Tworzenie pliku EXE z pliku .py w systemie Windows

1. Wprowadzenie

W systemie Windows możliwe jest przekształcenie pliku Python (.py) w samodzielny plik wykonywalny (.exe). Najczęściej wykorzystuje się do tego bibliotekę `pyinstaller`, która pakuje cały interpreter Pythona wraz ze skryptem i jego zależnościami.

2. Instalacja PyInstaller

```
pip install pyinstaller
```

3. Podstawowe użycie

Aby stworzyć plik .exe z pliku `main.py`, użyj polecenia:

```
pyinstaller --onefile main.py


Po zakończeniu w folderze `dist/` znajdziesz plik `main.exe`.
```

4. Popularne opcje PyInstaller

```
--onefile         # tworzy pojedynczy plik EXE (w przeciwnym razie
                 tworzy folder z wieloma plikami)
--noconsole       # ukrywa konsolę – użyteczne dla aplikacji GUI
--icon=app.ico    # dodaje ikonę do pliku EXE
--name=nazwa      # nadaje nazwę wyjściowemu plikowi
```

5. Przykład użycia

```
pyinstaller --onefile --noconsole --icon=logo.ico --name=MyApp app.py
```

6. Plusy i minusy podejścia EXE

Zalety:

```
+ Użytkownik końcowy nie potrzebuje instalować Pythona
+ Cała aplikacja w jednym pliku (z opcją --onefile)
+ Możliwość uruchamiania jako klasyczny program Windows
```

Wady:

```
- Plik EXE może być duży (20-80 MB)
- Czas uruchomienia może być wolniejszy (rozpakowanie do katalogu
tymczasowego)
- Programy antywirusowe mogą błędnie oznaczać EXE jako podejrzane
- Trudniejszy debugging
```

7. Czy to popularne w praktyce?

Tak – szczególnie w tworzeniu małych narzędzi CLI lub aplikacji desktopowych (np. z użyciem PyQt, Tkinter). Popularne w korporacjach i wewnętrznych rozwiązaniach, gdzie użytkownicy nie mają dostępu do Pythona.

8. Alternatywy

- cx_Freeze – podobne do pyinstaller, bardziej konfigurowalny
- nuitka – kompiluje do C/C++, wydajniejsze EXE
- py2exe – starsze, rzadziej używane obecnie
- pex/zipapp – dla środowisk linuksowych

]