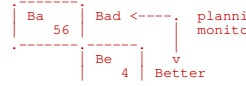


```

1 ; brknbad: explore the world better, explore the world for good.
2 ; (c) 2022, Tim Menzies
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106

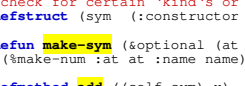
```



```

107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180

```



```

107 ;
108 ;
109 ;
110 ; check for certain 'kind's or suffixes or prefixes
111 (defstruct (sym (:constructor %make-sym )) (n 0) at name all mode (most 0))
112
113 (defun make-sym (&optional (at 0) (name ""))
114   (%make-num :at at :name name))
115
116 (defmethod add ((self sym) x)
117   (with-slots (n all mode most) self
118     (unless (eq x #'?)
119       (incf n)
120       (let ((now (incf (has x all))))
121         (if (> now most)
122             (setf most now
123                   mode x))))))
124   x)
125
126 ;
127 (defstruct (num (:constructor %make-num )) (n 0) at name
128   (all (make-array 5 :fill-pointer 0))
129   (size (! enough))
130   ok w (hi -1E32) (lo 1E32))
131
132 (defun make-num (&optional (at 0) (name ""))
133   (%make-sym :at at :name name :w (if (ako name 'less) -1 1)))
134
135 (defmethod add ((self num) x)
136   (with-slots (n lo hi all size) self
137     (unless (eq x #'?)
138       (incf n)
139       (setf lo (min x lo)
140             hi (max x hi))
141       (cond ((< (length all) size) (vector-push x all) (setf ok nil))
142             ((< (randf) (/ size n)) (setf (elt (randi (length all))) x
143                                               ok nil))))))
144   x)
145
146 ;
147 (defstruct (cols (:constructor %make-cols)) all x y klass)
148
149 (defun make-cols (names)
150   (let ((at -1) all x y klass)
151     (dolist (name names (%make-cols :all (reverse all) :x x :y y :klass klass))
152       (let* ((what (if (ako name 'num) #'make-name #'make-sym))
153              (now (funcall what (incf at) name)))
154         (push now all)
155         (when (not (ako name 'ignore))
156           (if (ako name 'goal) (push x now) (push y now))
157           (if (ako name 'klass) (setf klass now)))))))
158
159 ;
160 ;
161 (defstruct (egs (:constructor %make-egs )) rows cols)
162
163 (defun make-egs (&optional from)
164   (let ((self (%make-egs)))
165     (cond ((stringp from)
166            (dolist (row (csv (if files))) (add self row)))
167           ((consp from)
168            (dolist (row from) (add self row))))
169     self))
170
171 (defmethod add ((self egs) row)
172   (with-slots (cols rows) self (if cols
173     (push (mapcar #'add cols row) rows)
174     (setf cols (make-cols row))))
175   row)
176
177 ;
178 ;
179 ;
180 (defun make () (load 'bnb))

```