

# Laboratorium 4 – Zmienne środowiskowe, argumenty linii komend, przetwarzanie plików

## Cele dydaktyczne

1. Zapoznanie ze zmiennymi środowiskowymi i czytaniem parametrów z linii komend w języku Python.
2. Zapoznanie z uruchomieniem procesów i komunikacji z nimi.
3. Zapoznanie z przetwarzaniem danych w formatach CSV oraz JSON.
4. Zapoznanie z operacjami na systemie plików.

## Wprowadzenie

### Zmienne środowiskowe

Istnieją różne sposoby na sterowanie wykonaniem programów komputerowych. Zmienne środowiskowe są zmiennymi, których wartości są ustawiane poza programem, najczęściej przez funkcjonalności wbudowane w system operacyjny albo oprogramowanie zarządzające wykonywaniem usług. Zmienne środowiskowe składają się z par nazwa-wartość. Mogą przechowywać np. konfigurację aplikacji, co jest dobrą praktyką w tworzeniu aplikacji uruchamianych jako usługi•

Innym sposobem sterowania przebiegiem wykonania programu jest wykorzystanie argumentów linii komend. W języku Python dostępne są one na liście sys.argv. Pierwszy argument odpowiada nazwie skryptu, a kolejne reprezentują przekazane parametry. W kolejnych laboratoriach wykorzystane zostaną narzędzia wspierające tworzenie zaawansowanych CLI (ang. command-line interface).

## Zadania

1. Napisz skrypt, który wyświetli na wyjście standardowe listę wszystkich zmiennych środowiskowych.
  - a. Niech skrypt umożliwi uruchomienie go z dowolną liczbą parametrów linii komend. W takim przypadku, należy przefiltrować zmienne do wyświetlenia na wyjściu standardowym. Warunkiem wyświetlania zmiennej i jej wartości jest istnienie parametru, którego wartość zawiera się w nazwie zmiennej.
  - b. Zmienne powinny być wyświetlone w porządku alfabetycznym.
2. Napisz skrypt, który operuje na zmiennej środowiskowej PATH. Zmienna ta wykorzystywana jest w różnych systemach operacyjnych, m.in. Windows, Linux, Mac OS X. Zmienna ta zawiera katalogi, w których znajdują się pliki wykonywalne, które mogą być uruchamiane bez wpisywania pełnej ścieżki do pliku. Skrypt powinien umożliwić, z wykorzystaniem samodzielnie ustalonych parametrów linii komend, na realizację poniższych funkcjonalności :
  - a. Wypisanie na wyjście standardowe wszystkich katalogów znajdujących się w zmiennej środowiskowej PATH, każdy w osobnej linii.
  - b. Wypisanie na wyjście standardowe każdego katalogu znajdującego się w zmiennej środowiskowej PATH wraz z listą wszystkich plików wykonywalnych znajdujących się w tym katalogu.
3. Napisz własną, uproszczoną wersję uniksowego programu `tail`, który będzie wypisywał na wyjście standardowe ostatnie linie zadanego pliku lub danych przekazanych mu na wejście standardowe. Program powinien:
  - a. móc być wywołany z argumentem `--lines=n`, gdzie `n` jest liczbą naturalną określającą liczbę linii do wypisania.
    - i. w przypadku wywołania programu bez tego parametru, program powinien wypisać 10 ostatnich linii.
    - ii. w przypadku, gdy plik ma mniej linii, należy wypisać całą zawartość pliku.
  - b. móc być wywołany:
    - i. przekazując mu danych na wejście standardowe, np.

`cat plik.txt | python tail.py`
    - ii. z argumentem określającym ścieżkę pliku, który ma być wypisany np.

`python tail.py plik.txt`
    - iii. w przypadku wywołania łączącego te dwa sposoby, np.

```
cat plik.py | python tail.py plik.txt
```

program powinien zignorować dane z wejścia standardowego i wyświetlić dane z pliku.

- c. **Wersja rozszerzona (aby dostać 10 pkt):** program może dodatkowo przyjąć parametr `--follow`, którego dodanie sprawia, że po wypisaniu zawartości pliku nie kończy działania, lecz czeka na dodanie wierszy do pliku przez inne procesy, a następnie je wyświetla.
4. Napisz program w ulubionym języku programowania (dowolnym np. C, C++, Rust, Go, Java, Python, PHP, ...), który:
  - a. czyta z wejścia standardowego ścieżkę do pliku tekstowego
  - b. analizuje plik tekstowy pod kątem statystycznym, a następnie dla oblicza następujące informacje:
    - i. ścieżka do pliku,
    - ii. całkowita liczba znaków,
    - iii. całkowita liczba słów,
    - iv. liczba wierszy,
    - v. znak występujący najczęściej,
    - vi. słowo występujące najczęściej.
  - c. wynik obliczeń wypisywany jest na wyjście standardowe powinien w formacie `*.tsv`, `*.csv`, lub `*.json`
  - d. Następnie, napisz skrypt w języku Python, który:
    - i. przyjmuje jako argument linii komend ścieżkę do katalogu w systemie plików,
    - ii. z wykorzystaniem modułu `subprocess` uruchamia napisany powyżej program do obliczeń, przesyłając na wejście standardowe ścieżki do kolejnych plików,
    - iii. przetwarza dane wyjściowe kolejnych wywołań programu, zapisując wynik jako listę słowników,
    - iv. wypisuje na wyjście standardowe w dowolnym formacie:
      - liczbę przeczytanych plików, sumaryczną liczbę znaków, sumaryczną liczbę słów, sumaryczną liczbę wierszy, znak występujący najczęściej, słowo występujące najczęściej.
5. Z wykorzystaniem kanonicznych programów CLI i poleceń systemu operacyjnego (np. `ffmpeg`<sup>1</sup>, `mv`, `cp`, `os`, `subprocess`, `os.environ` itd.) oraz modułu `subprocess`, skonstruuuj skrypt `vidconvert.py`. Pomocnicze funkcjonalności (np. znajdowanie

---

<sup>1</sup> <https://www.ffmpeg.org/download.html>

plików, logowanie operacji, odczyt zmiennych środowiskowych) umieścić w osobnym module `utils.py`.

- a. Skrypt: `mediaconvert.py` – do konwersji plików multimedialnych powinien:
  - i. Przyjmować jako argument ścieżkę do katalogu zawierającego plik audio lub wideo.
  - ii. Dla każdego pliku
    - Wykonać konwersję do wybranego formatu przekazanego jako parametr korzystając z programu `ffmpeg` (wywołanie przez `subprocess`).
    - Nazwa wyjściowa powinna zawierać timestamp oraz oryginalną nazwę, np. `20250324-video123.webm`.
  - iii. Pliki wynikowe powinny być zapisywane do katalogu `CONVERTED_DIR`, którego lokalizację można ustawić poprzez zmienną środowiskową. Jeśli zmienna nie jest ustawiona, domyślnie należy zapisać do `converted/` w bieżącym katalogu roboczym.
  - iv. W katalogu docelowym zapisywana jest historia konwersji w pliku `history.json` lub `history.csv`, z informacjami:
    - Data i godzina konwersji
    - Ścieżka oryginalnego pliku
    - Format wyjściowy
    - Ścieżka pliku wynikowego.
- b. **Wersja rozszerzona (na max pkt.)** – Rozszerz skrypt `mediaconvert.py` o funkcjonalność konwersji obrazów z wykorzystaniem `ImageMagick`<sup>2</sup>.
  - i. Skrypt powinien automatycznie wykryć, czy przekazano obraz, czy plik audio/video i użyć odpowiednio `ffmpeg` lub `magick`.
  - ii. Plik z historią konwersji powinien mieć dodatkową kolumnę zawierającą ciąg znaków z użytym programem.

---

<sup>2</sup> <https://imagemagick.org/script/download.php>