***Импорт библиотек:***

* -tkinter as tk: Импортирует библиотеку Tkinter (стандартный графический интерфейс пользователя Python) и присваивает ей псевдоним tk для более удобного использования.
* -Модули из PIL: Импортирует необходимые модули из библиотеки Python Imaging Library (PIL), которая используется для работы с изображениями.
* Image: Обеспечивает основные функции для работы с изображениями, такие как открытие, обработка и сохранение.

-ImageTk: Позволяет отображать изображения PIL в приложениях Tkinter.

* io: Обеспечивает функции для работы с двоичными потоками данных, необходимые для загрузки и сохранения изображений.
* simpledialog из tkinter: Используется для отображения диалоговых окон, например, для запроса ввода пользователя.

***Главное окно:***

* root = tk.Tk(): Создает главное окно приложения.
* root.title("Графический редактор"): Устанавливает заголовок главного окна "Графический редактор" на русском языке.
* root.geometry("1200x800"): Устанавливает размер главного окна 1200 пикселей в ширину и 800 пикселей в высоту.
* Переключение темы:
* def toggle\_theme(): Определяет функцию для переключения между светлой и темной темами приложения. Эта функция будет реализована в следующих частях документации.

***Холст:***

* canvas\_width = 800 и canvas\_height = 600: Определяет ширину и высоту области рисования на холсте.
* canvas = tk.Canvas(root, width=canvas\_width, height=canvas\_height, bg="white"): Создает виджет холста для рисования внутри главного окна.
* canvas.pack(pady=20): Размещает виджет холста в главном окне с отступом 20 пикселей сверху и снизу.

***Переменные рисования:***

* draw\_color = "black": Устанавливает цвет рисования по умолчанию черным.
* eraser\_on = False: Изначально инструмент ластик отключен.
* eraser\_size = 10: Определяет размер кисти ластика.
* brush\_size = 3: Определяет размер кисти для рисования.
* original\_image = None: Хранит копию исходного изображения, загруженного на холст (изначально пустая).
* Масштабирование изображения:
* def scale\_image(scale\_percent):
* Масштабирует текущее отображаемое изображение на холсте.
* Проверяет, загружено ли исходное изображение (original\_image).
* Если да:
* Вычисляет новую ширину и высоту на основе предоставленного scale\_percent.
* Изменяет размер изображения с использованием original\_image.resize с сглаживанием для повышения качества.
* Создает новый объект PhotoImage из измененного изображения.
* Обновляет холст новым изображением с помощью canvas.create\_image и сохраняет ссылку на него в canvas.image.

***Фильтры изображений:***

* def apply\_grayscale():
* Применяет фильтр оттенков серого к загруженному изображению.
* Проверяет, загружено ли исходное изображение (original\_image).
* Если да:
* Преобразует изображение в оттенки серого с помощью ImageOps.grayscale.
* Создает новый объект PhotoImage из изображения в оттенках серого.
* Обновляет холст отфильтрованным изображением и сохраняет ссылку.
* def apply\_pink\_filter():
* Применяет розовый фильтр к загруженному изображению.
* Аналогично apply\_grayscale, но использует ImageEnhance.Color для увеличения насыщенности цвета в сторону розового.
* def apply\_blue\_filter():
* Применяет синий фильтр к загруженному изображению.
* Аналогично apply\_pink\_filter, но уменьшает насыщенность цвета в сторону синего.

***Обрезка изображения:***

* def crop\_image():
* Позволяет пользователям обрезать определенный регион из загруженного изображения.
* Проверяет, загружено ли исходное изображение (original\_image).
* Если да:
* Создает новое окно верхнего уровня (crop\_window) для обрезки.
* Создает новый холст (crop\_canvas) внутри crop\_window с теми же размерами, что и исходное изображение.
* Отображает исходное изображение на crop\_canvas с помощью PhotoImage.
* Инициализирует переменные start\_x и start\_y для хранения начальной точки выбора обрезки.
* def start\_crop(event):
* Записывает координаты клика (event.x и event.y) как начальную точку.
* def crop\_image\_done(event):
* Завершает процесс обрезки.
* Записывает финальные координаты клика (event.x и event.y) как конечную точку.
* Вычисляет левую, верхнюю, правую и нижнюю границы выбранной области обрезки.
* Извлекает выбранный регион с помощью original\_image.crop.
* Создает новый объект PhotoImage из обрезанного изображения.
* Обновляет основной холст обрезанным изображением и сохраняет ссылку.
* Уничтожает окно crop\_window.
* Связывает события с crop\_canvas:
* <Button-1>: Вызывает start\_crop для захвата начальной точки.
* <B1-Motion>: Создает красный прямоугольник для визуализации выбранной области обрезки, пока кнопка мыши удерживается.
* <ButtonRelease-1>: Вызывает crop\_image\_done для завершения обрезки при отпускании кнопки мыши.

***Отражение изображения:***

* def flip\_horizontal():
* Отражает загруженное изображение по горизонтали.
* Проверяет, загружено ли исходное изображение (original\_image).
* Если да:
* Создает отраженную по горизонтали версию изображения с помощью ImageOps.mirror.
* Создает новый объект PhotoImage из отраженного изображения.
* Обновляет холст отраженным изображением и сохраняет ссылку.

***Вращение изображения:***

* def rotate\_image():
* Позволяет пользователям вращать загруженное изображение на указанный угол.
* Проверяет, загружено ли исходное изображение (original\_image).
* Если да:
* Создает новое окно верхнего уровня (rotate\_window) для ввода угла вращения.
* Добавляет метку и поле ввода для ввода пользователем (angle\_label и angle\_entry).
* def rotate\_image\_done():
* Обрабатывает вращение, когда пользователь нажимает кнопку (не показано в этом фрагменте кода).
* Пытается преобразовать введенный угол (angle\_entry.get()) в целое число.
* Если успешно:
  + Поворачивает изображение на указанный угол с помощью original\_image.rotate.
  + Создает новый объект PhotoImage из повернутого изображения.
  + Обновляет холст повернутым изображением и сохраняет ссылку.

***Рисование прямоугольника:***

* def draw\_rectangle():
* Эта функция включает рисование прямоугольников на холсте.
* Она привязывает три события к холсту:
* <Button-1>: Вызывает start\_rectangle для захвата начальной точки прямоугольника, когда пользователь нажимает кнопку.
* <B1-Motion>: Вызывает draw\_rectangle\_on\_canvas, пока кнопка мыши удерживается, для обновления прямоугольника на основе текущей позиции курсора.
* <ButtonRelease-1>: Вызывает end\_rectangle для завершения рисования прямоугольника, когда кнопка мыши отпускается.
* def start\_rectangle(event):
* Записывает начальные координаты клика (event.x и event.y) как начальную точку для прямоугольника.
* def draw\_rectangle\_on\_canvas(event):
* Удаляет любой существующий прямоугольник на холсте (canvas.delete("rectangle")).
* Вычисляет ширину и высоту на основе начальной точки и текущей позиции курсора.
* Создает прямоугольник на холсте с помощью canvas.create\_rectangle с вычисленными размерами, цветом контура, установленным в draw\_color, и тегом "rectangle" для последующих ссылок.
* def end\_rectangle(event):
* Отвязывает все три события от холста после того, как прямоугольник нарисован.

***Рисование круга:***

* def draw\_circle():
* Эта функция похожа на draw\_rectangle, но для создания кругов.
* Она привязывает аналогичные события для обработки рисования кругов.
* start\_circle захватывает начальную точку.
* draw\_circle\_on\_canvas вычисляет радиус на основе начальной точки и текущей позиции курсора, затем создает овал на холсте с помощью canvas.create\_oval, чтобы представить круг.
* end\_circle отвязывает события после того, как круг нарисован.

***Панель меню:***

* menu\_bar = tk.Menu(root):
* Создает панель меню для приложения.
* root.config(menu=menu\_bar):
* Прикрепляет панель меню к главному окну.

***Меню "Файл":***

* file\_menu = tk.Menu(menu\_bar):
  + Создает подменю для файловых операций.
* file\_menu.add\_command(...):
  + Добавляет пункты меню для сохранения, открытия и очистки холста, вероятно, связанные с функциями, такими как save\_image и clear\_canvas (не представлены в этом коде).

***Меню "Вид":***

* view\_menu = tk.Menu(menu\_bar):
  + Создает подменю для параметров просмотра.
* view\_menu.add\_command(...):
  + Добавляет пункт меню для переключения темы (toggle\_theme).

***Меню "Правка":***

* edit\_menu = tk.Menu(menu\_bar):
  + Создает подменю для функций редактирования изображений.
* edit\_menu.add\_command(...):
  + Добавляет пункты меню для:
  + Масштабирования изображения (вызывает scale\_image с вводом пользователя из диалога).
  + Применения фильтра оттенков серого (apply\_grayscale).
  + Применения розового и синего фильтров (apply\_pink\_filter и apply\_blue\_filter).
  + Обрезки изображения (crop\_image).
  + Отражения по горизонтали (flip\_horizontal).
  + Вращения изображения (rotate\_image).

***Меню "Рисование":***

* draw\_menu = tk.Menu(menu\_bar):
  + Создает подменю для рисования фигур.
* draw\_menu.add\_command(...):
  + Добавляет пункты меню для рисования прямоугольников и кругов, связанные с соответствующими функциями (draw\_rectangle и draw\_circle).
* root.mainloop():
* Запускает главный цикл событий приложения, прослушивающий взаимодействие пользователя.