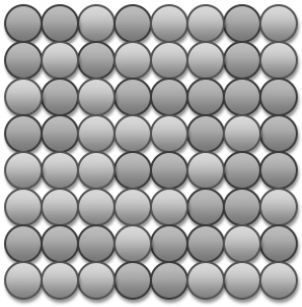

UNIT 2A
An Introduction to Programming

Arithmetic Expressions

- Mathematical Operators

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

- Order of Precedence
  {**} then {* / %} then {+ -}
- Use parentheses to force alternate precedence
  5 * 6 + 7 ≠ 5 * (6 + 7)
- Left associativity except for **
  2 + 3 + 4 = (2 + 3) + 4 2 ** 3 ** 4 = 2 **(3 ** 4)

Data Types

- Integers
  `4  15110  -53  0`
- Floating Point Numbers
  `4.0  -0.8  0.3333333333333333`
  `7.34e+014`
- Strings
  `"hello"  "A"  " "  ""  "7up!"`
- Booleans
  `true  false`

Variables

- All variable names must start with a lowercase letter.
- The remainder of the variable name (if any) can consist of any combination of uppercase letters, lowercase letters, digits and underscores (_).
- Variables are case sensitive.
  Example: `Value` is not the same as `value`.

Assignment Statements

- The lefthand side must contain a single variable.
- The righthand side can be any valid Ruby expression:
  - A numerical, string or boolean value.
    ```
    x = 45.2
    ```
  - A numerical expression.
    ```
    y = x * 15
    ```
  - A method (function) call.
    ```
    z = sqrt(15100)
    ```
  - Any combination of these:
    ```
    root1 = -b + sqrt(b**2 - 4*a*c)/(2 * a)
    ```


Methods

- Methods are used to capture small algorithms that might be repeated with different initial conditions.

```
def methodname(parameterlist)
    instructions
end
```

- `def` and `end` are reserved words and cannot be used as variable names.

Methods (cont'd)

- The name of a method follows the same rules as names for variables.
- The parameter list can contain 1 or more variables that represent data to be used in the method's computation.
  - A method can have 0 parameters.

```
def hello_world()
   print "Hello World!\n"
end
```

(`\n` is a newline character)

countertop.rb

parameter

```
def compute_area(side)
  square = side * side
  triangle = 0.5 * side / 2 * side / 2
  area = square – triangle
  return area
end
```

To run the function in `irb`:

```
load "countertop.rb"
compute_area(109)
```

argument
(run function with side = 109)

Methods (cont'd)

- To run a method, we say we "call" the method.
- A method can return either one answer or no answer to its "caller".
- The `hello_world` function does not return anything to its caller. It simply prints something on the screen.
- The `compute_area` function does return its result to its caller so it can use the value in another computation:

```
compute_area(109) + compute_area(78)
```


Methods (cont'd)

- Suppose we write `compute_area` this way:

```
def compute_area(side)
   square = side * side
   triangle = 0.5 * side/2 * side/2
   area = square – triangle
   print area
end
```

- Now this computation does not work since each function call prints but returns nothing:

```
compute_area(109) + compute_area(78)
```

escape.rb

(a function with two parameters)

```
def compute_ev(mass, radius)
  # computes escape velocity
  univ_grav = 6.67e-011
  return sqrt(2*univ_grav*mass/radius)
end
```

a comment

To run the function for Earth in `irb`:

```
load "escape.rb"
compute_ev(5.9742e+024, 6378.1)
```


Using predefined modules

- `Math` is a predefined module of methods that we can use without writing their implementations.

```
Math.sqrt(16)
Math::PI
Math.sin(Math::PI / 2)
```

- If we are going to use this module a lot, we can include it first and then leave off the module name when we call a function.

```
include Math
sqrt(16)
sin(PI / 2)
```