**Библиотека Streamlit:**

Запуск сервера - streamlit run <.\streamlit\_app.py>

**Оформление проекта Python:**

Virtual environment:

Запуск виртуального окружения - Scripts\activate

Создание файла requirements.txt 🡪 pip freeze > requirements.txt

**Docker:**

* Работа с виртуальным UBUNTU из образа Linux

docker run -it myubuntu # «myubuntu» название образа личного образа.

* Просмотр образов

docker ps

* Сохранение образа рабочего Ubuntu со всеми настройками и обновлениями:

docker commit -p container\_id new\_container\_name

пример:

sudo docker commit -p 524aa76baafb myubuntu

unminimize – команда для расширения урезанной версии линукс до полной

**GIT:**

**Работа с локальным репозиторием:**

1. git status – показывает статус файлов в проекте
2. git add . – добавляет файлы для отслеживания
3. git commit – добавляет файлы в локальный git
4. git log –oneline – выводит информацию о добавленных commit

**Работа с удаленным репозиторием:**

git init – инициализация git

git push -u origin main – публикация кода в репозиторий git

git remote add origin [git@github.com:NeurocoderAI/test\_ci\_cd.git](mailto:git@github.com:NeurocoderAI/test_ci_cd.git) - подключение к репозиторию GIT (указывается команда git remote add, потом указывается ветка в которую будет отправлен код, потом указывается ссылка на репозиторий)

Команда для объединения двух команд add и commit

git config --global alias.add-commit '!git add -A && git commit'

после добавления изменений в конфиг файл, можно использовать одну команду:

git add-commit -m 'My commit message'

Команда для удаления файлов, которые входят в файл gitignore

Linux Bash:

git ls-files -i -c --exclude-from=.gitignore | xargs git rm --cached

Windows Shell:

git ls-files -i -c --exclude-from=.gitignore | %{git rm --cached $\_}

Создание новой ветки в локальном git, и потом merge (слияние) в основную удаленную ветку.

После git clone на ветке, в которой лежит основной код выполняем команды:

git checkout -b myFeature dev

Выходим из той, и одновременно создаем копию в новой ветке.

Работаем с кодом, потом, коммитим изменения:

git commit -am “Your message”

Теперь слияние на локальном Гит:

git checkout dev

git merge –-no—ff myFeature

И пушим изменения на новую ветку и при желании ветку, в которой велась работа.

git push origin dev

git push origin myFeature

Работа с файлом .gitignore

Правила в вашем .gitignore файле применяются только к неотслеживаемым файлам. Поскольку файлы в этом каталоге уже были зафиксированы в вашем репозитории, вы должны отменить их фиксацию, создать коммит и отправить его на

git rm --cached `git ls-files -i -c --exclude-from=.gitignore`

git commit -m 'Removed all files that are in the .gitignore'

git push origin develope

Который удалит все файлы / папки, которые находятся в вашем git ignore, за исключением того, что вам придется выбирать каждый из них вручную

**CI/CD GitHub Actions:**

1. Создать папку .github
2. В ней создается папка workflows
3. В ней создается файл build\_and\_test.yml. Название может быть другим, расширение всегда .yml
4. В файл <\_\_\_>.yml помещается код для запуска GitHub Actions. Который состоит из имени задания, далее секция on в которой определяются действия пользователя, которые будут запускать действие, например git pull и далее запускаются действия.
5. Действия (jobs) состоят из шагов, первым действием будет запуск операционной системы, судеющими действиями будет инициализация переменной среды, инсталляция обновлений, зависимостей, потом идет запуск main.py

Ниже код:

name: Build and Test  
  
on:  
 push:  
 branches: [main]  
 pull\_request:  
 branches: [main]  
  
jobs:  
 test:  
 runs-on: ubuntu-latest  
  
 steps:  
 - name: Checkout code  
 uses: actions/checkout@v2  
 - name: Set up Python Environment  
 uses: actions/setup-python@v2  
 with:  
 python-version: '3.x'  
 - name: Install Dependencies  
 run: |  
 python -m pip install --upgrade pip  
 pip install -r requirements.txt  
  
 - name: Run Tests  
 run: |  
 python manage.py test

Действие тест так как для Phyton сборки не требуется.

Flecht – загружает данные из удаленного репозитория, и показывает измененный код в удаленной ветке, не изменяя локальный проект. Это безопасный способ ознакомиться с изменениями

Merge – слияние удаленной ветки с локальной.

LINUX:

Просмотр размера папок

sudo du -h --max-depth=1 var/lib/ | sort -rh

-h – вид отображения цифр, тут гигабайты

--max-depth=1 – Глубина просмотра

Var/lib/ - папка которую смотрим

Sort -rh - сортировка от большего к меньшему

**Python in Linux:**

Обновим индекс пакетов и запустим команду с целью обновления пакетов на актуальные релизы.

sudo apt update && sudo apt upgrade -y

Ключ “-y” означает обновить принудительно.

Проверка версии

python3 --version

sudo apt install python3-pip -y – установка PIP и далее необходимых пакетов

pip3 install matplotlib

sudo apt install build-essential libssl-dev libffi-dev python3-dev

Установка пакета для виртуальной среды:

sudo apt install python3-venv -y

Запуск виртуальной среды в конкретной папке

python3 -m venv test\_env

Включение виртуальной среды:

source /bin/activate

Выключение виртуальной среды:

Deactivate

Создается файл командой touch. Имя файла.py

Редактируется в редакторе VIM или NANO одноименными командами

Например: nano.Имя файла.py

В файле пишется код

Потом производится запуск командой:

python3 file\_name.py

------------------------------Команды LINUX-----------------------------------

cat /etc/os-release – Просмотр версии Linux

sudo apt-get update – Обновление пакетов обновлений

apt-get install openjdk-8-jre – Инсталяция java

sudo apt-get install gdebi – команда для установки утилиты, способной сохранять deb пакеты, типа дистрибутивы

PIP

Обновление PIP

python.exe -m pip install --upgrade pip

**Как спрятать ключ пайтон:**

import os

from openai import OpenAI

from dotenv import load\_dotenv

load\_dotenv()

print(os.environ.get("OPENAI\_API\_KEY")) #key should now be available

Ключь помещаем в файл .env