What makes your friends reliable?

- We have vested interests in each others success
- We have enough personal information for mutual assured destruction in the event of a fallout
- We have knowledge of each others thought processes enough to communicate effectively

What makes a system reliable?

- Formal verification and proofs. Thus using tools which perform some of these actions automatically (ie - functional programming languages) can help
- Systems designed and planned out thoroughly by competent engineers before being constructed by competent developers are generally more reliable as mistakes would have to be made in order for it to fail and competent individuals generally make fewer mistakes
- Manual overrides and openness of system. For example when using linux I'm able to diagnose any problems which occur (largely resulting from misuse) so that I can resolve it, however for other operating systems this isn't a given with the only solution being to wait in customer support for hours. This can also apply to hardware where one could for example fix a vehicle using only a screwdriver or something of that nature in some situations if needed

Can systems be programmed to achieve a specified degree of reliability?

- Yes, thankfully we have languages which allow developers to make guarantees such as no runtime type-errors. This allows us to make guarantees about the way a program will behave. However there is a certain level of reliability which is entirely outside the scope of the system which is not something that can be guaranteed. For example we cannot guarantee that a desktop application will never crash because the user could for example run out of memory, causing the program to stop working.

How do we make decisions based on information which we do not consider 100% reliable?

- Generally it's best to have safeguards and verifications so that the system will fail-safe instead of fail-deadly
- For example when allocating memory in C even though in nearly all circumstances malloc() will be successful it's good practice to verify this

Can we program systems to learn how to make good (perfect?) decisions in the face of less than perfectly reliable information?

- Even with imperfect information there is always a best decision
- Machines should never assume the information is not reliable especially when coming from a human operator lest we see something like HAL 9000 from 2001 space odyssey where it didn't trust the human operators, believing them to be weak links in the mission
- We have systems designed to account for error and variability, many of which operate by using multiple sensor readings and data sources to verify and factor into calculations which ultimately give a more reliable and accurate value. A great example of this is a kalman filter which when applied to an IMU combines readings from a gyroscope (which

drifts due to earth’s orbit) and an accelerometer (which has a high error rate) in order to produce more accurate data.