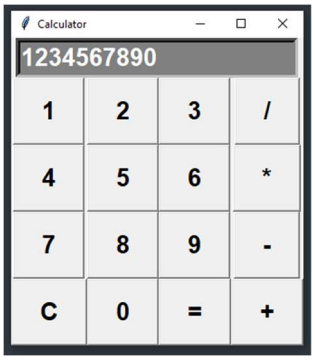
Zestaw 8

Zestaw jest dedykowany prostym aplikacjom z interfejsem graficznym tkinter. Aby zadania nie polegały wyłącznie na generowaniu okienka i jego elementów, towarzyszą im dodatkowe cele, ale wystarczająca jest minimalna realizacja założonych funkcji.

1. Kalkulator. Celem jest napisanie aplikacji, która będzie bardzo prostym kalkulatorem działającym na liczbach całkowitych i wykonującym podstawowe operacje arytmetyczne. Przykładowy wygląd:

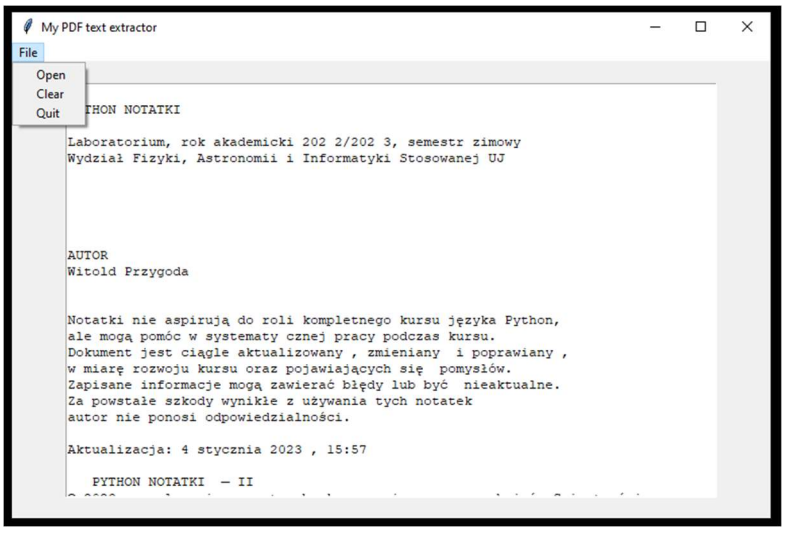


   Klawisz C oznacza czyszczenie wpisanych wcześniej wartości. Aby ułatwić i przyspieszyć napisanie programu, załączony jest plik zadanie1.py, który zawiera część kodu, którą najlepiej jest rozwinąć, dopisując brakujące rzeczy. Użyte widgety to `Entry` oraz `Button`. Aby nie powielać kodu, proponuję zamiast programować jakąś funkcję reakcji (poprzez argument command=), użyć reakcję na kliknięcie klawiszem myszy w któryś z `Button`s, za pomocą:

```
mainwindow.bind("<ButtonRelease-1>", mouse_button_release)
```

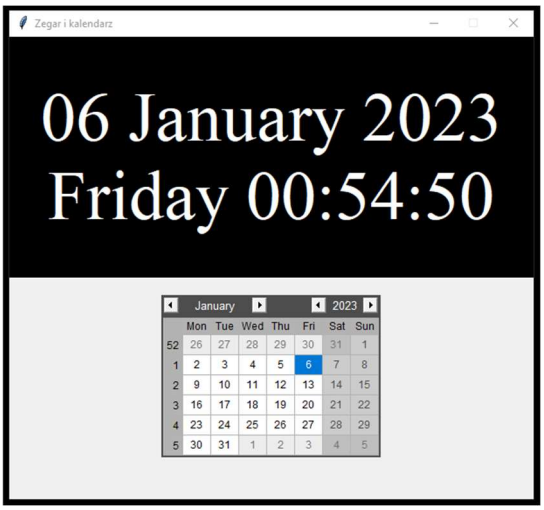
   [**zadanie za 2 pkt**].

2. Czytnik PDF. A dokładniej, program, który wydobywa ze wskazanego pliku PDF tekst i wyświetla go w widgecie Text. Spodziewany wygląd programu:



   Program powinien używać moduł PyPDF2, a do wczytywania plików filedialog. Należy stworzyć Menu tak jak na rysunku, szczegóły są podane w pliku zadanie2.py, w którym pozostawiłem cały kod wykonujący wczytanie (czyli kompletną funkcję open_pdf) [**zadanie za 2 pkt**].

3. Zegar i kalendarz. W prostej aplikacji będziemy odczytywać i odświeżać bieżącą datę i czas, a poniżej tego zegara wstawimy interaktywny kalendarz. Program powinien wyglądać podobnie do:



Użyte tutaj są moduły datetime, z którego możemy za pomocą odpowiednich metod i kodów formatujących pozyskać dowolnie informację o dacie i czasie
https://docs.python.org/3/library/datetime.html#strftime-and-strptime-format-codes
Użyty też jest moduł tkcalendar (należy zainstalować), szczegóły na temat proponowanej struktury programu są w pliku zadanie3.py [**zadanie za 1,5 pkt**].

4. Program do skalowania zdjęć. W tym przypadku użyty zostanie dodatkowo widget PhotoImage, a także moduł PIL (Python Imaging Library) oraz Image. Oczekiwany wygląd programu:

Ponieważ celem nie jest poświęcenie dużej ilości czasu na pisanie kodu obsługującego, więc cała mechanika programu została zachowana w pliku zadanie4.py. Jedyne, co w tym zadaniu należy zrobić, to zdefiniować i spozycjonować odpowiednio widoczne na obrazku widgety przycisków i pól, w które wpisujemy dane. Są one zakomentowane, trzeba po prostu kod uzupełnić. Natomiast warto obejrzeć uważnie cały kod, w jaki sposób wczytywane, skalowane i zapisywane są pliki (w tym przypadku) .png [**zadanie za 1,5 pkt**].

5. Rozwiążemy i narysujemy problem znany jako znalezienie największego pustego prostokąta (MER, Maxiumum Empty Rectangle). Na wejściu mamy kwadratowe pole składające się z 0 i 1, w którym 1 to pole zablokowane, a 0 to pole, które można użyć. W podanym wzorcu należy znaleźć prostokąt o największym możliwym polu powierzchni (lub 0, gdyby wszystkie pola były 1). Przykładowo, dla wejścia:
   5
   0 1 0 1 0
   0 0 0 0 0
   0 0 0 0 1
   1 0 0 0 0
   0 1 0 0 0
   gdzie 5 oznacza rozmiar kwadratu, a kolejne linie sam kwadrat z danymi, poprawna odpowiedź to 9 (zaznaczone kolorem niebieskim). Zadanie polega na napisaniu poprawnie działającego algorytmu (proszę przewidzieć wczytywanie wzorca z pliku, w formacie takim jak w przykładzie), oraz – narysowanie graficznie z pomocą tkinter i dowolnie wybranych widgetów tego wczytanego wzorca, z wyodrębnieniem znalezionego prostokąta. Wyodrębnić można w dowolny sposób, czy to przez kolor czcionki, czy przez linie otaczające – według uznania. W tym przypadku cały kod należy napisać samemu. Uwaga: aby pomóc w weryfikacji poprawności algorytmu, załączam dodatkowo testowy plik wejściowy zadanie5_input.txt z kwadratem o rozmiarze 100, dla którego poprawnie znaleziony największy prostokąt ma pole powierzchni 368 [**zadanie za 3 pkt**].