Tk является базовым классом любого Tkinter приложения.

Tkinter является событийно-ориентированной библиотекой. В приложениях такого типа имеется главный цикл обработки событий. В Tkinter такой цикл запускается **методом mainloop**. Для явного выхода из интерпретатора и завершения цикла обработки событий используется **метод quit.**

Таким образом **минимальное приложение на Tkinter** будет таким:

```
from tkinter import *
root = Tk()
root.mainloop()
```

Виджеты создаются вызовом конструктора соответствующего класса. Первый аргумент (как правило неименованный, но можно использовать имя *master*) это **родительский виджет**, в который будет упакован (помещён) наш виджет. Родительский виджет можно не указывать, в таком случае будет использовано главное окно приложения. Далее следуют **именованные аргументы**, конфигурирующие виджет. Это может быть используемый шрифт (font=...), цвет виджета (bg=...), команда, выполняющаяся при активации виджета (command=...) и т.д.

Полный список всех аргументов можно посмотреть в man options и man-странице соответствующего виджета.

Метод configure, config

Виджеты могут быть сконфигурированы во время создания, но иногда необходимо изменить конфигурацию виджета во время исполнения программы. Для этого используется **метод *configure* (или его синоним *config*)**.

Метод cget

Метод cget является **обратным** к методу configure. Он предназначен для получения информации о конфигурации виджета.

Метод destroy

Уничтожение виджета и всех его потомков.

Метод grab_*

Методы семейства grab_ предназначены для управления потоком события. Виджет, захвативший поток, будет получать все события окна или приложения.

- **grab_set** - передать поток данному виджету
- **grab_set_global** - передать **глобальный** поток данному виджету. В этом случае все события на дисплее будут передаваться этому виджету. Следует пользоваться очень осторожно, т.к. остальные виджеты всех приложений не будут получать события.
- **grab_release** - освободить поток
- **grab_status** - узнать текущий статус потока событий для виджета. Возможные значения: None, "local" или "global".

- **grab_current** - получить виджет, который получает поток

Метод focus_*

Методы семейства focus_ используются для управления фокусом ввода с клавиатуры. Виджет, имеющий фокус, получает все события с клавиатуры.

- **focus** (синоним focus_set) - передать фокус виджету.
- **focus_force** - передать фокус, даже если приложение не имеет фокуса. Используйте осторожно, поскольку это может раздражать пользователей.
- **focus_get** - возвращает виджет, на который направлен фокус, либо None, если такой отсутствует.
- **focus_displayof** - возвращает виджет, на который направлен фокус на том дисплее, на котором размещён виджет, либо None, если такой отсутствует.
- **focus_lastfor** - возвращает виджет, на который будет направлен фокус, когда окно с этим виджетом получит фокус.
- **tk_focusNext** - возвращает виджет, который получит фокус следующим (обычно смена фокуса происходит при нажатии клавиши Tab). Порядок следования определяется последовательностью упаковки виджетов.
- **tk_focusPrev** - то же, что и focusNext, но в обратном порядке.
- **tk_focusFollowsMouse** - устанавливает, что виджет будет получать фокус при наведении на него мышью. Вернуть нормальное поведение достаточно сложно.

Методы after, after_idle и after_cancel

Таймеры. С помощью этих методов вы можете отложить выполнение какого-нибудь кода на определённое время.

after - принимает два аргумента: время в миллисекундах и функцию, которую надо выполнить через указанное время. Возвращает идентификатор, который может быть использован в after_cancel.

after_idle - принимает один аргумент - функцию. Эта функция будет выполнена после завершения всех отложенных операций (после того, как будут обработаны все события). Возвращает идентификатор, который может быть использован в after_cancel.

after_cancel - принимает один аргумент: идентификатор задачи, полученный предыдущими функциями, и отменяет это задание.

Методы update и update_idletasks

Две функции, для работы с очередью задач. Их выполнение вызывает обработку отложенных задач.

update_idletasks выполняет задачи, обычно откладываемые "на потом", когда приложение будет простаивать. Это приводит к прорисовке всех виджетов, расчёту их расположения и т.д. Обычно эта функция используется если были внесены изменения в состояние приложения, и вы хотите, чтобы эти изменения были отображены на экране немедленно, не дожидаясь завершения сценария.

update обрабатывает все задачи, стоящие в очереди. Обычно эта функция используется во время "тяжёлых" расчётов, когда необходимо чтобы приложение оставалось отзывчивым на действия пользователя.

Методы eval и evalfile

Две недокументированные функции для выполнения кода на tcl. **eval** позволяет выполнить строку на языке программирования tcl, а **evalfile** - выполнить код, записанный в файл. В качестве аргументов принимают соответственно строку и путь к файлу. Данные функции полезны при использовании дополнительных модулей, написанных на tcl.