1. What is miniconda and what is the difference between it and Anaconda

**What Is Miniconda?**

Miniconda is a well-designed, free, smaller version of Anaconda that installs Conda (a package manager feature), including the other packages, python language, and other handy packages like pip, etc.

it includes all the Anaconda robust features except those 200 plus pre-installed applications.

**What Is Anaconda?**

Anaconda is a comprehensive distribution of python and R programming systems that makes **computing, data processing,** and **package managing** a lot easier for users.

2.

Choose Anaconda If:

- You need the 150 installments downloaded with the file and don't want to download each of the individual packages.
- You have enough disc space and time to download 750 plus files.
- You are new to python and Conda commands.

Choose Miniconda If:

- Only need to download some of the specifically required files or don't mind downloading 150 files one by one.
- If you don't have time and space for the extra provided programs noting that you won't need most of the 700 plus packages.

- You want swift, faster commands on Conda and don't require the Anaconda distribution.
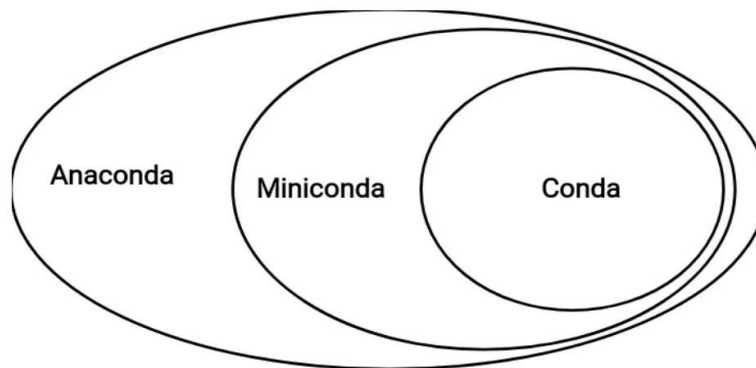


What is framework? why we use it? advantages and disadvantages

 framework is a structure that you can build software on. It serves as a foundation, so you're not starting entirely from scratch. Frameworks are typically associated with a specific programming language and are suited to different types of tasks.

Why do we use frameworks?

Using frameworks saves time and reduces the risk of errors. You don't need to write everything from the ground up, so there's less chance of introducing errors. Plus, frameworks have already been tested, so there's less to worry about. Other advantages include:

- More secure code
- Simpler testing and debugging
- Avoiding duplicate code
- Clean and easily adaptable code
- Able to focus on writing code specific to the project

- Can be extended

## PROS
### Efficiency

Jobs that generally would take you hours and hundreds of lines of code to compose, can now be done in minutes with pre-built functions. Development becomes a lot easier, so if it's much easier it's quicker, and subsequently effective.

### Security

An extensively utilized framework has big security applications. The big benefit is the neighborhood behind it, where users end up being long-lasting testers. If you find a vulnerability or a security hole, you can go to the framework's web site and let the team understand so they can fix it.

### Support

As any other distributed tool, a framework generally includes documents, a support group, or huge community online forums where you can acquire quick responses.

## Cons

### You learn the framework, not the language

I believe this to be the significant issue. If you're making use of a framework and you know very little about the language behind it, you will certainly find out the framework and not the language itself. The method you code jQuery is different from the way you code javascript. Simple put, if you know jQuery, it doesn't imply you understand javascript.
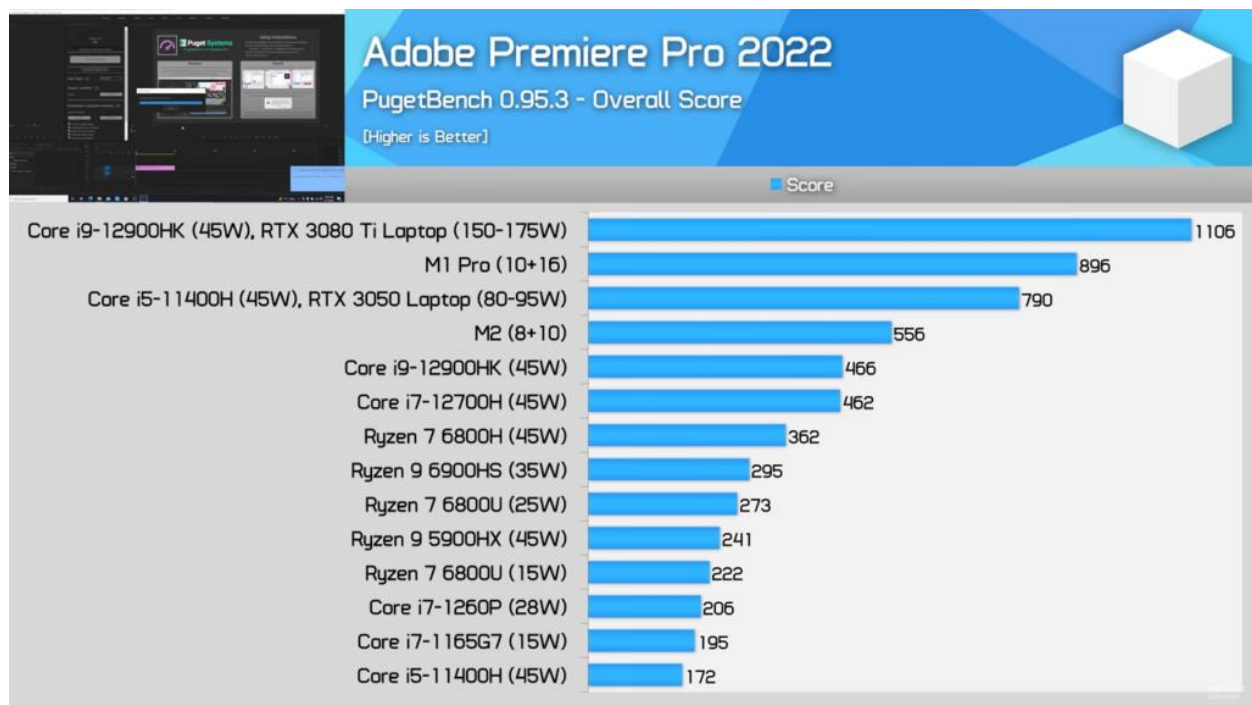
## Restriction

The framework's core behaviour cannot be modified, indicating that when you utilize a framework, you are required to respect its limitations and work the way it is required. Ensure you choose a framework that fits your needs.

## Code is public

Since the framework is readily available to everyone, it is likewise offered to people with bad intentions. It can be studied in order to know how things work and to discover flaws that can be utilized against you.

## Most popular 5 processors in laptops and 5 in mobiles? and why?



**Adobe Premiere Pro 2022**
PugetBench 0.95.3 - Overall Score
[Higher is Better]

■ Score

| Processor | Score |
|---|---|
| Core i9-12900HK (45W), RTX 3080 Ti Laptop (150-175W) | 1106 |
| M1 Pro (10+16) | 896 |
| Core i5-11400H (45W), RTX 3050 Laptop (80-95W) | 790 |
| M2 (8+10) | 556 |
| Core i9-12900HK (45W) | 466 |
| Core i7-12700H (45W) | 462 |
| Ryzen 7 6800H (45W) | 362 |
| Ryzen 9 6900HS (35W) | 295 |
| Ryzen 7 6800U (25W) | 273 |
| Ryzen 9 5900HX (45W) | 241 |
| Ryzen 7 6800U (15W) | 222 |
| Core i7-1260P (28W) | 206 |
| Core i7-1165G7 (15W) | 195 |
| Core i5-11400H (45W) | 172 |

| | Phone ∨ | Sell Gadgets ∨ | Buy Phone ∨ | Find New Gadget ∨ | |
|---|---|---|---|---|---|

Allow

| Rank | Processor Name | Phone | Centurion Mark | AnTuTu 8 | GeekBench 5* |
|---|---|---|---|---|---|
| #1 | Apple A15 Bionic | Apple iPhone 13 | | 801199 | 1722 / 4685 |
| #2 | Dimensity 9000 | Vivo X80 | | 1,049,081 | 1179/4189 |
| #3 | Snapdragon 8 Gen 1 | OnePlus 10 Pro | | 1,029,556 | 1259/4189 |
| #4 | Apple A14 Bionic | Apple iPhone 12 | 165 | 611843 | 1613 / 3909 |
| #5 | Snapdragon 888 Plus | Asus ROG Phone 5S | | 863552 | 1160 / 2834 |
| #6 | Exynos 2200 | Samsung Galaxy S22 Ultra | | 965,874 | 934/2925 |
| #7 | Snapdragon 888 | OnePlus 9 | 158 | 722907 | 1140 / 3745 |
| #8 | Dimensity 8100 | Realme GT Neo 3 | | 809443 | 1003/2607 |
| #9 | Exynos 2100 | Samsung Galaxy S21 | 156 | 698965 | 1085 / 3654 |
| #10 | Google Tensor | Google Pixel 6 Pro | | 676789 | 1056 / 2855 |

# what is hashtables ?

A hash table is a data structure that you can use to store data in key-value format with direct access to its items in constant time.

Hash tables are said to be *associative*, which means that for each key, data occurs at most once. Hash tables let us implement things like phone books or dictionaries; in them, we store the association between a value (like a dictionary definition of the word "lamp") and its key (the word "lamp" itself).

We can use hash tables to store, retrieve, and delete data uniquely based on their **unique key**.

# why we use hash tables in unordered list

The most valuable aspect of a hash table over other abstract data structures is its **speed** to perform insertion, deletion, and search operations. Hash tables can do them all in constant time.

For those unfamiliar with time complexity (big O notation), constant time is the fastest possible time complexity. Hash tables can perform nearly all methods (except *list*) very fast in O(1) time.

| Algorithm | Average | Worst case |
|-----------|---------|------------|
| List | O(n) | O(n) |
| Search | O(1) | O(n) |
| Insert | O(1) | O(n) |
| Delete | O(1) | O(n) |

Because of this efficiency, you'll find hash tables to be pretty dang useful for many use cases. And if you look carefully, you'll notice that they're actually implemented in a variety of places throughout your tools like your databases, caches, data-fetching libraries, and so on.

how to print the error type for the user in try catch

```python
try:
  print(x)
except:
  print("An exception occurred")
```

```python
In [1]: while True:
            try:
                x = int(input("Please enter a number: "))
                break
            except ValueError:
                print("Oops!  That was no valid number.  Try again...")

        Please enter a number: k
        Oops!  That was no valid number.  Try again...
        Please enter a number: 6
```