

Data types

Just a note that we could use some docs on:

- reference types
- value types (see also [User-defined value types aka structs](#) and I assume a struct keyword will be added too)
- null
- = vs. ==
- == vs. is
- is vs. isa
- typeof
- x.GetType()

== vs is vs isa

The "==" operation is deceptively simple. When a data structure has overridden the "Equals" method, .NET/Boo checks the values of one data structure against another and returns true if they are identical or false if they are not. By default, .NET provides "Equals" overloads for most of the builtin data-types like int, float, decimal, string, and etc. Using "==", two separate objects can be evaluated and it can be determined if, by value, they are equal or not.

The "isa" keyword determines if a variable is of a particular type. "isa" is useful in determining if a variable implements an interface, or derives from a common base class.

The "is" keyword serves a different, and somewhat more complex function. Sometimes you might want to go beyond the usual check of "are these two variables of equal **value?**" and see, "are these two variables actually **pointing to the same object?**" This is where "is" shines; it looks at the reference of two variables and determines if they both point to the same object!

This technique is normally used when you receive an object by poking a dictionary or invoking a method; if the identity of var1 and var2 is the same, then they are obviously equal--because they are the same object.

"==" example:

```
/*  
True,  
False,  
True!  
*/  
a = 0  
b = 0  
c = 4  
  

```

"isa" example:

```
/*
Prints out...
True,
True,
False!
*/
class Food:
    pass

class Sandwich(Food):
    pass

hamAndSwiss = Sandwich()

print hamAndSwiss isa Food
print hamAndSwiss isa Sandwich
print hamAndSwiss isa int
```

"is" example:

```
/*  
prints,  
True,  
False  
*/  
var1 = "hello, world!"  
var2 = var1  
var3 = "hey, carl!"  
print var1 is var2  
print var1 is var3 //2 different objects  
pointed at!
```