DYALOG

APL Germany

Fun with array notation and programmatic variable assignment

Adám Brudzewsky

Array Notation?

```
[
  'a'
  'b'
]
```

```
['a' ◇ 'b']
```

```
(
  'eh'
  'bee'
)
```

```
(
  a: 'eh'
  b: 'bee'
)
```

```
()
```

Traditional code

```
∇ seq←Range(from step to)
  seq←from,from+step×⍳step÷⍨to-from
∇
      Range 10 2 18
10 12 14 16 18
```

Traditional code

```
∇ seq←Range args;from;step;to
  :Select ≢args
  :Case 3 ⋄ (from step to)←args
  :Case 2 ⋄ (from step to)←1(↑,1,↓)args
  :Case 1 ⋄ (from step to)←0 1,args
  :EndSelect
  seq←from,from+step×⍳step÷⍨to-from
∇
```


Fun with array notation and programmatic variable assignment

Traditional code

```
∇ seq←Range args;from;step;to

  (from step to)←(58⍴args,0 1)[41 12 58]

  seq←from,from+step×⍳step÷⍨to-from
∇
```


Fun with array notation and programmatic variable assignment

Traditional code

```
∇ seq←Range args;from;step;to
  (from step to)←(58⍴args,0 1)[41 12 58]
  seq←from,from+step×⍳step÷⍨to-from
∇
```

```
      Range 10 18
10 11 12 13 14 15 16 17 18
```

Parameter namespace argument

```
∇ seq←Range params;from;step;to
  (from step)←0 1
  ⎕THIS ⎕NS params
  seq←from,from+step×⍳step÷⍨⍤⍨to-from
∇
      p←⎕NS⍬
      p.from←10 ⋄ p.to←18
      Range p
10 11 12 13 14 15 16 17 18
```


Fun with array notation and programmatic variable assignment

Parameter namespace argument

```
∇ seq←Range params;from;step;to
  (from step)←0 1
  ⎕THIS ⎕NS params
  seq←from,from+step×⍳step÷⍨⍥⍨to-from
∇
```

```
      p←(from:10 ⋄ to:18)
      Range p
10 11 12 13 14 15 16 17 18
```


Fun with array notation and programmatic variable assignment

Parameter namespace argument

```
∇ seq←Range params;from;step;to
  (from step)←0 1
  ⎕THIS ⎕NS params
  seq←from,from+step×⍳step÷⍨⍥⍨to-from
∇
```

```
      Range(from:10 ◇ to:18)
10 11 12 13 14 15 16 17 18
```


Fun with array notation and programmatic variable assignment

Parameter namespace argument

```
∇ seq←Range params;from;step;to

  ⎕THIS ⎕NS(from:0 ⋄ step:1)params
  seq←from,from+step×⍳step÷⍨to-from
∇
```

```
      Range(from:10 ⋄ to:18)
10 11 12 13 14 15 16 17 18
```


Fun with array notation and programmatic variable assignment

Parameter namespace argument

```
∇ seq←Range params

  ns←⎕NS(from:0 ⋄ step:1)params
  seq←ns.(from,from+step×⍳step÷⍨to-from)
∇
```

```
      Range(from:10 ⋄ to:18)
10 11 12 13 14 15 16 17 18
```

Parameter namespace argument

```
∇ seq←Range params

  ns←(from:0 ⋄ step:1)⎕NS params
  seq←ns.(from,from+step×⍳step÷⍨to-from)
∇
```

```
      Range(from:10 ⋄ to:18)
10 11 12 13 14 15 16 17 18
```


Fun with array notation and programmatic variable assignment

Parameter namespace argument

```
∇ seq←Range params

  seq←((from:0 ⋄ step:1)⎕NS params).(from,from+s

∇
```

```
      Range(from:10 ⋄ to:18)
10 11 12 13 14 15 16 17 18
```

Parameter namespace argument

```
∇ seq←Range params
  seq←((
    from:0
    step:1
  )⎕NS params).(from,from+step×⍳step÷⍨to-from)
∇
```

```
      Range(from:10 ⋄ to:18)
10 11 12 13 14 15 16 17 18
```


Fun with array notation and programmatic variable assignment

Parameter namespace argument

```
∇ seq←Range params
  seq←((
    from:0
    step:1
  )⍎⍕NS params).(from,from+step×⍳step÷⍨to-from)
∇
```

```
      Range(from:10 ⋄ to:18)
10 11 12 13 14 15 16 17 18
```


Fun with array notation and programmatic variable assignment

Parameter namespace argument

```
∇ seq←Range params
  seq←((
    from:0
    step:1   ⍝ must be positive
  )⎕NS params).(from,from+step×⍳step÷⍨to-from)
∇
```

```
      Range(from:10 ⋄ to:18)
10 11 12 13 14 15 16 17 18
```


Fun with array notation and programmatic variable assignment

Parameter namespace argument

```
Range←{((
     from:0
     step:1   ⍝ must be positive
   )⎕NS ⍵).(from,from+step×⍳step÷⍨to-from)}
```

```
      Range(from:10 ⋄ to:18)
10 11 12 13 14 15 16 17 18
```

Programmatic assignment?

```
'ns1' ⎕NS 'v1' 'v2'
⎕NS ns2
```

```
ns1 ⎕NS 'v1' 'v2'
⎕NS ns2 ns3 ns4
```

```
{ns⍎⍺,'←⍵'}⍤⍥⊂nv1 nv2
names{ns⍎⍺,'←⍵'}⍤⍥⊂vals
```

```
ns ⎕VSET nv1 nv2
ns ⎕VSET (↑names)vals
() ⎕VSET ...
```

💡 Programmatic retrieval

```
vals←ns⍎¨names
```

```
vals←ns ⎕VGET names
```

```
nvs←{⍵(⍎¨⍵)}¨⎕NL ¯2
```

```
nvs←⎕VGET ¯2
```

```
(names vals)←{⍵(⍎¨↓⍵)}⎕NL 2
```

```
(names vals)←⎕VGET 2
```


Fun with array notation and programmatic variable assignment

Import to namespace

```
      200↑⊃⎕NGET'/d/aplde2025a/people.csv'
first,last,sex,dob,address,city,country
Uriab,Morris,M,20230512,"23, 14th Str",Jingdezhe
Martha,Mitchell,F,20131129,"473, 27th Av",Lyon,F
Lilah,Scott,F,20110527,"3483, 73rd Av",Northavon
```


Fun with array notation and programmatic variable assignment

Import to namespace

```
(Δtable Δcols)←⎕CSV csvFile ⍬ ⍬ 1


Δtable[;Δcols⍳⊆'last' 'first']
  ↳≠⍨Δtable[;Δcols⍳⊆'country']∊⊂'Germany'
Barnes  Idena
Cooper  Clyde
```


Fun with array notation and programmatic variable assignment

Import to namespace

```
      (Δvals Δcols)←⎕CSV⍠2⊢csvFile 0 0 1
      ⎕VSET(↑Δcols)Δvals
      (last,⍪first)⌿⍨country∈⊂'Germany'
Barnes  Idena
Cooper  Clyde
```

Import to namespace

```
      {⎕VSET(↑ω)α}/ ⎕CSV⍠2⊢csvFile ⍬ ⍬ 1

      (last,⍪first)⌿⍨country∈⊂'Germany'
Barnes  Idena
Cooper  Clyde
```


Fun with array notation and programmatic variable assignment

Import to namespace

```
      db←{⍺⍺⍵⍵VSET(↑⍵)⍺}/⍵⍵CSV⍠2⊢csvFile ⍬ ⍬ 1

      db.((last,⍪first)≠⍨⍤⊂country∈⊂'Germany')
Barnes  Idena
Cooper  Clyde
```


Fun with array notation and programmatic variable assignment

Import to namespace

```
      db←{()⎕VSET(↑ω)α}/⎕CSV⍠2⊢csvFile ⍬ ⍬ 1
      Where←{⊢()⎕VSET ω∘/¨@2¨α ⎕VGET ¯2}
      db Where db.country∊⊂'Germany'
#.[Namespace]
```

Import to namespace

```
db←{⍵⍎⍺,'←⎕VSET(↑⍵)⍺'}/⎕CSV⍠2⊢csvFile ⍬ ⍬ 1
Where←{⊢⍵⍎⍺,'←⎕VSET ⍵⍥⍕¨@2¨⍺ ⎕VGET ¯2'}
View←{⊃{(↓⍺)⌿⍵}⍵⍎⍺,'←⎕VGET 2'}
View db Where db.country∊⊂'Germany'
```

address	city	country	do
38, Linda Dr, Suite 209	Bochum	Germany	20
203, Mandel Av	Pforzheim	Germany	20

Fun with array notation and programmatic variable assignment

Programmatic retrieval

```
values←ns⍎¨names


values←ns ⎕VGET names
values←ns ⎕VGET nameMat
name_value_pairs←ns ⎕VGET ¯2
(nameMat values)←ns ⎕VGET 2
```


Fun with array notation and programmatic variable assignment

💡 Default values

```
values←ns⍎¨names
```

```
values←ns ⎕VGET (name1 val1) (name2 val2)
values←ns ⎕VGET nameMat values
```

Parameter namespace argument

```
∇ seq←Range params;from;step;to

  ⎕THIS ⎕NS(from:0 ⋄ step:1)params
  seq←from,from+step×⍳step÷⍨to-from
∇
```

```
      Range(from:10 ⋄ to:18)
10 11 12 13 14 15 16 17 18
```


Fun with array notation and programmatic variable assignment

Default values

```
∇ seq←Range params;from;step;to
  (from step to)←params ⎕VGET(
    'from' 0
    'step' 1
    'to'
  )
  seq←from,from+step×⍳step÷⍨to-from
∇
```


Fun with array notation and programmatic variable assignment

Default values

```
∇ seq←Range params;from;step;to

  (from step to)←params ⎕VGET('from' 0 ⋄ 'step'
  seq←from,from+step×⍳step÷⍨to-from

∇
```


Fun with array notation and programmatic variable assignment

Default values

```
∇ seq←Range params;from;step;to
  :Trap 0
    (from step to)←params ⎕VGET('from' 0 ⋄ 'step
    seq←from,from+step×⍳step÷⍨⍥⍨to-from
  :Else
    ⎕SIGNAL⊂⍨2⎕VGET⍨⍥⎕DMX.⎕NS'EN' 'EM' 'Message'
  :EndTrap
∇
```


Fun with array notation and programmatic variable assignment

Default values

```
∇ seq←{debug}Range params;from;step;to
  :Trap 0↓⍨⎕VGET⊂'debug' 0
    (from step to)←params ⎕VGET('from' 0 ⋄ 'step
    seq←from,from+step×⍳step÷⍨to-from
  :Else
    ⎕SIGNAL⊂⍨2⎕VGET⍨⎕DMX.⎕NS'EN' 'EM' 'Message'
  :EndTrap
∇
```


Fun with array notation and programmatic variable assignment

Default values

```
∇ seq←{debug}Range params;from;step;to
  :Trap ⎕VGET⊂'debug' 0
    (from step to)←params ⎕VGET('from' 0 ⋄ 'step
    seq←from,from+step×⍳step÷⍨to-from
  :Else
    ⎕SIGNAL⊂⍨2⎕VGET⍨⎕DMX.⎕NS'EN' 'EM' 'Message'
  :EndTrap
∇
```


Fun with array notation and programmatic variable assignment

Summary

Matrix, Vector: `['a' ⋄ 'b']  ('eh' ⋄ 'bee')`

Namespace: `(a:'eh' ⋄ b:'bee')  ()`

Merge Namespace: `ns1 ⎕NS ns2  ⎕NS ns1 ns2`

Value Get:

```
⎕VGET names      ⎕VGET nameMat
⎕VGET ¯2         ⎕VGET 2
⎕VGET nv1 nv2    ⎕VGET nameMat values
```

Value Set: `⎕VSET nv1 nv2  ⎕VSET nameMat values`

Fun with array notation and programmatic variable assignment

Summary

Matrix, Vector: `['a' ⋄ 'b']  ('eh' ⋄ 'bee')`

Namespace: `(a:'eh' ⋄ b:'bee')  ()`

Merge Namespace: `ns1 ⎕NS ns2  ⎕NS ns1 ns2`

Value Get:

```
⎕VGET names        ⎕VGET nameMat
⎕VGET ¯2           ⎕VGET 2
⎕VGET nv1 nv2      ⎕VGET nameMat values
```

defaults

Value Set: `⎕VSET nv1 nv2  ⎕VSET nameMat values`