Slicing

A slicing operation is a simple way to extract a range of elements from a container. The boo compiler supports native slicing operations on lists, arrays and strings. Support for user defined slicing operations is planned but currently not implemented.

General Syntax

A slicing operation is applied to a container through the following syntax:

```
range = container[<firstIndexWanted> :
<firstIndexNotWanted> : <step>]
```

When **firstIndexWanted** is omitted it is assumed to be 0.

When **firstIndexNotWanted** is omitted it is assumed to be equals to `len(container)`.

When **step** is omitted it is assumed to be 1.

List Slicing

```
l = [1, 2, 3, 4]
assert l[0] == 1
assert l[0:1] == [1]
assert l[0:2] == [1, 2]
assert l[1:3] == [2, 3]
assert l[:] == [1, 2, 3, 4]  // easy way to
clone a list
```

Array Slicing

```
a = (1, 2, 3, 4)
assert a[0:1] == (1,)
assert a[:3] == (1, 2, 3)
assert a[:2] == (1, 2)
#assert a[::2] == (1, 3)      #Slicing step
not implemented yet
#assert a[-2:-1:-1] == (4, 3) #ditto
```

String Slicing

```
s = "bamboo"
assert "b" == s[0:1]
assert "boo" == s[3:]
assert "bo" == s[3:-1]
```

Differences between collection types

Consider the following example:

```
print([1, 2, 3].GetType()) // will print
"Boo.Lang.List"
print((4, 5, 6).GetType()) // will print
"System.Int32[]"
print(("1", "2", 3).GetType()) // will print
"System.Object[]"
print(("a", "b").GetType()) // will print
"System.String[]"
print(["foo", "bar"].GetType()) // will
print "Boo.Lang.List"
```

You must specify a parameter is a sliceable type when using it with methods:

```
// This code will NOT compile
class Test:
    def example(itens):
        i = 0
        itensLen = len(itens)
        while i < itensLen:
            print (itens[i])
            ++i

t = Test()
t.example([1, 2, 3, 4, 5])
```

the above code will result on compile time error *The type 'System.Object' does not support splicing'*. This is because the boo compiler cannot predict that you want to pass a collection to the method, and since the type *Object* is neither a collection or array, we got the error. To handle that, explicitly tell the compiler that you will pass a collection to the method:

```
// This code WILL compile
import System.Collections

class Test:
    def example(itens as IList):
        i = 0
        itensLen = len(itens)
        while i < itensLen:
            print (itens[i])
            ++i
t = Test()
t.example([1, 2, 3, 4, 5])
```