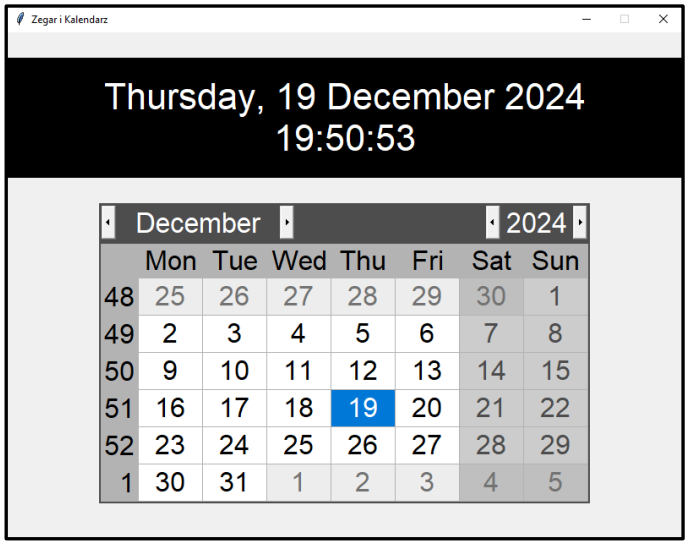
Zestaw 7

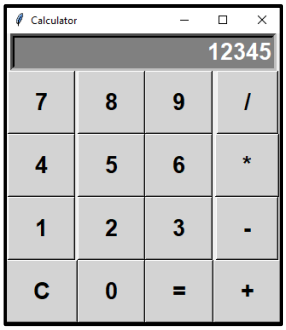
Zestaw dotyczy aplikacji z interfejsem graficznym `tkinter`. Należy zapoznać się albo z rozdziałem `TKINTER` w dokumencie „Python notatki", albo z wykładem https://ufkapano.github.io/algorytmy/lekcja10/index.html. Aby zadania nie polegały wyłącznie na generowaniu okienka i jego elementów, towarzyszą im dodatkowe cele, ale wystarczająca jest minimalna realizacja założonych funkcji. Programy nie mają testów, będą osobno uruchamiane i sprawdzane. O ile `tkinter` jest dobrym punktem startowym do nauki, to istnieją rozwiązania bardziej nowoczesne, lepiej wyglądające, dające większe możliwości. Wymienię kilka: `CustomTkinter`, `ttkbootstrap`, `kivy`, `PyQT`, `PySide`. Wybór konkretnej biblioteki będzie uzależniony również od rodzaju licencji, na jakiej jest ona udostępniana. Jeśli ktoś ma ochotę wykorzystać którąś z tych bibliotek, ale zachowując meritum poniższych zadań, to może tak zrobić. Ponieważ zadania wymagają pewnego nakładu pracy w poznanie i testowanie `tkinter`, są w tym zestawie cztery, każde za **1,25 pkt** (razem za całość 5 pkt).

1. Zegar i kalendarz. W prostej aplikacji będziemy odczytywać i odświeżać bieżącą datę i czas, a poniżej zegara wstawimy interaktywny kalendarz. Program powinien wyglądać mniej więcej tak:



   Z modułu `datetime` 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).
   Wymagania formalne Użyć plik `ZADANIE1/zadanie1.py` w repozytorium GitHub Classroom do uzupełnienia swoim kodem, w którym są szczegóły na temat proponowanej struktury programu. Proszę postarać się tak dobrać wielkości fontów i okienka, żeby całość wyglądała estetycznie.

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

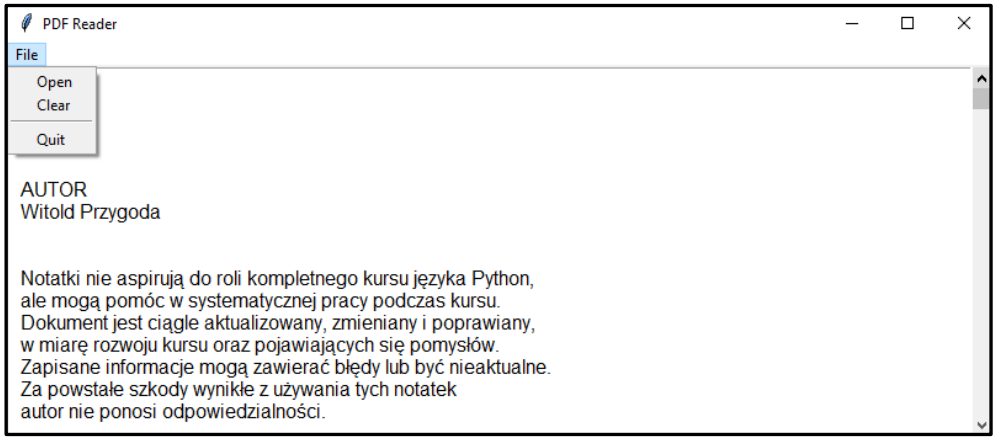


   Klawisz `C` oznacza czyszczenie wpisanych wcześniej wartości. Aby ułatwić i przyspieszyć napisanie programu, załączony jest plik `ZADANIE2/zadanie2.py`, zawierający część kodu, którą najlepiej jest rozwinąć, dopisując brakujące rzeczy. Użyte widgety to `Entry` oraz `Button`. Aby nie powielać kodu, proponuję zamiast programować jakąś funkcję reakcji (poprzez argument `command=`), użyć reakcję na kliknięcie klawiszem myszy w któryś z `Buttons`, za pomocą: `mainwindow.bind("<ButtonRelease-1>", mouse_button_release)`

W kodzie przyciski mogą wyglądać nierówno w zależności od systemu operacyjnego i domyślnego zarządzania rozmiarami przez `grid`. W `tkinter` domyślnie szerokość przycisków może różnić się w zależności od długości tekstu (np. "-" vs "+"). Rozmiary przycisków można zdefiniować precyzyjnie, ale nie jest to wymagane w tym zadaniu.

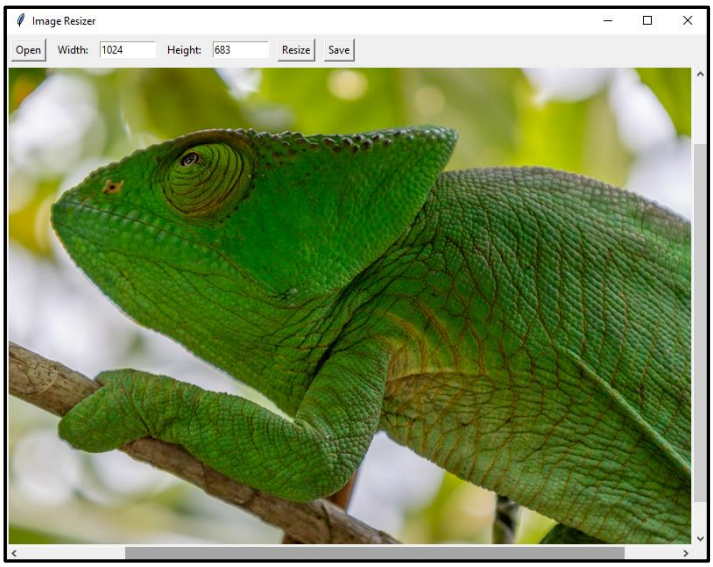
Wymagania formalne Użyć plik `ZADANIE2/zadanie2.py` w repozytorium GitHub Classroom do uzupełnienia swoim kodem, proszę zwrócić uwagę na obsługę błędów, to znaczy sytuacji, gdy ktoś np. będzie dzielił przez 0 – powinien zostać wyświetlony jakiś komunikat, `ERROR`.

3. Biblioteka `pymupdf` pozwala na wczytywanie plików w formacie PDF, jak również ekstrakcję ich zawartości. Celem zadania będzie napisanie programu, który wydobędzie ze wskazanego pliku PDF tekst i wyświetla go w widgecie `Text`. Spodziewany wygląd programu:



Program powinien używać moduł `fitz`, a do wczytywania plików `filedialog`. Należy stworzyć Menu tak jak na rysunku.
Wymagania formalne Użyć plik `ZADANIE3/zadanie3.py` w repozytorium GitHub Classroom do uzupełnienia swoim kodem. W pliku jest cały kod wykonujący wczytanie (czyli kompletna funkcja `open_pdf`). Reszta opisana w komentarzach.

4. Program do wyświetlania i skalowania zdjęć. W tym przypadku użyty zostanie `Pillow`, moduł do pracy z obrazami w Pythonie. Pillow to zaktualizowana wersja starej biblioteki PIL (Python Imaging Library). Aby korzystać z tego modułu, należy go zainstalować: `pip install Pillow`. Moduł Pillow pozwala na otwieranie, manipulację i zapisywanie obrazów w wielu formatach, a `ImageTk` umożliwia wyświetlanie obrazów w aplikacjach `tkinter`. Oczekiwany wygląd programu:



Ponieważ celem nie jest poświęcenie dużej ilości czasu na pisanie kodu obsługującego, więc cała mechanika programu została zachowana w pliku. Aplikacja powinna umożliwiając dynamiczne wyświetlanie, zmianę rozmiaru i zapis obrazów z odpowiednim zarządzaniem paskami przewijania. Precyzyjniej wymieniając, zadaniem jest uzupełnienie brakujących fragmentów kodu w aplikacji do zmiany rozmiaru obrazów w `tkinter`. Trzeba dodać paski przewijania do siatki (`grid`) w funkcji `setup_ui`, tak aby były widoczne na `Canvas` i działały poprawnie podczas wyświetlania dużych obrazów. Następnie, w funkcji `resize_handler`, obsłużyć proces zmiany rozmiaru obrazu. Sprawdzić, czy wczytano obraz, a wartości w polach `width` i `height` są poprawnymi

liczbami. Po zmianie rozmiaru obrazu, wyświetlić go na `Canvas`. Kolejnym krokiem jest uzupełnienie funkcji `save_handler`. W tej funkcji zaimplementować możliwość zapisania zmienionego obrazu do nowego pliku w lokalizacji wybranej przez użytkownika, wykorzystując `filedialog.asksaveasfilename`. Ostatnim zadaniem jest skonfigurowanie `scrollregion` dla `Canvas` w funkcji `display_image`, aby scroll bary pojawiały się automatycznie, gdy wczytany obraz przekracza wymiary dostępnego obszaru `Canvas`. Dzięki temu użytkownik będzie mógł przewijać obraz, jeśli jego rozmiar jest większy niż okno aplikacji.

Wymagania formalne Użyć plik `ZADANIE4/zadanie4.py` w repozytorium GitHub Classroom do uzupełnienia swoim kodem. W pliku odszukać miejsca opisane do uzupełnienia w komentarzach.