Anaconda and **miniconda** are software distributions that are widely used in data science to simplify package management and deployment.

In this Answer, we will cover the differences between the two applications.

Differences

Both Anaconda and Miniconda uses Conda as the package manager.

There are essentially two main differences:

1. **Number of packages:** Anaconda comes with over 150 data science packages, whereas miniconda comes with only a handful.
2. **Interface:** Anaconda has a graphical user interface (GUI) called the Navigator, while miniconda has a command-line interface.

In other words, miniconda is a *mini* version of Anaconda. Miniconda ships with just the repository management system and a few packages. Whereas, with Anaconda, you have the distribution of some 150 built-in packages.

---

2 - https://github.com/zedr/clean-code-python

---

3 - A framework exists to provide a base from which you can build your next project with great success. Providing an architecture, abstraction, and some initial code towards your goals—it can provide a way to save time and strengthen a project throughout every stage of development.

From this strong base, engineers have an opportunity to extend and develop in a direction that maintains good working practices while keeping close to the structures and patterns prescribed by the framework.

One point of confusion that often exists, even among developers, is the difference between a library and a framework.

A reasonable working definition we can apply is that a library should provide one or more collections of functionality and helpers to link your

project against. A framework, in contrast, should provide a skeleton of code for you to build your application from.

Most frameworks are likely to include one or more libraries, but they will also prescribe or at least suggest some architectural patterns to assist with developing a robust and reliable application.

Advantages of Using A Framework for Application Development:

1- Simplifies Many Tasks and Challenges

For complex, high-traffic, or exceptionally rich systems and applications, a framework could be considered all but essential. Laying out strong architectural foundations creates a well thought out application that is easy to build on and maintain without losing application performance or control of its structure.

2- Focused Application Development

3- Standardized Coding Practices.

Framework can bring is its ability to create a well-defined structure to follow throughout the lifetime of a project.

4- Testing and Debugging

5- Community

Perhaps one of the most under-appreciated advantages of developing an application from a highly capable framework is the community of developers, engineers, and users available to support it.

Disadvantages of Using a Framework for Application Development

1- Framework Limitations

2- Unnecessary Extras

---

4 => Most popular 5 processors in laptops and 5 in mobiles, and why?

For PC

1. AMD Ryzen 9 5900HX
2. Intel Core I9-9900K
3. AMD Ryzen 7 5800H
4. AMD Ryzen 7 4800H
5. AMD Ryzen 9 4900H
6. AMD Ryzen 9 4900HS
7. Intel Core I9-9980HK
8. Intel Xeon W-10885M
9. Intel Core I7-10870H
10. AMD Ryzen 7 4800HS

=For mobiles

1. Apple A15 Bionic
2. Dimensity 9000 Plus
3. Snapdragon 8 Plus Gen 1
4. Dimensity 9000
5. Snapdragon 8 Gen 1
6. Apple A14 Bionic
7. Snapdragon 888 Plus
8. Exynos 2200
9. Snapdragon 888
10. Dimensity 8100

---

5 => Here are some tips for optimizing recursive code:

1. Memoization: Memoization is the technique of caching the results of expensive function calls and returning the cached result when the same inputs occur again. This can be particularly useful in recursive functions where the same inputs are used repeatedly. By

memoizing intermediate results, you can avoid redundant computations and speed up your code.

2. Tail recursion: Tail recursion is a special case of recursion where the recursive call is the last operation performed in the function. This means that the function does not need to keep track of any state between recursive calls, and can be optimized by the compiler into an iterative loop. Some programming languages, such as Scheme and Scala, automatically optimize tail-recursive functions.
3. Divide and conquer: Recursive algorithms that use a divide-and-conquer approach can often be faster than their iterative counterparts, especially when working with large data sets. For example, the quicksort algorithm is a recursive algorithm that uses divide and conquer to sort an array, and is often faster than alternative iterative sorting algorithms.
4. Avoid unnecessary work: Recursive functions can sometimes perform unnecessary work, especially if they explore all possible paths through a problem space. By using heuristics or pruning techniques, you can often eliminate unnecessary branches of the recursion and speed up your code.

- An unordered_set is implemented using a hash table where keys are hashed into indices of a hash table so that the insertion is always randomized. All operations on the unordered_set takes constant time O(1) on an average which can go up to linear time O(n) in worst case which depends on the internally used hash function, but practically they perform very well and generally provide a constant time lookup operation.
- The unordered_set can contain key of any type – predefined or user-defined data structure but when we define the key of type user define the type, we need to specify our comparison function according to which keys will be compared.

https://poe.com/s/E6r1nX7mRYer7v4nGzeb

```
try:
    # some code that might raise an exception
    result = 1/0
except ZeroDivisionError as error:
    print("Error type:", type(error).__name__)
    print("Error message:", str(error))
```