Programming Paradigms

CT331 Week 6 Lecture 3

Finlay Smith

Finlay.smith@universityofgalway.ie

Predicates

A predicate is any function that returns a Boolean value #t or #f.

Scheme predicates conventionally have names ending with a question mark.

Exceptions are primitives such as <, >, >=, <= etc.

Examples include:
negative?
null?
number?
even?
equal?
symbol?
Note: not, and, or are keywords

Control

Program control can be carried out by use of `if` built-in function.

Format:

```
(if <expr> <expr> <expr>)
```

The first ‹expr› is always evaluated. If it produces a non-#f value, then the second ‹expr› is evaluated for the result of the whole if expression, otherwise the third ‹expr› is evaluated for the result.

Control

```
> (if (> 2 3)
      "bigger"
      "smaller")
"smaller"
```

Control

More complex program control can be carried out by use of `cond` built-in function.

`cond` can consist of multiple condition-action pairs.

Format:

```
(cond  (clause-1)
               (clause-2)
               . . .
               (clause-n)
               (else (expression))
)
```

where each clause is:

`(clause) = ((condition) (expression))`

Control

```
(cond  [(condition1) (expression1)]
       [(condition2) (expression2)]
       [(condition3) (expression3)]
       [(condition4) (expression4)]
       [else (expression)]
)
```

“ In Racket, parentheses and square brackets are actually interchangeable, as long as (is matched with) and [is matched with].”

- Racket docs

The conditions are evaluated from top to bottom.

(condition1) is evaluated. If it is true, (expression1) is evaluated and the result returned.

As a default, #t will be returned if there is no (expression1).

If (condition1) is false, evaluation continues to the next clause which is evaluated in the same way.

Lisp - Aside: Print

> (display “Hello world!”)
Hello world!

> (printf "The answer is ~a" (add 4 5))
The answer is 9

Normally no need to use print – returning a value will display the result fro the top calling function

See: https://docs.racket-lang.org/reference/Writing.html

Lisp - Aside: Begin

It is not always possible to be totally functional. Printing is one example of this.

```
(begin
        (display “Ok, here we go!”)
        (some function)
        (display “wow, that was fun”))
```

Generally don’t use begin as it isn’t functional – you won’t need it for any of the questions in the assignment.

See: https://docs.racket-lang.org/reference/begin.html?q=begin#%28form._%28%28quote._~23~25kernel%29._begin%29%29

Links and refs:

https://racket-lang.org/

https://docs.racket-lang.org/reference/index.html

In particular: http://docs.racket-lang.org/reference/pairs.html

Cond: https://docs.racket-lang.org/guide/syntax-overview.html#%28part._.Conditionals_with_if__and__or__and_cond%29