Imperative Asteroid with Objects

- The "Hello World" program…

Load system module

Sentence terminator

```
1    load system "io".¬
2    ¬
3    println("Hello World!").¬
```

Iteration

- 'while', 'for', 'loop' constructs are all supported

```
1    -- compute the factorial¬
2    ¬
3    load system "io".¬
4    load system "type".¬
5    ¬
6    function fact with n:%integer do¬
7    ··let val = 1.¬
8    ··while n > 1 do¬
9    ····let val = val*n.¬
10   ····let n = n-1.¬
11   ··end¬
12   ··return val.¬
13   end¬
14   ¬
15   let x = tointeger(input("Enter a positive integer: ")).¬
16   println ("The factorial of "+x+" is "+(fact x)).¬
```

Constraint pattern

Iteration

Assignment

Type conversion

Data Structures

- Built-in lists
  - [1,2,3]
- Built-in tuples
  - (x,y)
- Element access
  - a@i

```
1    -- the bubble sort
2    load system "io".
3
4    function bubblesort with a:%list do
5      ...
6    ..loop
7    ....let i = 0.
8    ....let swapped = false.
9    ....for i in 0 to len(a)-2 do
10   ......if a@(i+1) <= a@i do
11   ........let (a@i,a@(i+1)) = (a@(i+1),a@i).
12   ........let swapped = true.
13   ......end
14   ....end
15   ....if not swapped do
16   ......break.
17   ....end
18   ..end
19
20   ..return a.
21   end
22
23   let l = [6,5,3,1,8,7,2,4].
24   println("unsorted array: "+l).
25   println("sorted array: "+(bubblesort l)).
```

Structures & Objects

- Asteroid is object-based
- Bundle operations with data
- No inheritance
  - Construct new objects from other objects via object composition (and traits in the case of Rust)
- New languages with a full object-oriented type system are waning
  - Of the three "big" new languages (Rust, Go, Swift) only Swift supports OO, the others are object-based.

Structures

```
1    -- rectangle structure
2    load system "io".
3
4    structure Rectangle with
5      data xdim.
6      data ydim.
7    end
8
9    let r = Rectangle(4,2). -- default constructor
10   println ("Rectangle with x="+r@xdim+" and y="+r@ydim).
```

- Structures consist of 'data' fields and are associated with a 'default constructor
- Member access is via the '@' operator

Structure

```
1   -- rectangle structure
2   load system "io".
3   load system "type".
4
5   structure Rectangle with
6   ··data xdim.
7   ··data ydim.
8
9   ··-- member function
10  ··function area with none do
11  ····return this@xdim*this@ydim.
12  ··end
13
14  ··-- constructor
15  ··function __init__ with (x %if isscalar(x), y %if isscalar(y)) do
16  ····let this@xdim = x.
17  ····let this@ydim = y.
18  ··end
19  end
20
21  let r = Rectangle(4,2).
22  println ("The area of Rectangle ("+r@xdim+","+r@ydim+") is "+r@area()).
```

- Member functions
- Object identity is given with the 'this' keyword
- Member functions are called on objects with the '@' operator
  - E.g., r@area()

Types in Asteroid

- Asteroid has a set of **primitive data types**:
  - integer
  - real
  - string
  - boolean
- Asteroid arranges these data types in a **type hierarchy** in order to facilitate automatic type promotion:
  - boolean < integer < real < string

Types in Asteroid

- Asteroid supports two more data types:
  - list
  - tuple
- These are **structured data types** in that they can contain entities of other data types.
- Lists and tuples themselves are also embedded in type hierarchies, although very simple ones:
  - list < string
  - tuple < string
- That is, any list or tuple can be viewed as a string. This is very convenient for printing lists and tuples.

Types in Asteroid

- Finally, Asteroid supports one more type, namely the **none** type.
- The none type has a constant named conveniently 'none'.
- The empty pair of parentheses () can often be used as a convenient short-hand for the constant none.
- The none data type does not belong to any type hierarchy.

Assignments

- Explore the links in the github readme
- Do Assignment #1 – see BrightSpace