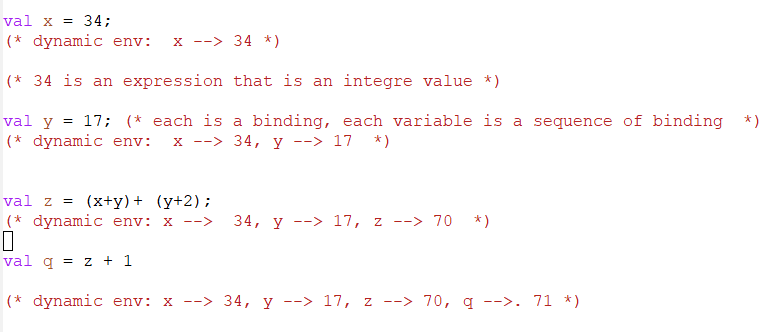
**Week1**

**ML variable and bindings and expressions**

* C-X C-F pass in path name (to create a new file)
* C-X C-S save the file
* C-c C-s sml terminal
* What is a sequence of binding?
  + You can use earlier
    - We evaluate in the current dynamic environment



* + Can you use later binding?
    - NO
    - Advantages?
* Type checking happens in static env before evaluation happens in dynamic env
  + Static env type checks
  + Doesn’t actually run program
* The two branches of if and else must have a same type
* Semantics
  + Syntax is how to write something
  + Semantic is what that something means
    - Type-checking
    - Evaluation
  + For variable bindings:
    - Type check expression and extend static env
    - Evaluate expression and extend dynamic env
  + For each kind of expression they have their own type checking and evaluation rules

**rules for expression**

* Expression can be the result of evaluating some other expressions
  + 3 < 2 is an expression which gives another expression that is Boolean
* In other words, expression can be build from other expressions
* Definition of expression is recursive because for every kind of expression
  + What is syntax (how is it written down)
  + What type checking (what it cause to fail)
  + What are the evaluation rule
* Variables (simples expression)
  + Syntax:
    - Sequence of letters, digits, not starting with digits
  + Type checking:
    - This is when we are using it
    - Look up whatever value it has in current static env
      * If not there, fail
  + Evaluation
    - Look up variable in dynamic env
* Addition
  + Syntax:
    - An expression E1 + e2, where e1,e2 are expressions
  + Type checking:
    - If e1 and e2 to have type int then e1+e2 has type int
    - If either 2 doesn’t have same type then e1+e2 does not type check
    - Answer for large is built out of answer for small
  + Evaluation
    - If e1 evaluates v1 and e2 to v2 then e1+e2 evaluates to sum of v1+v2;
* Conditionals
  + Syntax:
    - If e1 then e2 else e3
    - Where, if, then, an else are keywords
    - E1,e2 and e3 are sub expressions
  + Type checking
    - E1 must be Boolean
    - E2 and e3 can be of any type but must be the same
  + Evaluation
    - First evaluates to e1 to v1
    - If true, evaluates e2 which is whole expression result
    - Else, evaluates to e3, which is whole expression result

Values

* Every value is an expression, but not all expressions are values
* Every value evaluates to itself

**The REPL and Error Messages**

* What is a REPL?
  + We used REPL using “use”
  + It takes content of a file. It binds all the results at a batch
  + Read-eval-print-loop
* Errors:
  + Mistake can be
    - Syntax / type checking / evaluation

**Shadowing**

* Extend dynamic env means to add it to the env
* There is no concept of assignment. Namely, new binding is created each time and shadows the earlier binding
* If you use multiple times on a file then
  + Introduce same bindings again
  + Wrong code is correct
  + Unexpected behavior
* So it’s better to rerun REPL

**Functions Informally**

* Allow variables in functions
  + like a method in OOP, take arg, consume and return
* type of function is:
  + type arg1 \* type arg n 🡪 result type
    - ML figured it out by looking at the function body
* Inside the function body you can call the function itself
* In ML, unless function has 1 arg you need parenthesis
* Like variable, function binding can use function in earlier binding

**Functions formally**

* Syntax
  + Fun x0 ( x1: t1, … , xn : tn) = e
* Evaluation
  + A function is a value
  + Namely, add it to the env, so that later expression can call it
  + We evaluate when we call it
* Type checking
  + Adds binding: x0 : (t1 \* ... \* tn ) -> t if: