CSC 171
Introduction to Computer Programming
Lecture 2

Dr. Stephen Bloch
sbloch@adelphi.edu
http://www.adelphi.edu/sbloch/class/171/

Review: Recipe for defining functions

1) **Contract**
   specify function name
   num/types of args,
   type returned

2) **Examples**
   write down examples with known right answers

3) **Function skeleton**

4) **Function body**

5) **Testing**

Review:
Why do we need a recipe?

Programming requires creativity, and sometimes that's enough.

But when programs get big and complex, it helps to have a structure to follow.

Analogous to rules of counterpoint, harmony, etc. in music. Sonata, rondo, virelai, fugue, …

Helps avoid "blank page syndrome".

Syntax Rules So Far

1) **How to call a function**

   (*function-name expression expression* …)

   Ex: (+ 1 2 3 4 5)

2) **How to define a variable**

   (define *variable-name expression*)

   Ex: (define bigger (* big big big))

3) **How to define a function**

   (define (*func-name param-name* …)

   expression)

   Ex: (define (cube num) (* num num num))

(There will be c. 6 syntax rules for the whole semester. So you're halfway done!)

Another example: temperature conversion

; f->c takes a number of degrees Fahrenheit and returns the corresponding number of degrees Celsius.

Another example: temperature conversion

; f->c takes a number of degrees Fahrenheit and returns the corresponding number of degrees Celsius.

“Examples of f->c:”

(f->c 32) “should be” 0)

(f->c 212) “should be” 100

(f->c -40) “should be” -40

Another example: temperature conversion

; f->c takes a number of degrees Fahrenheit and returns the corresponding number of degrees Celsius.

```
(define (f->c Fahrenheit)
  … Fahrenheit …)
```

"Examples of f->c:"

(f->c 32) "should be" 0)

(f->c 212) "should be" 100

(f->c -40) "should be" -40

Another example: temperature conversion

; **f->c** takes a number of degrees Fahrenheit and returns the corresponding number of degrees Celsius.

```
(define (f->c Fahrenheit)
  (* 5/9 (- Fahrenheit 32)))
```

"Examples of f->c:"

(f->c 32) "should be" 0)

(f->c 212) "should be" 100

(f->c -40) "should be" -40

When you're Writing a Function…

- Function names *cannot* contain spaces.
- You *must* spell the function name *exactly* the same way in contract, examples, and definition.
- You *must* spell the parameter name *exactly* the same way in header and body.
- You *cannot* assume that the input will be any specific number (like 2, or 7, or …)
- Refer to the input only by the parameter name (e.g. **days**), so it works on *whatever* input is actually provided (stuck into the envelope).

Different kinds of information

- Numbers
  - Integers, e.g. 3, -29, 12870985701928745
  - Fractions, e.g. 3/4, 23897/38904
  - “floating-point” numbers, e.g. #i3.1415926
- Booleans: true and false
- Strings, e.g. “should be 7”, “Dr. Bloch”
- Pictures (we’ll play with this on Monday)

Objects, Types, and Functions

English

- Verb (action)
  - go, eat, buy, give
- Proper noun (specific thing)
  - Adelphi, Rover, Dr. Bloch
- Pronoun (holder for thing)
  - him, her, it
- Improper noun (kind of thing)
  - school, dog, professor

Programming language

- Function
  - +, sqrt, cube, hrs->mins
- Literal
  - 3, "hello", true
- Variable
  - big, num, age, …
- Data type
  - number, boolean, string, …

In-class exercise

- The function “days->hours” takes in a number of days and returns how many hours are in it. Write contract, examples, and skeleton.
- The function “difference-of-cubes” takes in two numbers and returns the difference between their cubes. Write contract, examples, and skeleton.

In-class exercise

- The function “over-18?” takes in a person’s age (a number of years) and returns a true or false indicating whether that person is over 18. Write contract, examples, and skeleton.
- The function “salutation” takes in an hour of the day (1-24) and returns either “Good morning”, “Good afternoon”, or “Good evening” as appropriate. Write contract, examples, and skeleton.