	Threading	**Processing**
	• Потоки можно сравнить с мини-процессами, которые "живут" внутри процессов • Делят общее пространство памяти и имеют доступ к общим глобальным переменным • Два потока не могут исполнять код одновременно в одной программе	• Создаются ОС для запуска программ • Процессы включают в себя потоки • Два процесса могут исполнять код одновременно в одной программе
Достоинства	Позволяет программе оставаться отзывчивой, пока один поток ожидает ввода, а другой одновременно запускает графический интерфейс.	Отдельное пространство памяти у каждого процесса (безопасность)
	Все потоки процесса имеют доступ к его глобальным переменным. Если глобальная переменная изменяется в одном потоке, она видна и другим потокам. Поток также может иметь свои собственные локальные переменные.	Предотвращает ограничения GIL(Python Global Interpreter Lock — это своеобразная блокировка, позволяющая только одному потоку управлять интерпретатором Python) для cPython
	Запуск нескольких потоков в программе рационализирует использование процессора, поскольку время простоя процессора уменьшается.	Процессы дочерних процессов можно прервать/уничтожить
	Повышает скорость вычислений, поскольку каждое ядро или процессор обрабатывает отдельные потоки одновременно.	Сетевые механизмы хорошо приспособлены к IPC (межпроцессное взаимодействие), поскольку они работают с процессами, выполняющимися на различных узлах сети, а не на одном и том же узле.

		Модуль multiprocessing предоставляет механизмы, пригодные для осуществления IPC по сети.
		Нагрузка распределяется между несколькими процессорами, что повышает надежность. Если один процессор выходит из строя, его отказ может немного замедлить скорость системы, но система будет работать бесперебойно.
Недостатки	Не подходит для однопроцессорной системы — многопоточность затрудняет достижение производительности с точки зрения скорости вычислений в однопроцессорной системе по сравнению с производительностью в многопроцессорной системе.	Увеличение объема памяти
	Проблема безопасности — все потоки в программе совместно используют одни и те же данные, следовательно, всегда существует проблема безопасности, потому что любой неизвестный поток может изменить данные.	Операционная система защищает процессы от взаимного влияния. Процессы, нуждающиеся в обмене данными, должны явно организовать такой обмен посредством механизмов межпроцессного взаимодействия
	Требуется синхронизация — синхронизация требуется, чтобы избежать взаимного исключения. Это приводит к увеличению использования памяти и процессора.	Передача информации между процессами происходит медленнее, чем между потоками, т.к. несколько процессов не могут использовать одно пространство памяти. В Python передача информации между процессами происходит

		методом "пиклинга" (процесс преобразования объекта в поток байтов).
	Не прерывается/уничтожается	Процессы более энергозатратны, чем потоки, потому что открытие/закрытие процессов занимает больше времени
	Даже хотя некий процесс может содержать множество потоков, отдельная недопустимая операция в пределах одного потока может отрицательно сказываться на имеющейся обработке всех прочих потоков в этом процессе и может вызывать в результате крушение всей программы целиком.	
	Разделение одних и тех же ресурсов может выступать неким преимуществом, для совместного использования ресурсов также требуется аккуратное рассмотрение подробностей с тем, чтобы совместные данные вычислялись правильно и их обработка была корректной. Интуитивно непонятные проблемы, которые могут быть вызваны небрежной координацией потоков включают в свой состав взаимные блокировки, зависания и состояние конкуренции	