Execution Efficiency

- The speed and ease with which a particular program can execute. Also the ability of a program to take advantage of the hardware and the speed of runtime execution..
- Execution Efficiency is important in language design because it affects the speed at which a program executes.
- A positive example of Execution Efficiency in a programming language is how Java programs compile faster than C programs. This is because java programs run on a virtual machine.
- A negative example of Execution Efficiency in a programming language is how Python has dynamic typing, meaning functions can take different data types at runtime. This is more inefficient than specifying the data types like in C++.

Programmer Efficiency, which encompasses Expressiveness and Maintainability

- The ability of a programmer or a development team to implement software in a programming language. How efficient it is for a programmer to express complex ideas in the language and write code. A sub principle is maintenance efficiency. Maintenance efficiency is how readable and simple to maintain the code is. Expressiveness is the ability to express concepts in a simple and direct manner.
- Programmer Efficiency is important in language design because it allows programs to be finished at a faster rate.
- A positive example of Programmer Efficiency in a programming language is the ability to increment a variable in C++ such as ‘number1++’ instead of typing out ‘number1 = number1 + 1;’. This reduces the amount of time a programmer has to spend writing a line of code; improving programmer efficiency.
- A negative example of Programmer Efficiency in a programming language is that C++ programs can be complicated to understand and debug.

Regularity, which encompasses Generality, Orthogonality and Uniformity

- Regularity is how well a certain language integrates the features of that language. It shouldn't have any crazy restrictions and it should be consistent and easy to use. Regularity issues can be broken up into a few subcategories: Generality, Orthogonality, and Uniformity.
- Generality is combining the ideas of different constructs of a language into simpler concepts. This can include ideas like functions, classes, and structures in C++ because these operate in a similar way. In C++, functions are not general. There are no local functions.
- Orthogonality allows for the use of a small set of primitive constructs in a language that can be combined in a relatively small number of ways. This is closely related to simplicity because the more orthogonal a language is, the fewer rules there are to remember. In the Vax programming language, a single add instruction can have arbitrary operands. This is a positive example of orthogonality. A poor example of orthogonality is in the IBM assembly language. There are different ways to add memory to a register in IBM assembly.
- Uniformity in a language means that similar features of that language should look similar and behave similarly. This is important so programming is easy for the user to learn and pick up quickly. A good example of this is C++. It is easy to learn and is taught in intro programming courses. Pascal is not uniform with its treatment of while and repeat loops.

Reliability and Security

- Reliability is the ability to consistently perform according to its specifications. Reliability of programming languages deals with things such as crashing if a user inputs weird data. This is important because users expect to have reliable performance when using a certain language. If errors and crashes keep occurring then the language is not reliable and can slow down user performance. Reliability is sometimes traded for speed. A good reliable language is PHP due to it being open source and easy to work with. C can sometimes be less reliable due to security flaws that can be exploited.
- Security in language design makes sure programs don't do damage that isn't expected. This is important because you don't want a program you are writing to hurt your

computer. Java is an example of good security because it runs on a virtual machine (sandbox mode).so it doesn't hurt your computer. C on the other hand compiles programs down to machine code which can harm the computer you are working on.

Extensibility

- Implementation that takes future growth into consideration.
- Ability to adapt to future change while having minimal or no change to the system's internal structure and data flow.
- Not everything can be planned in advance, the goal is to have a light framework that can be adjusted over time so that fewer problems are encountered.
- Programming languages are constantly being updated to support new programming practices and solutions to known errors that are currently in the system.
- The changes from Python 2 to Python 3 is an example that can be seen as good or bad
- Python 3 has received many changes that have been a problem for Python 2 and is significantly better.
- Python 3 is not able to use code from Python 2 as it has been significantly changed to the point where the styles of programming are too different.

Portability

- The ability to transfer software between environments or computers without it deviating from the original purpose or functionality.
- Being able to transfer software across different environment is essential, because it means that one is able to use the same software on various machines without have any problems.
- An example of programming languages making use of portability are libraries.
    - Libraries are able to be used across many environments and still provide the same functionalities needed for a programming language.
- An example of programming language where portability fails does not necessarily rely on the programming language itself. There are many times where portability is a problem.

- Portability itself is not necessarily the problem as it relies heavily on the hardware components as well software too.
- There are some requirements that must be met before Portability can be possible.

2. **Security and Efficiency**

Security and Efficiency can both conflict in the topics explained. One significant example would be the programming language C. C is a lower level programming language that is still able to work efficiently, which is why it is still widely used today. However, although it is efficient to use, there are many security problems with it. Since it is an older language, it does not have many of the protections of modern day programming languages. Also there are less checks in C, which can cause an overflow.

3. **Parallelized Data Mining Software / Big Data**

Design features that would be considered critical when choosing a language for a parallelized data mining application would be the ability to take advantage of multiple cores and multiple threads. This means that imperative programming languages such as Java and C++ would not be great for parallel execution. Parallelized data mining programs and algorithms are still relatively new. As it stands now, C++ for example, still only allows parallelization of individual structures such as cycles. As our knowledge and experience stand now, C++ still seems like the most viable programming language to design a Parallelized Data Mining application.