Thinking in APL: Array-oriented Solutions (Part 2)

Richard Park

Previously

Thinking in APL: *Array-oriented Solutions* **Part 1**

dyalog.tv/Webinar/?v=myoK22rq1jk

Heuristics

Metzger, R.C., 1981.

APL thinking finding array-oriented solutions.

ACM SIGAPL APL Quote Quad, *12*(1), pp.212-218.

Heuristics

Metzger, R.C., 1981.

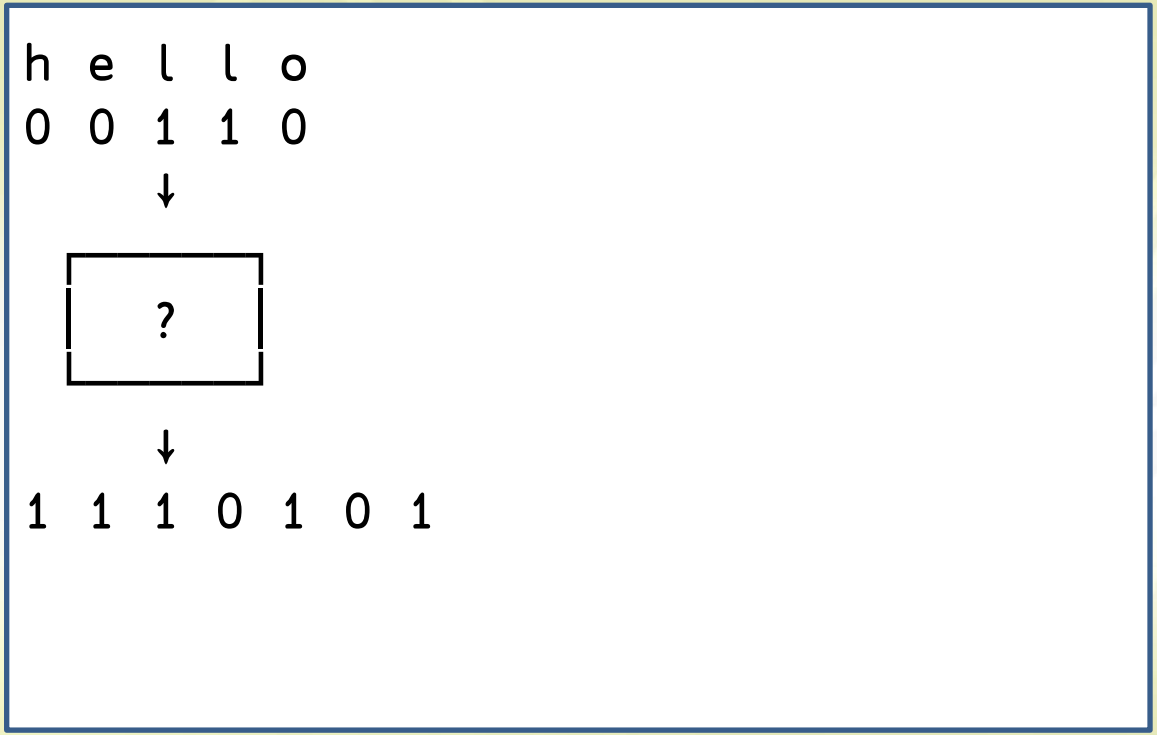
APL thinking finding array-oriented solutions.

1) Value First, Then Shape;
2) Shape First, Then Value;
3) **Data Transformation;**
4) **Loop First;**
5) Think Big;
6) Function Listing;
7) Synonym Search.

Black Box / Data Transformation

```
      b←'l'=t←'hello' ⋄ ↑t b
h e l l o
0 0 1 1 0
      ⎕←e←ExpansionVector b
1 1 1 0 1 0 1
      e\t
hel l o
```

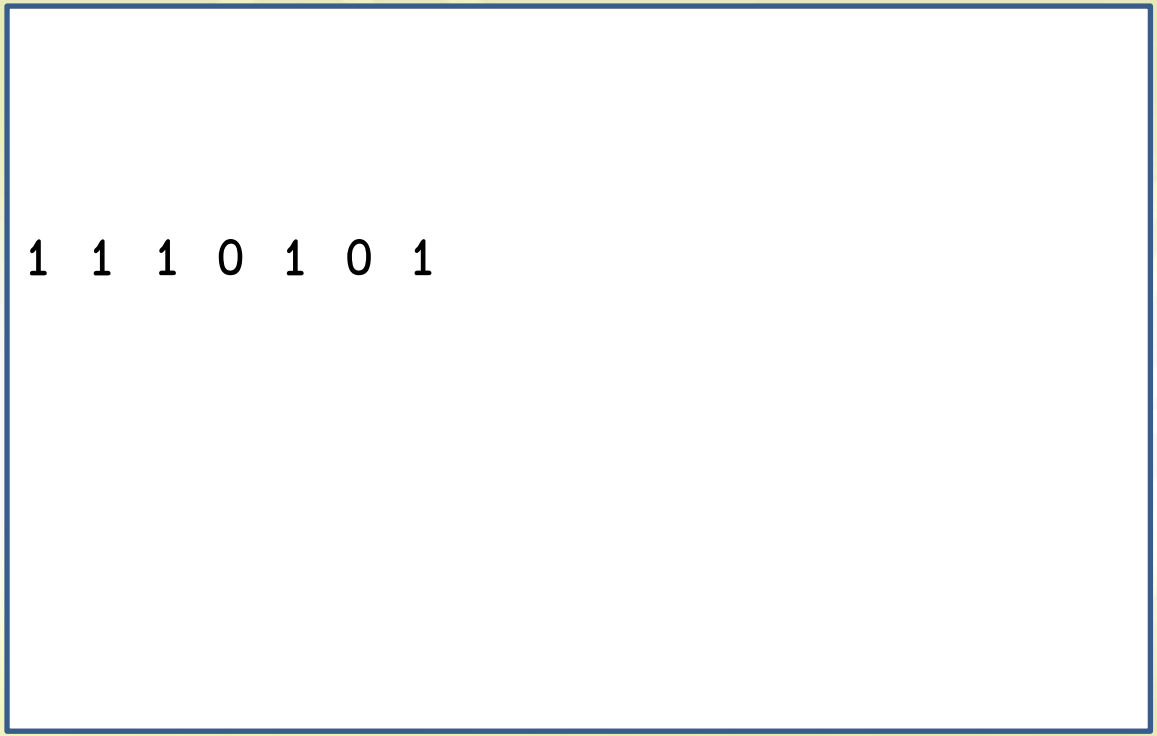
Black Box / Data Transformation



Black Box / Data Transformation

```
h e l l o
0 0 1 1 0

1 1 1 0 1 0 1
```

Black Box / Data Transformation




```
h e l l o
0 0 1 1 0
```

Black Box / Data Transformation


```
h e l l o
0 0 1 1 0
```

Black Box / Data Transformation

```
1
1
1  0

1  0

1
```

```
h  e  l  l  o
0  0  1  1  0
```

Black Box / Data Transformation

```
  1
  1
1 0
1 0
  1
```

```
h e l l o
0 0 1 1 0
```

Black Box / Data Transformation

```
1 1
1 1
1 0
1 0
1 1
```

```
h e l l o
0 0 1 1 0
```

Black Box / Data Transformation

```
1  1     1
1  1     1
1  0     1  0
1  0     1  0
1  1     1
```

```
h  e  l  l  o
0  0  1  1  0
```

Black Box / Data Transformation

```
1 1     1 1
1 1     1 1
1 0     1 0
1 0     1 0
1 1     1 1
```

```
h e l l o
0 0 1 1 0
```

Black Box / Data Transformation

```
1 1     1 1     1
1 1     1 1     1
1 0     1 0     1 0
1 0     1 0     1 0
1 1     1 1     1
```

```
h e l l o
0 0 1 1 0
```

Black Box / Data Transformation

```
1  1      1  1        1
1  1      1  1        1
1  0      1  0     1  0
1  0      1  0     1  0
1  1      1  1        1
```

```
h  e  l  l  o
0  0  1  1  0
```

Black Box / Data Transformation

```
1 1     1 1     0 1
1 1     1 1     0 1
1 0     1 0     1 0
1 0     1 0     1 0
1 1     1 1     0 1
```

```
h e l l o
0 0 1 1 0
```

Black Box / Data Transformation

```
1 1     1 1     0 1     1
1 1     1 1     0 1     1
1 0     1 0     1 0     1 0
1 0     1 0     1 0     1 0
1 1     1 1     0 1     1
```

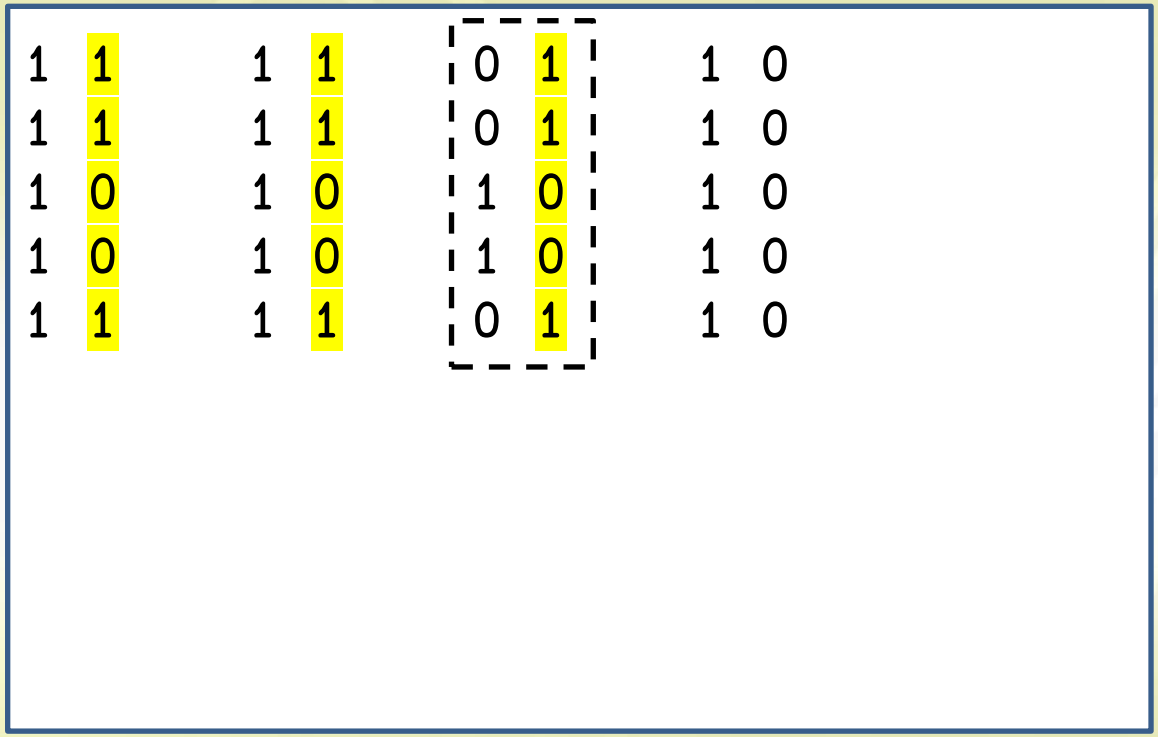
```
h e l l o
0 0 1 1 0
```

Black Box / Data Transformation

```
1 1   1 1   0 1   1 0
1 1   1 1   0 1   1 0
1 0   1 0   1 0   1 0
1 0   1 0   1 0   1 0
1 1   1 1   0 1   1 0
```

```
h e l l o
0 0 1 1 0
```

Black Box / Data Transformation



h e l l o
0 0 1 1 0

Black Box / Data Transformation

```
0 1
0 1
1 0
1 0
0 1
```

```
h e l l o
0 0 1 1 0
```

Black Box / Data Transformation

```
0 1
0 1
1 0
1 0
0 1

  ↓?

1 1 1 0 1 0 1
```

```
h e l l o
0 0 1 1 0
```

Black Box / Data Transformation

```
0 1       ⍵ Remove 0
0 1       ⍵ Remove 0
1 0       ⍵ Keep
1 0       ⍵ Keep
0 1       ⍵ Remove 0

  ↓

1 1 1 0 1 0 1
```

```
h e l l o
0 0 1 1 0
```

Black Box / Data Transformation

```
0 1       ⍵ Remove 0 → 0 1
0 1       ⍵ Remove 0 → 0 1
1 0       ⍵ Keep     → 1 1
1 0       ⍵ Keep     → 1 1
0 1       ⍵ Remove 0 → 0 1

  ↓

1 1 1 0 1 0 1
```

```
h e l l o
0 0 1 1 0
```

Black Box / Data Transformation

```
0 1          0 1
0 1          0 1
1 0   /⍨      1 1
1 0          1 1
0 1          0 1

  ↓

1 1 1 0 1 0 1
```

```
h e l l o
0 0 1 1 0
```

Black Box / Data Transformation

```
0 1          0 1
0 1          0 1
1 0   /⍨⍤⍥,  1 1
1 0          1 1
0 1          0 1

  ↓

1 1 1 0 1 0 1
```

```
h e l l o
0 0 1 1 0
```

Black Box / Data Transformation

```
0 1          0 1
0 1          0 1
1 1  / ö,    1 0
1 1          1 0
0 1          0 1

  ↓

1 1 1 0 1 0 1
```

```
h e l l o
0 0 1 1 0
```

Black Box / Data Transformation

```
{(,⍵,[1.5]1)

      /

0 1 0 1 1 0 1 0 0 1}

  ↓

1 1 1 0 1 0 1
```

```
h e l l o
0 0 1 1 0
```

Black Box / Data Transformation

```
{(,⍵,[1.5]1)

      /

           (,⍵,[1.5]~⍵)}

  ↓

1 1 1 0 1 0 1
```

```
h e l l o
0 0 1 1 0
```

Black Box / Data Transformation

```
{(,⍵,[1.5]1)

      /

           (,⍵,[1.5]~⍵)}

  ↓

1 1 1 0 1 0 1
```

```
h e l l o
0 0 1 1 0
```

Black Box / Data Transformation

```
{(,⍵,[1.5]1)/(,⍵,[1.5]~⍵)}

      ((⍪,1⍨⍤⍤)⊢⍤⍤⍥⊂/⍤⍤⊂,(⍪,~))

   ↓

1  1  1  0  1  0  1
```

```
h  e  l  l  o
0  0  1  1  0

aplcart.info
```

Black Box / Data Transformation

```
h e l l o
0 0 1 1 0
↓
1 1 1 0 1 0 1

⍝ What is the mapping?
0 → 1
1 → 1 0
```

Black Box / Data Transformation

```
h e l l      o
0 0 1 1      0
↓ ↓          ↓
1 1 1 0 1 0 1

⍝ What is the mapping?
0 → 1   ⍝   ~
1 → 1 0

      ~0 0 1 1 0
1 1 0 0 1
```

Black Box / Data Transformation

```
h e l l      o
0 0 1 1      0

1 1 1 0 1 0 1

⍝ What is the mapping?
0 → 1 ⍝ ~
1 → 1 0
```

Black Box / Data Transformation

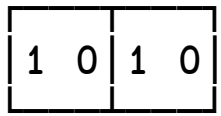
```
h e   l   l o
0 0   1   1 0
      ↓   ↓
1 1 1 0 1 0 1

⍝ What is the mapping?
0 → 1   ⍝   ~
1 → 1 0
```

Black Box / Data Transformation

```
h e   l   l o
0 0   1   1 0
      ↓   ↓
1 1 1 0 1 0 1

⍝ What is the mapping?
0 → 1   ⍴   ~
1 → 1 0   ⍴ 1,~
```

Black Box / Data Transformation

```
h e   l   l o
0 0   1   1 0
      ↓   ↓
1 1 1 0 1 0 1

⍝ What is the mapping?
0 →   1   ⍴   ~
1 → 1 0   ⍴ 1,~

      1,¨0 0
```



Black Box / Data Transformation

```
h e   l   l o
0 0   1   1 0

1 1 1 0 1 0 1

⍝ What is the mapping?
0 → 1   ⍝   ~
1 → 1 0 ⍝ 1,~

          ~0 0 1 1 0
1 1 0 0 1
```

Black Box / Data Transformation

```
h e   l   l o
0 0   1   1 0

1 1 1 0 1 0 1

⍝ What is the mapping?
0 → 1   ⍴   ~
1 → 1 0   ⍴ 1,~

          @~~0 0 1 1 0
1 1 0 0 1
```

Black Box / Data Transformation

```
h e   l   l o
0 0   1   1 0

1 1 1 0 1 0 1

⍝ What is the mapping?
0 → 1   ⍝   ~
1 → 1 0 ⍝ 1,~

      1,¨@~~0 0 1 1 0
1 1 0 0 1
```

Black Box / Data Transformation

```
h e   l   l o
0 0   1   1 0

1 1 1 0 1 0 1

⍝ What is the mapping?
0 → 1   ⍴   ~
1 → 1 0 ⍴ 1,~

      1,¨@~~0 0 1 1 0
```

1	1	1	0	1	0	1

Black Box / Data Transformation

```
h e   l   l o
0 0   1   1 0

1 1 1 0 1 0 1

⍝ What is the mapping?
0 → 1   ⍝   ~
1 → 1 0   ⍝ 1,~

      ∊1,¨@~~0 0 1 1 0
1 1 1 0 1 0 1
```

Black Box / Data Transformation

```
h e   l   l o
0 0   1   1 0

1 1 1 0 1 0 1

⍝ What is the mapping?
0 → 1
1 → 1 0
```

Black Box / Data Transformation

```
h e   l   l o
0 0   1   1 0

1 1 1 0 1 0 1

⍝ What is the mapping?
0 →   1   ⍝ 1↑1
1 → 1 0   ⍝ 2↑1

            1+0 0 1 1 0
1 1 2 2 1
```

Black Box / Data Transformation

```
h e   l   l o
0 0   1   1 0

1 1 1 0 1 0 1

⍝ What is the mapping?
0 → 1   ⍝ 1↑1
1 → 1 0 ⍝ 2↑1

      ∊1↑⍨⍨1+0 0 1 1 0
1 1 1 0 1 0 1
```

Loop First

```
    ∇  r←Loop bits;bit
[1]     r←0
[2]     :For bit :In bits
[3]         ⎕←r,←(bit⍴1),~bit
[4]     :EndFor
    ∇

    LOOP FIRST

       Good APL programmers avoid looping.
```

Loop First: How big is the output array?

```
      ⎕←b←'l' = 'hellolo'
0 0 1 1 0 1 0

      +/b    ⍝ One 0 per 1
3
      ≠b     ⍝ One 1 per element
7
      (+/+≠)b
10
```

Loop First: How big is the output array?

```
b ← 0 0 1 1 0 1 0          ⍝ Input bits
    1 2 3 4 5 6 7
p ← 1 1 1 1 1 1 1 1 1 1    ⍝ Pre-allocated
r ← 1 1 1 0 1 0 1 1 0 1    ⍝ Output
    1 2 3 4 5 6 7 8 9 10


      ⍸b
3 4 6
      ⍸~r
4 6 9
      (⍸~r) - (⍸b)
1 2 3
```

Loop First

```
⍝ (where 1s in b) + (integers to sum of b)
      (⍸b)        +       ⍳+/b


      1⍴⍨(≢⍤++/)b
1 1 1 1 1 1 1 1 1 1
      (⍸b) + ⍳+/b
4 6 9
      ~@((⍸b) + ⍳+/b) ⊢ 1⍴⍨(≢⍤++/)b
1 1 1 0 1 0 1 1 0 1
```

Performance

```
 ]defs

 Old ← {(,⍵,[1.5]1)/,⍵,[1.5]~⍵}
 New ← (⍪,1⍨)⊢⍤⊃/⊃,⍪,~

 Cat ← ∊1,⍤@~~
Repl ← ∊1 0⍨⍤@~⍤⊃~
Take ← {∊1↑⍤⍨⍨1+⍵}

Calc ← {~@((⍳⍵)+⍳+/⍵)⊢1⍴⍨(≠++/)⍵}
```

Performance

```
     ∇  r←Loop b;bit
[1]      r←0
[2]      :For bit :In b
[3]          ⎕←r,←(bit⍴1),~bit
[4]      :EndFor
     ∇
```

Performance

```
      □ ← b ← 'l'='hello'
0 0 1 1 0

]runtime -c "Old b" "New b" "Cat b" "Repl b" "Take b" "Calc b" "Loop b"

  Old b  → 1.2E¯6 |    0% □□□□□□□□□
  New b  → 1.1E¯6 |   -7% □□□□□□□□□
  Cat b  → 1.9E¯6 |  +56% □□□□□□□□□□□□□□□
  Repl b → 1.7E¯6 |  +40% □□□□□□□□□□□□□
  Take b → 1.8E¯6 |  +54% □□□□□□□□□□□□□□□
  Calc b → 2.3E¯6 |  +91% □□□□□□□□□□□□□□□□□□
  Loop b → 5.8E¯6 | +386% □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
```

```
      ]defs

 Old ← {(,⍵,[1.5]1)/,⍵,[1.5]~⍵}
 New ← (⍪,1⍨)⊢⍤⊖/⊖,⍪,~

 Cat ← ∊1,¨@~~
Repl ← ∊1 0⍨¨@~⍤⊖~
Take ← {∊1↑¨⍨1+⍵}

Calc ← {~@((⍳⍴⍵)+⍳+/⍵)⊢1⍴⍨¨(≠++/)⍵}

Loop
```

Performance

```
      ⎕ ← b ← 'l'='hello'
0 0 1 1 0

]runtime -c "Old b" "New b" "Cat b" "Repl b" "Take b" "Calc b"

  Old b  → 1.3E¯6 |    0% ⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕
  New b  → 1.2E¯6 |  -11% ⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕
  Cat b  → 1.9E¯6 |  +41% ⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕
  Repl b → 1.7E¯6 |  +25% ⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕
  Take b → 1.7E¯6 |  +31% ⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕
  Calc b → 2.4E¯6 |  +84% ⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕
```

```
      ]defs

 Old ← {(,ω,[1.5]1)/,ω,[1.5]~ω}
 New ← (⍪,1⍨)⊢⍤/⍤,⍪,~

 Cat ← ∊1,¨@~~
Repl ← ∊1 0⍨¨@~⍤~
Take ← {∊1↑¨⍨1+ω}

Calc ← {~@((⍸ω)+⍳+/ω)⊢1⍴⍨(≢++/)ω}
```

Performance

```
      ⎕←30↑ b ← 1=?1e6⍴2
1 0 0 0 1 0 0 1 1 0 0 0 1 0 1 0 1 1 0 0 1 0 0 1 0 1 0 1 0 1

]runtime -c "Old b" "New b" "Cat b" "Repl b" "Take b" "Calc b"

  Old b  → 1.6E¯2 |    0% ⎕⎕⎕⎕
  New b  → 1.8E¯2 |  +12% ⎕⎕⎕⎕⎕
  Cat b  → 1.1E¯1 | +566% ⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕
  Repl b → 4.6E¯2 | +186% ⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕
  Take b → 1.5E¯1 | +866% ⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕⎕
  Calc b → 3.3E¯3 |  -80% ⎕
```

```
      ]defs

 Old  ← {(,⍵,[1.5]1)/,⍵,[1.5]~⍵}
 New  ← (⍪,1⍨)⊢⍤/⍥,⍪,~

 Cat  ← ∊1,¨@~~
Repl  ← ∊1 0⍨¨@~⍤~
Take  ← {∊1↑¨⍨1+⍵}

Calc  ← {~@((⍸⍵)+⍳+/⍵)⊢1⍴⍨(≢++/)⍵}
```

Summary

Data Transformations

Play with Data / New Perspective

What is the Mapping?

Try a Loop

Speaking & Listening

Array-oriented Functional Programming by Aaron W Hsu, Dhaval Dalal and Morten Kromberg at #FnConf18

https://www.youtube.com/watch?v=Gsj_7tFtODk

Thinking Out Loud in APL

Search: "RikedyP Perl Weekly Challenge"

youtube.com/playlist?list=PLR-QNHF170ul6n2jMU9wwSzHg5McGWhAs

Next week

British APL Association Open Session

Thursday 24th September 16:00 BST

britishaplassociation.org/webinar-schedule-2020/

Next Dyalog Webinar

Rank and Dyadic Transpose

Thursday 1st October 16:00 BST

Dyalog.tv

Dyalog '20 Online

Monday 9 - Tuesday 10 November

Register: dyalog.com/user-meetings/dyalog20.htm