Compilers

Lab Session 1

Antoine Van Muylder
Sander Huyghebaert

VUB VRIJE UNIVERSITEIT BRUSSEL

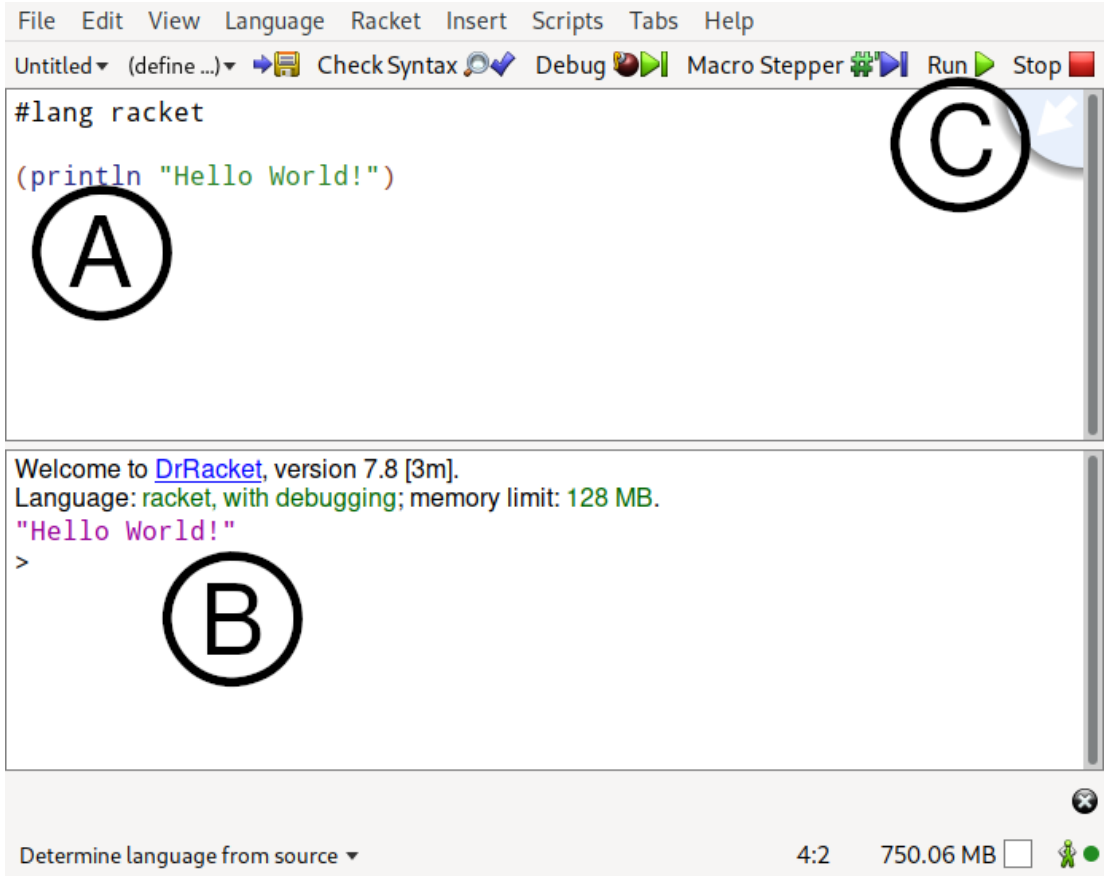
DRRACKET

INSTALLATION

https://download.racket-lang.org/

DRRACKET

OVERVIEW



- A: Definitions window (#lang racket)
- B: Read-Eval-Print-Loop
- C: Run your program

DRRACKET

SHORTCUTS

Some useful shortcuts:

- F1: Documentation
- F5: Run program
- F6: Syntax check
- Ctrl-i: Reindent all code

Some Racket features. . .

QUOTING

Used for nested lists.

```
'((M00 M01 M02) (M10 M11 M12) (M20 M21 M22))
'(begin (assign x 10) (assign z #t))
'() ;;empty list
'(1 2 . (3 4)) ;; => as '(1 2 3 4)
```

QUASIQUOTING & UNQUOTING

Like quote forms but with variables

```
(define var 47)
`,var  ;; => to 47.
`(1 2 3 ,var) ;; => to '(1 2 3 47)
(define alist `(4 ,(+ var 3))  )
;;unquote inside quasiquote "`" with the comma ","
```

Last expression `alist` evaluates to `'(4 50)`.

UNQUOTE SPLICING

To insert elements of the list, not the list itself

```
(define upper-half '(5 6 7 8))
`(1 2 3 4 ,@upper-half) ;; => '(1 2 3 4 5 6 7 8)
`(1 2 3 4 ,upper-half) ;; => '(1 2 3 4 (5 6 7 8))

(define (get-rich-quick money)
  `(,@money ,@money ,@money))

(get-rich-quick '(20))
;; => '(20 20 20)
(get-rich-quick '(1 10 50))
;; => '(1 10 50 1 10 50 1 10 50)
```

BASIC PATTERN MATCHING

To analyse nested expressions and extract subexpressions

```
(define (is-lt-1? num) ; function definition
  (match num
    [0 #t]
    [1 #t]
    [_ #f])) ; _ matches anything

(is-lt-1? 0)       ;; => #t
(is-lt-1? 1)       ;; => #t
(is-lt-1? 42)      ;; => #f
(is-lt-1? 'hello) ;; => #f
```

EXTRACT SUBEXPRESSIONS

```
(match (list 1 2 3)
  [(list a b 3) b]) ;; => 2
```

Or using a quasi pattern:

```
(match (list 1 2 3)
  [`( ,a ,b 3) b]) ;; => 2.
```

ELLIPSES IN PATTERN

```
(define (2nd-elem alist)
  (match alist
    [`(,_ ,e ,rest ...) e]))
```

Comparison with this pattern: `` `( ,_ ,e ,rest ) `` ?

GUARDED PATTERNS

```
(match expr
  [`(,binop ,x ,rest ...)
   #:when (binop? binop) 'its-a-binop]
  [_ #f])
```

SUMMARY OF PATTERNS/SEXP CONSTITUANTS

- Backtick (`)
- Unquoting (,)
- Ellipses (...)
- Splicing (,@)

RACKUNIT

UNIT TESTING

```
;; place the require at the top of your file
(require rackunit)

(module+ test ;; open submodule "test"
  (check-eq? (eval 1) 1)
  (check-eq? (eval '(+ 1 2)) 3)
  (check-eq? (eval '(+ 10 (- 5 2))) 13))
```