# ✅**Основы CI/CD, Git:**

* ✅Waterfall.
* ✅Agile.
* ✅DevOps.
* ✅Основы CI/CD: задачи, принципы, этапы, платформы.
* ✅Системы контроля версий. Git, архитектура, настройка, команды

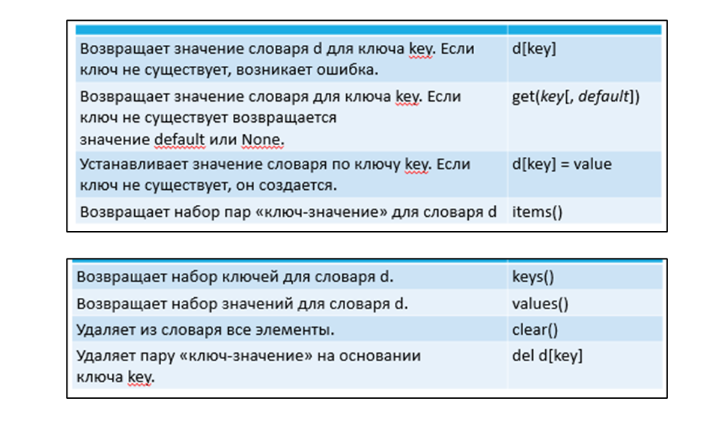
# ✅**Docker:**

* ✅Развертывание приложений.
* ✅Docker.
* ✅Назначение.
* ✅Архитектура.
* ✅ Основные команды.
* ✅ Dockerfile.
* ✅ Docker Compose.
* ✅ Хранилища данных.
* ✅Docker Networking.

# **Общая характеристика языка Python:**

* ✅История Python.
* ✅Установка, реализации, справка.
* ✅IDE для Python.
* ✅Структура программы.
* ✅Синтаксис Python.(макс длина стркои – 79, комментарии - 72)
* ✅Линтеры(flake8 and pylint) - **это** инструмент, который анализирует исходный код, чтобы отметить ошибки программирования, ошибки, стилистические ошибки и подозрительные конструкции.(ctrl + alt + L)
* ✅Переменные
* ✅ Управление памятью.(алгоритм сборщика мусора на подсчете ссылок)

# **Базовые типы данных и операции:**

* ✅Классификация типов данных.
* **Скалярные (неделимые):**
* Числа (целое, вещественное).
* Логический тип.
* NoneType.
* **Структурированные (составные) / коллекции:**
* Последовательности: строка, список, кортеж, числовой диапазон, bytes и bytearray.
* Отображения: словарь.
* Множества.
* ✅ (изменяемые(mutable): list, dict , set, bytearray – неизменяемые(unmutable) – int, float, decimal.Decimal, complex, str, tuple, frozensetm bytes)
* 
* ✅Логический тип. - bool
* ✅NoneType. – None аналог NULL
* ✅Числа.
* ✅Коллекции:
  + последовательность,
  + списки,
  + строки,
  + кортежи,
  + отображения,
  + множества,
  + числовой диапазон,
  + Bytearray – изменяемая последовательность байтов
  + Bytes – неизменяемая последовательность байт
* ✅Функции и модули.
* ✅Общие функции объектов.
* ✅Поверхностное и глубокое копирование.
* ✅ Взаимное преобразование.
* ✅ Приоритет операций.

# **Управляющие конструкции, функции:**

* ✅Условный оператор.
* ✅ Циклы:
  + while,
  + for.
* ✅Перемещение по коллекциям.
* ✅ Коллекционные включения.
* ✅ Прерывание и продолжение циклов.
* ✅ Функции.
* ✅ Вложенные функции.
* ✅ Замыкание — **это функция,** которая ***динамически* генерируется другой функцией,** и они обе могут изменяться и запоминать значения переменных, которые были созданы вне функции.
* ✅ Области видимости(LEGB)
* ✅ Параметры функции: позиционные и ключевые.
* ✅ Возврат значений.
* ✅ Функция как аргумент функции.
* ✅ Анонимные лямбда–функции.
* ✅ Функции как метод.
* ✅ Документирование кода.

# **Итераторы, генераторы, декораторы, обработка исключений:**

* ✅Назначение,
  + Синтаксис
  + и функции итераторов.
* ✅Назначение,
  + синтаксис
  + и функции генераторов,
  + генераторные выражения.
* ✅Назначение,
  + Синтаксис
  + и функции декораторов,
  + декораторы функций,
  + декораторы классов,
  + параметризированные декораторы.
* ✅Категории ошибок и их обработка.
* ✅Понятие исключения.
* ✅Конструкция try.
* ✅Возбуждение исключений raise. – явно raise ValueError(“”)
* ✅Утверждения assert. – отладка asser ==10
* ✅Определение собственного типа исключения

# **Объектно-ориентированное программирование и классы:**

* ✅Основные понятия и принципы ООП.
  + Класс.
  + Специальные методы.
  + Aтрибуты.
  + Наследование и полиморфизм.
  + Специальные (магические) методы.
  + Композиция(агрегирование).
  + Геттеры и сеттеры.
  + Свойства.
  + Методы.
  + Множественное наследование.
  + Метаклассы – опр поведение другого класса
  + Примеси (миксины – функцион возможности классов – однако запрещает создавать свои экземпляры – юзается при множеств наследовании) - DataMixins

# **Файлы и сериализация данных:**

* ✅Файловый объект,
* ✅менеджер контекста,
* ✅двоичные и текстовые файлы.
* ✅Основные свойства и методы файлов.
* ✅Сериализация и десериализация.
* ✅Модуль pickle,
* ✅библиотека cloudpickle.
  + CSV- ,
  + JSON- и YAML-файлы.
* ✅Модуль – shelve.
* ✅Работа с папками и файлами с помощью модуля OS.
* ✅Модуль zipfile.

# ✅**Модули, пакеты, стандартная библиотека:**

* ✅Модули.
* ✅Пакеты.
* ✅Классификация.
* ✅Подключение и использование модулей и пакетов.
* ✅Специальные атрибуты.
* ✅Стандартная библиотека Python: math, random, statistics, дата/время модули, locale, calendar, time, pytz, zoneinfo, модули для работы с платформой и операционной системой.
* ✅Регулярные выражения.
* ✅Визуализация данных: matplotlib.

# **10)** ✅**Сетевое и веб-программирование:**

* ✅Модель OSI.
* 7 лвл:  
  приложения – 7
* Представления – 6
* Сеансовый – 5
* Сетевой – 4
* Транспортный - 3
* Канальный – 2
* Физический -1
* ✅Понятие сетевого протокола.
* Формат данных, правила их передачи между двумя узлами на одном уровне называются **сетевым протоколом**.
* ✅Стек протоколов TCP/IP.
* Прикладной- 4
* Транспортный - 3
* Межсетевой - 2
* Канальный -1
* ✅Коммуникация в TCP/IP.
* Их функции доступны через соответствующее API.
* **передача информации по протоколу TCP/IP** состоит из четырех этапов:
* протокол ТСР: разбиение информации на нумерованные пакеты;
* протокол IP: передача пакетов получателю;
* протокол TCP на стороне получателя: проверка комплектности полученных пакетов;
* протокол TCP: восстановление искомой информации.
* ✅Протокол IP.
* ✅Протоколы UDP
* ✅и TCP.
* ✅URI,
* ✅URL,
* ✅URN,
* ✅DNS.
* ✅Протокол HTTP,
* ✅HTTPS.
* ✅Django.

# **11)** ✅**Параллельное программирование:**

* ✅Процесс,
* **Исполняемый экземпляр какой-либо программы**, имеет **собственное ядро** и следовательно **выделенную ему память**
* ✅параллелизм,
* Конкурентность - задачи совместно
* Параллелизм - задачи отдельно
* ✅multiprocessing,
* Модуль многопроцессорной обработки данных **позволяет организовать параллелизм вычислений** за счет создания подпроцессов и использования ядра процессора вместо потоков.
* ✅очереди,
* ✅каналы данных,
* Каналы данных (Data Channels) являются **механизмом обмена данными между различными параллельными компонентами программы**
* Coroutine (корутины), или сопрограммы — это **блоки кода, которые работают асинхронно, то есть по очереди**. которые могут приостанавливать свое выполнение и передавать управление другим сопрограммам, не блокируя поток выполнения. Они обозначаются ключевым словом `async` перед объявлением функции. Внутри сопрограммы можно использовать ключевое слово `await` для ожидания завершения других асинхронных операций.

**2. \*\*Событийный цикл (event loop)\*\*:**

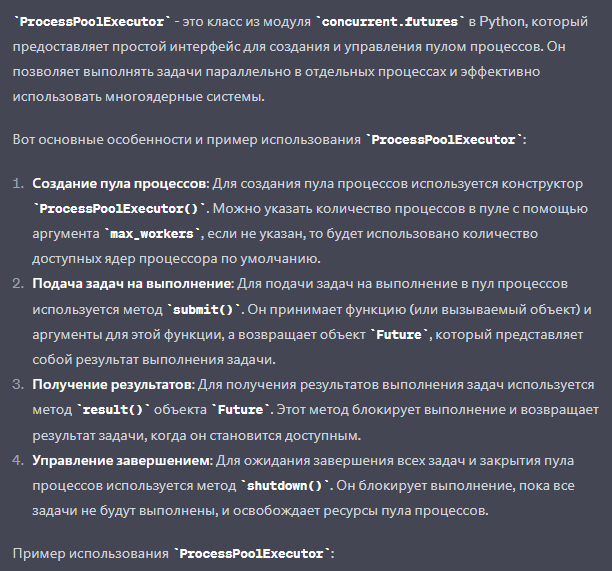
Это центральный компонент `asyncio`, который обрабатывает события и управляет выполнением сопрограмм. Он отслеживает активные сопрограммы, вызывает их выполнение, ожидает завершения операций ввода-вывода и передает управление другим сопрограммам.

**3. \*\*Асинхронные операции (async operations)\*\*:**

 Это операции, которые выполняются асинхронно в рамках `asyncio`. Например, операции ввода-вывода (чтение/запись файлов, сетевые запросы), ожидание задержки (с помощью `asyncio.sleep()`) и другие асинхронные вызовы.

**4. \*\*Асинхронные объекты (async objects)\*\*:**

 Это объекты, которые представляют собой асинхронные операции или ресурсы, их можно использовать в сопрограммах для выполнения асинхронных операций. Например, асинхронные сокеты, асинхронные файлы и т.д.

* ✅Pool.
* Пул (pool) в программировании - это **контейнер для повторно используемых ресурсов**. Он используется для экономии на создании ресурса. Идея в том, что после завершения использования ресурса он не уничтожается - а возвращается в пул, откуда его можно быстро достать для повторного использования.
* В языке программирования Python модуль `multiprocessing` предоставляет класс `Pool`, который представляет собой пул процессов и упрощает параллельное выполнение задач. **Пул процессов позволяет одновременно запускать несколько процессов для выполнения функций или методов.**
* ✅Поток,
* НАИМЕНЬШАЯ ЕДИНИЦА ОБРАБОТКИ ОС
* ✅многопоточность,
* ✅threading.
* ✅Параллелизм и конкурентность.
* ✅GIL.
* ✅Асинхронность – asyncio.
* ✅ ProcessPoolExecutor.
* 
* 