COMPSCI 367

The Practice of Artificial Intelligence

A.L.I.C.E.'S CREATOR SUDDENLY REALIZES THAT HE HAS THE LOEBNER PRIZE IN THE BAG.
EITHER THAT, OR A HUGE COMMERCIAL SUCCESS.
ALICE> so...what are you wearing?
COPYRIGHT©2004 J.D. "Illiad" Frazer HTTP://WWW.USERFRIENDLY.ORG/

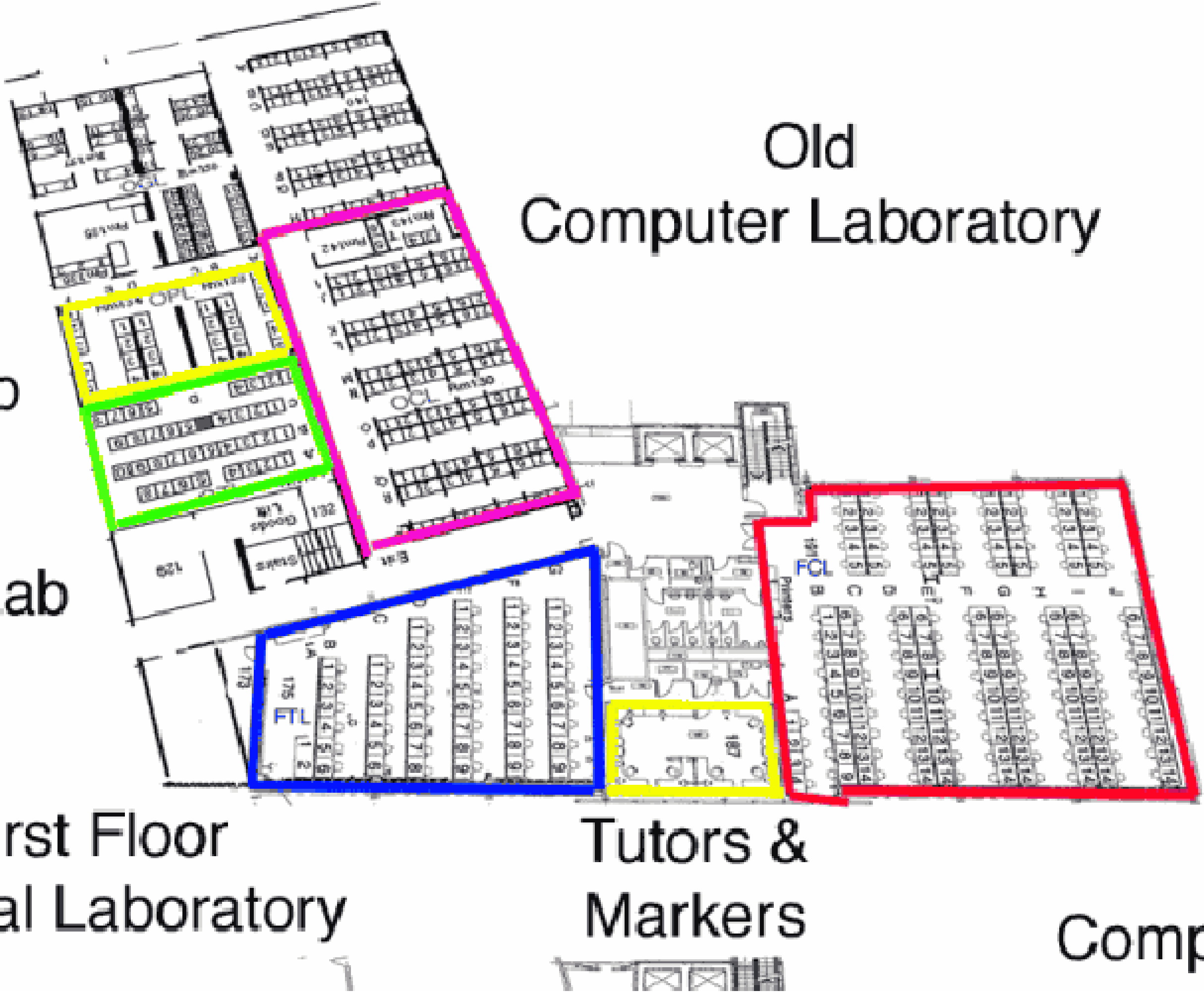
COMPSCI 367 Tutorial 1

- Introduction
- Clips Intro

Introduction

- Jonathan Rubin
- jrubin01@gmail.com
- Office Hours: Tuesday 12 – 1pm
- Room 187 (see next slide)

Introduction (cont...)



Old
Computer Laboratory

Old
Project Lab

Old
Tutorial Lab

First Floor
Tutorial Laboratory

Tutors &
Markers

First Floor
Computer Laboratory

1

Introduction (cont...)

- Tutorials will mainly focus around assignments.
  - Rule-based expert system (CLIPS)
  - Machine Learning (WEKA)
  - Planning (Prolog)

Introduction (cont...)

- Based on content by Carl Schultz

(last years 367 tutor)

  - http://www.cs.auckland.ac.nz/compsci367s1c/lectures/Pat.d/

Introduction to Clips

CLIPS Documentation

- User's Guide

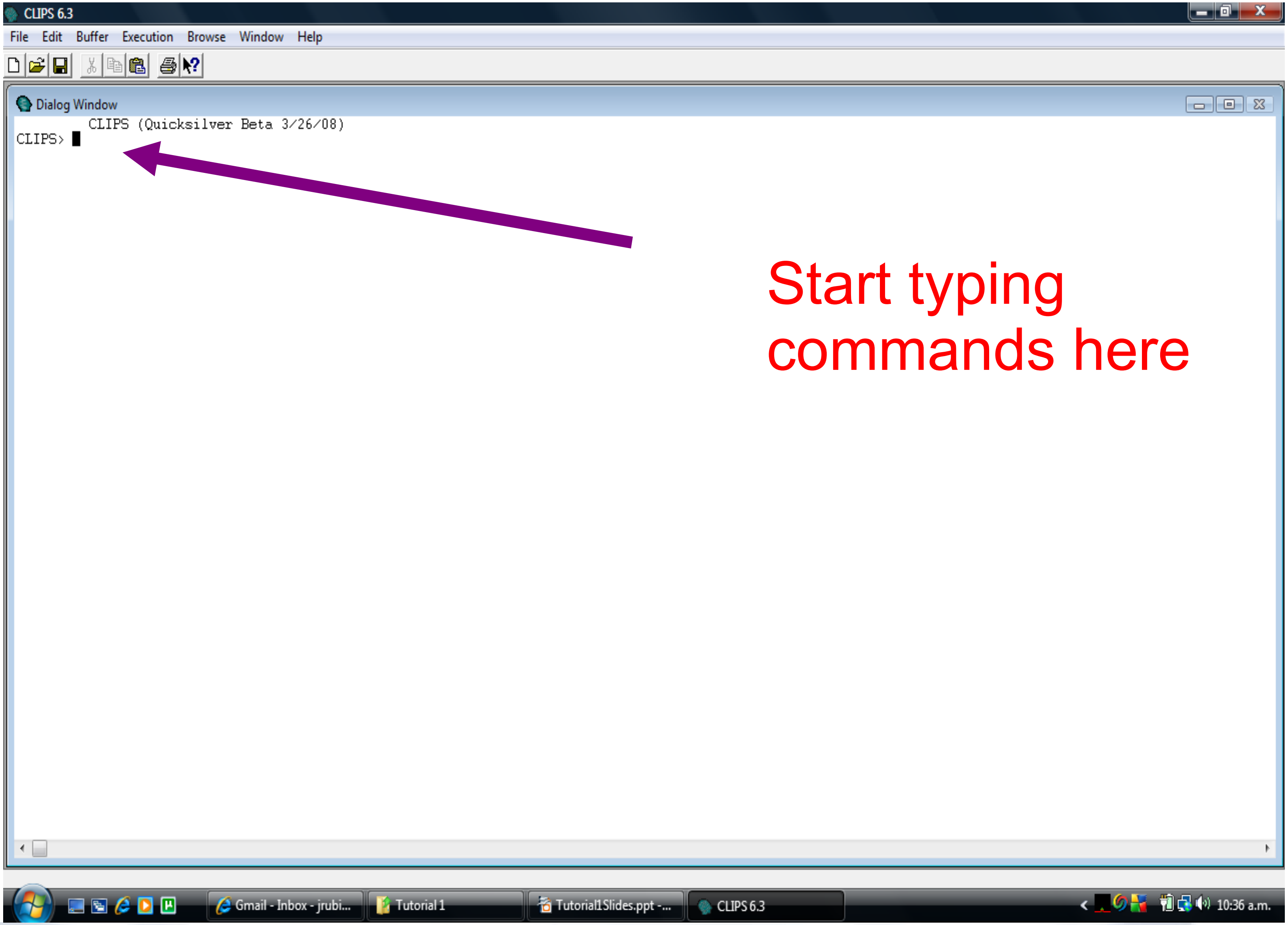
http://www.cs.auckland.ac.nz/compsci367s1c/resources/clips/documentation/usrguide.pdf

- Reference Manual
  - Volume I: Basic programming guide

http://www.cs.auckland.ac.nz/compsci367s1c/resources/clips/documentation/bpg.pdf

  - Volume II: Advanced Programming Guide
  - Volume III: Interfaces Guide

Clips Environment



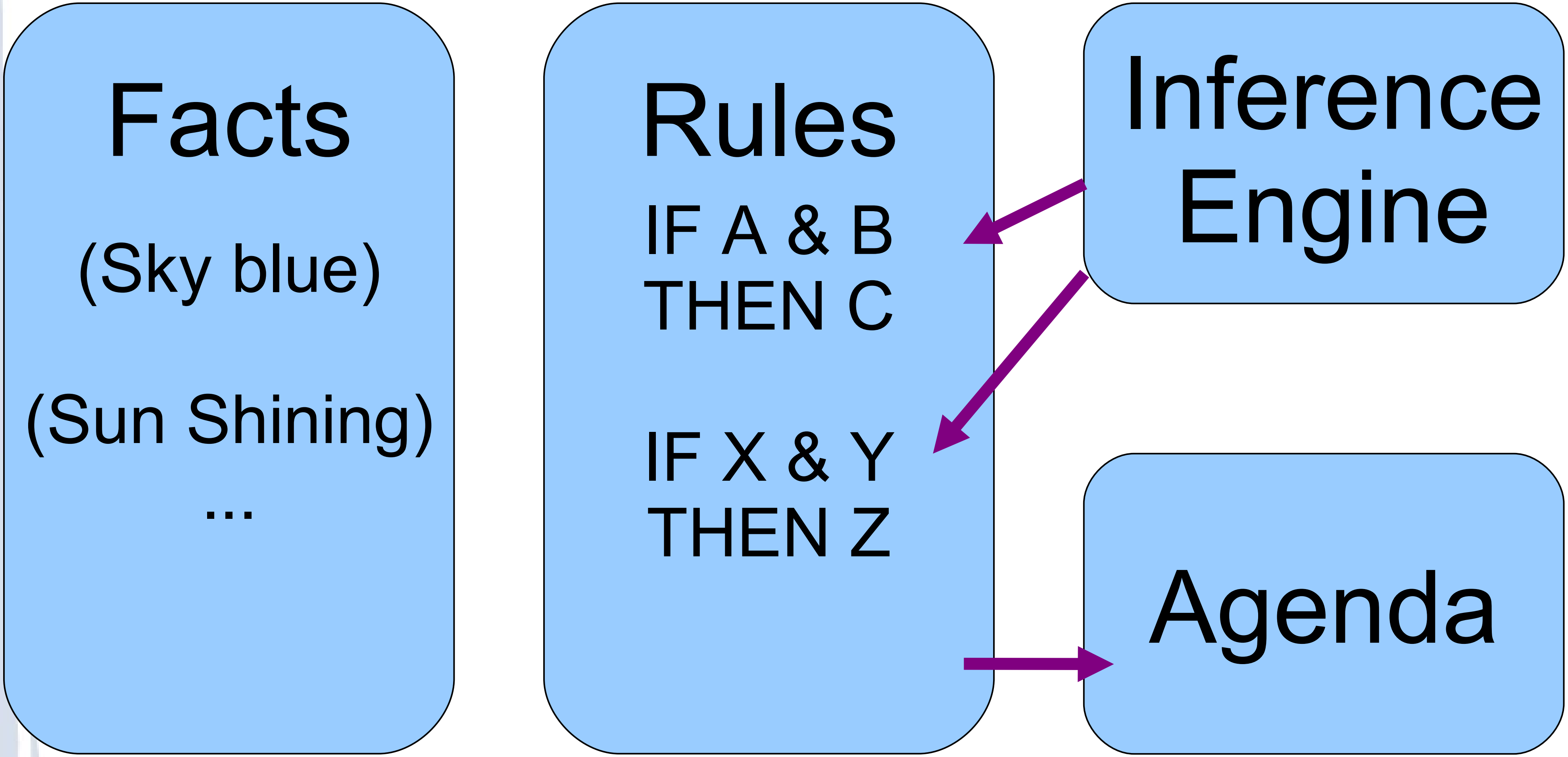
```
         CLIPS (Quicksilver Beta 3/26/08)
CLIPS> 
```

Start typing commands here

Clips Overview

- Facts
  - Heuristic knowledge based on experience
- Rules
  - Knowledge base
- Inference Engine
  - Decides which rules should be executed and when

Clips Overview



Facts

(Sky blue)

(Sun Shining)

...

Rules

IF A & B
THEN C

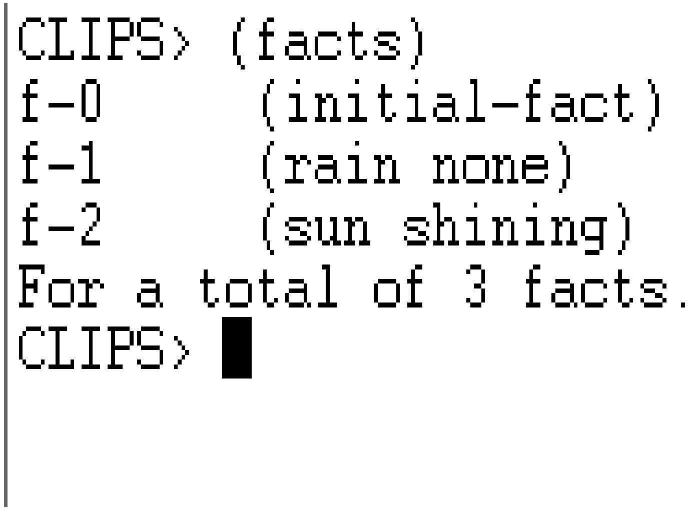
IF X & Y
THEN Z

Inference Engine

Agenda

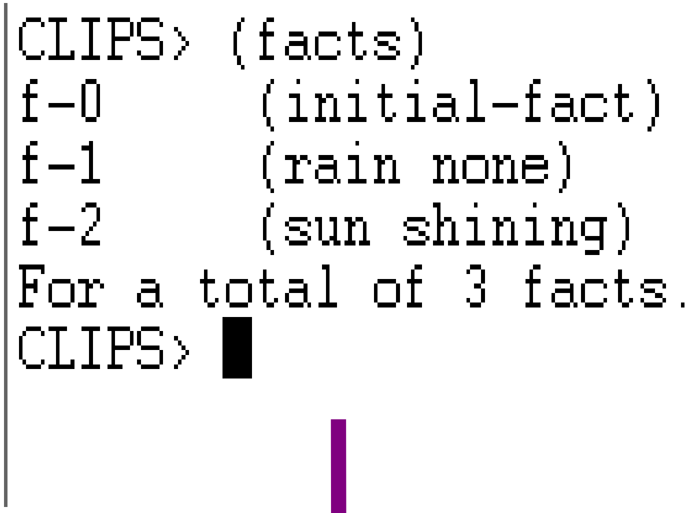
Facts

- (assert (rain none))
- (assert (sun shining))
- (facts)



```
CLIPS> (facts)
f-0     (initial-fact)
f-1     (rain none)
f-2     (sun shining)
For a total of 3 facts.
CLIPS> █
```

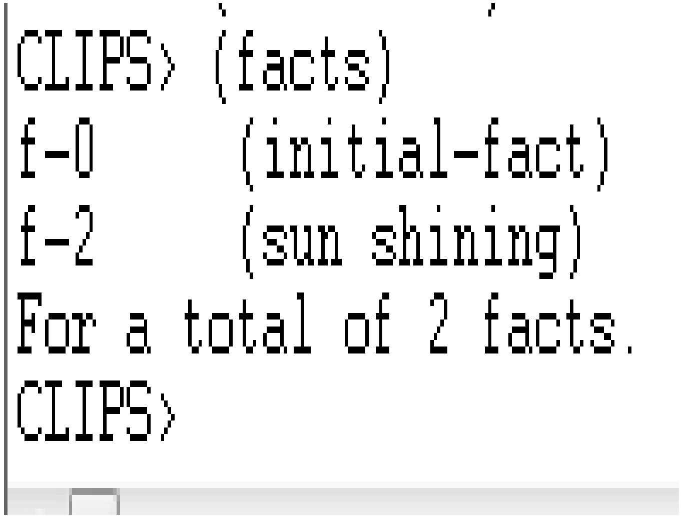
Facts (cont...)



```
CLIPS> (facts)
f-0     (initial-fact)
f-1     (rain none)
f-2     (sun shining)
For a total of 3 facts.
CLIPS> 
```

- (retract 1)
- (facts)

- (clear)



```
CLIPS> (facts)
f-0     (initial-fact)
f-2     (sun shining)
For a total of 2 facts.
CLIPS>
```

Deffacts

- (clear) will remove facts

Deffacts name

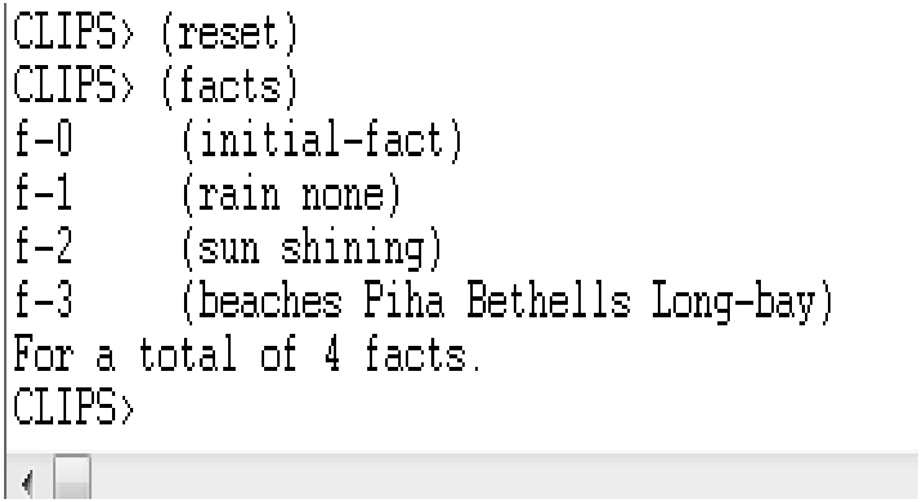
- (deffacts weather-facts

```
(rain none)
(sun shining)
(beaches Piha Bethells Long-bay))
```

No assert required

Deffacts (cont...)

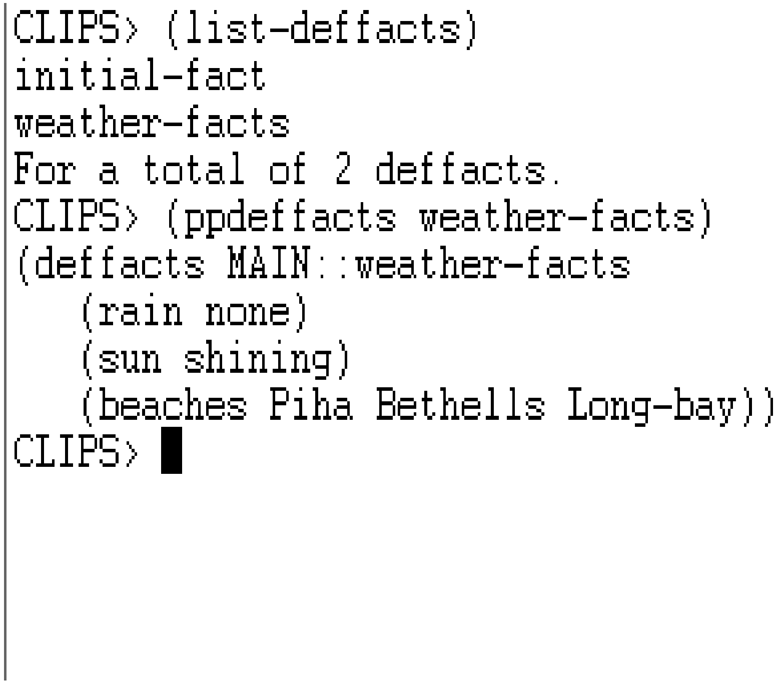
- (reset) – keeps rules and deffacts



```
CLIPS> (reset)
CLIPS> (facts)
f-0     (initial-fact)
f-1     (rain none)
f-2     (sun shining)
f-3     (beaches Piha Bethells Long-bay)
For a total of 4 facts.
CLIPS>
```

Deffacts (cont...)

- (list-deffacts)
- (ppdeffacts weather-facts)



```
CLIPS> (list-deffacts)
initial-fact
weather-facts
For a total of 2 deffacts.
CLIPS> (ppdeffacts weather-facts)
(deffacts MAIN::weather-facts
   (rain none)
   (sun shining)
   (beaches Piha Bethells Long-bay))
CLIPS> █
```

Defrules

- Defines an “IF THEN” rule
- E.g:

```
(defrule beach-day “should we go to the beach”
(rain none)
(sun shining)
=>
(assert (beach-day true)))
```

Patterns

Actions

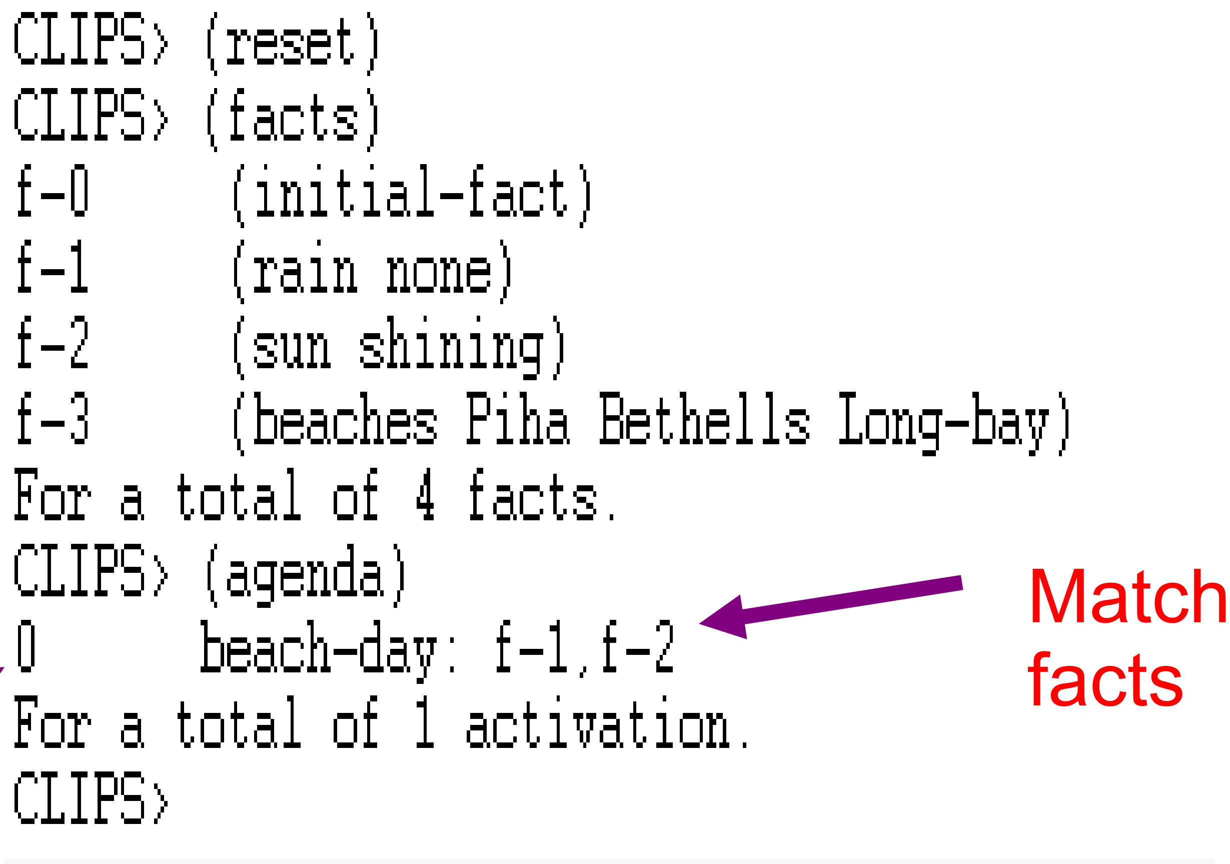
Rule Activations

- Clips attempts to match the patterns of rules against the facts in the facts-list.
- If pattern entities match then the rule is activated and put on the agenda.
- Agenda: Collection of activations. 0 or more activations may be on the agenda.

Defrules (cont...)

- (rules)
  - List of rules
- (ppdefrule *“rule-name”*)
  - Pretty Print a rule
- (undefrule *“rule-name”*)
  - Remove a rule

Agenda



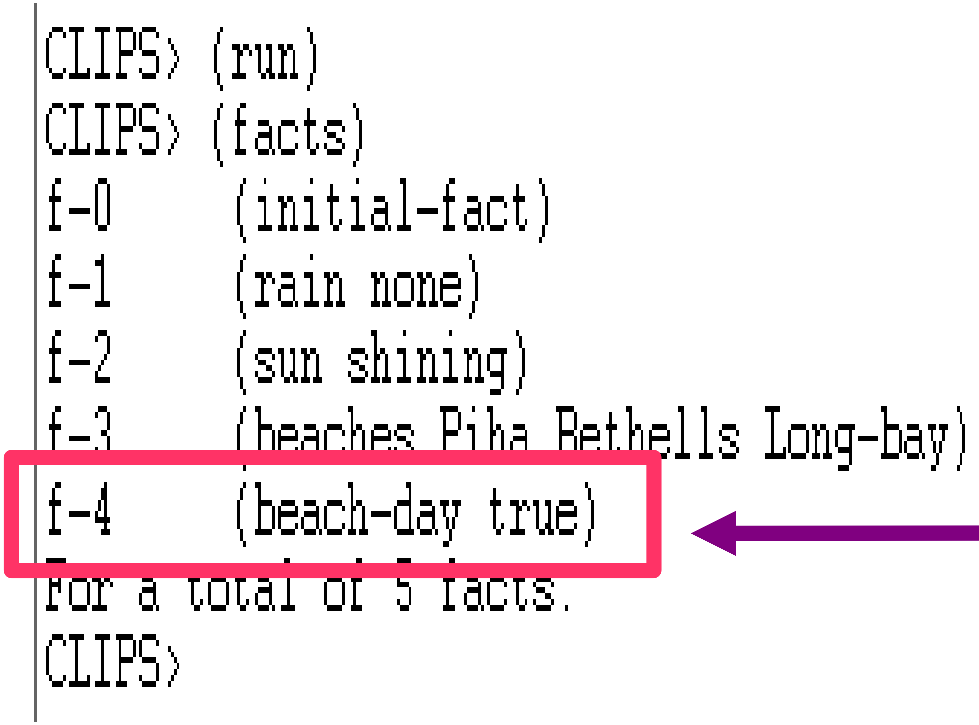
```
CLIPS> (reset)
CLIPS> (facts)
f-0     (initial-fact)
f-1     (rain none)
f-2     (sun shining)
f-3     (beaches Piha Bethells Long-bay)
For a total of 4 facts.
CLIPS> (agenda)
0      beach-day: f-1,f-2
For a total of 1 activation.
CLIPS>
```

Matching facts

Salience
{-10,000 to 10,000}

Agenda (cont...)

- Need to (run) for rule to fire:



```
CLIPS> (run)
CLIPS> (facts)
f-0     (initial-fact)
f-1     (rain none)
f-2     (sun shining)
f-3     (beaches Piha Bethells Long-bay)
f-4     (beach-day true)
For a total of 5 facts.
CLIPS>
```

New fact

Agenda (cont...)

- What happens if we (run) again???
- Nothing Happens!!!
- Why?
  - A rule is activated if its patterns are matched by:
    - 1) A brand new pattern entity that did not exist before or,
    - 2) A pattern entity that did exist before, but was retracted and reasserted, i.e. a “clone” of the old pattern entity, and thus now a new pattern entity.

Variables

- Values will be bound to variables within rules
- Single-field variable:
  - ?<variable-name>
  - e.g. ?x, ?colour, ?value etc....
- Multifield variable:
  - $?<variable-name>
  - e.g. $?colours, $?values

Variables (cont...)

- Single-field Variable
- (defrule display-weather “Displays the weather”

(sun ?sunValue)

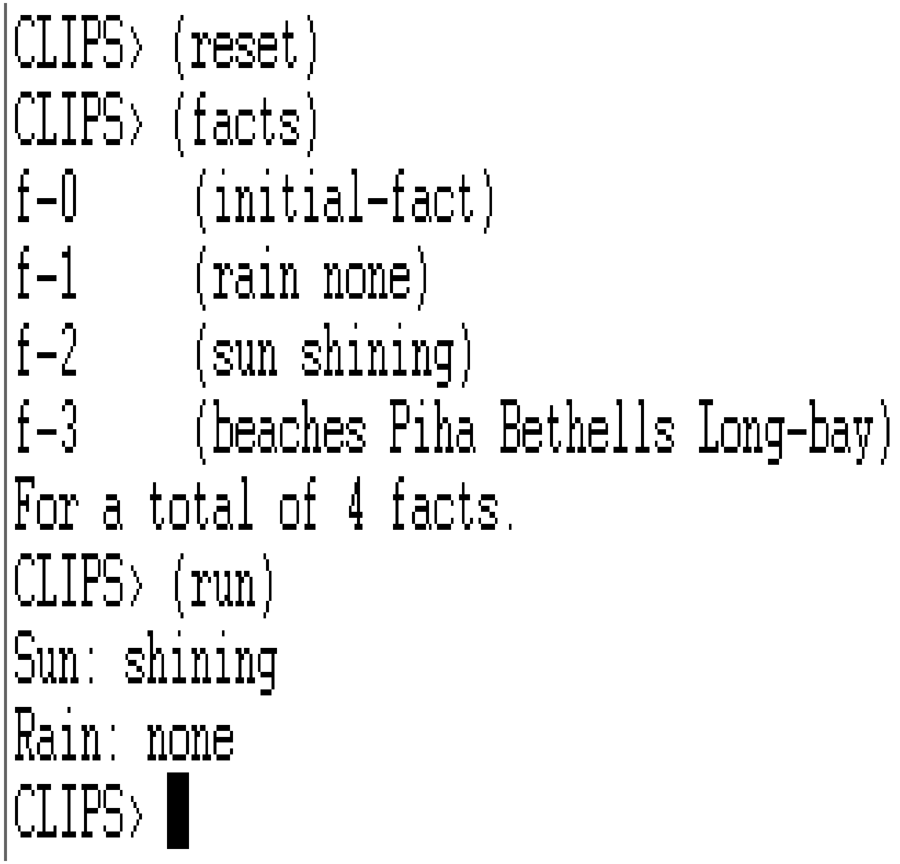
(rain ?rainValue)

=>

(printout t “Sun: ” ?sunValue crlf “Rain: ” ?rainValue crlf))

Print to the screen (t is standard output)

Variables (cont...)



```
CLIPS> (reset)
CLIPS> (facts)
f-0     (initial-fact)
f-1     (rain none)
f-2     (sun shining)
f-3     (beaches Piha Bethells Long-bay)
For a total of 4 facts.
CLIPS> (run)
Sun: shining
Rain: none
CLIPS> █
```

Variables (cont...)

- Multifield Variable
- (defrule display-beaches

```
   (beaches $?allBeaches)
=>
   (printout t ?allBeaches crlf))
```

No need for $ on RHS of rule

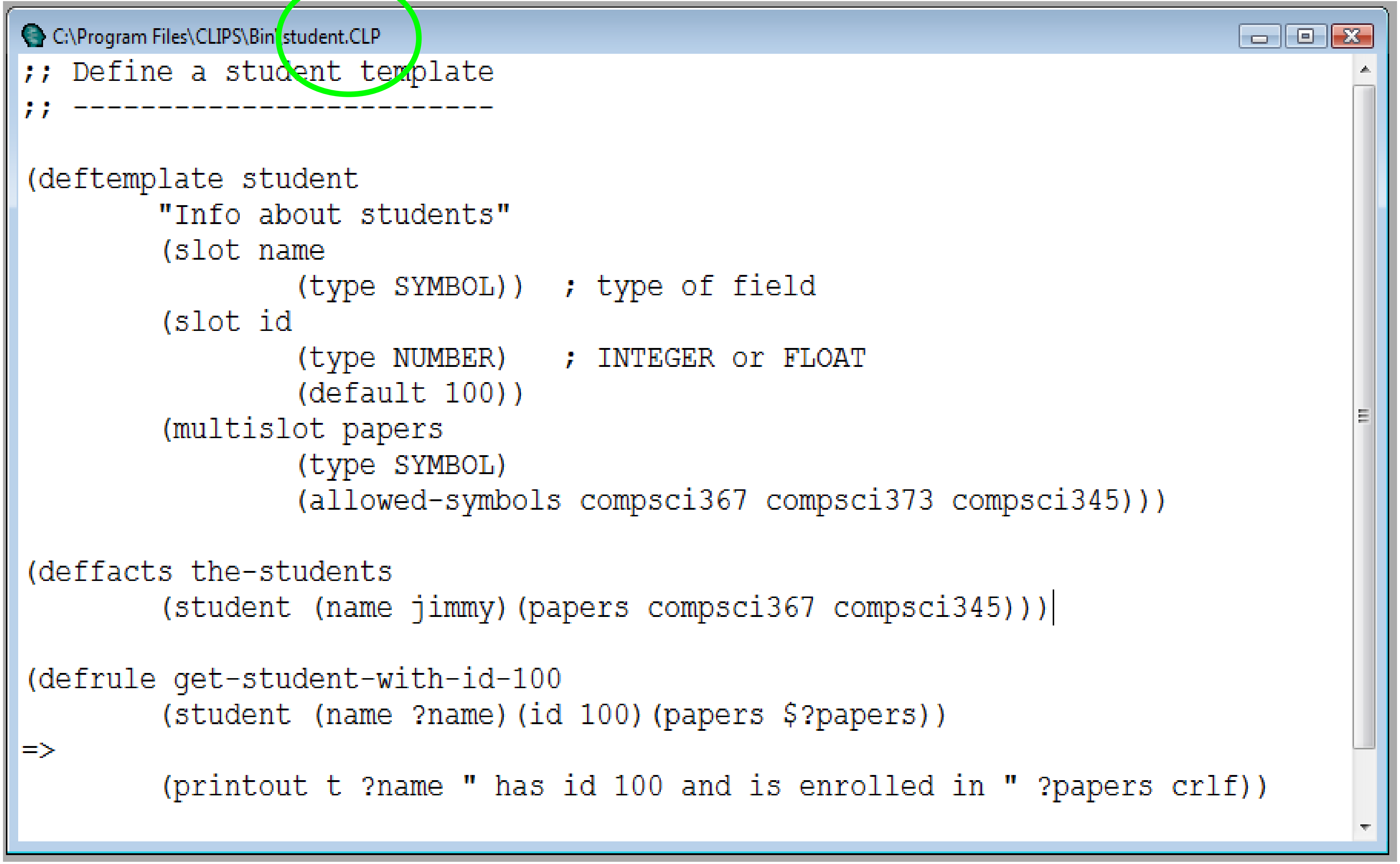
- Output:

(Piha Bethells Long-bay)

Deftemplates

- Adds structure to facts and rules
- Consists of named fields
  - Slot: single field
  - Multislot: zero or more fields
- Allows type declarations: SYMBOL, STRING, NUMBER ...

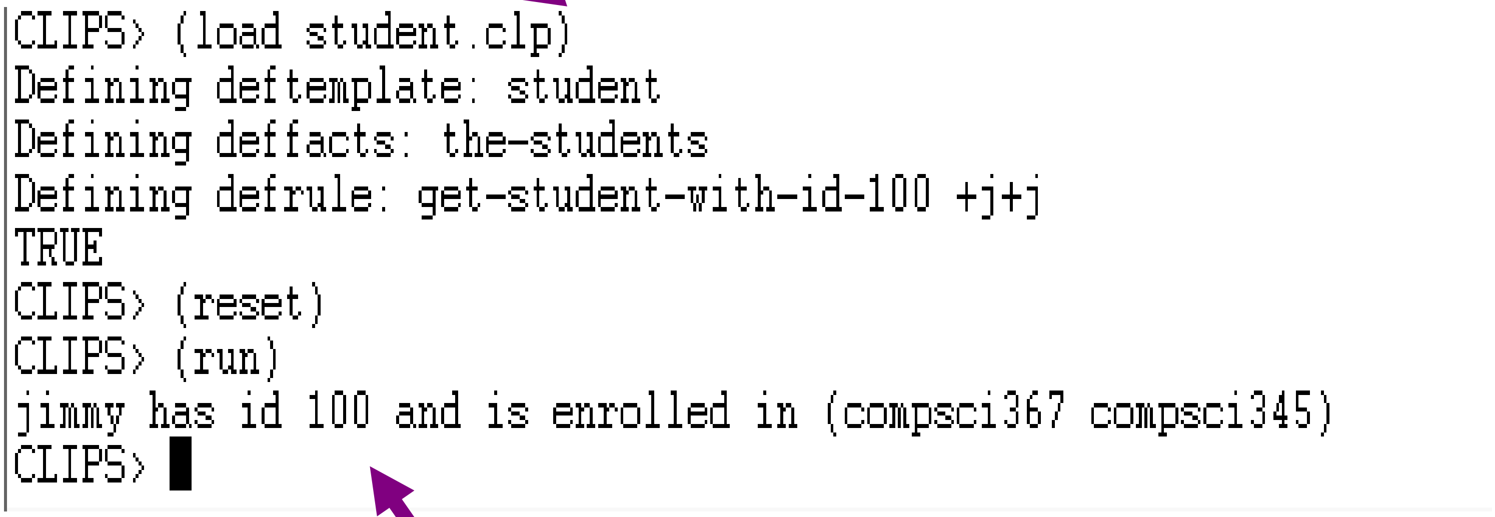
Deftemplates (cont...)



```
;; Define a student template
;; -------------------------

(deftemplate student
        "Info about students"
        (slot name
                (type SYMBOL))   ; type of field
        (slot id
                (type NUMBER)    ; INTEGER or FLOAT
                (default 100))
        (multislot papers
                (type SYMBOL)
                (allowed-symbols compsci367 compsci373 compsci345)))

(deffacts the-students
        (student (name jimmy)(papers compsci367 compsci345)))

(defrule get-student-with-id-100
        (student (name ?name)(id 100)(papers $?papers))
=>
        (printout t ?name " has id 100 and is enrolled in " ?papers crlf))
```

Deftemplates (cont...)

Load the saved clp file



```
CLIPS> (load student.clp)
Defining deftemplate: student
Defining deffacts: the-students
Defining defrule: get-student-with-id-100 +j+j
TRUE
CLIPS> (reset)
CLIPS> (run)
jimmy has id 100 and is enrolled in (compsci367 compsci345)
CLIPS>
```

Default Value

TODO

- Download & Install Clips (if you haven't already)
- Write, Save and Load some sample clips programs.
  - student, family etc....
- Think about how to represent a decision tree in clips for Assignment 01.