Loosely Typed vs Strongly Typed Languages

Loosely Typed (Weakly Typed)

- Types are more flexible.
- Implicit type conversions (coercions) often happen automatically.

Strongly Typed

- Enforces strict rules about type usage.
- Rarely allows implicit conversions.

Key Differences

Feature / Behavior	Loosely Typed Languages	Strongly Typed Languages
Type Enforcement	Flexible, implicit type conversions allowed	Strict, explicit type matching enforced
Example Languages	JavaScript, PHP, Perl, C, C++	Java, C#, Rust, Haskell, Python, Ruby
Variable Decl.	Can hold different types / unsafe casts	Variables must match declared type
Implicit Conversion	Common (e.g., string + number → string)	Rare or disallowed
Error Detection	More runtime errors (caught late)	More compile/runtime errors (safer)
Code Safety	Less predictable	More predictable

Examples

Loosely Typed (JavaScript)

```
let value = 5; // number
value = "hello"; // now a string, no error
console.log(5 + "5"); // "55" (string concatenation)
```

Strongly Typed (Java)

```
int value = 5;
value = "hello"; // Compile-time error
System.out.println(5 + "5"); // "55" but requires explicit handling
```

Strongly Typed (Python)

```
x = 5
print(x) # 5
x = "hello"
print(x) # hello
print("5" + 5) # TypeError (no implicit coercion)
```

Loosely Typed (C++)

```
#include <iostream>
using namespace std;
int main() {
    int x = 5;
    double y = 2.5;
    int z = x + y; // implicit conversion (y → int)
    cout << z; // prints 7 (data loss!)
}
```

Strongly Typed (Rust)

```rust
fn main() {
    let x: i32 = 5;
    let y: f64 = 2.5;
    let z = x + y; // Compile-time error (no implicit coercion)
    println!("{}", z);
}
```

Strongly Typed (Haskell)

```haskell
main = do
  let x = 5 :: Int
  let y = 2.5 :: Double
  print (x + y) -- Type error: no implicit conversion
```

Strongly Typed (C#)

```csharp
int x = 5;
x = "hello"; // Compile-time error
Console.WriteLine(5 + "5"); // "55" but requires string coercion rules
```

Loosely Typed (PHP)

```php
<?php
$x = 5;
$x = "hello";
echo $x; // hello
echo "5" + 5; // 10 (implicit coercion!)
?>
```

Loosely vs Strongly Typed

- **Strong typing**: Enforces strict type rules, catching errors early and improving reliability.
- **Loose typing**: Allows variables to change types, making coding faster and more flexible.
- **Best Fit**:
  - Strong typing → Large, complex, safety-critical systems.
  - Loose typing → Quick prototyping, small scripts, experimentation.

Statically Typed vs Dynamically Typed

Feature / Behavior	Statically Typed Languages	Dynamically Typed Languages
Type Binding	Types checked at compile time	Types checked at runtime
Variable Decl.	Must declare type explicitly (or via inference)	No need to declare types explicitly
Example Languages	Java, C, C++, Rust, Go, Haskell, C#	Python, JavaScript, Ruby, PHP
Error Detection	Many errors caught before execution	Errors appear only when that code runs
Flexibility	Less flexible, but safer	More flexible, can change types at runtime

---

Examples

Statically Typed (Go)

```go
package main
import "fmt"
func main() {
    var number int = 10
    // number = "hello" // Compile-time error
    fmt.Println(number)
}
```

Statically Typed (Java)

```java
int number = 10;
number = "hello"; // Compile-time error (type mismatch)
```

Statically Typed (C++)

```
int number = 10;
number = "hello"; // Compile-time error
```

Dynamically Typed (Python)

```
x = 10
print(x)
x = "hello"
print(x) # Allowed (type changes at runtime)
```

Dynamically Typed (JavaScript)

```
let x = 10;
x = "hello"; // Allowed (dynamic typing)
```

Visual Diagram: Type System Quadrants

```
             Strongly Typed            Loosely Typed
Statically   | Java, C#, Rust, Haskell | C, C++ (unsafe casts, coercion)
Typed        |                         |
             |                         |
Dynamically  | Python, Ruby            | JavaScript, PHP, Perl
Typed        |                         |
```

Summary

- **Loosely vs Strongly Typed** → Focuses on type enforcement.
- **Statically vs Dynamically Typed** → Focuses on when types are checked.