**Difference Between Statically & Dynamically Typed Languages**

## Statically-Typed Languages

A language is **statically-typed** if the type of a variable is known at **compile-time** instead of at run-time. Common examples of statically-typed languages include Java, C, C++, FORTRAN, Pascal and Scala.

In Statically typed languages, once a variable has been declared with a type, it cannot ever be assigned to some other variable of different type and doing so will raise a type error at compile-time(some IDE’s generally shows a Red Cross mark denoting the error).

## **Java Example**

Statically-typed languages require you to declare the data types of your variables before you use them

int data;  
data = 50;  
data = “Hello World!”; // causes an compilation error

Note: the statement above (which binds an integer value, and then binds a string value to the same variable name **data**) is illegal. But in a dynamically-typed language this sequence of statements is perfectly fine.

## Advantages:

* A large class of errors are caught in the early stage of development process.
* Static typing usually results in compiled code that executes more quickly because when the compiler knows the exact data types that are in use, it can produce optimized machine code (i.e. faster and/or using less memory).

# Dynamically type

# A language is ****dynamically-typed**** if the type of a variable is checked during ****run-time****. Common examples of dynamically-typed languages includes JavaScript, Objective-C, PHP, Python, Ruby, Lisp, and Tcl.

In Dynamically typed languages, variables are bound to objects at run-time by means of assignment statements, and it is possible to bind the same variables to objects of different types during the execution of the program.

Dynamic type checking typically results in less optimized code than static type checking. It also includes the possibility of run time type errors and forces run time checks to occur for every execution of the program (instead of just at compile-time).

## Python Example

Dynamically-typed languages do not require you to declare the data types of your variables before you use them

data = 10;  
data = “Hello World!”; // no error caused

The above statement assigns a new value to the same variable **data**of different data type than the one assigned earlier. The program will execute successfully without any error. This is characteristic to dynamic-typed programming languages.

## Advantages:

* Implementations of dynamically type-checked languages generally associate each run time object with a type tag (i.e., a reference to a type) containing its type information. This run-time type information (RTTI) can also be used to implement dynamic dispatch, late binding, down-casting, reflection, and similar features.
* The absence of a separate compilation step means that you don’t have to wait for the compiler to finish before you can test your code changes. This makes the debug cycle much shorter and less cumbersome.