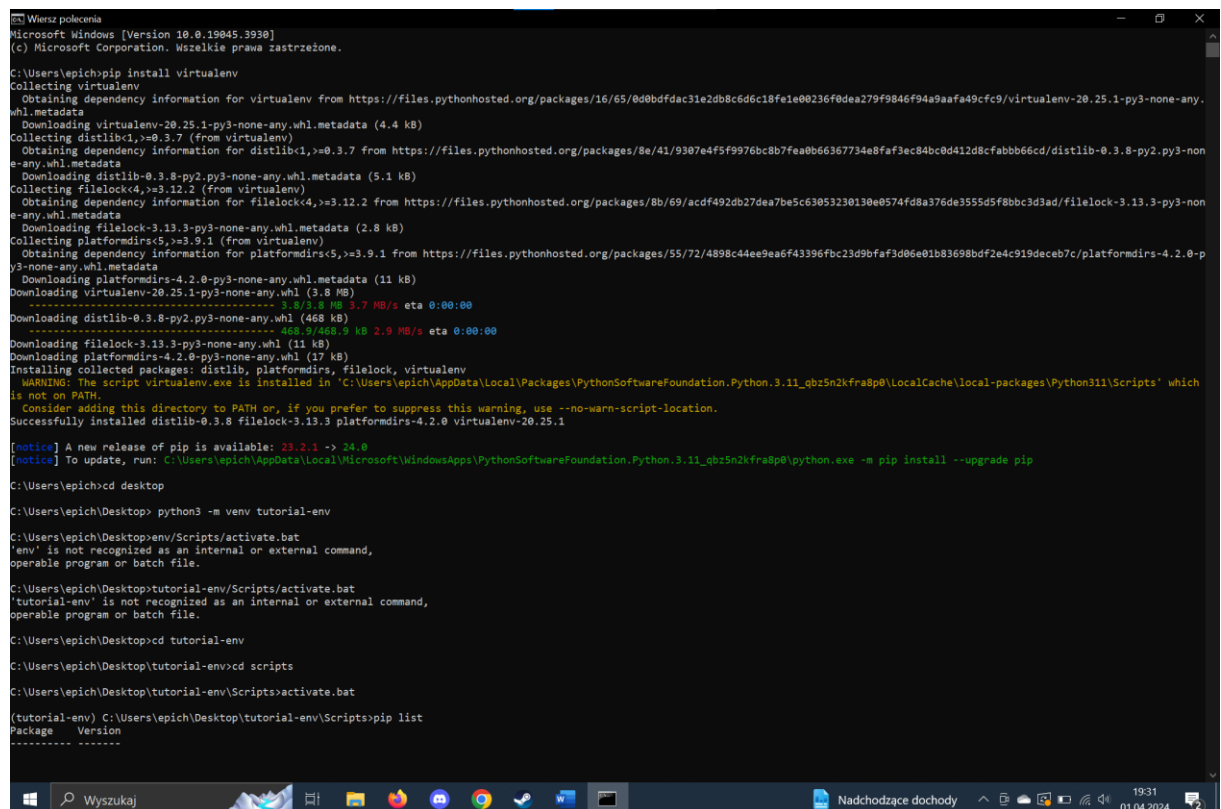
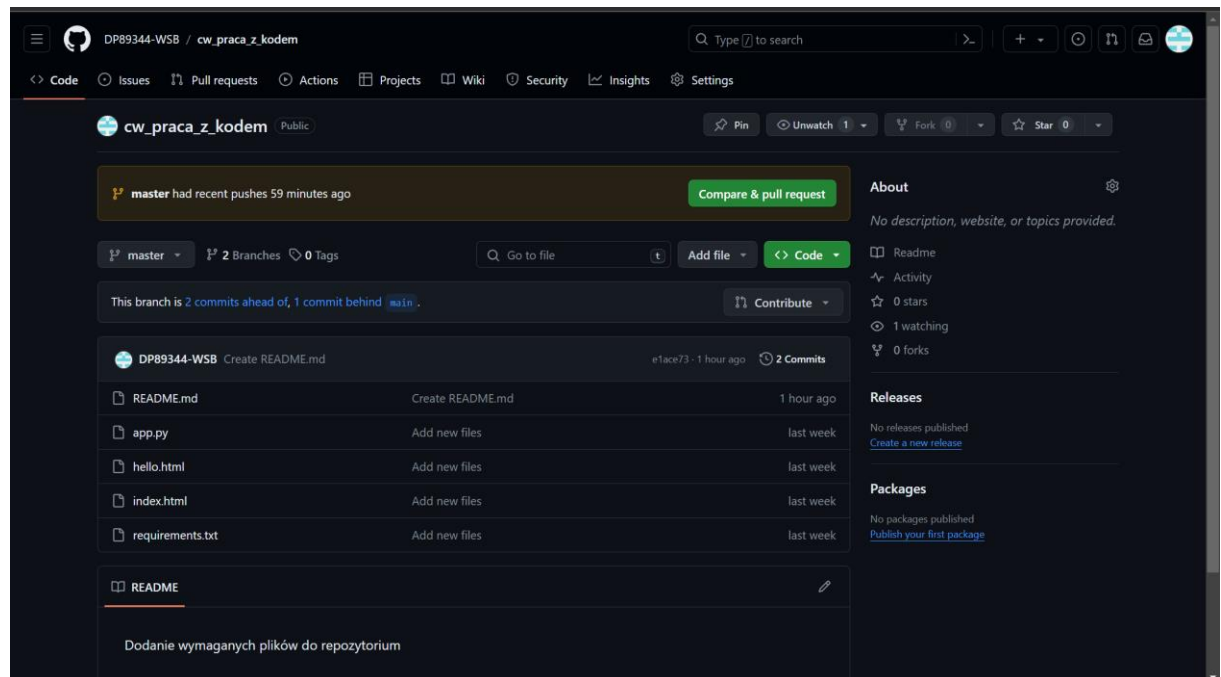
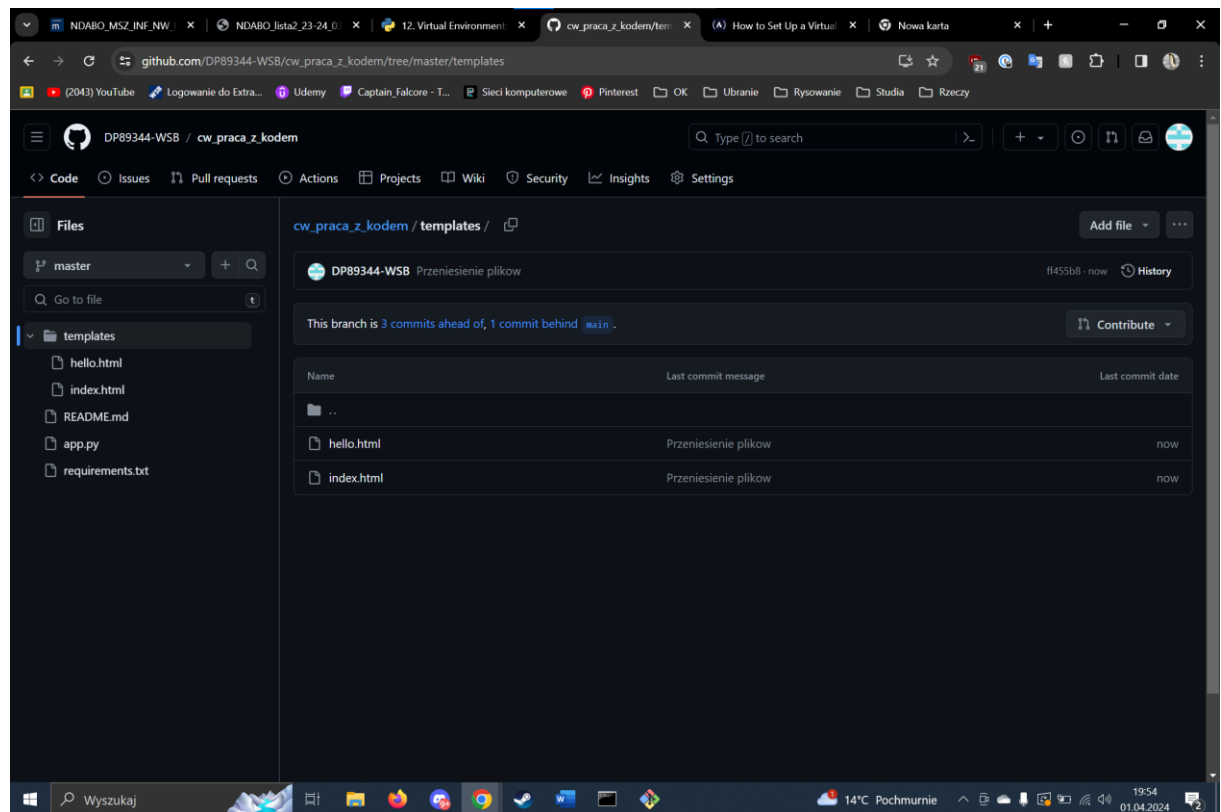


Link do repozytorium – [https://github.com/DP89344-WSB/cw\\_praca\\_z\\_kodem.git](https://github.com/DP89344-WSB/cw_praca_z_kodem.git)

1. Dodanie plików app.py, index.html, hello.html, requirements.txt do repozytorium na githubie oraz utworzenie wirtualnego środowiska Python i aktywowanie go.



## 2. Utwórz folder templates wraz z umieszczeniem tam plików index.html oraz hello.html.



### 3. Utworzenie pliku Makefile w folderze głównym projektu. Dodanie dwóch komend (pip install -r requirements.txt, python -m flask run) i wrzucenie zmian do repozytorium na githubie

```
epich@DamianLP MINGW64 ~/github_DP89344/cw_praca_z_kodem (master)
$ nano Makefile

epich@DamianLP MINGW64 ~/github_DP89344/cw_praca_z_kodem (master)
$ cat Makefile
pip install -r requirements.txt
python -m flask run

epich@DamianLP MINGW64 ~/github_DP89344/cw_praca_z_kodem (master)
$ git add -A
warning: in the working copy of 'Makefile', LF will be replaced by CRLF the next time Git touches it
epich@DamianLP MINGW64 ~/github_DP89344/cw_praca_z_kodem (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   Makefile
        new file:   __pycache__/app.cpython-311.pyc

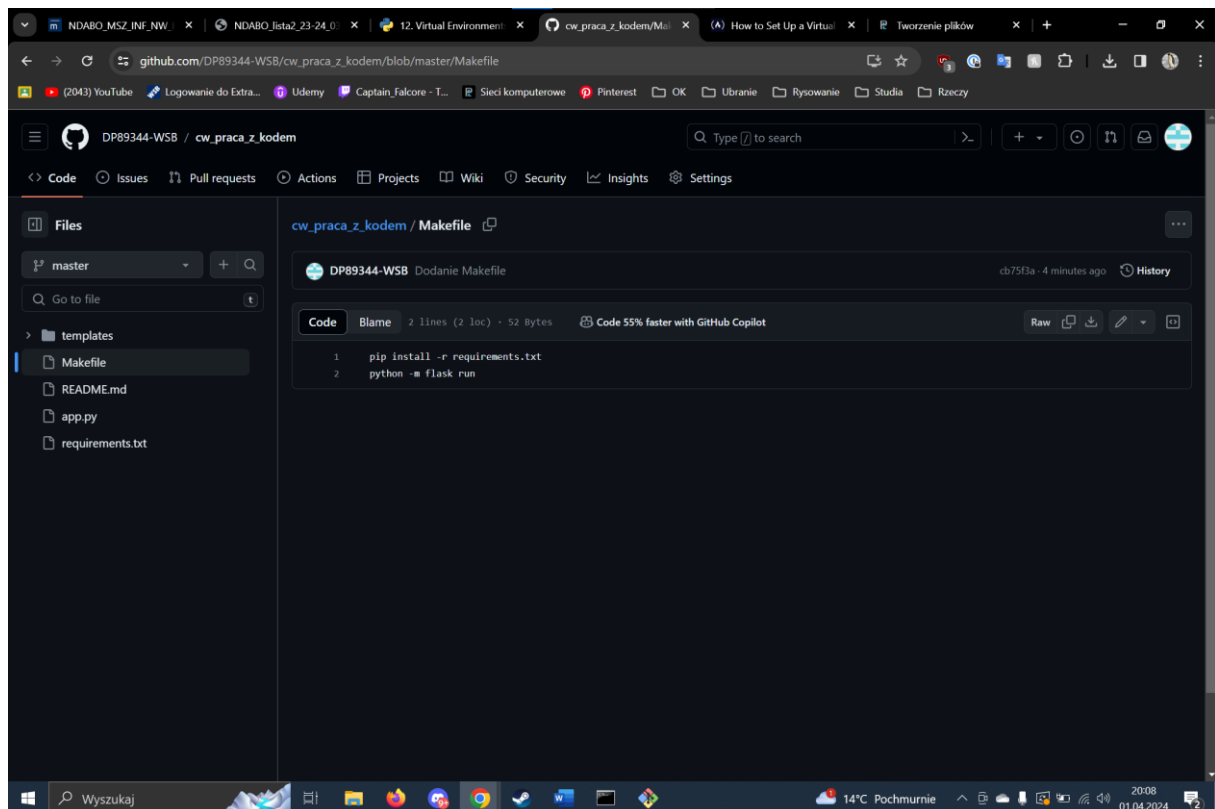
epich@DamianLP MINGW64 ~/github_DP89344/cw_praca_z_kodem (master)
$ git commit -m "Dodanie Makefile"
[master cb75f3a] Dodanie Makefile
 2 files changed, 2 insertions(+)
 create mode 100644 Makefile
 create mode 100644 __pycache__/app.cpython-311.pyc

epich@DamianLP MINGW64 ~/github_DP89344/cw_praca_z_kodem (master)
$ git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 12 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (5/5), 1.11 KiB | 1.11 MiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/DP89344-WSB/cw_praca_z_kodem.git
ff455b8..cb75f3a master -> master

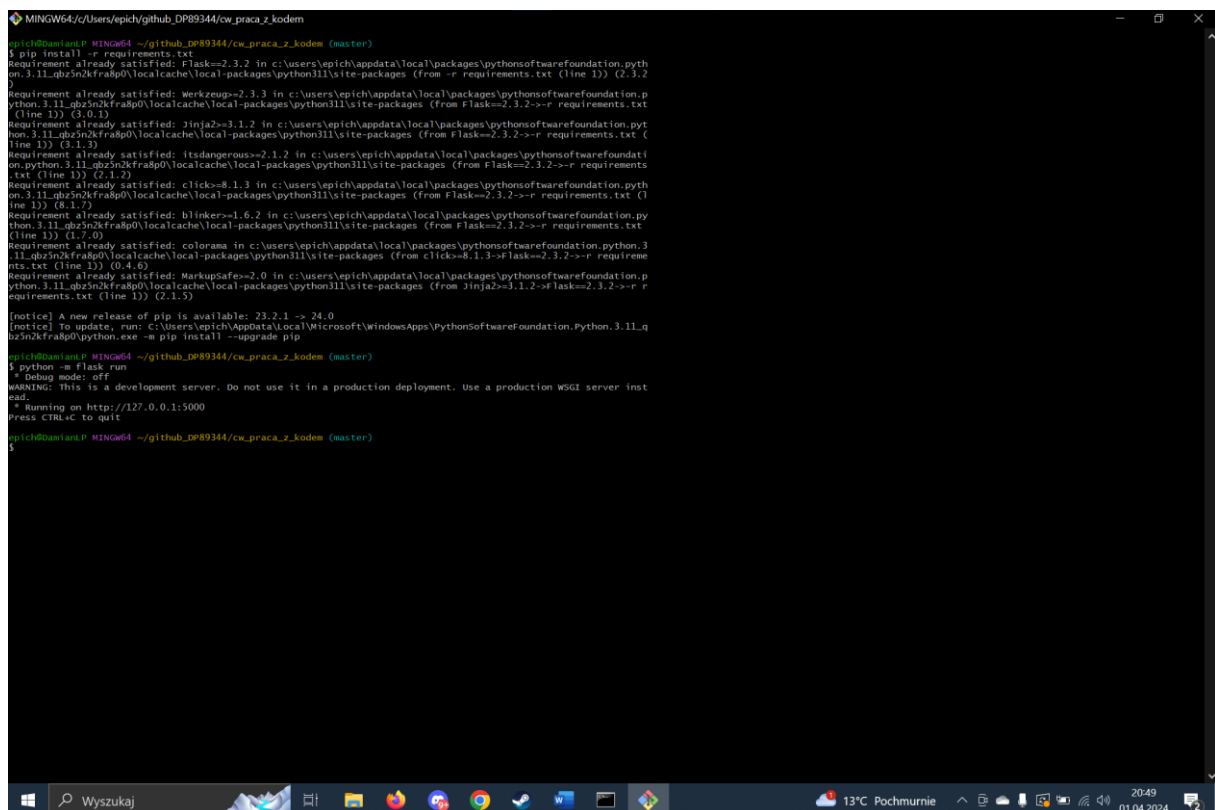
epich@DamianLP MINGW64 ~/github_DP89344/cw_praca_z_kodem (master)
$ ls
Makefile  README.md  __pycache__/  app.py  requirements.txt  templates/
```

```
MINGW64/c/Users/epich/github_DP89344/cw_praca_z_kodem
GNU nano 7.2 makefile Modified
install:
    pip install -r requirements.txt

run:
    python -m flask run
```

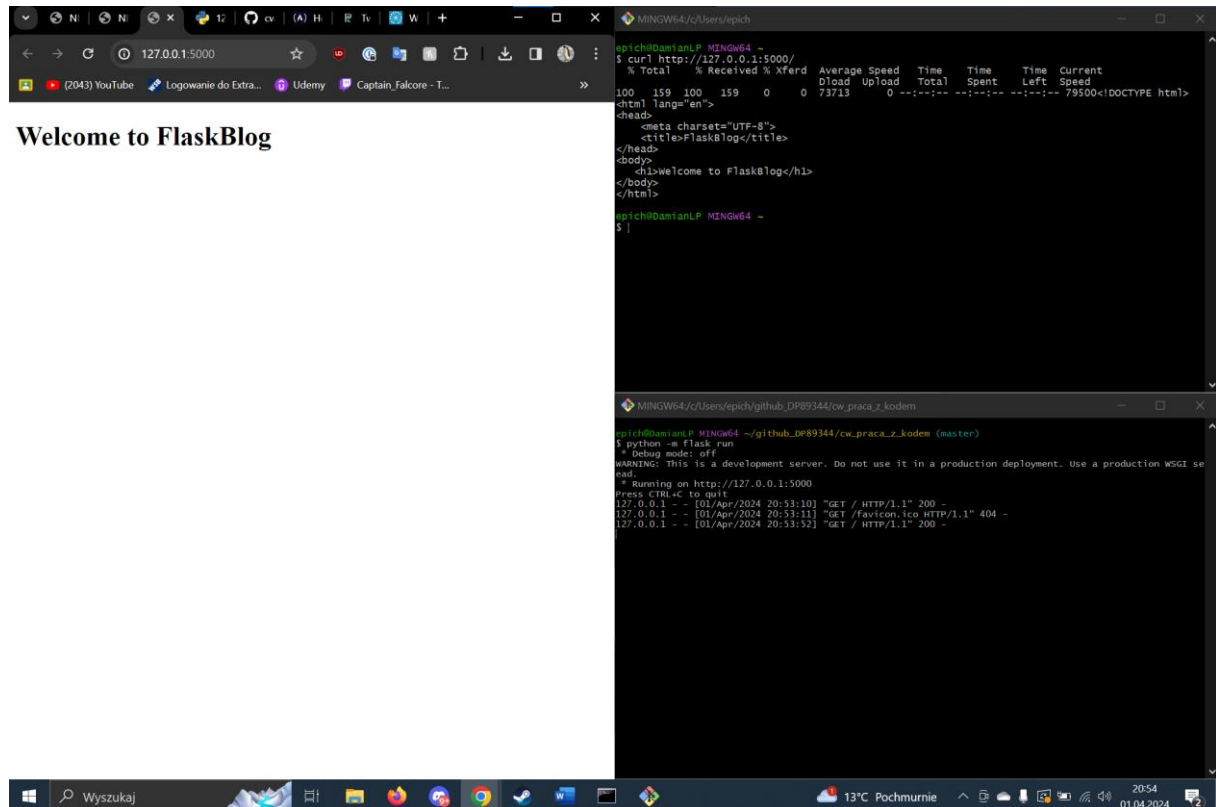


#### 4. Uruchomienie komend z punktu 3 za pomocą Makefile . Sprawdzenie działania komend.

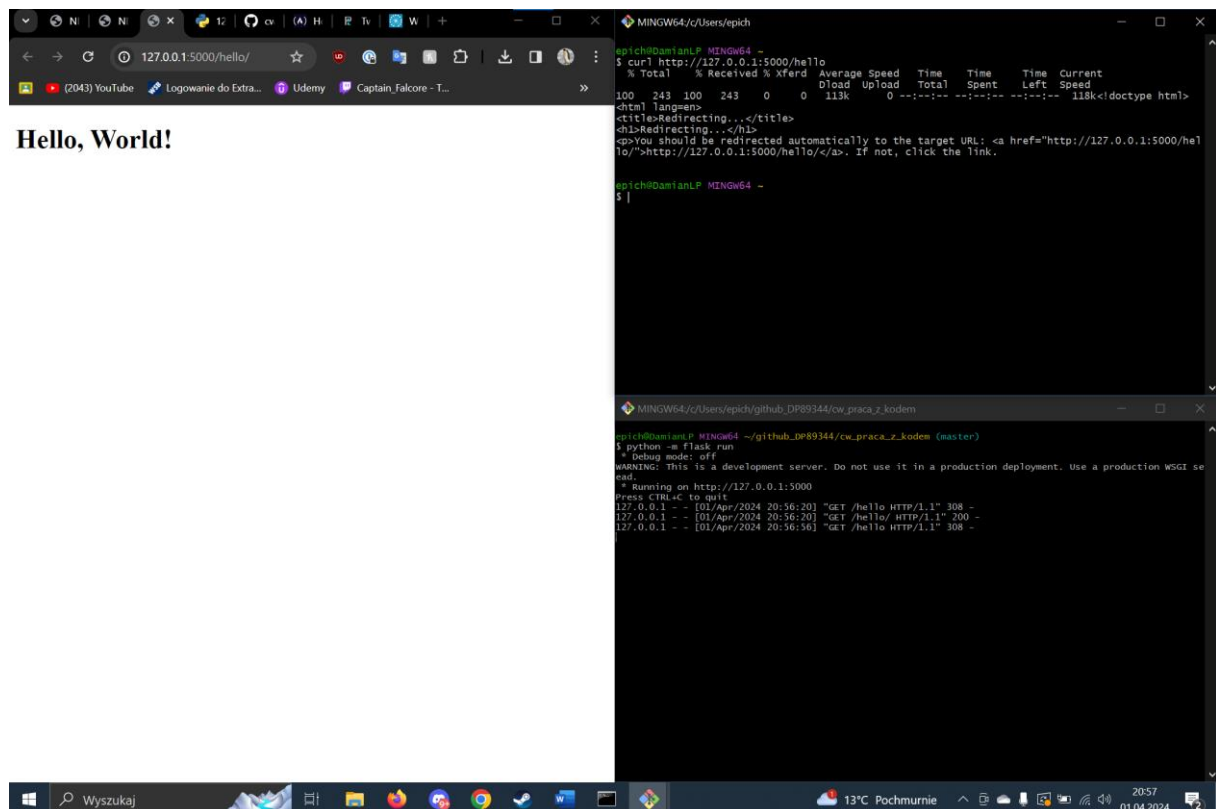


#### 5. Wejście na link ze strony i wykonanie poleceń w terminalu

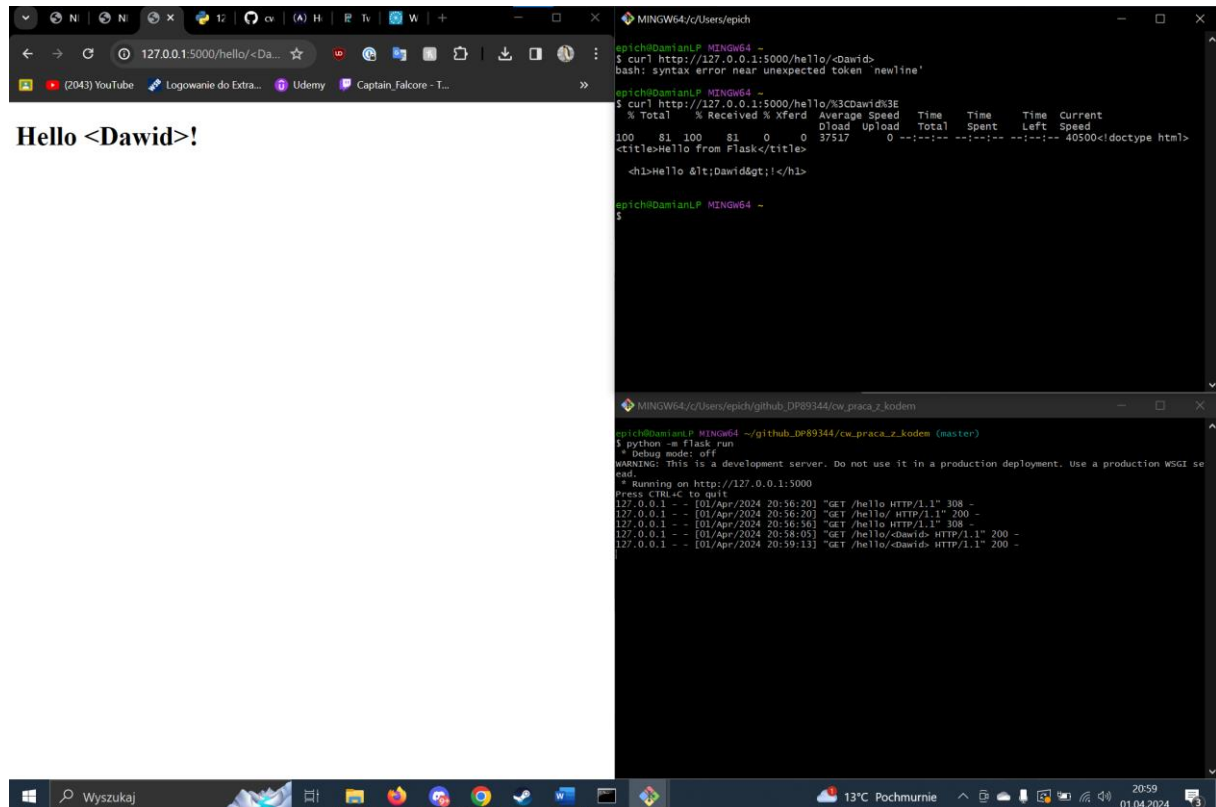
a) Wejście na adres `http://127.0.0.1:5000/` i wykonanie w terminalu polecenie: `curl http://127.0.0.1:5000/`



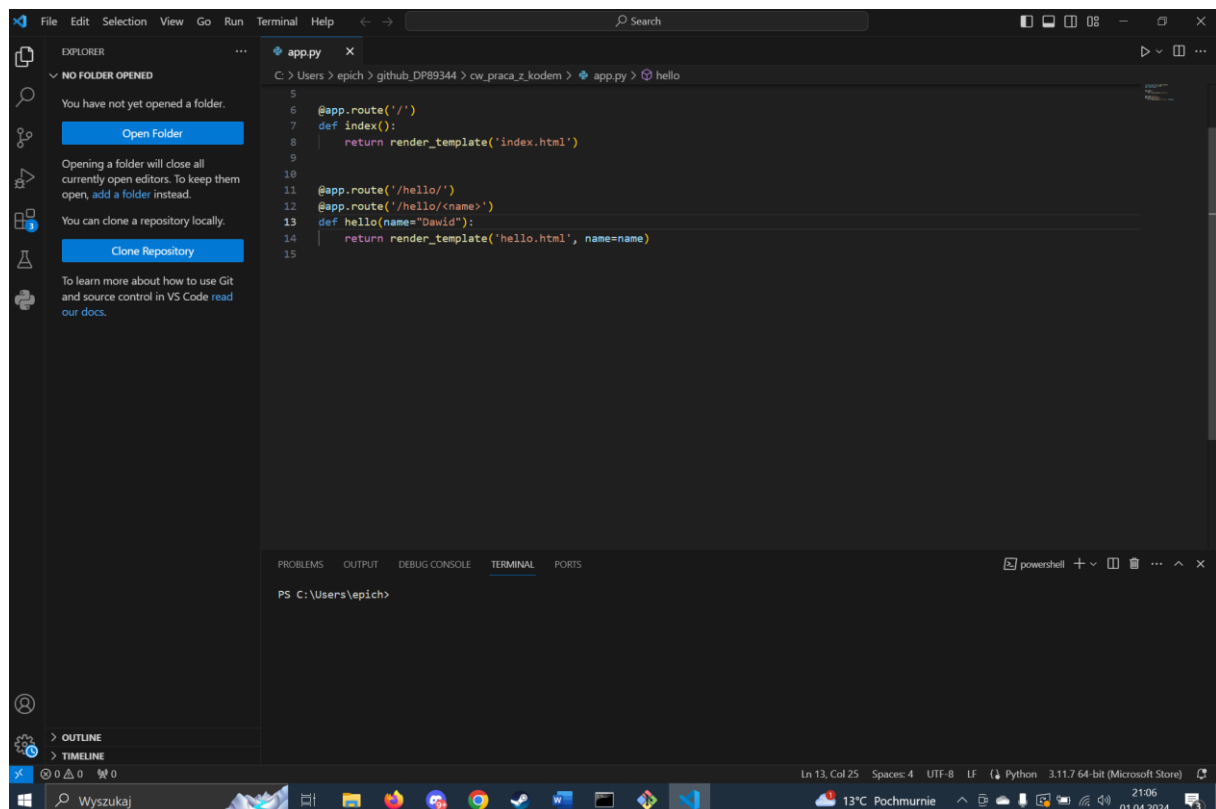
b) Wejście na adres `http://127.0.0.1:5000/hello` i wykonanie w terminalu polecenie: `curl http://127.0.0.1:5000/hello`

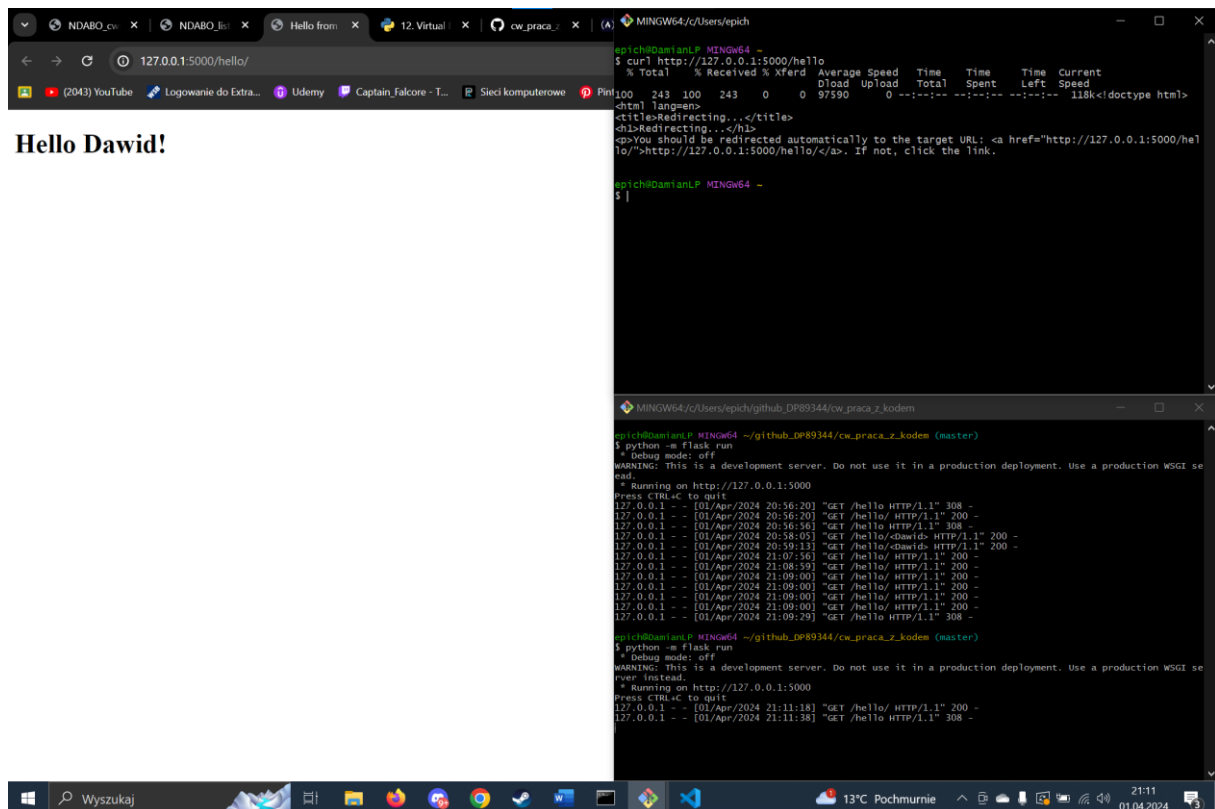


c) Wejście na adres `http://127.0.0.1:5000/hello<Dawid>` i wykonanie w terminalu polecenie: `curl http://127.0.0.1:5000/hello/$3CDawid%3E`

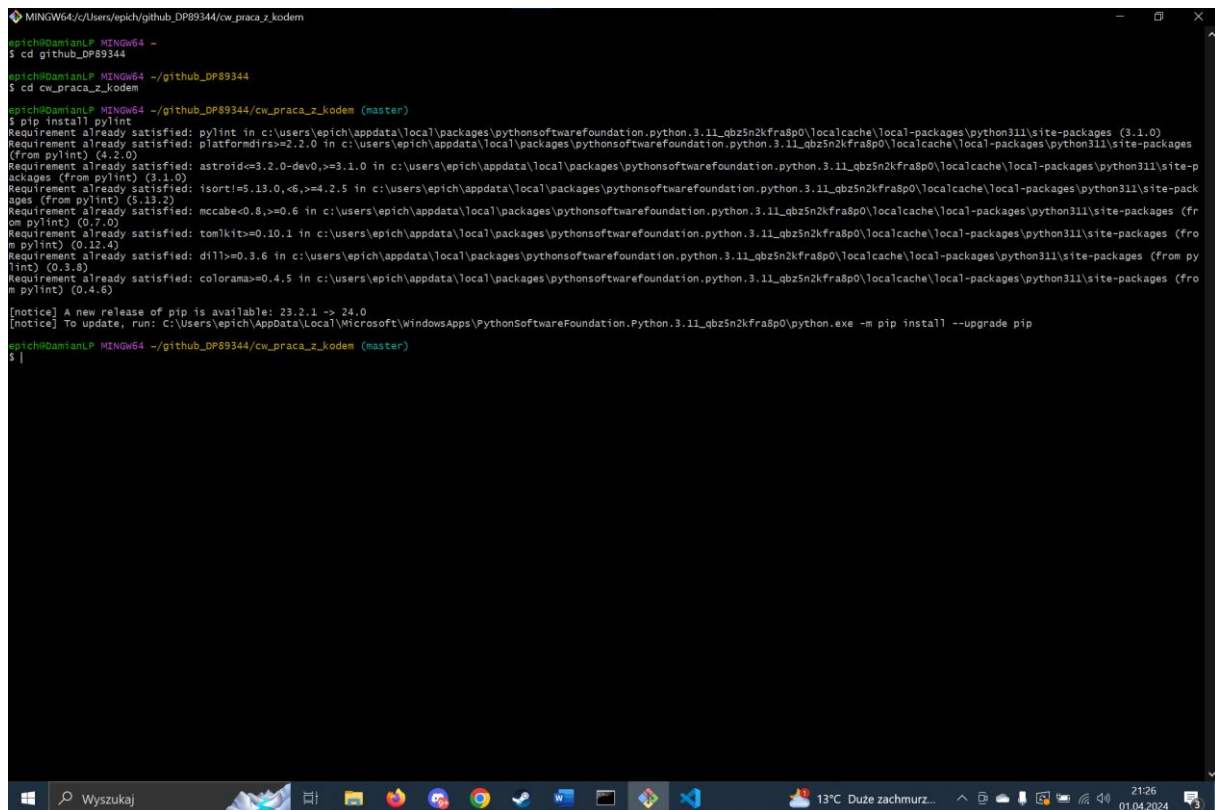


6. W pliku `app.py` został zamieniony parametr „name” na moje imię i po uruchomieniu aplikacji słowo World zamieniło się na moje imię

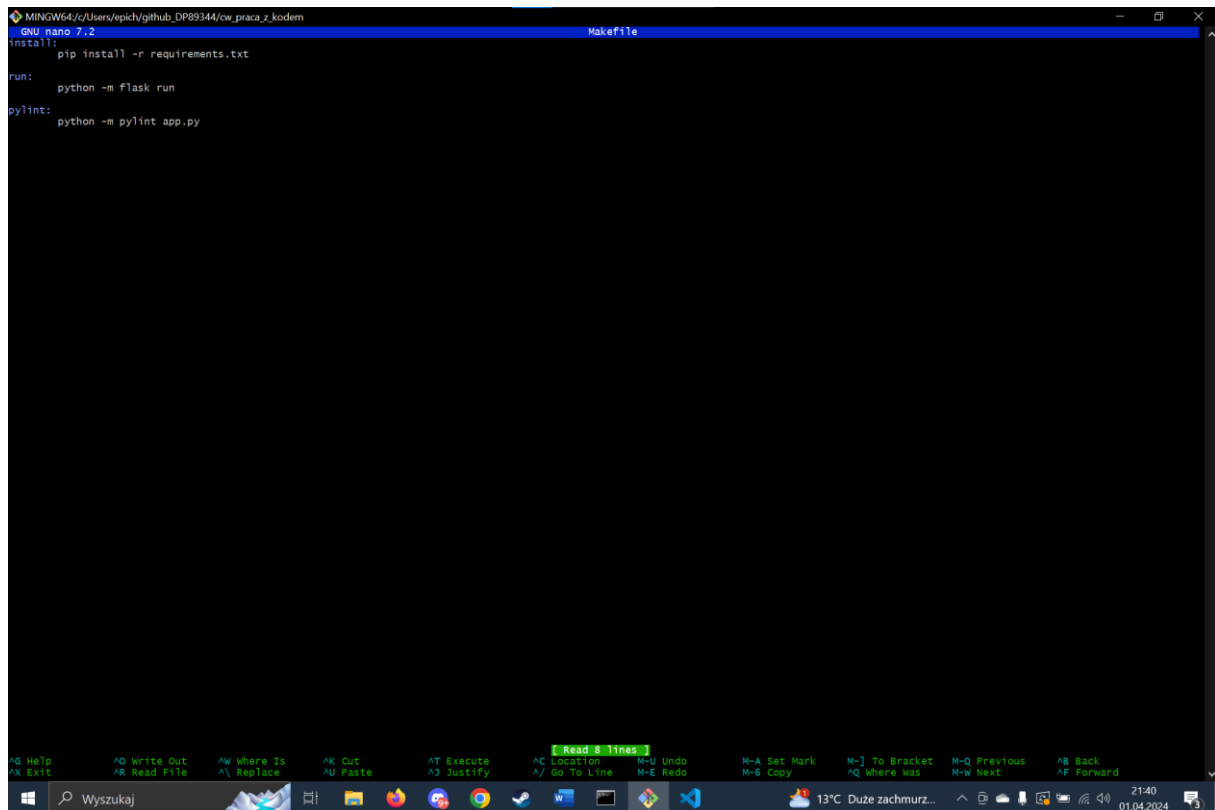




## 7. Instalacja narzędzia pylint



## 8. Dodanie w pliku Makefile Komendy sprawdzającej kod: pylint



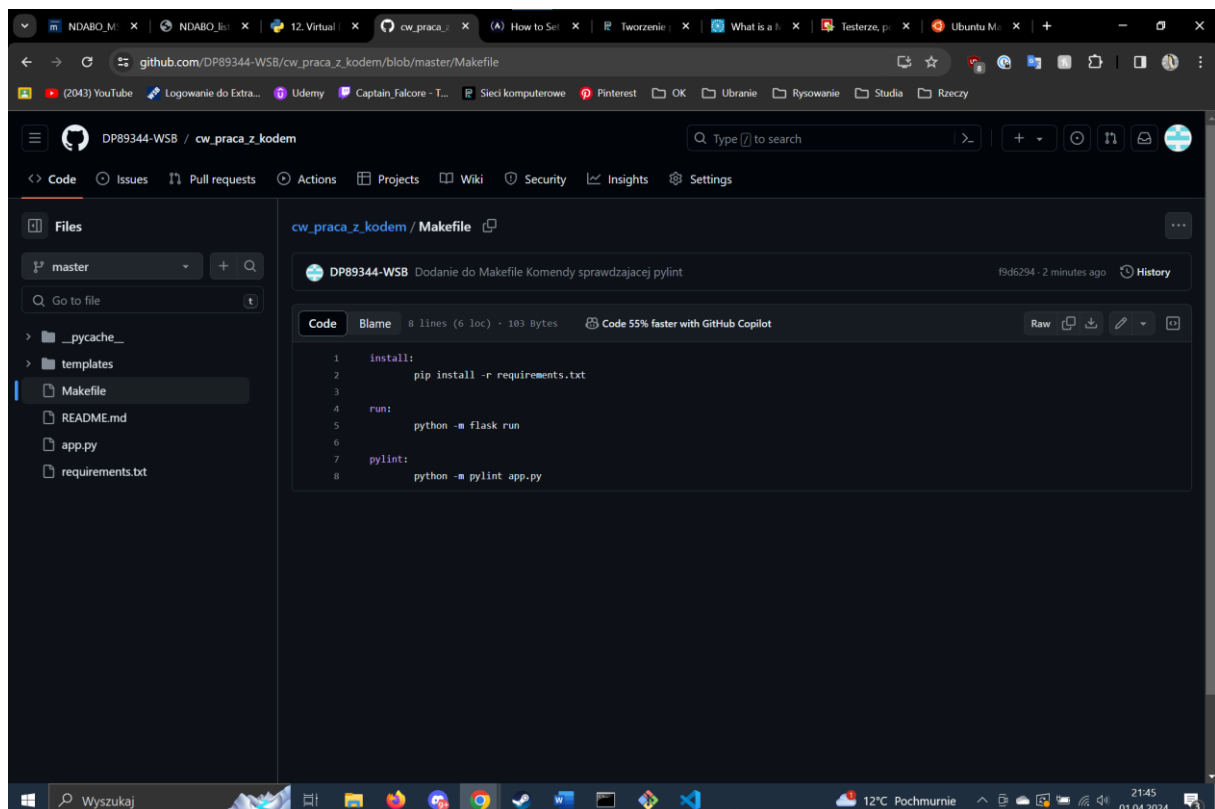
A screenshot of a terminal window titled "MINGW64: c:/Users/epachy/github\_DP89344/cw\_praca\_z\_kodem". The terminal is running GNU nano 7.2 and editing a file named "Makefile". The content of the Makefile is as follows:

```
install:
    pip install -r requirements.txt

run:
    python -m flask run

pylint:
    python -m pylint app.py
```

The terminal window has a status bar at the bottom showing various system icons, the temperature (13°C), and the time (21:40 on 01.04.2024).



A screenshot of a web browser showing the GitHub repository page for "DP89344-WSB / cw\_praca\_z\_kodem". The page displays the "Makefile" file content, which matches the terminal window above:

```
1 install:
2     pip install -r requirements.txt
3
4 run:
5     python -m flask run
6
7 pylint:
8     python -m pylint app.py
```

The page also shows a commit message "Dodanie do Makefile Komendy sprawdzającej pylint" and a commit hash "f9d6294". The left sidebar shows the repository structure with files like "Makefile", "README.md", "app.py", and "requirements.txt". The bottom status bar shows the temperature (12°C) and the time (21:45 on 01.04.2024).



## 9. Wykonanie polecenia za pomocą Makefile z pkt 8

```
MINGW64~/Users/epich/github_DP89344/cw_praca_z_kodem
app.py:13:0: C0116: Missing function or method docstring (missing-function-docstring)
app.py:2:0: W0611: Unused escape imported from markupsafe (unused-import)

-----
Your code has been rated at 4.29/10

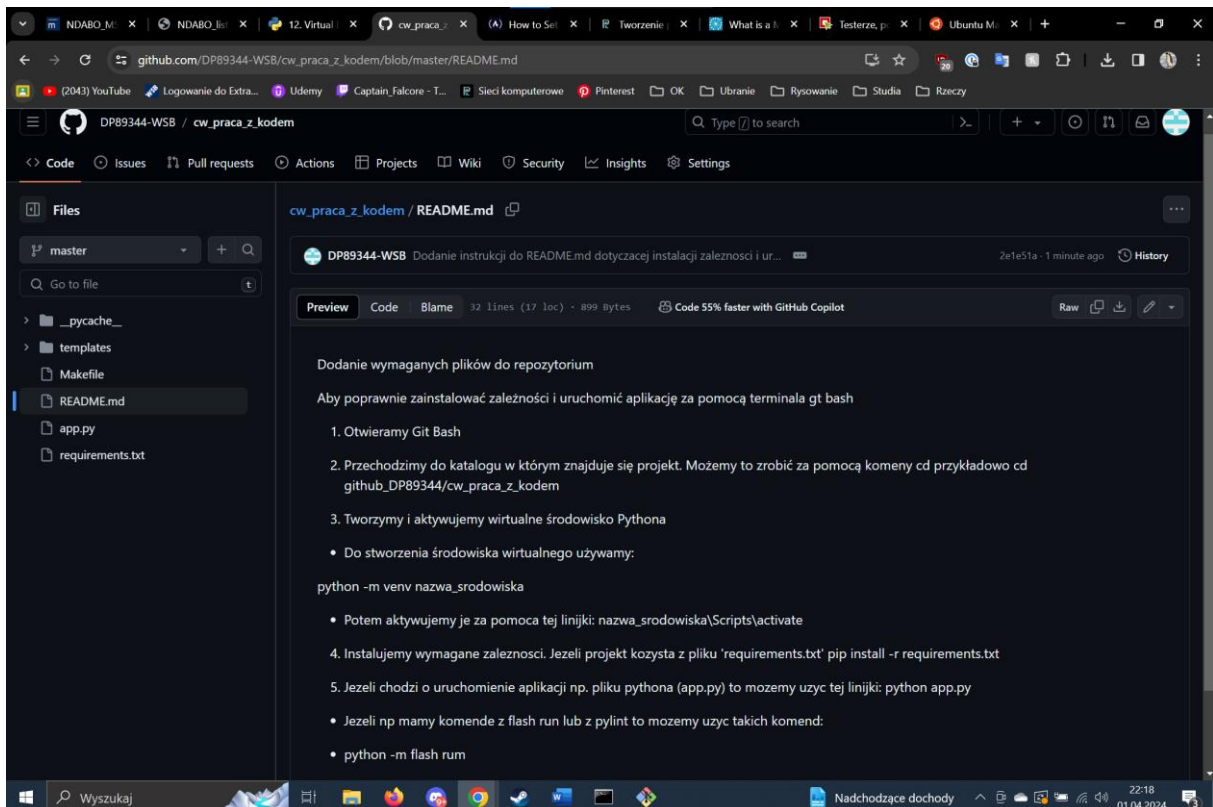
epich@DamianLP MINGW64 ~/github_DP89344/cw_praca_z_kodem (master)
$ nano Makefile
epich@DamianLP MINGW64 ~/github_DP89344/cw_praca_z_kodem (master)
$ nano Makefile
epich@DamianLP MINGW64 ~/github_DP89344/cw_praca_z_kodem (master)
$ python -m make run
C:\Users\epich\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundation.Python.3.11_qbz5n2kfra8p0\python.exe: No module named make
epich@DamianLP MINGW64 ~/github_DP89344/cw_praca_z_kodem (master)
$ git add -A
warning: in the working copy of 'Makefile', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'app.py', LF will be replaced by CRLF the next time Git touches it
epich@DamianLP MINGW64 ~/github_DP89344/cw_praca_z_kodem (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   Makefile
        new file:   __pycache__/app.cpython-311.pyc
        modified:   app.py
epich@DamianLP MINGW64 ~/github_DP89344/cw_praca_z_kodem (master)
$ git commit -m "Dodanie do Makefile komendy sprawdzajacej pylint"
[master f9d6294] Dodanie do Makefile komendy sprawdzajacej pylint
 3 files changed, 5 insertions(+), 2 deletions(-)
 create mode 100644 __pycache__/app.cpython-311.pyc
epich@DamianLP MINGW64 ~/github_DP89344/cw_praca_z_kodem (master)
$ git push
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 12 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 1.21 KiB | 1.21 MiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/DP89344-WSB/cw_praca_z_kodem.git
 7558764..f9d6294 master -> master
epich@DamianLP MINGW64 ~/github_DP89344/cw_praca_z_kodem (master)
$ python -m pylint app.py
***** Module app
app.py:10: C0114: Missing module docstring (missing-module-docstring)
app.py:7:0: C0116: Missing function or method docstring (missing-function-docstring)
app.py:13:0: C0116: Missing function or method docstring (missing-function-docstring)
app.py:2:0: W0611: Unused escape imported from markupsafe (unused-import)

-----
Your code has been rated at 4.29/10 (previous run: 4.29/10, +0.00)

epich@DamianLP MINGW64 ~/github_DP89344/cw_praca_z_kodem (master)
$ |
```

## 10. Dodanie instrukcji w README.md dotyczącej instalowania zależności oraz uruchomienia aplikacji



The screenshot shows the GitHub repository page for `DP89344-WSB/cw_praca_z_kodem`. The `README.md` file is open, displaying the following content:

```
Dodanie wymaganych plików do repozytorium

Aby poprawnie zainstalować zależności i uruchomić aplikację za pomocą terminala gt bash

1. Otwieramy Git Bash

2. Przechodzimy do katalogu w którym znajduje się projekt. Możemy to zrobić za pomocą komendy cd przykładowo cd github_DP89344/cw_praca_z_kodem

3. Tworzymy i aktywujemy wirtualne środowisko Pythona

• Do stworzenia środowiska wirtualnego używamy:

python -m venv nazwa_srodowiska

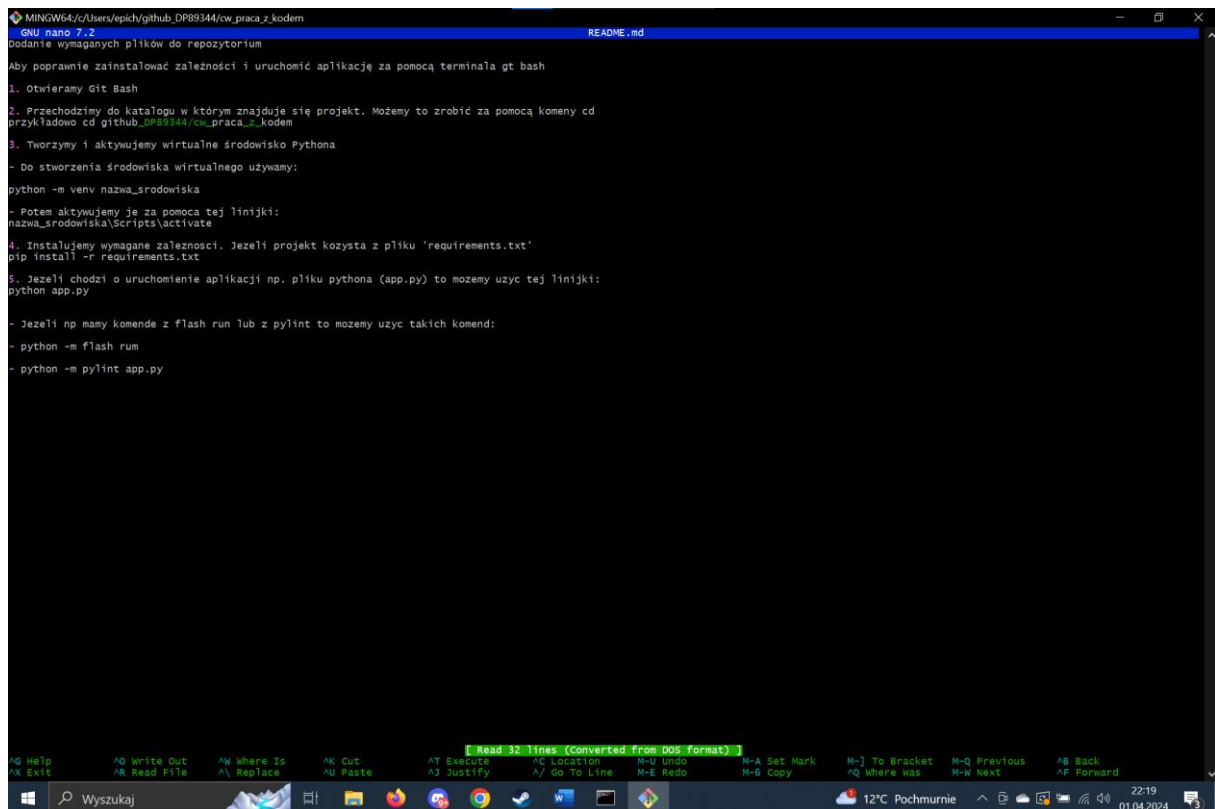
• Potem aktywujemy je za pomocą tej linii: nazwa_srodowiska/Scripts/activate

4. Instalujemy wymagane zależności. Jeżeli projekt korzysta z pliku 'requirements.txt' pip install -r requirements.txt

5. Jeżeli chodzi o uruchomienie aplikacji np. pliku pythona (app.py) to możemy użyć tej linii: python app.py

• Jeżeli np. mamy komendę z flash run lub z pylint to możemy użyć takich komend:

• python -m flash run
```



```
MINGW64/c/Users/epich/github_DP89344/cw_praca_z_kodem
GNU nano 7.2 README.md
podanie wymaganych plików do repozytorium
Aby poprawnie zainstalować zależności i uruchomić aplikację za pomocą terminala gt bash
1. Otwieramy Git Bash
2. Przechodzimy do katalogu w którym znajduje się projekt. Możemy to zrobić za pomocą komendy cd
przykładowo cd github_DP89344/cw_praca_z_kodem
3. Tworzymy i aktywujemy wirtualne środowisko Pythona
- Do stworzenia środowiska wirtualnego używamy:
python -m venv nazwa_srodowiska
- Potem aktywujemy je za pomocą tej linijki:
nazwa_srodowiska\Scripts\activate
4. Instalujemy wymagane zależności. Jeżeli projekt korzysta z pliku 'requirements.txt'
pip install -r requirements.txt
5. Jeżeli chodzi o uruchomienie aplikacji np. pliku pythona (app.py) to możemy użyć tej linijki:
python app.py
- Jeżeli np. mamy komendę z flash run lub z pylint to możemy użyć takich komend:
- python -m flash run
- python -m pylint app.py
Read 32 lines (converted from DOS format)
^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location ^H Undo ^A Set Mark ^N To Bracket ^Q Previous ^B Back
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/_ Go To Line ^E Redo ^E Copy ^Q Where Was ^W Next ^F Forward
```

**\*10. W projekcie na GitHubie ważne jest README, ponieważ to pierwsze źródło informacji dla osób odwiedzających repozytorium. Zapewnia krótkie wprowadzenie do projektu, jego celu, funkcjonalności i sposób użycia. Przykładowo:**

- **Ułatwia rozpoczęcie pracy:** Nowi użytkownicy mogą szybko zrozumieć i zacząć korzystać z projektu i jakie są jego główne funkcje
- **Dokumentuje sposób instalacji i uruchomienia:** README zawiera instrukcje dotyczące instalacji zależności, konfiguracji środowiska oraz uruchomienia aplikacji
- **Promuje współpracę:** Może zawierać informacje na temat sposobu wsparcia projektu, takie jak wytyczne dotyczące zgłaszania problemów, proponowania poprawek kodu czy tworzenia zgłoszeń pull request.
- **Podkreśla przejrzystość i transparentność:** Może zawierać informacje na temat licencji, wymagań systemowych i historii zmian, co pomaga użytkownikom zrozumieć zasady korzystania z projektu i jego rozwój.
- **Pozycjonuje projekt:** Dzięki README potencjalni użytkownicy mogą szybko ocenić, czy projekt spełnia ich potrzeby i czy warto z nim dalej pracować.