```
to be compiled
                                                                                                                 (def a (if 1
                                                                                                                                  (if true (setv c 2)
compile =====
 compile-atom =====
                                                                                                                                          (setv d 4))
compile-f: <function compile_expression at 0x10d49c500>
atom: (u'def' u'a' (u'if' 11 (u'if' u'True' (u'setv' u'c' 21) (u'setv' u'd' 41)) 31))
                                                                                                                                  3))
compile-atom =====
            <function compile def expression at 0x10d49c578>
atom: (u'def' u'a' (u'if' 1L (u'if' u'True' (u'setv' u'c' 2L) (u'setv' u'd' 4L)) 3L))
                                                                                                compile-def-expression doing
                                                                                                (setv result (.compile self result)) where result is the (if 1 (if true (setv c 2) (setv d 4)) 3)) form
compile-atom ===
compile-f: <function compile_expression at 0x10049c500>
atom: (u'if' 1L (u'if' u'rrue' (u'setv' u'c' 2L) (u'setv' u'd' 4L)) 3L)
compile—f: <function compile_if at 0x10d49c488> atom: (u'if' lL (u'if' u'True' (u'setv' u'c' 2L) (u'setv' u'd' 4L)) 3L)
                                                                                     compile-if doing
                                                                                      [condition (.compile self (.pop expression 0))]
 compile =====
compile-atom =====
compile-f: <function compile_integer at 0x10d49c6e0>
atom result 1: {'nodes': [1L], 'value': 1L}
                                                                                                                                      compile-atom doing (+ (Result) ret))
                                                                                                                                      Because result mlast `Number{1] is not a instance of Result
result before addition: {'stmts': [], 'temp_vars': [], '_Result_used_expr': False, '_expr': None}
result to be added: {'nodes': [III, 'value': IL} value': IL, 'value': IL, 'value': IL, 'value': IL, 'value': IL, 'value': IL, 'temp_vars': [], '_Result_used_expr': False, '_expr': None}
add result mlast `Number{1} to (Result) to convert it to a Result object
result to be added: {'stmts': [], 'temp_vars': [], '_Result_used_expr': False, '_expr': <class Number 1 at 0x10d315c90e}}
The messages display twice. First time it's adding the raw Number class to (Result), second time it's adding
                                                                                                                                      the converted Result object)
                                                                                    ompile-if doing
compile-atom =====
compile-f: <function compile_expression at 0x10d49c500> atom: (u'if' u'True' (u'setv' u'c' 2L) (u'setv' u'd' 4L))
                                                                                  [body (.compile self (.pop expression 0))]
                                                                                  remember the if expression is (if 1 (if true (setv c 2) (set d 4)) 3))
compile-atom =====
compile-f: <function compile if at 0x10d49c488>
                                                                                  where body is (if true (setv c 2) (setv d 4)) while the orel is just 3
atom: (u'if' u'True' (u'setv' u'c' 2L) (u'setv' u'd' 4L))
compile =====
                                                                    compile-if doing
compile-atom =====
compile-f: <function compile_symbol at 0x10d49c848>
                                                                    [condition (.compile self (.pop expression 0))]
                                                                    for (if true (setv c 2) (setv d 4))
atom result 1: {'nodes': [u'True']}
result before addition: {'stmts': [], 'temp_vars': [], '_Result__used_expr': False, '_expr': None} result to be added: {'nodes': [u'True']} result before addition: {'stmts': [], 'temp_vars': [], '_Result__used_expr': False, '_expr': None} result to be added: {'stmts': [], 'temp_vars': [], 'Result__used_expr': False, '_expr': <class Id True at 0x10d3f5510e} result after addition: {'stmts': [], 'temp_vars': [], '_Result__used_expr': True, '_expr': <class Id True at 0x10d3f5510e}
compile =====
                                                                                                   [body (.compile self (.pop expression 0))]
compile—atom ======
compile—f: <function compile_expression at 0x10d49c500>
                                                                                                   remember now the if expression is (if true (setv c 2) (setv d 4))
                                                                                                   where body is (setv c 2) while the orel is (setv d 4)
atom: (u'setv' u'c' 2L)
compile-atom ======
compile-f: <function compile_setv_expression at 0x10d49c5f0>
atom: (u'setv' u'c' 2L)
                                                                                    compile-setv doing (setv result (.compile self result))
compile =====
compile-f: <function compile integer at 0x10d49c6e0>
atom result 1: {'nodes': [2L], 'value': 2L}
```

compile-setv doing (setv ld-name (.compile self name))

the setv expression is (setv c 2)

result before addition: {'stmts': [], 'temp_vars': [], '_Result__used_expr': False, '_expr': None}

compile =====

compile-f: <function compile symbol at 0x10d49c848>

result to be added: {'nodes': [2L], 'value': 2L}
result to be added: {'nodes': [2L], 'value': 2L}
result before addition: {'stmts': [], 'temp_vars': [], 'Result_used_expr': False, '_expr': None}
result to be added: {'stmts': [], 'temp_vars': [], 'Result_used_expr': False, '_expr': <class Number 2 at 0x10d3f5150>}
result after addition: {'stmts': [], 'temp_vars': [], 'Result_used_expr': True, '_expr': <class Number 2 at 0x10d3f5150>}

```
atom: c
 atom result 1: {'nodes': [u'c']}
result before addition: {'stmts': [], 'temp_vars': [], '_Result__used_expr': False, '_expr': None} result to be added: {'nodes': [u'c']}
                                                                                                                                                                                                                                                                                                                                compile-atom converting Id c to result object
result to be added: {'nodes': [u'c']}
result before addition: {'stmts': [], 'temp_vars': [], '_Result_used_expr': False, '_expr': Kone}
result to be added: {'stmts': [], 'temp_vars': [], '_Result_used_expr': False, '_expr': <class Id c at 0x1003f53d0-}
result after addition: ('stmts': [], 'temp_vars': [], '_Result__used_expr': True, '_expr': <class Id c at 0x1003f53d0-}
                                                                                                                                                                                                                                                                                                                                    compile-setv doing (+= result (Set (mlast.Id [st-name]) [result.force-expr])).
                                                                                                                                                                                                                                                                                                                                    where the whole expresion is (setv c 2)
result before addition: ('stmts': [], 'temp_vars': [], 'Result_used_expr': True, 'expr': <class Number 2 at 0x10d3f5150+)
result to be added: ('nodes': [c-class Id [c-class Id c at 0x10d3f5200+] at 0x10d3f5200+
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  compile-setv doing (+= result ld-name)
result before addition: {'stmts': [<class Set nodes: [<class Id c at 0x10d3f53d0>] at 0x10d3f53d0>] at 0x10d3f5750>], 'temp_vars': [], 'Result_used_expr': True, '_expr': None} result to be added: {'stmts': [], 'temp vars': [], 'Result_used_expr': True, 'expr': class Id c at 0x10d3f53d0>] at 0x10d3f53d0>] at 0x10d3f53d0>] at 0x10d3f53f0>]
  result after addition: 's'ms's'mts': [class Ed nodes; - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 176, - 17
 atom result 1: {'stmts': [<class Set nodes: [<class Id (at 0x10d3f53d0>) at 0x10d3f53d0>], [<class Number 2 at 0x10d3f5150+]] at 0x10d3f5750>], 'temp_vars': [], '_Result__used_expr': True, '_expr': <class Id (at 0x10d3f53d0>) at 0x10d3f53d0>]
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       ompile-if finished [body (.compile self (.pop expression 0))]
 atom result 1: {'stmts': [<class Set nodes: [<class Id (<class Id c at 0x10d3f53d0»] at 0x10d3f53d0», [<class Number 2 at 0x10d3f5150»]] at 0x10d3f5150»], 'temp_vars': [], '_Result_used_expr': True, '_expr': <class Id c at 0x10d3f53d0»} use the whole if expression is (\f true (setv c 2) (setv d 4))
                                                                                                                                                           compile-if doing orel part
  compile-atom =====
                                                                                                                                                          where the whole if expression is (if true (setv c 2) (setv d
  compile-f:
                              <function compile expression at 0x10d49c500>
  atom: (u'setv' u'd' 4L)
 compile-atom =====
compile-f: <function compile_setv_expression at 0x10d49c5f0>
atom: (u'setv' u'd' 4L)
 compile =====
 compile—atom ======
compile—f: <function compile integer at 0x10d49c6e0>
 atom result 1: {'nodes': [4L], 'value': 4L}
result before addition: {'stmts': [], 'temp_vars': [], '_Result__used_expr': False, '_expr': None} result to be added: {'nodes': [41], 'value': 41} result before addition: {'stmts': [], 'temp_vars': [], '_Result__used_expr': False, '_expr': None} result to be added: {'stmts': [], 'temp_vars': [], 'Result__used_expr': False, '_expr': <class Number 4 at 0x10d3f5e10>} result after addition: {'stmts': [], 'temp_vars': [], 'Result__used_expr': True, '_expr': <class Number 4 at 0x10d3f5e10>}
 compile =====
  compile-atom =====
 compile-f: <function compile symbol at 0x10d49c848>
 atom result 1: {'nodes': [u'd']}
result before addition: {'stmts': [], 'temp_vars': [], '_Result__used_expr': False, '_expr': None} result to be added: {'nodes': [u'd']}
                                                                                                                                                                                                                                                                                             compile-atom converting Id d to result object
 result to De added: ('nodes': |u'd'|)
result before addition: ('stmts': [], 'temp_vars': [], '_Result_used_expr': False, '_expr': Kone}
result to De added: ('stmts': [], 'temp_vars': [], '_Result_used_expr': False, '_expr': <class Id d at 0x10d3f59d0>)
result after addition: ('stmts': [], 'temp_vars': [], '_Result_used_expr': True, '_expr': <class Id d at 0x10d3f59d0>)
 result before addition: {'stmts': [], 'temp_vars': [], '_Result_used_expr': True, '_expr': <class Number 4 at 0x10d3f5e10>} result to be added: {'nodes': [<class Id [<class Id d at 0x10d3f5900>] at 0x10d3f5909, [<class Number 4 at 0x10d3f5e10>]]}
result before addition: {'stmts': [], 'temp_vars': [], '_Result__used_expr': True, '_expr': <class Number 4 at 0x10d3f5e10>] result to be addet: {'stmts': [cclass Set nodes: |cclass Id |cclass Id | at 0x10d3f5e10>] at 0x10d3f5e
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    compile-setV doing (+= result ld-name)
result before addition: {'stmts': [<class Set nodes: [<class Id (<class Id d at 0x10d3f5900>] at 0x10d3f5900>] at 0x10d3f5900>] at 0x10d3f5900>], 'temp_vars': [], '_Result__used_expr': True, '_expr': None} where the S result to be added: '\stmts': [], 'temp_vars': [], 'Result__used_expr': True, '_expr': <class Id d at 0x10d3f5900>] at 0x10d3f5900>] at 0x10d3f5900>], 'temp_vars': [], 'Result__used_expr': True, '_expr': <class Id d at 0x10d3f5900>] at 0x10d3f5900>], 'temp_vars': [], 'Result__used_expr': True, '_expr': <class Id d at 0x10d3f5900>]
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    where the setw expression is (setv d 4)
 atom result 1: {'stmts': [<class Set nodes: [<class Id (<class Id (at 0x1003f5900>) at 0x1003f5900>} at 0x1003f5900>, [<class Number 4 at 0x1003f5900], 'temp_vars': [], '_Result_used_expr': True, '_expr': <class Id d at 0x1003f5900>} compile-if finshed orel part
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             where the whole if expression is (if true (setv c 2) (setv d 4
 atom result 1: {'stmts': [<class Set nodes: [<class Id (<class Id d at 0x10d3f59d0>] at 0x10d3f59d0>] at 0x10d3f59d0>], 'temp vars': [], 'Result used expr': True, 'expr': <class Id d at 0x10d3f59d0>]
result before addition: {'stmts': [], 'temp_vars': [], '_Result_used_expr': False, '_expr': None} result to be added: {'nodes': [[<class Id hua_anon.var_l at &xIdd3f580=], []]} result before addition: {'stmts': [], 'temp_vars': [], 'Result_used_expr': False, '_expr': None}
                                                                                                                                                                                                                                                                                                            compile-if doing (setv ret (+ (Result) (mlast.Local [var]) ret))
                                                                                                                                                                                                                                                                                                            where var is generated anonymous variable
```

but how body's result is stored in expr? Rember the body is a sety expression. In compile-assign we explicitly store the result of body in the "name" and append the "name" to the result of compiling in the following line: (+= result ld-name)

compile-if doing (setv ret (+ (Result) (mlast.Local [var]) ret)) remember the if expression is (if 1 (if true (setv c 2) (set d 4)) 3))

compile-if doing (setv ret (+ (Result) (mlast.Local [var]) ret)) where var is generated anonymous variable result to be added: {'stmts': [<class Local nodes: [[<class Id _hua_anon_var_1 at 0x10d3f5690>], []] at 0x10d3f5990>], 'temp_vars': [], '_Result__used_expr': False, '_expr': None} result after addition: ('stmts': |<class Local nodes: [[<class Id _hua_anon_var_1 at 0x10d3f5690>], []] at 0x10d3f5990>], 'temp_vars': [], '_Result__used_expr': True, '_expr': None} compile-if doing (setv ret (+ (Result) (mlast.Local [var]) result before addition: {'stmts': [<class Local nodes: [[<class Id _hua_anon_var_1 at 0x10d3f5690>], []] at 0x10d3f5990>], 'temp_vars': [], '_Result__used_expr': True, '_expr': None} result to be added: ("stmts: [], 'temp vars': [], 'Result_used expr': True, 'expr': <alass Id True at 0x100375510>) result_used_expr': True, 'expr': <alass Id True at 0x100375900>], 'ltemp_vars': [], 'Result_used_expr': True, 'expr': <class Id True at 0x100375510>) pile-if doing (+= body (mlast.Set [var] [body.force-expr])) result before addition: {'stmts': [<class Set nodes: [<class Id < at 0x10d3f53d0+] at 0x10d orel (mlast.Set [var] [orel.force-expr])) result before addition: {'stmts': [<class Set nodes: [<class Id [<class Id dat 0x1003f5900>] at 0x1003f5900>] at 0x1003f5900>] at 0x1003f5900>], 'temp_vars': [], '_Result__used_expr': True, '_expr': <class Id d at 0x1003f5900>] [], '_Result__used_expr': True, '_expr': None} result before addition: ('stmts': [<class Local nodes: [[<class Id hua_anon_var_1 at 0x100375590>], []] at 0x100375590>], 'temp_vars': [], 'Result_used_expr': True, 'expr': <class Id True at 0x100375590>] at 0x100375590>], []] at 0x100375590>], 'temp_vars': [], 'Result_used_expr': True, 'expr': <class Id True at 0x100375590>] at 0x100375590>], 'cclass Set nodes: [{<class Id hua_anon_var_1 at 0x100375590>], []] at 0x100375590>], 'cclass Number 2 at 0x100375590>] at 0x100375590>], 'cclass Set nodes: [{<class Id hua_anon_var_1 at 0x100375590>], 'cclass Id c at 0x100375590>], 'cclass Number 2 at 0x100375590>], 'cclass Set nodes: [{<class Id hua_anon_var_1 at 0x100375590>], 'cclass Number 2 at 0x100375500>], 'cclass Set nodes: [{<class Id Add at 0x100375590>], 'cclass Id at 0x100375500>], 'ccl at 0x1003753009]] at 0x1003753009], ctcluss set noues: [<class Id | cclass ad | cclass Id | cclass ad | cclass Id | cclass ad | cclass | c compile-if doing (+= ret (apply Result | | f"expr" expr-name "temp yars" | expr-name var|1)). If's expr is the anonymous variable introduced at the beginning, why temp-yars is done this way? Compile-1r doing (+= ret (apply Kespir expr-name var); | x-result before addition: ('stats': (class Local nodes: [<class Id Jua anno. var] at 0x1003755000-), [| at 0x1003755000-], | x-result before addition: ('stats': (class Id | nua anno. var] at 0x1003755000-), | x-result before addition: ('stats') at 0x1003755000-), | x-result befo hua_anon_var_1 at 0x10d3f5590-], [cclass Id d at 0x10d3f5900-], at 0x10d3f5900-], at 0x10d3f5900-]] at 0x10d3f5900-], at atom result 1: ('stmts': [<class Local nodes: [[<class Id _hua_anon_var_1 at 0x1003f56000-], []] at 0x1003f55000-, <class If nodes: [<class Id True at 0x1003f5500-, [<class Set nodes: [<class Id _cat 0x1003f5500-] at 0x1003f55000-]] at 0x1003f55000-]] at 0x1003f55000-]] at 0x1003f55000-], (*class Id _cat 0x1003f55000-], (*class Id _cat 0x1003f55000-], (*class Id _cat 0x1003f55000-]] at 0x1003f55000-], (*class Id _cat 0x1003f55000-], (*class Id _cat 0x1003f55000-]] at 0x1003f55000-], (*class Id _cat 0x1003f5000-], (*c atom result 1: ('stmts': [<class Local nodes: [[<class Id _hua_anon_var_1 at 0x1003f56000-], [[] at 0x1003f55000-, <class If nodes: [<class Id True at 0x1003f5500-, [<class Set nodes: [<class Id (<class Id _hua_anon_var_1 at 0x1003f5600-)] at 0x1003f5600-]] at 0x1003f5600-]] at 0x1003f5600-], [<class Id _hua_anon_var_1 at 0x1003f5600-], [<class Id _hua_anon_var_1 at 0x1003f5600-]] at 0x1003f5600-]] at 0x1003f5600-]] at 0x1003f5600-], [<class Id _hua_anon_var_1 at 0x1 remember the if expression is (if 1 (if true (setv c 2) (set d 4)) 3)) compile-if doing [orel (.compile self (.pop expression 0))] compile-atom ===== remember the if expression is (if 1 (if true (setv c 2) (set d 4)) 3)) compile-f: <function compile integer at 0x10d49c6e0> atom result 1: {'nodes': [3L], 'value': 3L} result before addition: {'stmts': []. 'temp vars': []. ' Result used expr': False. ' expr': None} result before addition: ('Stmts: 1), temp vars: [], _mesult_used_expr: raise, _expr: mone; result to be added: ('nodes: [Bl], 'value': 3L] result before addition: ('stmts': [], 'temp_vars': [], '_Result_used_expr': False, '_expr': <class Number 3 at 0x10d3f5610>} result to be added: ('stmts': [], 'temp_vars': [], '_Result_used_expr': False, '_expr': <class Number 3 at 0x10d3f5610>} result after addition: ('stmts': [], 'temp_vars': [], '_Result_used_expr': True, '_expr': <class Number 3 at 0x10d3f5610>} result before addition: {'stmts': [], 'temp_vars': [], '_Result_used_expr': False, '_expr': None} compile-if doing (setv ret (+ (Result) (mlast.local [var]) ret))
result to be added: {'nodes': [[c-lass Id _hua_anon.var_2 at 0xi0d3750900-], []] result_used_expr': raise, '_expr': None}
result before addition: {'stmts': [], 'temp_vars': [], 'Result_used_expr': raise, '_expr': None}
result to be added: {'stmts': [-class local nodes: [[c-lass Id _hua_anon.var_2 at 0xi0d3750900-], []] at 0xi0d375000-], 'temp_vars': [], 'Result_used_expr': False, '_expr': None}
result after addition: {'stmts': [-class local nodes: [[c-lass Id _hua_anon.var_2 at 0xi0d375000-], []] at 0xi0d375000-], 'temp_vars': [], 'Result_used_expr': True, '_expr': None}

result before addition: {'stmts': [<class Local nodes: [<class Id hua anon_var_1 at 0x10d3f5500b-], [<class Id nodes: [<class Id rough of 10x10d3f5500b-], [<class Set nodes: [<class Id hua anon_var_1 at 0x10d3f5500b-], [<class Id cat 0x10d3f5500b-], [<class Set nodes: [<class Id dat 0x10d3f5500b-], [<class Id

result before addition: {'stmts': [<class Local nodes: [[<class Id _hua_anon_var_2 at 0x1003f5d90-], []] at 0x1003f5d90-], 'temp_vars': [], '_Result__used_expr': True, '_expr': Moment of doing result to be added: ('stmts': [], 'temp_vars': [], '_Result__used_expr': True, '_expr': <class Number 1 at 0x1003f5c90-}, 'temp_vars': [], '_Result__used_expr': True, '_expr': <class Number 1 at 0x1003f5c90-}, 'temp_vars': [], '_Result__used_expr': True, '_expr': <class Number 1 at 0x1003f5c90-}

hua_anon_var_1 at 0x1003f5690>], [<class Id d at 0x1003f5900>]] at

compile-if doing (+= body (mlast.Set [var] [body.force-expr])) remember the if expression is (if 1 (if true (setv c 2) (set d 4)) 3))

compile-if doing (+= orel (mlast.Set [var] [orel.force-expr]))

result before addition: {'stmts': [], 'temp_vars': [], '_Result__used_expr': True, '_expr': <class Number 3 at 0x10d3f5610>} result to be added: {"nodes: [[<] sets I], _kesult_used_expr: rue, _expr: <class number 3 at 0x1003750109} result to be added: {"nodes: [[<] ss I d hua anno yar_2 at 0x1003750409, [<] class Number 3 at 0x1003750109] result before addition: {"stmts: [], "hespult_used_expr': True, _expr': <class Number 3 at 0x1003750109] at 0x1003751009], 'temp_vars': [], "Result_used_expr': False, '_expr': None} result to be added: {\stmts: [<] starts: [class Set nodes: [[<] lass Id_hua_anon_var_2 at 0x1003750409], |<class Number 3 at 0x1003750109] at 0x1003751909], 'temp_vars': [], '_Result_used_expr': False, '_expr': None} result after addition: {\stmts: [<] starts: [<] starts: [], '_Result_used_expr': True, _expr': None} result after addition: {\stmts: [] starts: [], '_Result_used_expr': True, _expr': None} result after addition: {\stmts: [] starts: [], '_Result_used_expr': True, _expr': None} result after addition: {\stmts: [] starts: [], '_Result_used_expr': True, _expr': None} result after addition: {\stmts: [] starts: [], '_Result_used_expr': True, _expr': None} result after addition: {\stmts: [] starts: [], '_Result_used_expr': True, _expr': None} result after addition: {\stmts: [] starts: [], '_Result_used_expr': True, _expr': None} result after addition: {\stmts: [] starts: [], '_Result_used_expr': True, _expr': None} result after addition: {\stmts: [] starts: [], '_Result_used_expr': True, _expr': None} result after addition: {\stmts: [] starts: [], '_Result_used_expr': True, _expr': None} result after addition: {\stmts: [] starts: [], '_Result_used_expr': True, _expr': None} result after addition: {\stmts: [] starts: [], '_Result_used_expr': True, _expr': None} result after addition: {\stmts: [] starts: [], '_Result_used_expr': True, _expr': None} result after addition: {\stmts: [] starts: [], '_Result_used_expr': True, _expr': None} result after addition: {\stmts: [] starts: [], '_Result_used_expr': True, _expr': None} result after addition: {\stmts: [] starts: [], '_Result_used_expr': True, _expr': None} result aft compile-if doing (+= ret (mlast.If ret.force-expr body.stmts orel.stmts)) result before addition: ('stats': [<class Local nodes: [<class Id _hua_anon_var_2 at 0x1003755090-], [class Id _hua_anon_var_1 at 0x100375909-], [cl remember the if expression is (if 1 (if true (setv c 2) (set d 4)) 3)) result before addition, ('stmts': [<class Local nodes: [<class Id hua anon var 2 at 0x10d375900-], []] at 0x10d375700-], (<class Id nodes: [<class Id hua anon var 1 at 0x10d375900-], (<class Id hua anon var 1 at 0x10d375900-), (<class Id hua anon var 1 at 0x10d375900-), (<class Id cat 0x10d375900-), (<class Id cat 0x10d375900-), (<class Id hua anon var 1 at 0x10d375900-), (<class Id hua anon var 2 at 0x10d375900-), (<class Id hua anon var 1 at 0x10d375900-), atom result 1: {'stmts': [<class Local nodes: [[<class Id _hua_anon_var_2 at 0x1003f55090, (<class If nodes: [<class Id _hua_anon_var_1 at 0x1003f55090, []] at 0x1003f55000, [<class If nodes: [<class Id _hua_anon_var_1 at 0x1003f55000], []] at 0x1003f55000, [<class Id True at 0x1003f55100, [<class Id _hua_anon_var_1 at 0x1003f55000], [<class Id _hua_anon_var_2 at 0x1003f55000], atom result 1: {'stmts': [<class Local nodes: [{<class Id _hua_anon_var_2 at 0x1003f5500>, []] at 0x1003f5500>, (<class If nodes: [<class Id True at 0x1003f5510>, [<class Et nodes: [<class Id _hua_anon_var_1 at 0x1003f5500>], []] at 0x1003f5500>, (<class Id True at 0x1003f5510>, [<class Et nodes: [<class Id _hua_anon_var_1 at 0x1003f5500>], []] at 0x1003f5500>, (<class Id True at 0x1003f5510>, (<class Id _hua_anon_var_1 at 0x1003f5500>], []] at 0x1003f5500>, (<class Id _hua_anon_var_1 at 0x1003f5500>), []] at 0x1003f5500>, (<class Id _hua_anon_var_1 at 0x1003f5500>), [<class Id _hua_anon_var_2 at 0x1003f5500>], [<class Id _hu _hua_anon_var_2 at 0x10d3f5d90>], '_Result__used_expr': True, '_expr': <class Id _hua_anon_var_2 at 0x10d3f5710>} compile-if doing final step remember the if expression is (if 1 (if true (setv c 2) (set d 4)) 3)) compile-atom ===== compile-f: <function compile_symbol at 0x10d49c848> compile-assign (setv ld-name (.compile self name)) atom result 1: {'nodes': [u'a']} In the between process, we rename the temp-vars _hua_anon_var_2 to name instead of assign the result to name. Compare the temp-vars before and after result before addition: {'stmts': [], 'temp_vars': [], '_Result__used_expr': False, '_expr': None} result to be added: {'nodes': [u'a']} result_used_expr: ratse, _expr: wone;
result to be added: {'nodes': [u'a']}
result before addition: {'stmts': [], 'temp_vars': [], '_Result_used_expr': False, '_expr': None}
result to be added: {'nodes': [u'a']}
result to be added: {'stmts': [], 'temp_vars': [], '_Result_used_expr': False, '_expr': «class Id a at 0x10d3f5c00>}
result after addition: {'stmts': [], 'temp_vars': [], '_Result_used_expr': True, '_expr': «class Id a at 0x10d3f5c00>} compile-assign (+= result ld-name) result before addition: ('stmts': [<class Local nodes: [(<class Id a at 0x100375900-], []] at 0x100375700-, (<class If nodes: [<class Id nodes: [(<class Id nodes: [(

atom result 1: {'stmts': [<class Local nodes: [[<class Id a at 0x10d3f5909>], []] at 0x10d3f5909>, <class If nodes: [<class Id True at 0x10d3f5510>, [<class Local nodes: [<class Id _hua_anon_var_1 at 0x10d3f5509>], []] at 0x10d3f5909>, <class If nodes: [<class Id _hua_anon_var_1 at 0x10d3f5509>], []] at 0x10d3f5909>, <class If nodes: [<class Id _hua_anon_var_1 at 0x10d3f5509>], []] at 0x10d3f5909>, <class Id True at 0x10d3f5909>, <class Id _hua_anon_var_1 at 0x10d3f5909>], []] at 0x10d3f5909>, <class Set nodes: [[<class Id _hua_anon_var_1 at 0x10d3f5909>], [<class Number 4 at 0x10d3f5909>], [<class Id _hua_anon_var_1 at 0x10d3f5909>], [<class

atom result 1: {'stmts': {class Local nodes: [{class Id a at 0x1003f5909>, []] at 0x1003f5909>, {class If nodes: {class Id True at 0x1003f5509}, {class Local nodes: [{class Id hua anon_var_1 at 0x1003f5509>}, {class Id c at 0x1003f5509>}, {class Id class Id class

local a

if 1 then
local _hua_anon_var_1
if True then