Birthday Cards Generator - documentation

This application was made as a tool to easily create birthday cards. The algorithm randomly generates a birthday card choosing from 12 background images, 4 frames, and 4 different wishes. The choice depends on the gender of the person provided by the user.

The project has been split into several, separate files and directories:

1. Content directories:

- Background_imgs - contains 12 images, backgrounds for the cards
- Fonts - contains 2 possible fonts (only 1 is used at the time)
- Frames - contains 4 frames, all in png format
- Wishes - contains 4 text files with wishes

2. constans.py:

- mainFont - color of the font
- urlBasic - api request url
- fntSrc - source of the font
- dictionaries - each one is a dictionary of dictionaries, key is a name of the item, values are dict that contains params, every dict has 'buffer' key used as a starting point in *getItem* function:
    - backgroundImages (stores backgrounds): params: source - direct source to file that stores image; gender - str that indicates for whom it should be used
    - Wishes (stores wishes): params: *source* - direct source to file that stores text of the wishes; *gender* - str that indicates for whom it should be used, *fntSize* - font size; *shadowColor* - font shadow color in RGB format
    - Frames (stores frames): params: *source* - direct source to file that stores image, *gender* - str that indicates for whom it should be used

3. model_io.py - contains functions to read .txt files:

- read_wishes - takes file in txt format and returns it as str
- load_wishes - opens text file from a given source (file_handle) and calls read_wishes function, returns the text of wishes as str.

4. generator.py - contains functions responsible for creating Cards:

- getItem - randomly chooses a key from the given dictionary based on the given gender. Returns the *key* (str) of the chosen item
- setBaseImage - gets a random frame and combines it with the background. Creates base for the text, and gets info about the font. *R*eturns: combined *background (*PIL.Image), *base* for the text (PIL.Image), and *font (ImageFont)*
- drawText - draws wishes text on the center of the card using given font**.** *R*eturns PIL.Image, ready to be combined.
- combineImage - combines base (generated with setBaseImage function) and image with wishes text (generated with drawText function). *R*eturns: birthday card in PIL.Image format.
- imageConversion - converts PIL.Image. *R*eturns QPixmap.
- generate_Card - most important function, calls functions mentioned above to generate birthday card with the style based on the person given as an argument. *R*eturns birthday card as a PIL.Image (easy to save into PNG file) and QPixmap (Easy to display using QLabel).
- class Person - used to store information about the jubilarian. Param name (str) and method gender. Method gender uses API to assume the gender of the person.

5. GUI related files:

- mainMenu.ui, generatorWindow.ui, dialog.ui - files generated in QtDesigner,
- ui_mainMenu.py, ui_dialog.py, ui_generatorWindow.py - files converted with pyside2-uic

6. Dialog windows:

- dialog.py - responsible for showing generated birthday card with options to close the window and save the image as BirthdayCard.png in the repository.
- generatorWindow.py - responsible for generator window, shows editLine to allow user input name of the person and generate card by clicking the button. The used validator doesn't allow to input anything else but letters, empty names also won't be accepted.

7. gui.py:

It is the main file that is used to start the application. Displays MainWindow with QStack widget. Index[0] shows menu and calendar. Index[1] shows info about the author and credits.
goCredits button takes the user to the author&credits section. *goGenerator* button takes the user to the generator dialog window.
We can generate multiple images without the need to restart the app.

8. Tests:

- test_generator.py - tests functions of algorithm core
- test_model_io.py - tests loading and reading txt files functions
- there aren't many tests due to the fact, that most of GUI functionalities and the image combining were tested empirically, in usage.