**Sterowanie witryną w2g za pomocą stream deck – z wykorzystaniem biblioteki selenium oraz wzorca client-server.**

Przygotowanie aplikacji

Spis rzeczy:

[I) Testowanie rozwiązań z użyciem biblioteki “dotenv” do wykorzystania zmiennych środowiskowych. 1](#_Toc82521874)

[II) Aplikacja z wykorzystaniem biblioteki “socket” (wzorzec client <-> server); biblioteki “selenium” (do zarządzania wybranym serwisem internetowym. 3](#_Toc1008181289)

[a. Problemy związane z blokowaniem localhostu. 3](#_Toc1605172334)

[b. Struktura programu 3](#_Toc182999018)

[c. Plik server.py, nasłuchiwanie poleceń 3](#_Toc582140929)

[d. Plik client.py, wysyłanie poleceń do servera. 4](#_Toc792431962)

[e. Podprogramy server.py 4](#_Toc1866328856)

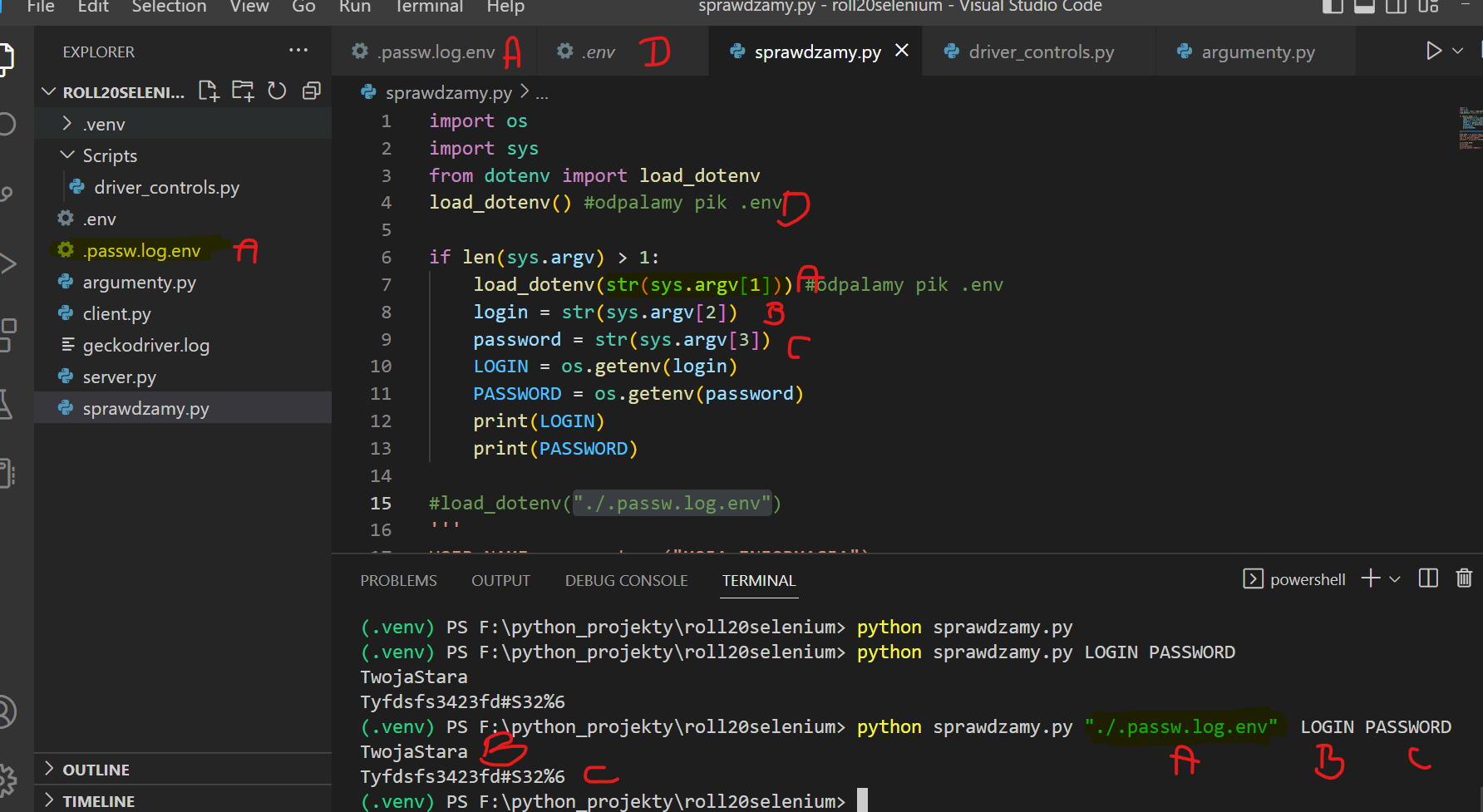
# Testowanie rozwiązań z użyciem biblioteki “dotenv” do wykorzystania zmiennych środowiskowych.

Cel: wykorzystanie zmiennych środowiskowych do przechowywania danych do logowania

1. Korzystanie z **environment variables** z pliku; za pomocą os i sys. Dodanie ich do środowiska poprzez komendę np. **$env:LOGIN** tak by można później z tej zmiennej korzystać w programie. Dodaje się w ten sposób wszytkie zmienne z pliku



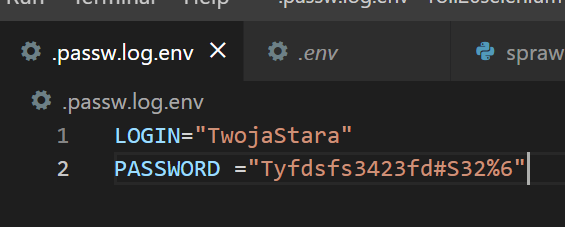
1. Zmienne te przechowują np. Login i hasło. Poniżej także hierarchia plików.



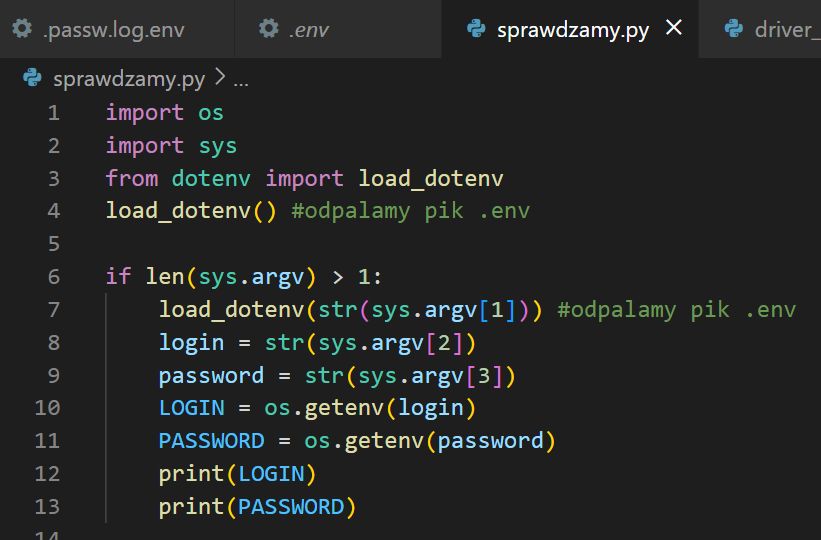
A – nazwa pliku oraz w komendzie ścieżka do pliku jako argument

B i C nazwy zmiennych zapisanych w pliku

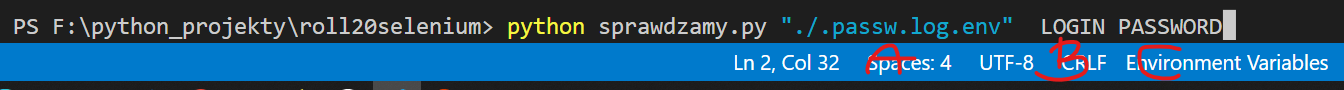
Zawartość pliku ‘.passw.log.env’

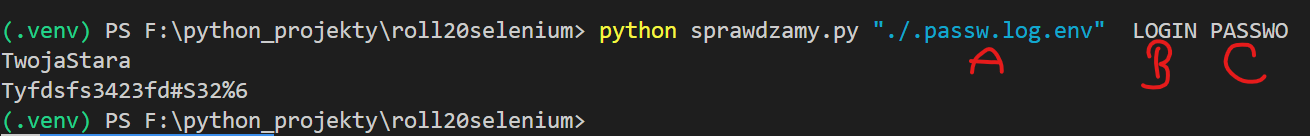


Zawartość pliku sprawdzamy.py



Wywołanie pliku “sprawdzamy.py” wraz z argumentami: ścieżka do pliku z **A** z environment variables **“./.passw.log.env”; B** zmienną **LOGIN; C** zmienną **PASSWORD**





Po wywołaniu pliku wyświetlane są przypisane do zmiennych wartości.

**Wymagania:** zainstalowana biblioteka dotenv

**Algorytm:**

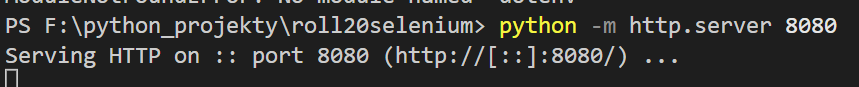
1. importuj biblioteki os, sys oraz funkcje load\_dotenv z biblioteki dotenv (trzeba ja wcześniej zainstalować)
2. Sprawdzić czy plik został wywołany z dodatkowymi argumentami (pierwszy, czyli sys.argv[0] to nazwa wywołanego pliku), jeżeli tak przejdź do kroku 3, w przeciwnym przypadku przejdź do kroku 6.
3. Załaduje plik z environment variables, użyj do tego ścieżki do pliku podanej w sys.argv[1] (**A**); przejdź do kroku 4
4. Przypisz do zmiennych login i password wartości podane w odpowiednio w sys.argv[2] i sys.argv[3] (**B** i **C**)
5. Wyświetl wartości zmiennych login i password
6. Zakończy program

# Aplikacja z wykorzystaniem biblioteki “socket” (wzorzec client <-> server); biblioteki “selenium” (do zarządzania wybranym serwisem internetowym.

**Cel:** Aplikacja, której celem będzie zarządzanie serwisem internetowym (w2g) za pomocą komend wysyłanych od clienta do servera. W ten sposób za pomocą przycisków w stream decku możliwe będzie zarządzanie playlistami na witrynie.

## Problemy związane z blokowaniem localhostu.

Aby programy server.py i client.py mogły się swobodnie komunikować (unikając blokowania przez zabezpieczenia systemu jak firewall) należy uruchomić w powłoce np. Cmd polecenie “p**ython –m http.server 8080**”. Tym samym powołamy server http, który będzie pośredniczył w komunikacji pomiędzy naszymi programami.



Wybrany port 8080 można zmienić na dowolny inny z dostępnego zakresu.

Od teraz można uruchomić plik server.py a następnie wysyłać do niego komunikaty z client.py (choć komunikacja może być jak najbardziej dwustronna).

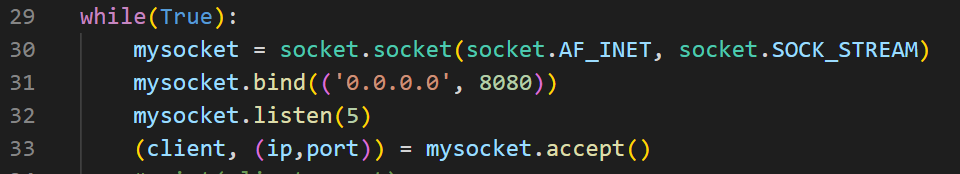
## Struktura programu

xxxxxxxxxxxxxx

## Plik server.py, nasłuchiwanie poleceń

Plik server.py będzie miał szereg zadań w aplikacji:

* Będzie przyjmować podczas uruchomienia argumenty z polecenia cmd/powershell - na podstawie, których poda do podprogramu dane do logowania oraz ścieżkę do pliku z loginem i hasłem, którego program użyje później do zalogowania się do witryny internetowej.
* Program server.py będzie trwał w pętli, w której będą wykonywane następujące zadania:
  + Nasłuchiwania sygnałów pochodzących od clienta.
  + Przekazywanie poleceń do podprogramów zarządzających witryną internetową



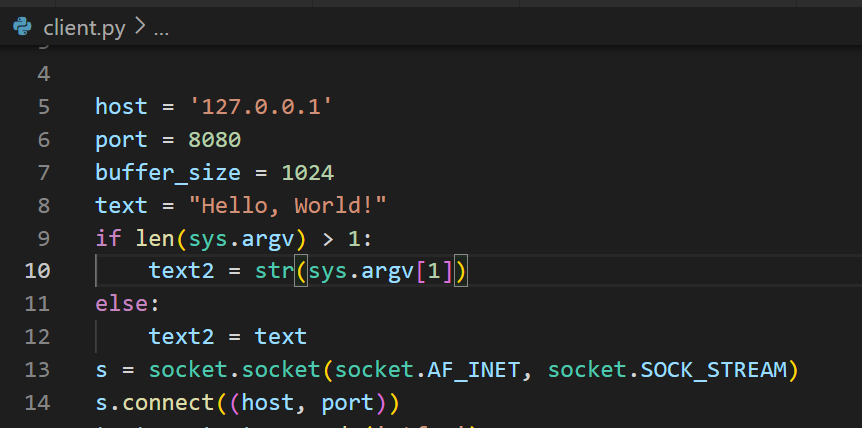
Wykorzystanie **socket** w programie server.py do ustanowienia połączenia z http.server na porcie 8080. Adres jest podany jako 0.0.0.0.

## Plik client.py, wysyłanie poleceń do servera.

Program client.py ma za zadanie wysłać informacje do serwera. Informacje te to polecenie, które program ma wykorzystać do sterowania serwisem internetowym w2g. W tym celu podczas wywołania w konsoli cmd otrzymuje także argument – komendę. Przykładowo **python client.py HORROR** będzie oznaczać, że witryna ma przełączyć się na playlistę o nazwie “horror” i uruchomić ją. Client będzie przesyłał również komendy, które pozwolą włączyć następny utwór lub zatrzymać/wznowić odtwarzanie.

Ideą jest, że urządzenie stream deck będzie miało przypisane pod przyciski odpowiednio multi-akcje, w których będzie wywoływana konsolo cmd, wywoływane polecenie cmd a następnie zamykana konsola.

Client będzie mógł wysłać polecenie kończące pracę serwera.

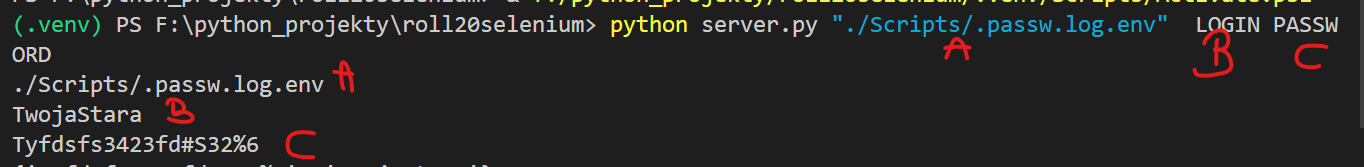


W pliku **client.py** ustalamy połączenie z lokalną maszyną, której adres to 127.0.0.1, port nasłuchu 8080 (musi być ten sam co **http.servera** oraz jaki nasłuchuje serwer.py). Metoda socket.connect(), ustanawia połączenie z hostem na zadanym porcie.

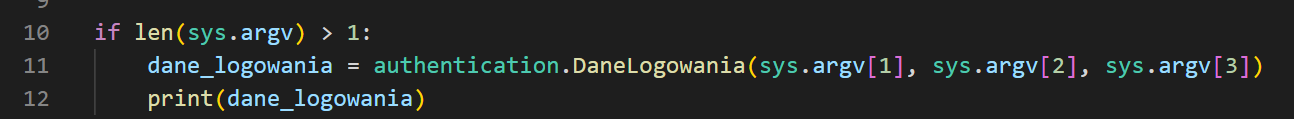
## Podprogramy server.py

W przeciwieństwie do clienta, serwer będzie korzystać z pracy wielu podprogramów, których zadaniem będzie obsługa:

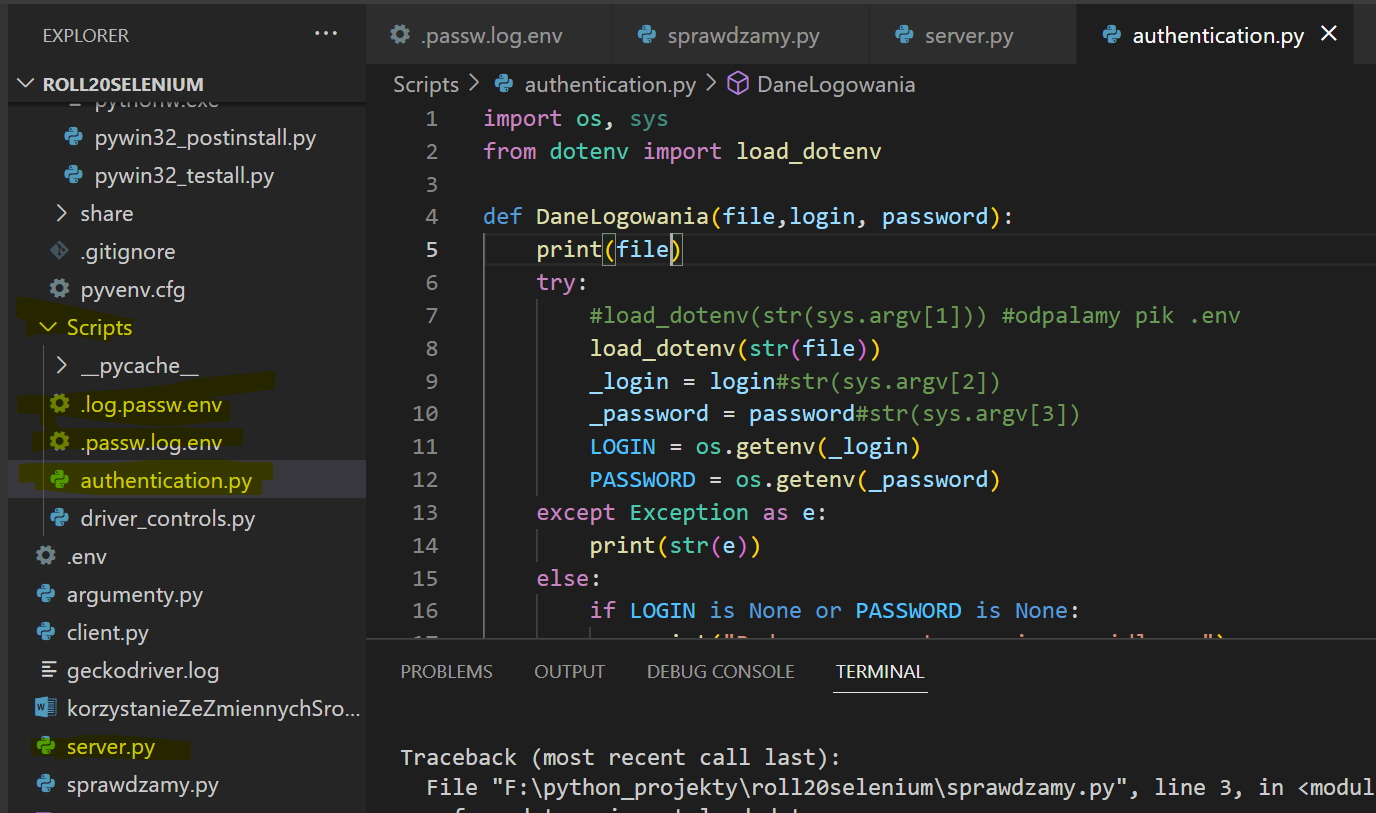
* Odebrania wartości argumentów zawierających dane dotyczących zmiennych środowiskowych przekazanych do serwera podczas jego uruchamiania. Pierwszym argumentem będzie ścieżka do pliku ze zmiennymi środowiskowymi, kolejnym login, ostatnim hasło. {wykorzystanie biblioteki **os, dotenv**}: polecenie cmd podczas uruchamiania pliku głównego server.py > **python server.py “./Scripts/.passw.log.env” LOGIN PASSWORD**; na poniższej ilustracji podprogram również wypisał wartości argumentów przekazanych podczas wywołania polecenia. Istotnym jest podanie ścieżki do pliku env z poziomu programu głównego.



*cmd*



*server.py*



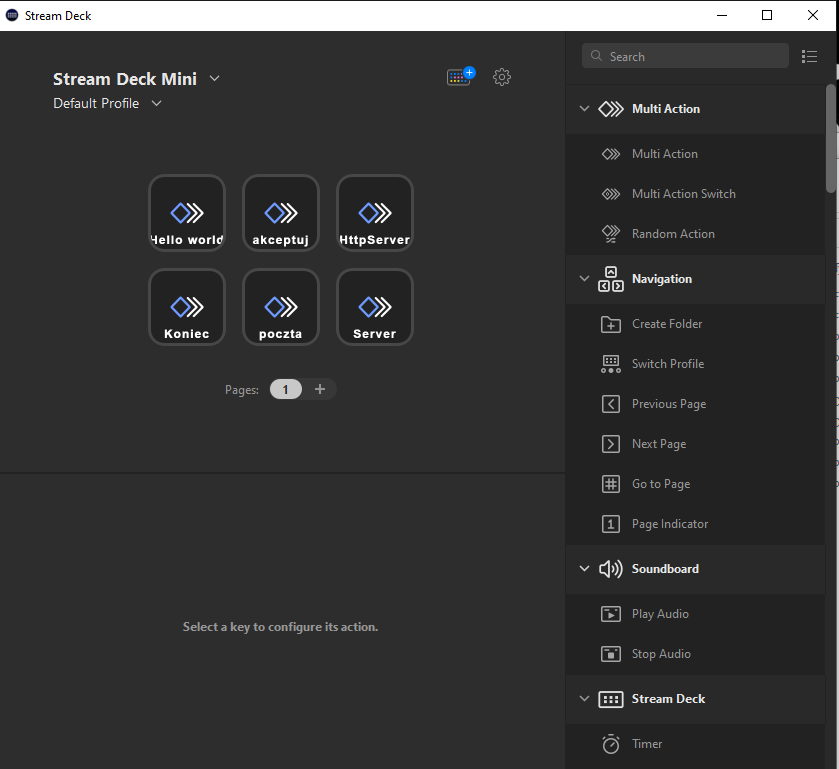
*authentication.py*

* Uruchamianie “webdrivera Firefox” i otwieranie przeglądarki na stronie w2g; logowanie się do serwisu. {wykorzystanie biblioteki **selenium**}. Po uzyskaniu hasła i loginu z poprzedniego podprogramu – podprogram driver\_contrls.py użyje ich do zalogowania się na witrynie.
* Odbieranie poleceń sterujących witryną. Serwer będzie nasłuchiwał i odbierał dane od clienta; jeżeli dane będą prawidłowe (tzn. Będą się zawierać w dostępnych komendach) zostaną przekazane do podprogramu i użyte do sterowania witryną w2g.
* Komendy sterujące zarządzaniem serwisem w2g będą znajdować się w podprogramie playercontrols.py

# Wykorzystanie stream deck do kierowania pracą programu i sterowania witryną w2g

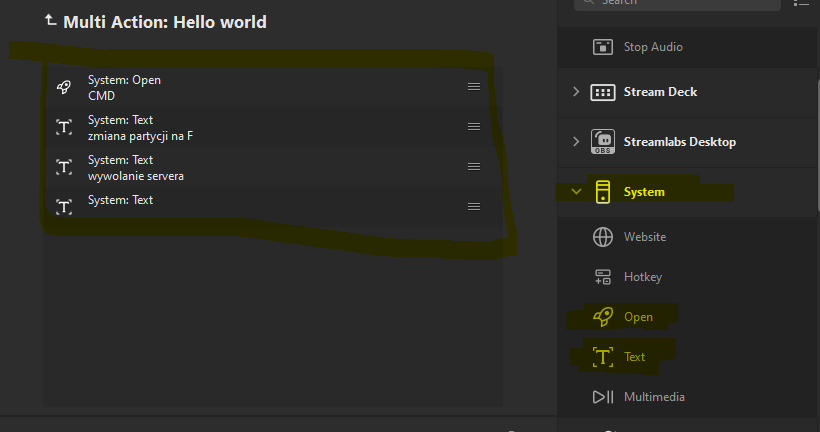
Ideą jest aby za pomocą konsoli stream deck kierować pracą programu. Do przycisków konsoli przypisane będą funkcję, które będą uruchamiać http.server na porcie np. 8080; wywoływać plik server.py z argumentami; oraz wywoływać plik client.py z odpowiednim argumentem.

W tym celu do programowania przycisków wykorzystana będzie funkcja multi action z modułu multi action w oprogramowaniu strem deck.



**Powyższe przykłady dotyczyły sterowania witryną www.wp.pl**

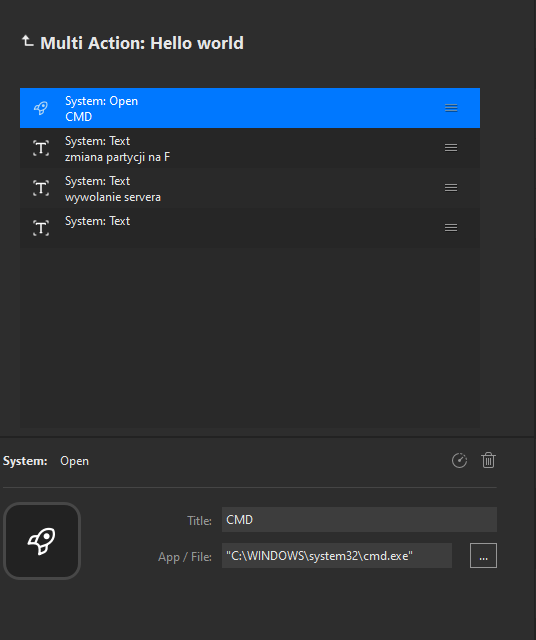
Następnie w każdej multiakcji podajemy sekwencję akcji jakie mają się wykonać:



W powyższym przykładzie kolejność będzie następująca: uruchom konsole CMD, wywołaj w niej polecenia przekazane przez System: Text

Przykładowo dla multiakcji **Hello world** (nieśmiertelnej i uniwersalnej) będzie to wyglądać następująco:

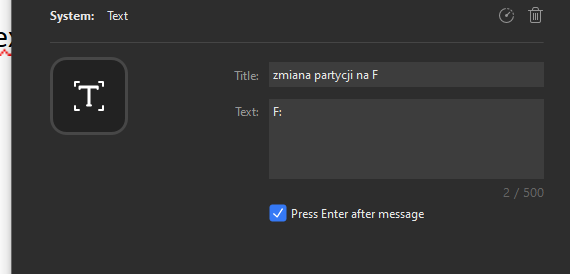
* Akcja uruchomienia konsoli cmd



Podajemy ścieżkę do uruchamianej aplikacji cmd.exe w polu App/File:

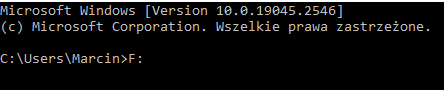
* Akcja System:Text zmieniająca partycję systemową (opcjonalnie)

Ponieważ w tym przypadku pliki z programem znajdują się na innej partycji F, należy do niej przejść w wywołanej konsoli.

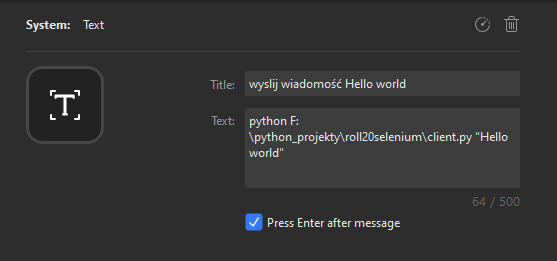


Istotnym jest aby zaznaczyć pole Press Enter after message, by wykonać polecenie w wywołanej konsoli.

To samo polecenie w konsoli cmd wyglądałoby tak:

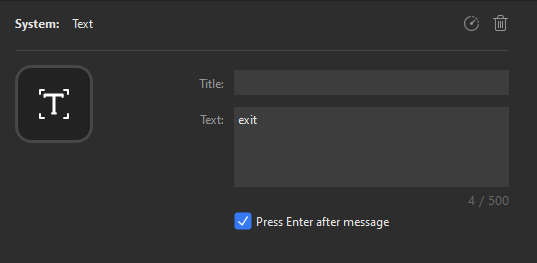


* Po przejściu (jeżeli to wymagane) wywołuje się w konsoli kolejne polecenie (kolejna akcja w multiakcjach przypisanych do przycisku.



Jest to polecenie python <ścieżka do wywoływanego pliku.py> argument (będzie to sys.argv[1]), oraz zaznaczanie ENTER po wysłaniu tego tekstu. W tym momencie konsoli zostanie uruchomiony program client.py i do nasłuchującego servera zostanie przekazany argument “Hello world”. Jeżeli taka komenda by miała jakąś funkcję w zarządzaniu witryną w2g, to server.py wykorzystał ją do zarządzania witryną.

* Ostatnia akcja zamyka wywołaną konsolę cmd poleceniem exit



Unikamy tym samym mnożących się okienek z wywołaniem programu client.py