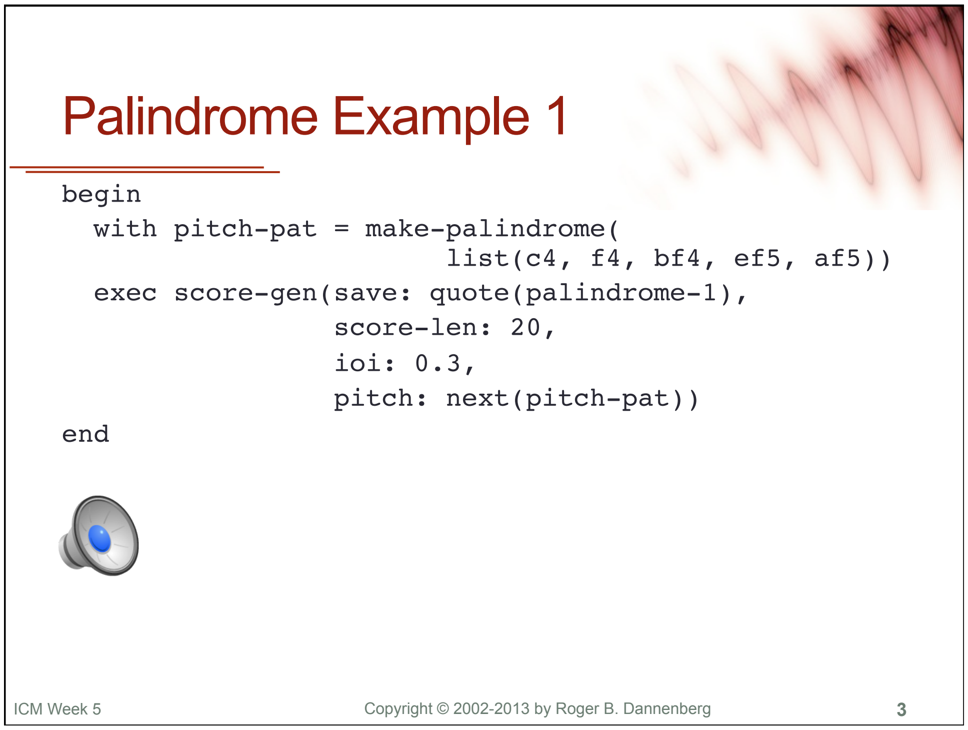
MORE PATTERNS

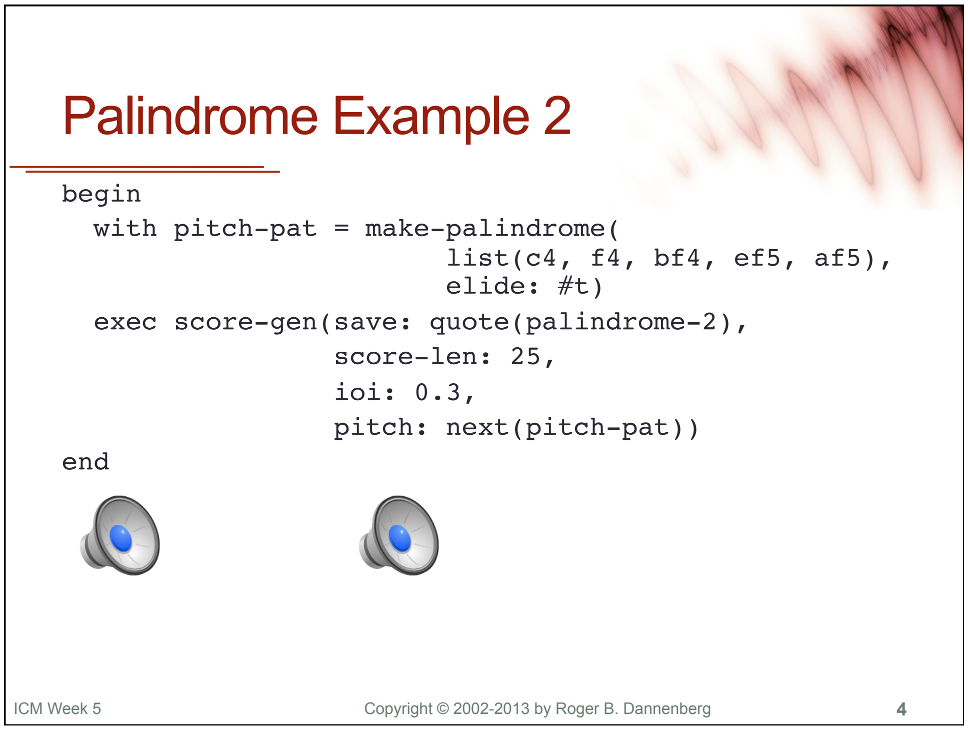
make-palindrome

- `make-palindrome(`*`list-or-pattern`*`,`
  `                for: `*`number-or-pattern`*`,`
  `                elide: `*`keyword`*`)`
- Output elements of input list forward then reverse order
- Length of period given by `for:`
  - Default is full forward-backward traversal
- Elide:
  - :first – A,B,C becomes A,B,C,C,B,A,B,C,C,B,... (elide the final A)
  - :last – A,B,C becomes A,B,C,B,A,A,B,C,B,A,...
    (elide the duplicate of C)
  - #t – A,B,C becomes A,B,C,B,A,B,C,B,... (elide first and last)
  - #f – A,B,C becomes A,B,C,C,B,A,A,B,C,C,B,A,... (no elision)
- Every period:
  - Update any pattern inputs



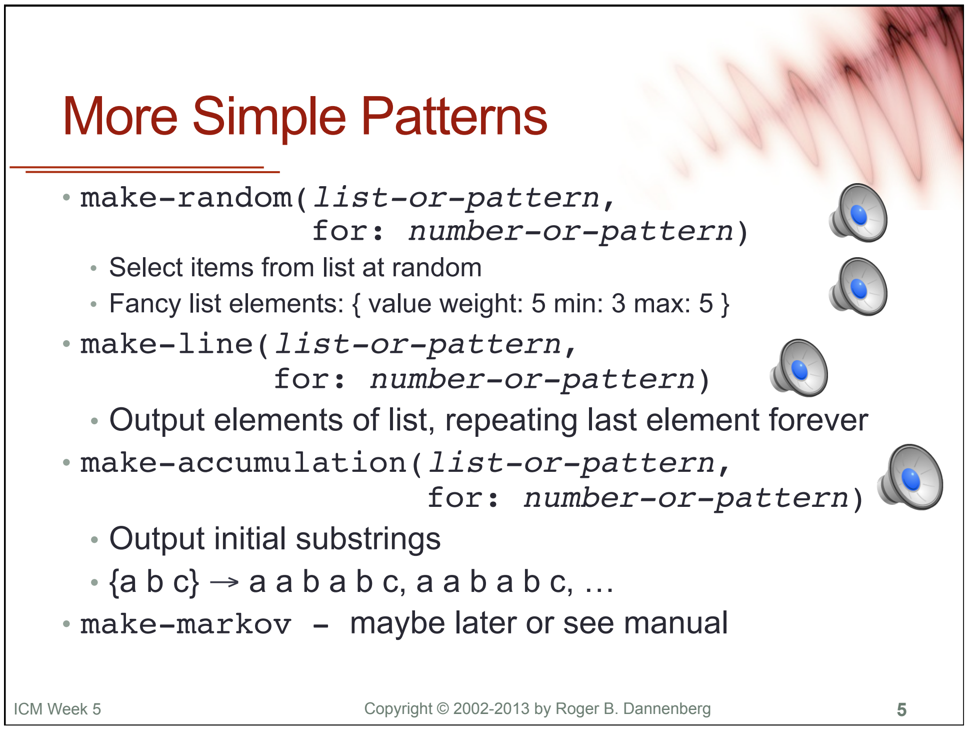
Palindrome Example 1

```
begin
  with pitch-pat = make-palindrome(
                      list(c4, f4, bf4, ef5, af5))
  exec score-gen(save: quote(palindrome-1),
                 score-len: 20,
                 ioi: 0.3,
                 pitch: next(pitch-pat))
end
```



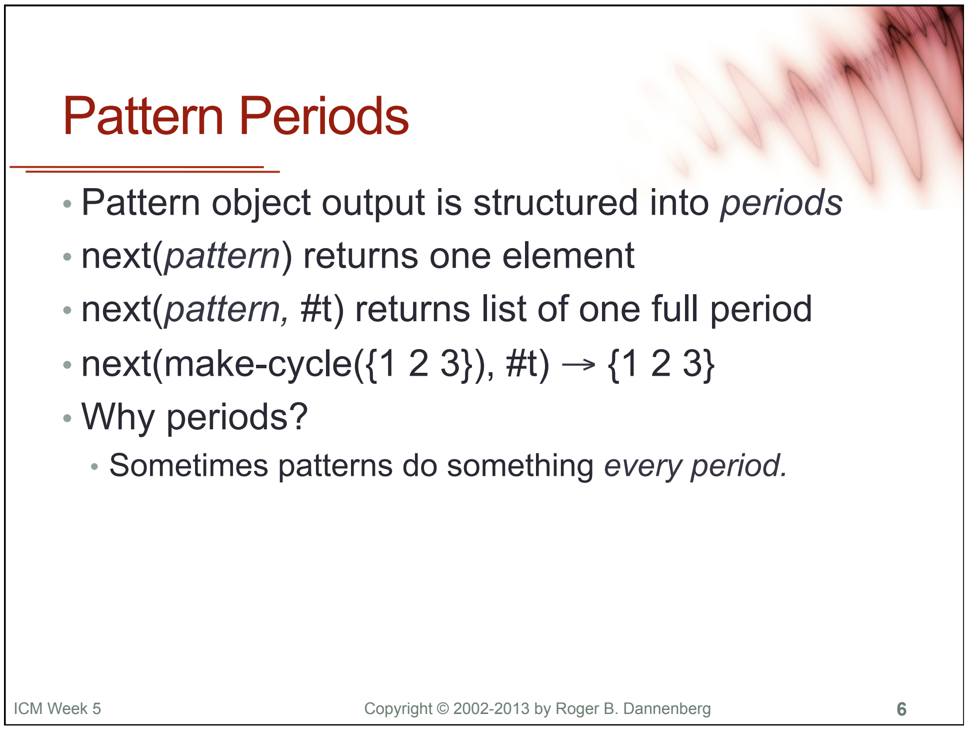
Palindrome Example 2

```
begin
  with pitch-pat = make-palindrome(
                      list(c4, f4, bf4, ef5, af5),
                      elide: #t)
  exec score-gen(save: quote(palindrome-2),
                 score-len: 25,
                 ioi: 0.3,
                 pitch: next(pitch-pat))
end
```



More Simple Patterns

- `make-random(`*`list-or-pattern`*`,`
  `for:` *`number-or-pattern`*`)`
  - Select items from list at random
  - Fancy list elements: { value weight: 5 min: 3 max: 5 }
- `make-line(`*`list-or-pattern`*`,`
  `for:` *`number-or-pattern`*`)`
  - Output elements of list, repeating last element forever
- `make-accumulation(`*`list-or-pattern`*`,`
  `for:` *`number-or-pattern`*`)`
  - Output initial substrings
  - {a b c} → a a b a b c, a a b a b c, …
- `make-markov –` maybe later or see manual



Pattern Periods

- Pattern object output is structured into *periods*
- next(*pattern*) returns one element
- next(*pattern,* #t) returns list of one full period
- next(make-cycle({1 2 3}), #t) → {1 2 3}
- Why periods?
  - Sometimes patterns do something *every period.*

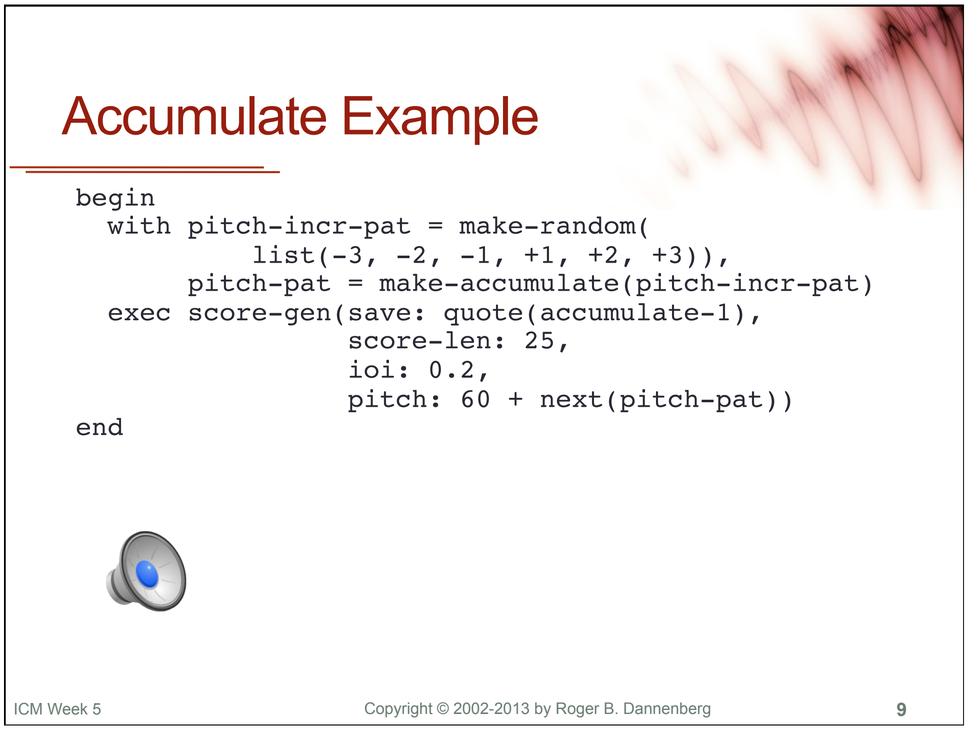
make-cycle

- `make-cycle(`*`list-or-pattern`*`,`
  ```
               for: number-or-pattern)
  ```
- Output elements of input list in sequence
- Length of period given by `for:`
  - Default is the length of the input list
- Every period:
  - Update list-or-pattern to next period
  - Update number-or-pattern to next value

Patterns of Patterns - 1

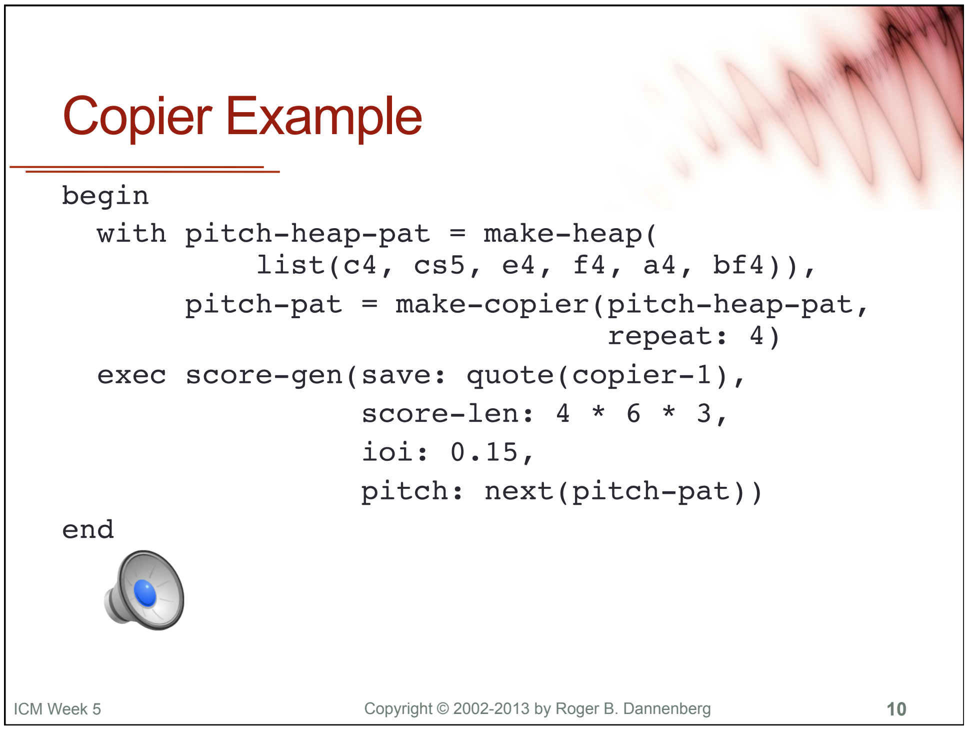
-
  ```
  make-accumulate(pattern,
                  max: expr, min: expr,
                  for: number-or-pattern)
  ```
  - Sum successive elements from input pattern

-
  ```
  make-copier(pat, repeat: expr-or-pat,
              merge: boolean,
              for: number-or-pattern)
  ```
  - Copy each period *repeat* times,
  - merge to one period if *merge* is true



Accumulate Example

```
begin
  with pitch-incr-pat = make-random(
           list(-3, -2, -1, +1, +2, +3)),
       pitch-pat = make-accumulate(pitch-incr-pat)
  exec score-gen(save: quote(accumulate-1),
                 score-len: 25,
                 ioi: 0.2,
                 pitch: 60 + next(pitch-pat))
end
```



Copier Example

```
begin
  with pitch-heap-pat = make-heap(
           list(c4, cs5, e4, f4, a4, bf4)),
       pitch-pat = make-copier(pitch-heap-pat,
                               repeat: 4)
  exec score-gen(save: quote(copier-1),
                 score-len: 4 * 6 * 3,
                 ioi: 0.15,
                 pitch: next(pitch-pat))
end
```

Patterns of Patterns - 2

- `make-length(`*`list-or-pattern,`*
  `number-or-pattern``)`
  - Regroup input sequence to specified period lengths
- `make-window(`*`pattern, window-size,`*
  `window-skip``)`
  - Output *window-size* elements,
  - then advance *window-skip*
  - make-cycle({a b c d}), 3, 1 → a b c b c d c d a d a b a b c …

Window Example

```
begin
  with pitch-line-pat = make-line(
           list(c4, cs4, e4, f4, a4, bf4)),
       pitch-pat = make-window(pitch-line-pat, 3, 1)
  exec score-gen(save: quote(window-1),
                 score-len: 17,
                 ioi: 0.2,
                 pitch: next(pitch-pat))
end
```

```
make-window(pitch-pat, 9, 3)
```