Automat biletowy MPK.

Aplikacja składa się z dwóch plików: automat.py oraz views.py (oraz automat\_test.py do testów).

Wszystkie założenia są zgodne z opisem projektu, poza pewnymi wyjątkami. Główną różnicą jest to, że w moim programie zwrot monet odbywa się na zasadzie, że najpierw zwracane są największe nominały, a nie dokładnie te same monety jakie zostały wrzucone. Różnicą jest też to, że funkcja zwracająca monety nie rzuca wyjątku, jeśli nie może wydać monet, zamiast tego go obsługuje wyświetlając odpowiedni komunikat na ekran oraz zwracając wrzuconą wartość monet:

<https://github.com/mathtev/ticket-dispenser/blob/f65c92be59661632e1cff3e95afc571b75285cf3/automat.py#L268-L278>

Wydaje mi się, że główną wadą projektu jest użycie zwykłej listy do przechowywania obiektów Moneta, gdzie monety o tym samym nominale się powtarzają. Zamiast tego mógłbym użyć collections.defaultdict(int) do zliczania, ile jest monet każdego typu.

Interfejs składa się z trzech stron. Strony są tworzone w klasie Application dziedziczą po obiekcie kontenera typu tk.Frame i są kontrolowane przez objekt klasy Application:

<https://github.com/mathtev/ticket-dispenser/blob/f65c92be59661632e1cff3e95afc571b75285cf3/views.py#L52-L60>

Strony są umieszczane na wierzchu za pomocą komendy tkraise():

<https://github.com/mathtev/ticket-dispenser/blob/f65c92be59661632e1cff3e95afc571b75285cf3/views.py#L69>

Monety są wczytywane do automatu zaraz po uruchomieniu aplikacji z pliku lista.csv.

W oknie głównym użytkownik może wybrać dany bilet za pomocą znaków + -. Po wyborze biletów przycisk ‘Kupuję i płacę’ prowadzi do okna, w którym można wpłacić monety naciskając odpowiedni przycisk ze zdjęciem, następnie trzeba wybrać liczbę monet wprowadzając ich liczbę do formularza. Domyślnie jest to 1. W każdej chwili jest możliwość powrotu do wyboru biletów, a także możliwość zwrotu pieniędzy. Po naciśnięciu przycisku zapłać pokazywana jest strona z napisem ‘Dziękujemy’ oraz przycisk na wyjście z programu. Dosyć łatwo można zaimplementować w programie funkcje, które by go resetowały (komentarze TODO w pliku views.py).

Wszystkie testy przebiegły pomyślnie za wyjątkiem trzeciego co wynika z obsługi błędu w funkcji wydaj resztę (pierwszy link). Aby testy zadziałały nie można zmieniać pliku lista.csv:

<https://github.com/mathtev/ticket-dispenser/blob/ab1e2ce9aa0aa4cb563914b84e9d3cb086a3680a/automat_test.py#L53-L72>

1. Program jest zgodny z opisem, raczej nie posiada dodatkowych funkcji.
2. Wszystkie testy przebiegły pomyślnie za wyjątkiem 3, gdyż nie rzucam wyjątku, co wynika bardziej z różnic w programie.
3. Konstrukcje Pythona:

-Wyrażenia lambda:

<https://github.com/mathtev/ticket-dispenser/blob/ab1e2ce9aa0aa4cb563914b84e9d3cb086a3680a/views.py#L180>

<https://github.com/mathtev/ticket-dispenser/blob/ab1e2ce9aa0aa4cb563914b84e9d3cb086a3680a/views.py#L191>

<https://github.com/mathtev/ticket-dispenser/blob/ab1e2ce9aa0aa4cb563914b84e9d3cb086a3680a/views.py#L281>

<https://github.com/mathtev/ticket-dispenser/blob/ab1e2ce9aa0aa4cb563914b84e9d3cb086a3680a/views.py#L295>

-List comprehensions:  
<https://github.com/mathtev/ticket-dispenser/blob/2001b98e551087e8e78dfef2f3e64fa48ad618e1/automat.py#L6>

<https://github.com/mathtev/ticket-dispenser/blob/2001b98e551087e8e78dfef2f3e64fa48ad618e1/automat.py#L353>

<https://github.com/mathtev/ticket-dispenser/blob/2001b98e551087e8e78dfef2f3e64fa48ad618e1/automat.py#L135>

-Wyjątki:

<https://github.com/mathtev/ticket-dispenser/blob/2001b98e551087e8e78dfef2f3e64fa48ad618e1/automat.py#L224-L233>

<https://github.com/mathtev/ticket-dispenser/blob/2001b98e551087e8e78dfef2f3e64fa48ad618e1/automat.py#L199-L200>

<https://github.com/mathtev/ticket-dispenser/blob/2001b98e551087e8e78dfef2f3e64fa48ad618e1/automat.py#L267-L276>

i więcej

-podział programu na przynajmniej dwa pliki, np. jeden do obsługi logiki i drugi do obsługi interfejsu

jest

1. wyróżniające elementy

raczej nie ma

1. Czytelność i udokumentowanie kodu

Czytelność: mam nadzieję, że jest

udokumentowanie: jest

aktywność na github: jest