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.

Program można zgłosić jako zrobiony, jeśli spełnione są poniższe warunki:

1. Program jest zgodny z podaną specyfikacją.
2. Został przetestowany.
3. Student go rozumie i potrafi wyjaśnić.

UWAGA:

Wykonywanie zadań przy użyciu czatu GPT będzie traktowane jako praca niesamodzielna i będzie skutkować oceną niedostateczną.

- Zadbaj o to, aby każda funkcja w programie miała tylko jedną odpowiedzialność.
- Zadbaj o rozdzielenie funkcji przetwarzających dane od funkcji najwyższego poziomu wypisujących tekst na wyjście standardowe.
- W przypadku, gdy kilka funkcjonalności wymaga skorzystania z samych funkcji, umieść je w osobnym module, który będzie ponownie użyty.
- Przed potencjalnie źle sformatowanymi danymi zabezpiecz się wykorzystując mechanizm wyjątków.

- **z wykorzystaniem *mechanizmu obsługi wyjątków* zadbaj o poprawność działania funkcji**
- **do zadań przygotować 3 – 5 testów sprawdzających poprawność działania**
- **do prowadzącego wysłać pliki z kodem źródłowym oraz rzuty ekranu z przygotowanymi testami**

Wprowadzenie

Zmienne środowiskowe

Istnieją różne sposoby na sterowaniem 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

Zadanie_1: Napisz skrypt, który umożliwia uruchomienie go z dowolną liczbą parametrów linii komend:

1.1. wyświetla na wyjście standardowe listę wszystkich zmiennych środowiskowych.
1.2. Wyświetla na wyjście standardowe podaną listę zmiennych środowiskowych. W takim przypadku, należy przefiltrować zmienne do wyświetlenia na wyjściu standardowym. Warunkiem wyświetlenia zmiennej i jej wartości jest istnienie parametru, którego wartość zawiera się w nazwie zmiennej.
1.3. Zmienne powinny być wyświetlone w porządku alfabetycznym.

Punkty: 1

Zadanie_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.

Punkty: 1

Zadanie_3:

Napisz własną, uproszczoną wersję uniksowego programu head, który będzie wypisywał na wyjście standardowe początkowe linie zadanego pliku lub danych przekazanych mu na wejście standardowe. Program powinien:

3.1. móc być wywołany z argumentem `--lines=n`, gdzie n jest liczbą naturalną określającą liczbę linii do wypisania.
- w przypadku wywołania programu bez tego parametru, program powinien wypisać 10 pierwszych linii.
- w przypadku, gdy plik ma mniej linii, należy wypisać całą zawartość pliku.

3.2. móc być wywołany:
- przekazując mu danych na wejście standardowe, np.

```
cat plik.txt | python tail.py (w SO Linux)

type plik.txt | python tail.py (w SO Windows)
```

- z argumentem określającym ścieżkę pliku, który ma być wypisany np.

```
python tail.py plik.txt
```

- w przypadku wywołania łączącego te dwa sposoby, np.

```
cat plik.py | python tail.py plik.txt (w SO Linux)

type plik.py | python tail.py plik.txt (w SO Windows)
```

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

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.

Punkty: 2

Zadanie_4:

4.1. 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:
   - ✓ ścieżka do pliku,
   - ✓ całkowita liczba znaków,

- ✓ całkowita liczba słów,
- ✓ liczba wierszy,
- ✓ znak występujący najczęściej,
- ✓ słowo występujące najczęściej.

c) wynik obliczeń wypisywany jest na wyjście standardowe powinien w formacie *.csv

4.2. Następnie, napisz skrypt w języku Python, który:

a) przyjmuje jako argument linii komend ścieżkę do katalogu w systemie plików,

b) z wykorzystaniem modułu subprocess uruchamia napisany powyżej program do obliczeń, przesyłając na wejście standardowe ścieżki do kolejnych plików,

c) przetwarza dane wyjściowe kolejnych wywołań programu, zapisując wynik jako listę słowników,

d) 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.

Punkty:2

Zadanie_5:

Z wykorzystaniem kanonicznych programów CLI i poleceń systemu operacyjnego (np. `ffmpeg`[1], `mv`, `cp`, `os`, `subprocess`, `os.environ` itd.) oraz modułu `subprocess`, skonstruuj skrypt `vidconvert.py`. Pomocnicze funkcjonalności (np. znajdowanie plików, logowanie operacji, odczyt zmiennych środowiskowych) umieść 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

[1] https://www.ffmpeg.org/download.html

- ✓ Ścieżka oryginalnego pliku
- ✓ Format wyjściowy
- ✓ Ścieżka pliku wynikowego.

Wersja rozszerzona (na max pkt z zadania. (1p)) – Rozszerz skrypt `mediaconvert.py` o funkcjonalność konwersji obrazów z wykorzystaniem ImageMagick[2].

a) Skrypt powinien automatycznie wykryć, czy przekazano obraz, czy plik audio/video i użyć odpowiednio ffmpeg lub magick.
b) Plik z historią konwersji powinien mieć dodatkową kolumnę zawierającą ciąg znaków z użytym programem.

Punkty:4

[2] https://imagemagick.org/script/download.php