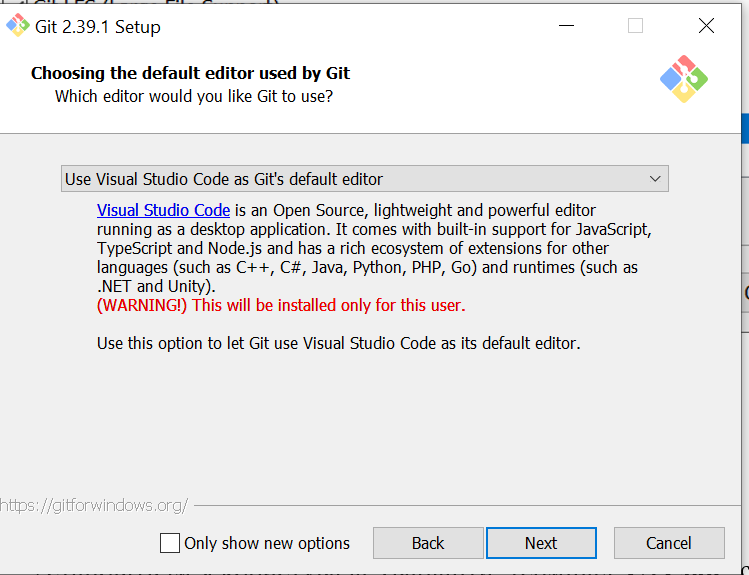
2.

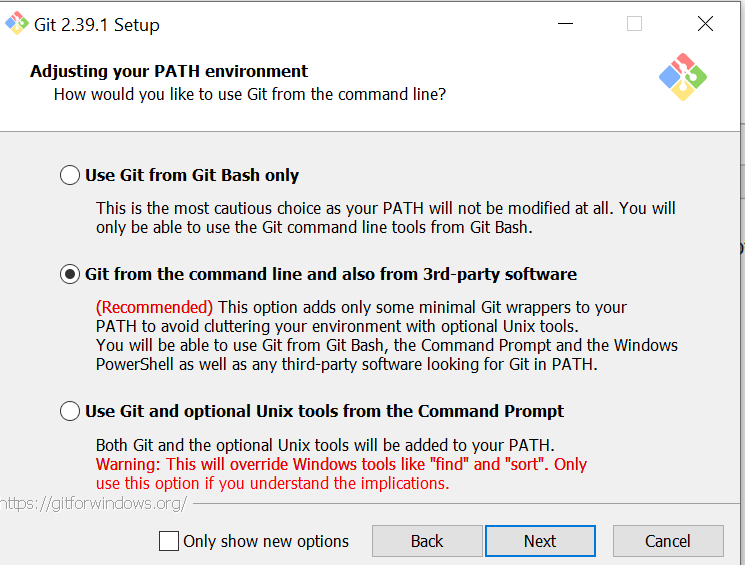


3.



4. Next і т.д.



в кінці Install

5. Перейдіть в термінал VS Code

**Завдання 3.**

1. Установіть ім’я користувача (варіанти: термінал **VS Code**, cmd, Power shell)
2. **Важливо!!!** вказувати пошту і профіль, яку вказували в **Git-Hub**

Для зручності використовуєте вбудований термінал VSCode

Введіть в терміналі почергово команди:

$ git config --global user.name "Name Surname"

$ git config --global user.email surname@example.com

Перше, що вам слід зробити після установки Git - вказати ваше ім'я та адресу електронної пошти. Це важливо, тому що кожен комміт в Git містить цю інформацію, і вона включена в комміти, що передаються вами, і не може бути далі змінена:

Якщо вказана опція --global, то ці настройки досить зробити тільки один раз, оскільки в цьому випадку Git буде використовувати ці дані для всього, що ви робите в цій системі. Якщо для якихось окремих проєктів ви хочете вказати інше ім'я або електронну пошту, можна виконати цю ж команду без параметра --global в каталозі з потрібним проєктом.

2. Щоб подивитися всі встановлені налаштування і дізнатися де саме вони задані, використовуйте команду:

$ git config --list --show-origin

**Завдання 4.**

1. Виконати ініціалізацію локального репозиторію в папці проєкту:

* *Зайдіть в створену папку і перевірте, чи створили папку .git і в*ивчіть вміст папки .git. Переконайтеся, що відсутній файл індексу .git / index
* Переконайтеся, що покажчик HEAD вказує на гілку main (або master)

1. Додайте в проєкт файл Readme.txt.

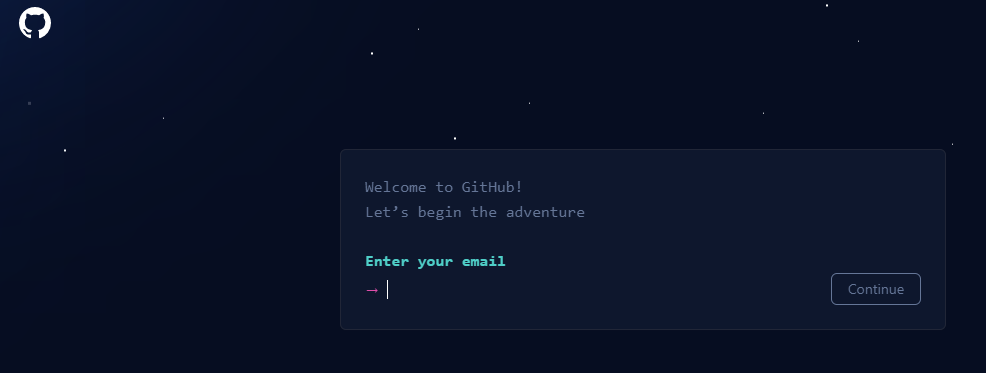
* Перегляньте статус локального репозиторію і запишіть його в файл Readme.txt
* Виконайте додавання файла Readme.txt до індексування
* Запишіть в Readme.txt вміст папки *.git / objects*;
* Запишіть зміст кожного git-об’єкта в файл Readme.txt;
* Виконайте додавання файла Readme.txt до індексування;
* Зробіть комміт. (*Коментар до коммітів записуйте довільно, але так щоб він відповідав на питання: Що робить даний комміт*);

1. Додайте в проєкт файл gitHistory.txt

* Перевірте статус і запишіть його в gitHistory.txt; Внесіть зміни до файлу і знов подивіться статус.
* Виконайте додавання файла до індексування, зробіть комміт;
* Продивіться історію коммітів і запишіть її в gitHistory.txt;
* Виконайте додавання файла до індексування, зробіть комміт;

**Завдання 5. Реєстрація в GitHub**

1. Перейдіть на сайт **GitHub**: <https://github.com/>
2. Зареєструйтесь в **GitHub.**

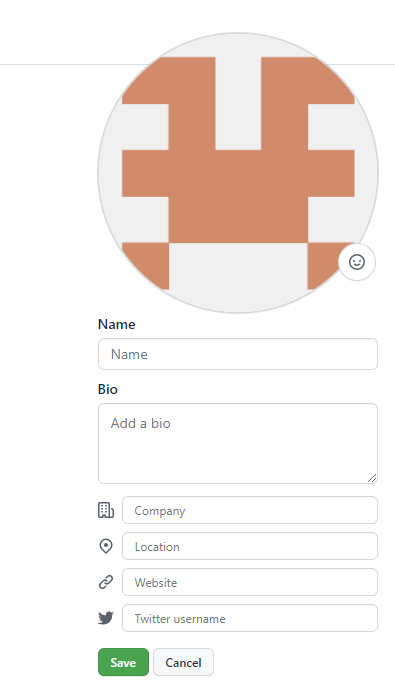


**Важливо!!!** вказувати пошту і профіль, яку вказували в **Git**

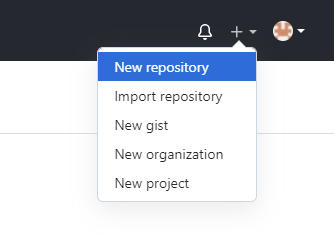
**(**user.email [Surname@example.com](mailto:Surname@example.com)

user.name "Name Surname"**)**

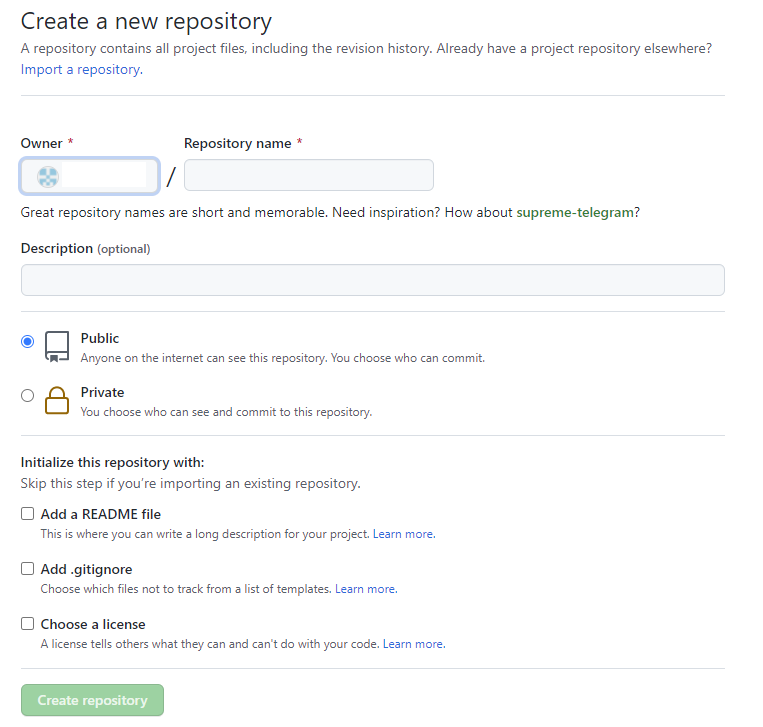
1. Заповніть профіль:



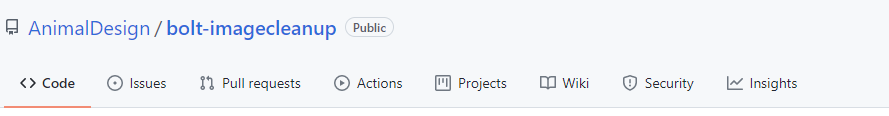
**Завдання 6. Створення репозиторію на GitHub:**

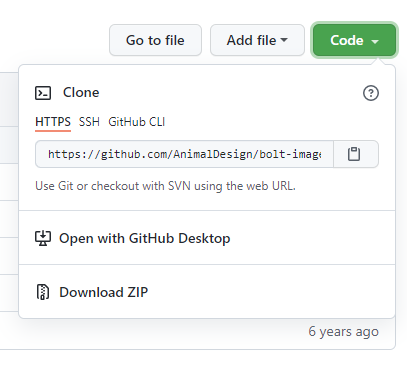


- Введіть назву: **projectName *(ім’я має бути таке саме як і локального проєкту) (*Public – за замовчуванням)**

****

1. Перейдіть на головну сторінку вашого профілю на Git-Hub вибрати вкладку Code, в ній міститься посилання на ваш репозиторій



* 1. Вибрати під вкладку Code і скопіювати адрес

**Завдання 7. Підключення і відправка файлів на віддалений репозиторій.**

1. Додати зв'язок з віддаленим репозиторієм.
2. Завантажити локальний проєкт на Github і переконатися, що всі файли загрузились.
3. В локальному проєкті створити на основі гілки master, ще одну гілку newBranch.

* Додати в гілку index.py;
* Виконайте додавання файла до індексування, зробіть комміт;
* Відправити гілку на Github;
* Зробити PullRequest і merge гілок;
* Завантажте зміни із зазначеного віддаленого репозиторію в локальний в головну гілку;

**Завдання 8. Клонування віддаленого репозиторію.**

* Створіть новий репозиторій на Github ;
* Склонуйте його до локального репозиторію;
* Своріть файл і виконайте додавання файла до індексування, зробіть комміт;
* Відправте до віддаленого репозиторію.

Підготувати звіт про виконану роботу: **посилання на GitHub.**

**Контрольні питання**

1. Що таке система контролю версій?
2. Перерахуйте основні команди git
3. Опишіть структру git проєкту та його складові
4. При яких умовах файл стає відслідкованим і що відбуваєтся з git

Об’єктом при цьому?

1. Як перевірити статус каталогу файлів. Які стани характерні для git?
2. Що таке репозиторій. Які типи репозиторіів ви знаєте. Які сучасні репозиторії використовують git ?

**Критерій прийому:**

Посилання на Git-hub.

Виконані всі завдання.

Теоретичний захист лр

**Оцінювання:**

Максимальний бал – 5