What is the difference between statically typed and dynamically typed languages?

Statically typed languages

A language is statically typed if the type of a variable is known at compile time. For some languages this means that you as the programmer must specify what type each variable is (e.g.: Java, C, C++); other languages offer some form of *type inference*, the capability of the type system to deduce the type of a variable (e.g.: OCaml, Haskell, Scala, Kotlin)

The main advantage here is that all kinds of checking can be done by the compiler, and therefore a lot of trivial bugs are caught at a very early stage.

Examples: C, C++, Java, Rust, Go, Scala

Dynamically typed languages

A language is dynamically typed if the type is associated with run-time values, and not named variables/fields/etc. This means that you as a programmer can write a little quicker because you do not have to specify types every time (unless using a statically-typed language with *type inference*).

Examples: Perl, Ruby, Python, PHP, JavaScript

Most scripting languages have this feature as there is no compiler to do static type-checking anyway, but you may find yourself searching for a bug that is due to the interpreter misinterpreting the type of a variable. Luckily, scripts tend to be small so bugs have not so many places to hide.

Most dynamically typed languages do allow you to provide type information, but do not require it. One language that is currently being developed, Rascal, takes a hybrid approach allowing dynamic typing within functions but enforcing static typing for the function signature.