Существуют различные версии для Python 2 и Python 3 и для импорта должен быть заисан по-разному(Tkinter/tkinter).

Ограничения:

Tkinter основан на инструментарии Tk, который поддерживает несколько устаревший вид виджетов. По умолчанию, он не поддерживает некоторые сложные виджеты, например, деревья или окна с закладками и др.

Если количество виджетов становится больше чем позволяет отобразить окно с максимальными размерами доступными для Вашего экрана монитора, то возникает проблема доступа к скрытой области окна.

Асинхронные события происходят независимо друг от друга (не синхронизированно друг с другом), полностью в одном потоке.

В отличие от многопоточности, в асинхронных программах программист контролирует, когда и как происходит произвольное вытеснение

В Python 3.7 асинхронные операции предоставляются модулем **asyncio**.

***Awaiting*** - код, который сам освобождает свои ресурсы для других частей системы.

Любой объект, который можно ожидать прерывание своего процесса выполнения, называется awaitable.

Ключевое слово **await** приостанавливает выполнение текущей подпрограммы (coroutine) и вызывает указанное ожидание awaitable.

В Python 3.7 есть три ожидаемых объекта (awaitable) – **coroutine**, **task** и **future**.

**Coroutine** в asyncio – это любая функция Python, в определении которой указан префикс **async**.

**task** в **asyncio** – это объект, который оборачивает **coroutine**, предоставляя методы для контроля ее выполнения и запроса ее статуса. **task** может быть создан с помощью **asyncio.create\_task()** или **asyncio.gather()**.

**future** в **asyncio** – это низкоуровневый объект, который выполняет роль заполнителя для данных, которые еще не были рассчитаны или получены. Он может обеспечить пустую структуру для последующего заполнения данными и механизм обратного вызова, который срабатывает, когда данные готовы.

В **asyncio** event loop (цикл обработки событий) управляет планированием и передачей ожидаемых объектов. **event loop**требуется для использования **awaitables**

Подпрограмма **asyncio.sleep(delay)** блокируется на секунды задержки. Это используется для имитации блокировки ввода-вывода.

Канонической точкой входа в программу **asyncio** является **asyncio.run(main())**, где main () – подпрограмма (coroutine) верхнего уровня.

Задача – это awaitable (ожидаемый) объект, который оборачивается вокруг подпрограммы (coroutine).

Задачи имеют несколько полезных методов для управления подпрограммами (coroutine). В частности, вы можете запросить отмену задачи, вызвав метод **.cancel()**. Задача будет запланирована для отмены в следующем проходе цикла событий.

Объекты awaitable могут быть собраны в группу, с помощью команды **asyncio.gather(awaitables)**.

**Asyncio.gather()** возвращает объект **awaitable**, представляющее собранные awaitable значения.

Особенности: позиционируется для быстрого написания GUI-приложений.