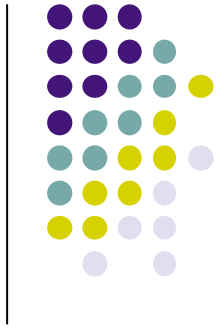
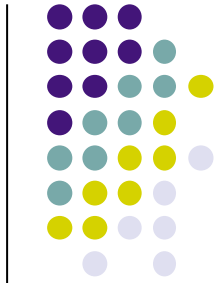
Higher-order programming and records



- This lesson gives the final two concepts we need to complete the functional paradigm and its kernel language
  - Higher-order programming
  - Record data structures
- Higher-order programming is the ability to use functions (and procedures) as first-class entities in the language
  - As inputs and outputs of other functions
  - This is an enormously powerful ability that lies at the foundation of data abstraction (including object-oriented programming)
- Record data structures are a general compound data type that allows symbolic indexing
  - This is useful both for symbolic programming and for data abstraction

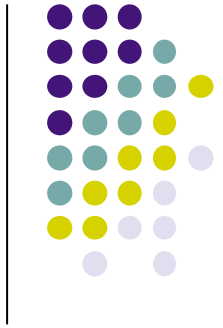
Higher-order programming



- Higher-order programming is based on two concepts
  - Contextual environment
  - Procedure value

- We introduce the contextual environment by means of a small exercise on static scope
  - This exercise shows naturally the need for the contextual environment

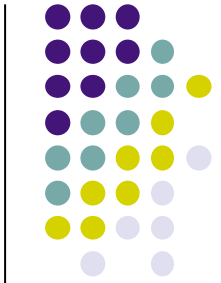
An exercise on static scope



What does this program display?

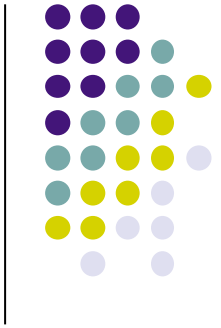
```
local P Q in
   proc {P} {Browse 100} end
   proc {Q} {P} end
   local P in
      proc {P} {Browse 200} end
      {Q}
   end
end
```

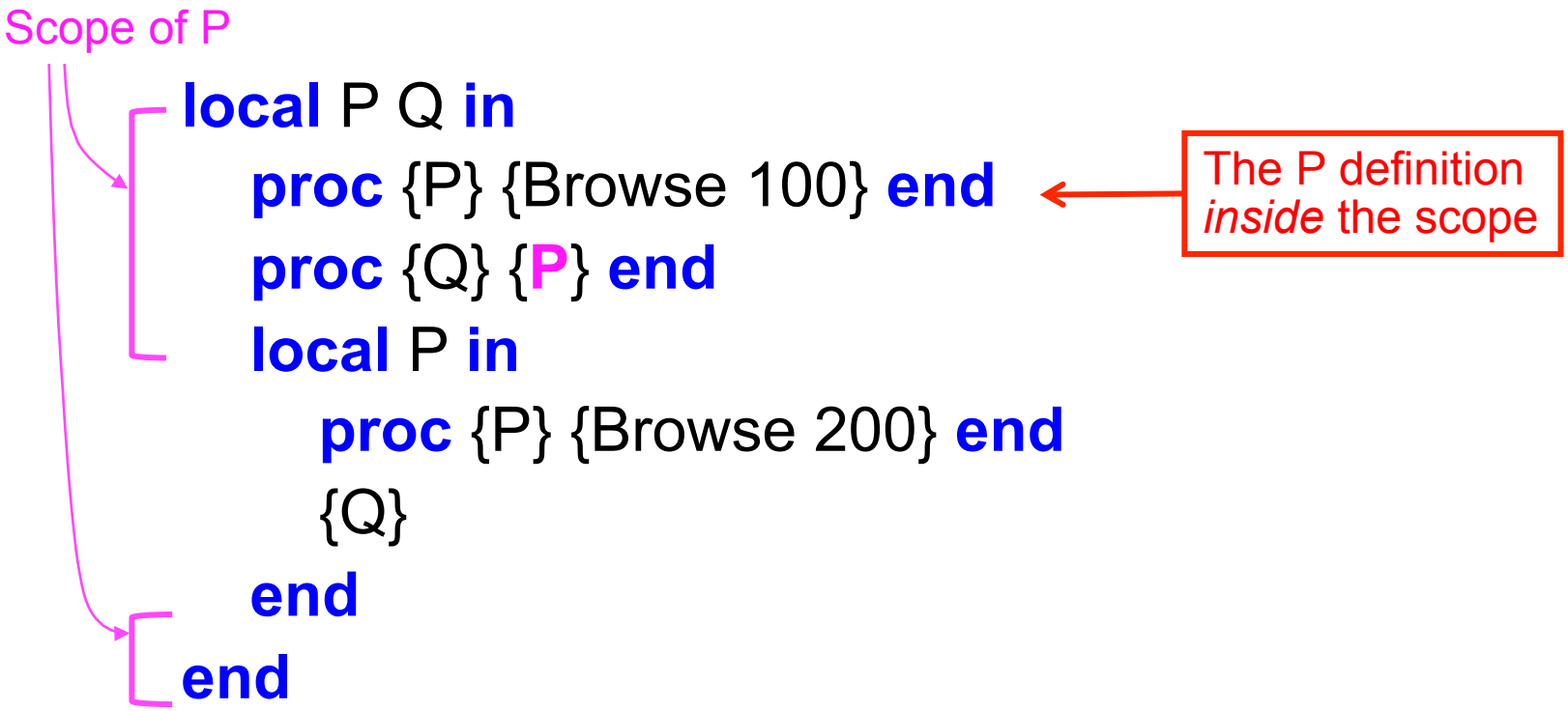
What is the scope of P?



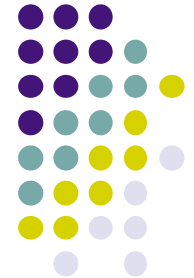
```
local P Q in
   proc {P} {Browse 100} end
   proc {Q} {P} end
   local P in
      proc {P} {Browse 200} end
      {Q}
   end
end
```

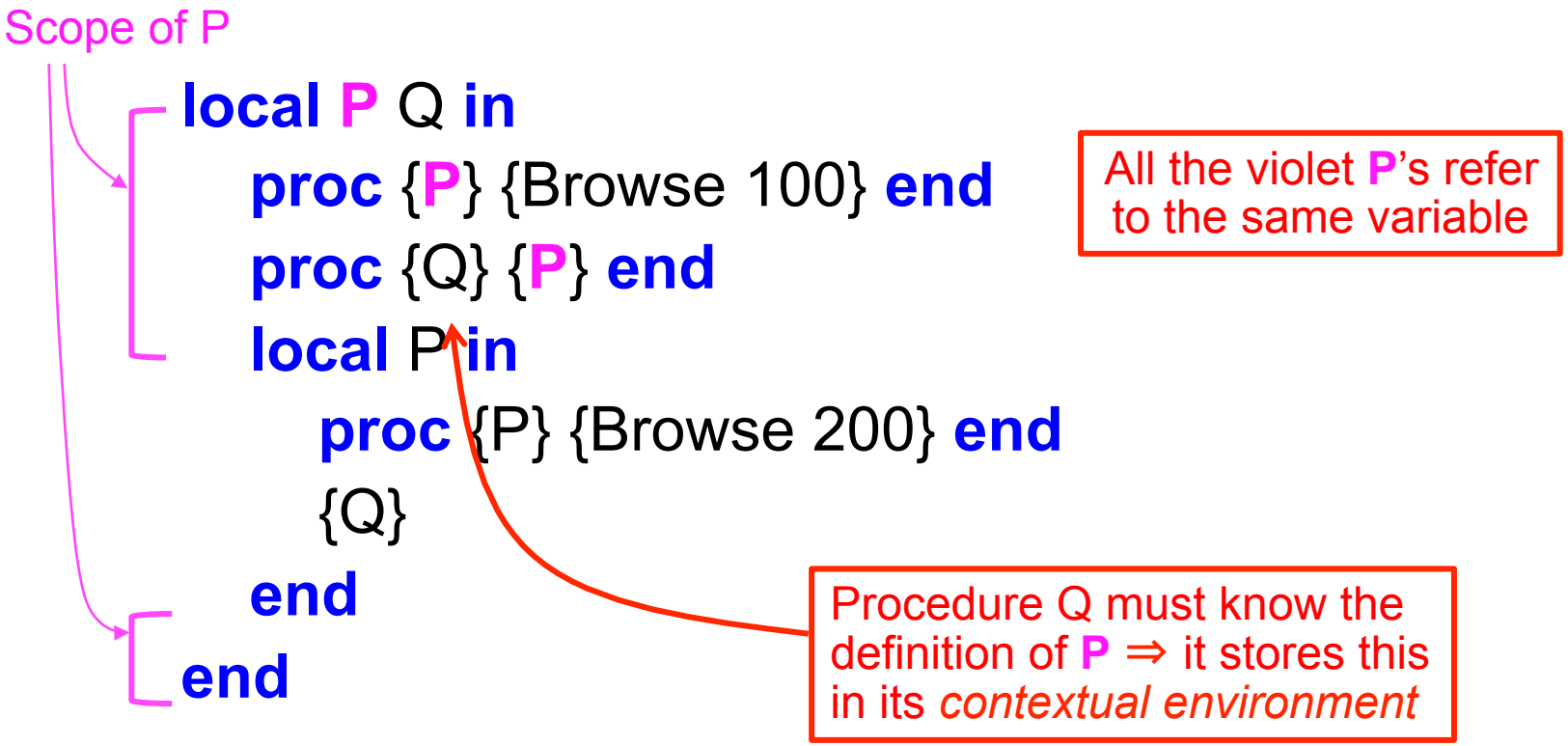
What is the scope of P?



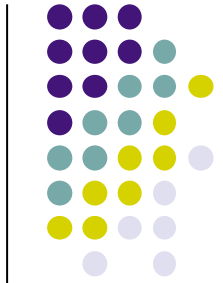


Contextual environment of Q





Contextual environment



- The contextual environment of a function (or procedure) contains all the identifiers that are used *inside* the function but declared *outside* of the function

```
declare
A=1
proc {Inc X Y} Y=X+A end
```

- The contextual environment of Inc is $E_C = \{A \rightarrow a\}$
  - Where *a* is a variable in memory: *a=1*