Computer Programming

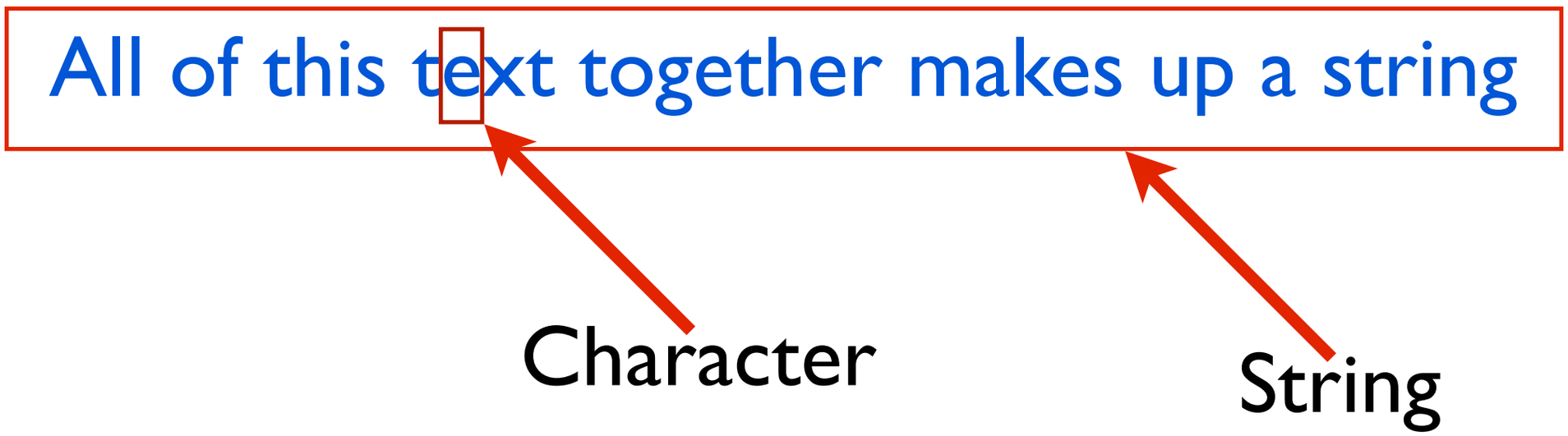
Lesson 2 - Simple Data Types and Logic

Review

Standard Types

Strings

- Sequence of Characters
- Denoted by single or double quote



Numbers

- **integers** - can be written without a decimal component (-50, -2, 10, 20, 15, 17, 1024)
- **floating-point** - with decimals (10.0, 14,6, -85.6734)
- **rational** - represented as a quotient of two integers (5/7, Rational(2, 3))
- **complex numbers** - expressed in the form a + b*i*, where a and b are real numbers and *i* is imaginary

Basic Math

How Numbers Interact

Most of the time, numbers work the way you'd expect.

- Addition (+)
- Subtraction (-)
- Multiplication (*)
- Division (/)
- Modulo (%)

```
1 + 2                 # => 3
1 + 2.0               # => 3.0
1.0 + 2               # => 3.0
1.0 + Complex(1,2)    # => (2.0+2i)
1 + Rational(2,3)     # => (5/3)
1.0 + Rational(2,3)   # => 1.6666666666666665
```

Basic Math

How Numbers Interact

Modulo (%) - Calculates the remainder of a division operation

```
puts 10%3 #=> 1

puts 10%5 #=> 0
```

Ranges

- Two dot (..) - inclusive
- Three dot (...) - exclusive of high value

```
digits = 0..9
digits.include?(5)          # => true
digits.min                  # => 0
digits.max                  # => 9
digits.reject {|i| i < 5 } # => [5, 6, 7, 8, 9]
digits.inject(:+)           # => 45
```

Logic

- == - exactly equal to
- > - greater than
- < - less than
- <= - less than or equal to
- >= - greater than or equal to
- != - not equal to

Control Structures

- if...elseif...else

```
name = "Shawn"

if name == "Shawn"
   puts "Hello Shawn"
end
```

Control Structures

- if...elseif...else

```
if name == "Shawn"
  puts "Hello Shawn"
else
  puts "Hello Someone else"
end
```

Control Structures

- if...elseif...else

```
name = "Fred"

if name == "Shawn"
   puts "Hello Shawn"
elsif name == "Joe"
   puts "Hello Shawn"
else
   puts "Hello Someone else"
end
```

Loops/Iterators

```
3.times          { print "X " }
1.upto(5)        {|i| print i, " " }
99.downto(95)    {|i| print i, " " }
50.step(80, 5) {|i| print i, " " }
```

produces:

```
X X X 1 2 3 4 5 99 98 97 96 95 50 55 60 65 70 75 80
```

```
10.downto(7).with_index {|num, index| puts "#{index}: #{num}"}
```

produces:

```
0: 10
1: 9
2: 8
3: 7
```

FizzBuzz

Write a program that prints the integers from 1 to 100. But for multiples of three print "Fizz" instead of the number and for the multiples of five print "Buzz". For numbers which are multiples of both three and five print "FizzBuzz".

Shawn Wallace

shawn@the-wallaces.net
@ShawnWallace