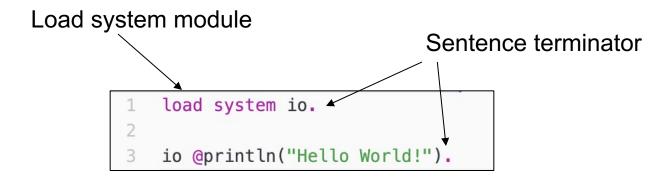
Imperative Asteroid

The "Hello World" program...



Iteration

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

```
-- compute the factorial
                                   Function argument
    load system io.
    load system type.
5
    function fact with n do
    let val = 1.
                              Iteration
      while n > 1 do
      let val = val*n. .
                               Assignment
     let n = n-1.
10
      end
     return val.
12
                        Type conversion
13
    end
14
    let x = type @tointeger(io @input("Enter a positive integer: ")).
15
    io @println ("The factorial of "+x+" is "+(fact x)).
16
```

Function Calls

 In Asteroid function calls are constructed by juxta positioning a function with a value, e.g.

fact 3.

no parentheses necessary.

Data Structures

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

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

Structures & Objects

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

Structures

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

Stru

```
-- rectangle structure
    load system io.
    load system type.
    structure Rectangle with
      data xdim.
 6
      data ydim.
      -- member function
10
      function area with none do
        return this@xdim * this@ydim.
      end
13
14
      -- constructor
15
      function __init__ with (x if type @isscalar(x), y if type @isscalar(y)) do
16
        let this@xdim = x.
17
        let this@ydim = y.
18
      end
19
    end
20
    let r = Rectangle(4,2).
    io @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()

Structures: Rust & Go

```
#[derive(Debug)]
struct Rectangle {
    width: u32,
    height: u32,
}

impl Rectangle {
    fn area(&self) -> u32 {
        self.width * self.height
    }
}
```

Rust

```
type rect struct {
    width int
    height int
}

func (r *rect) area() int {
    return r.width * r.height
}
```

Asteroid Exercises

- Ex1: Write an Asteroid program that prints out the integers 10 through 1.
- Ex2: Write an Asteroid program that has a structure for the type 'Circle' that holds the coordinates of the center of a circle and its radius.
 - Your program should instantiate various object circles and print them out using the 'io @println' statement.
 - Add a member function to your Circle structure that computes the circumference of the given circle using 2*pi*r. Your program should instantiate a number of circles and print out their circumference.

- 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

- Asteroid has two more built-in 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.

```
let l = [6,5,3,1,8,7,2,4].
println("unsorted array: "+1).
```

- Using the 'structure' keyword Asteroid also supports user defined types.
 - The name of the structure becomes a new type available in the program.

```
1 -- user defined types¬
2 structure Person with¬
3 · data name.¬
4 · data profession.¬
5 end¬
6 ¬
7 let p:%Person = Person("Fred","Carpenter").¬
```

- 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 be used as a short-hand for the constant none.
 - The none data type does not belong to any type hierarchy.

Running Asteroid

- Install the interpreter on your machine
 - See https://asteroid-lang.org
- Run Asteroid in the cloud
 - https://replit.com/@lutzhamel/asteroid

Assignments

- Reading: MPL Chap 6
- Do Assignment #1 see BrightSpace