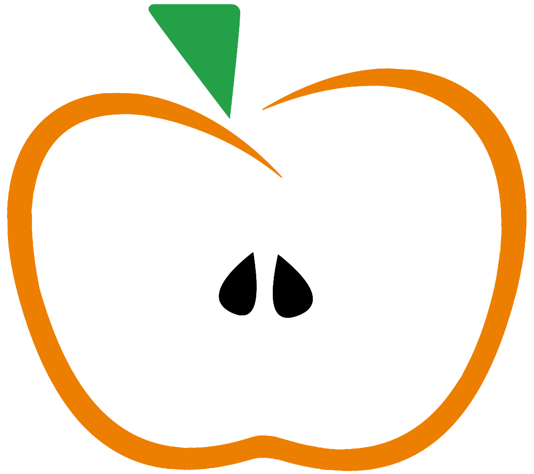
APL Seeds 2023

But How Will I Remember All Those Squiggles?! — APL Mnemonics

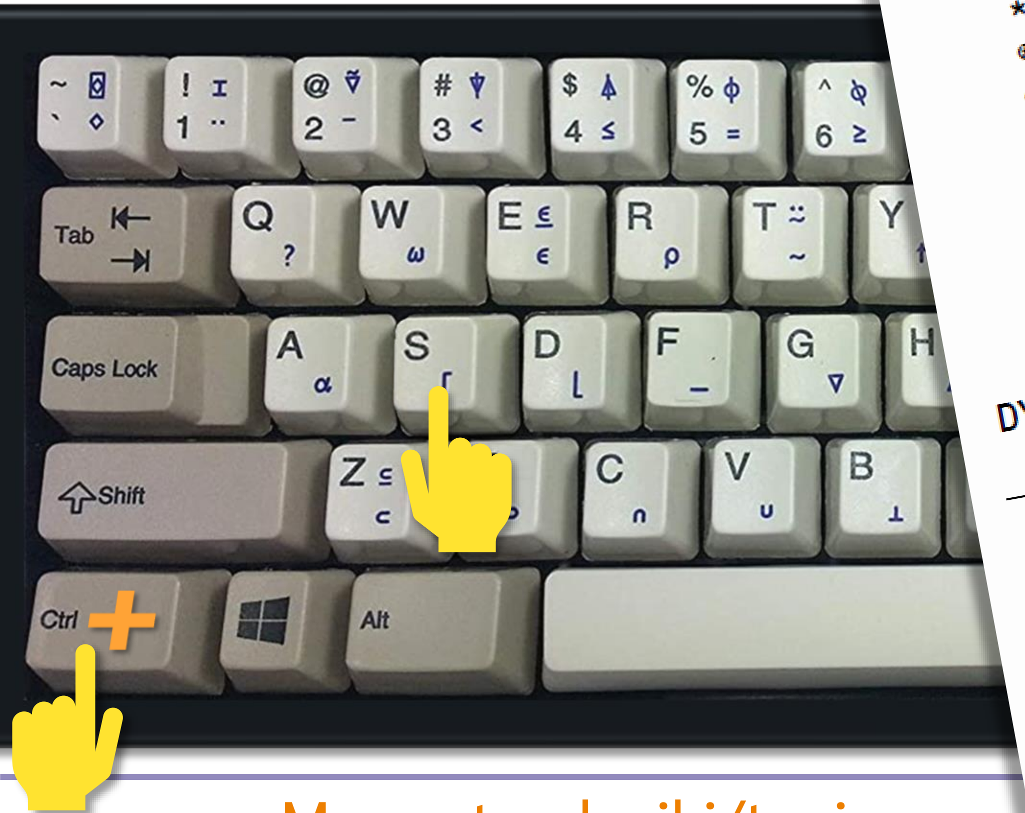
Adám Brudzewsky

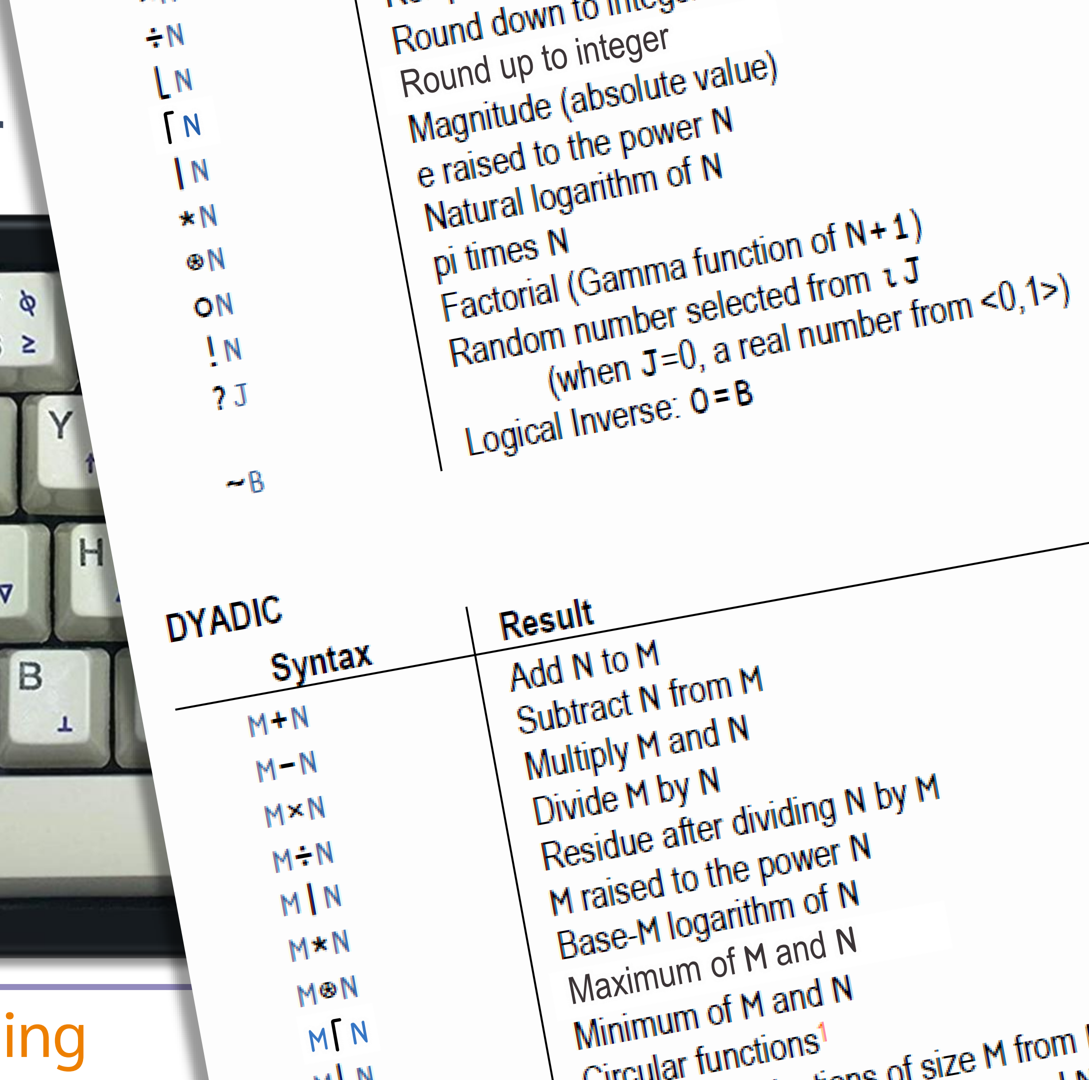


Mnemonics

noun A system that helps to remember things.

Lots to remember





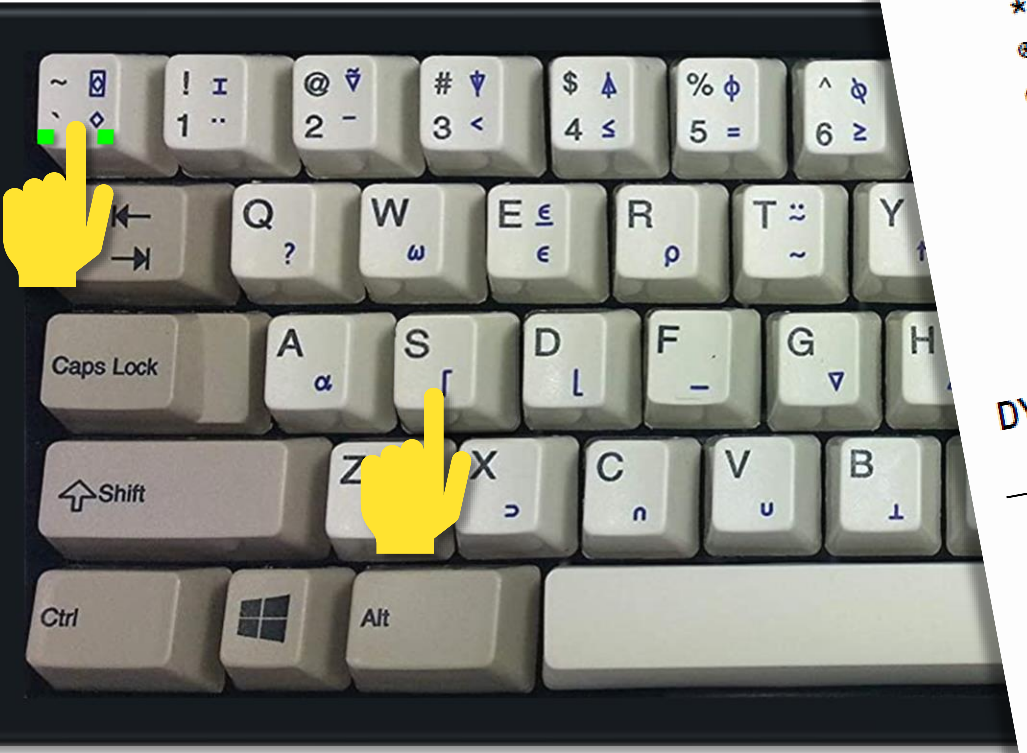
÷N	
⌊N	Round down to integ
⌈N	Round up to integer
|N	Magnitude (absolute value)
*N	e raised to the power N
⍟N	Natural logarithm of N
○N	pi times N
!N	Factorial (Gamma function of N+1)
?J	Random number selected from ⍳J (when J=0, a real number from <0,1>)
~B	Logical Inverse: 0=B

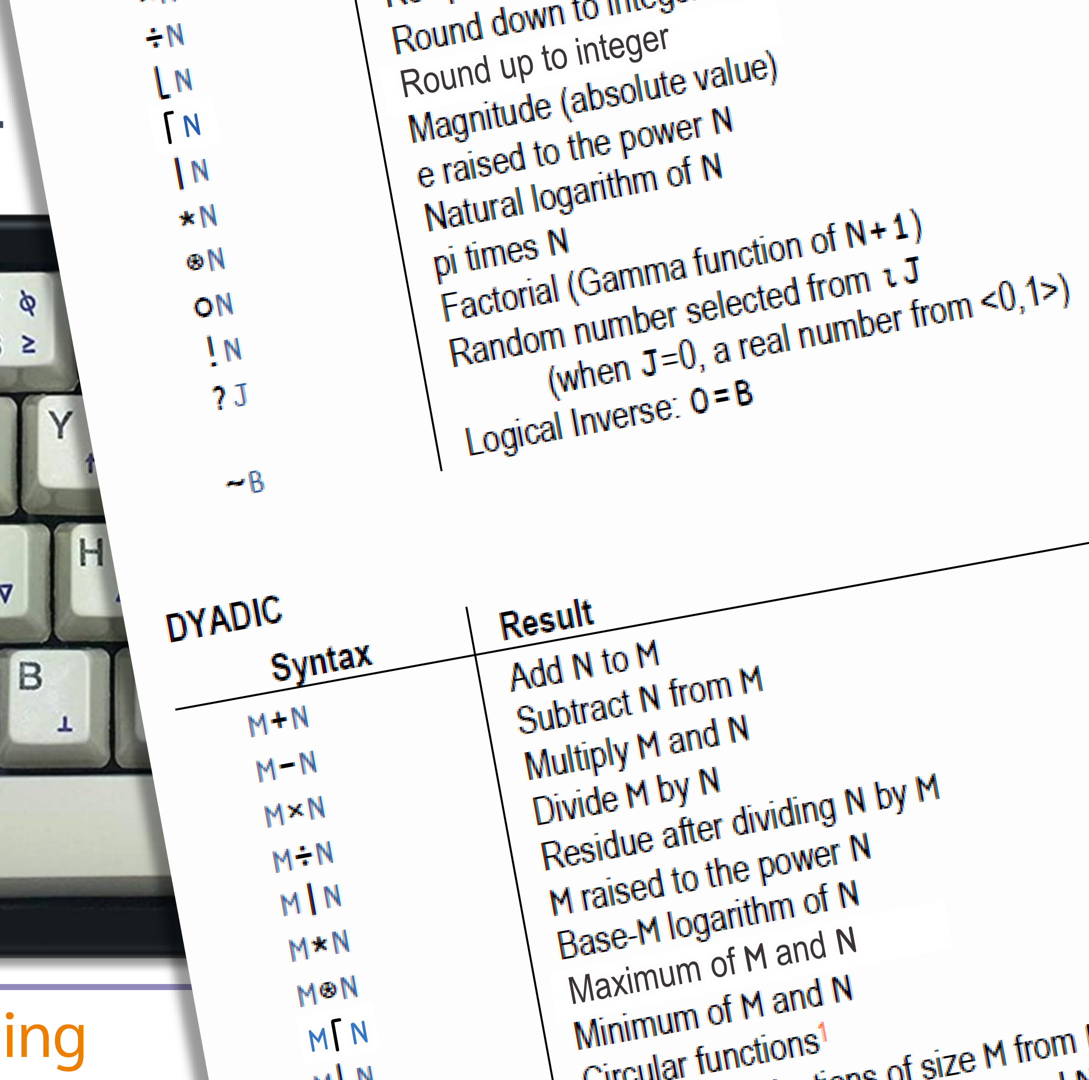
DYADIC

Syntax	Result
M+N	Add N to M
M−N	Subtract N from M
M×N	Multiply M and N
M÷N	Divide M by N
M|N	Residue after dividing N by M
M*N	M raised to the power N
M⍟N	Base-M logarithm of N
M⌈N	Maximum of M and N
M⌊N	Minimum of M and N
	Circular functions[1]

More at apl.wiki/typing

Lots to remember





÷N	
⌊N	Round down to integer
⌈N	Round up to integer
|N	Magnitude (absolute value)
*N	e raised to the power N
⍟N	Natural logarithm of N
○N	pi times N
!N	Factorial (Gamma function of N+1)
?J	Random number selected from ⍳J (when J=0, a real number from <0,1>)
~B	Logical Inverse: 0=B

DYADIC Syntax	Result
M+N	Add N to M
M−N	Subtract N from M
M×N	Multiply M and N
M÷N	Divide M by N
M|N	Residue after dividing N by M
M*N	M raised to the power N
M⍟N	Base-M logarithm of N
M⌈N	Maximum of M and N
M⌊N	Minimum of M and N
	Circular functions[1]

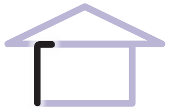
More at apl.wiki/typing

Lots to remember

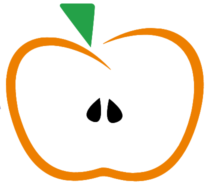
C *for* Cap

(intersection)

⌈N	Round up to integer
⌈	**Seiling** and wall
M⌈N	Maximum of M and N

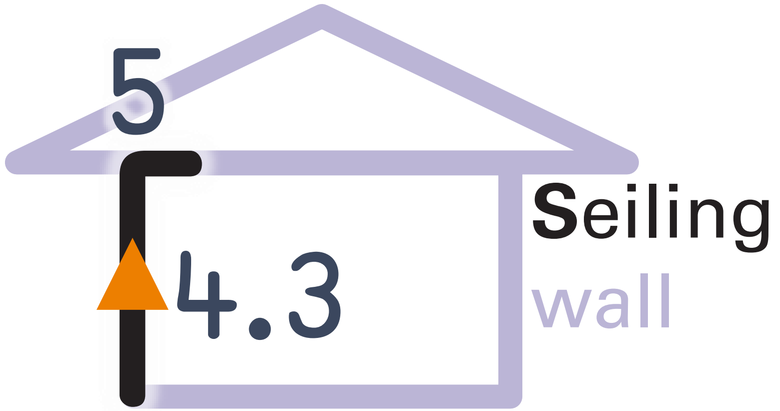


Ceiling *with* C

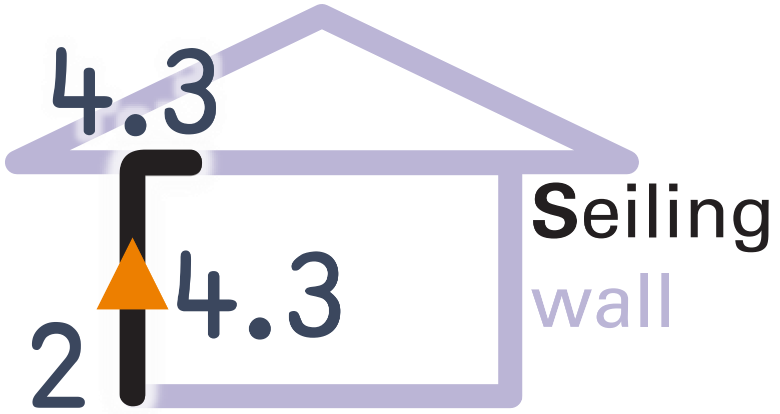


Lots to remember

⌈N Round up to integer



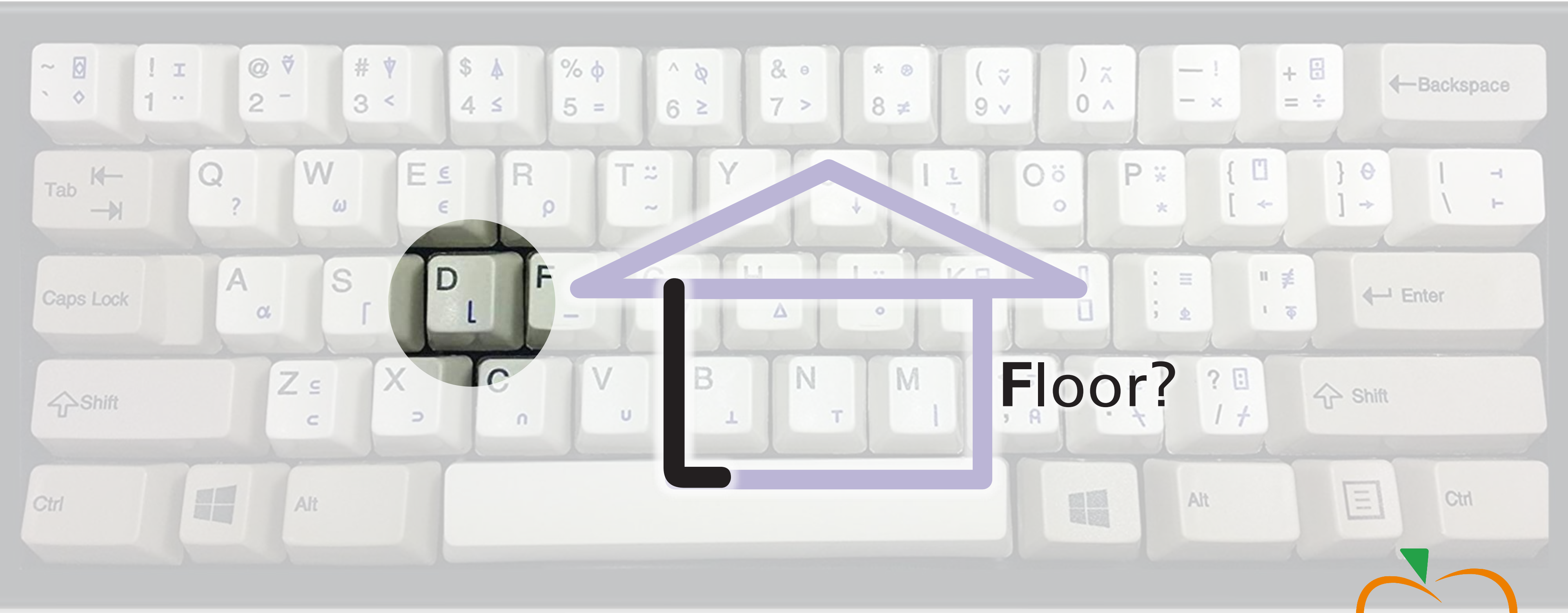
Lots to remember



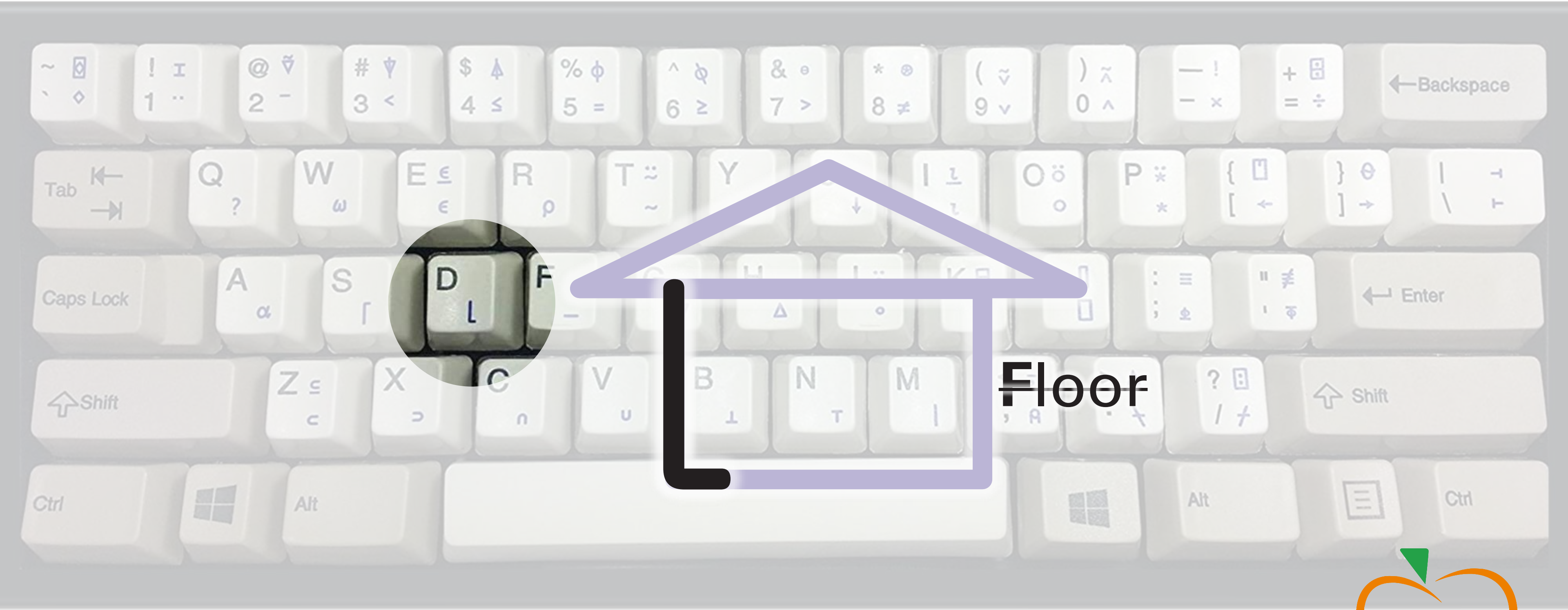
M⌈N   Maximum of M and N

Lots to remember

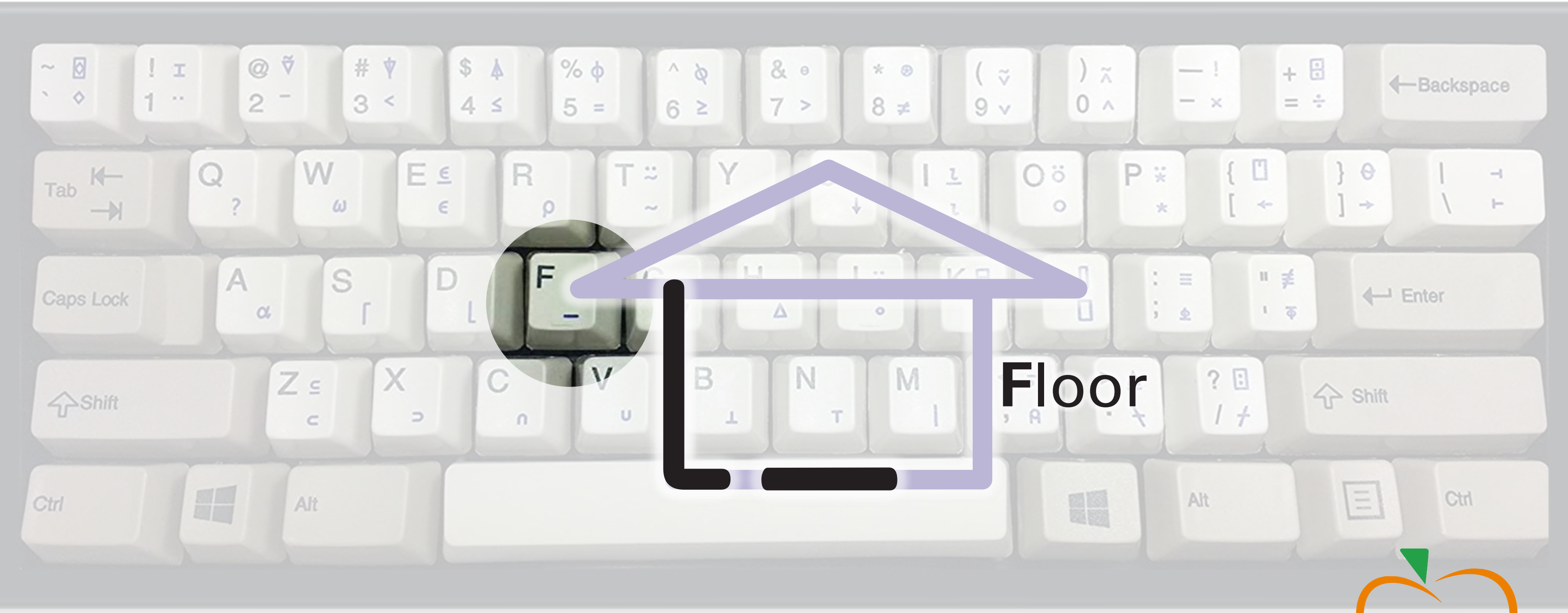
Lots to remember



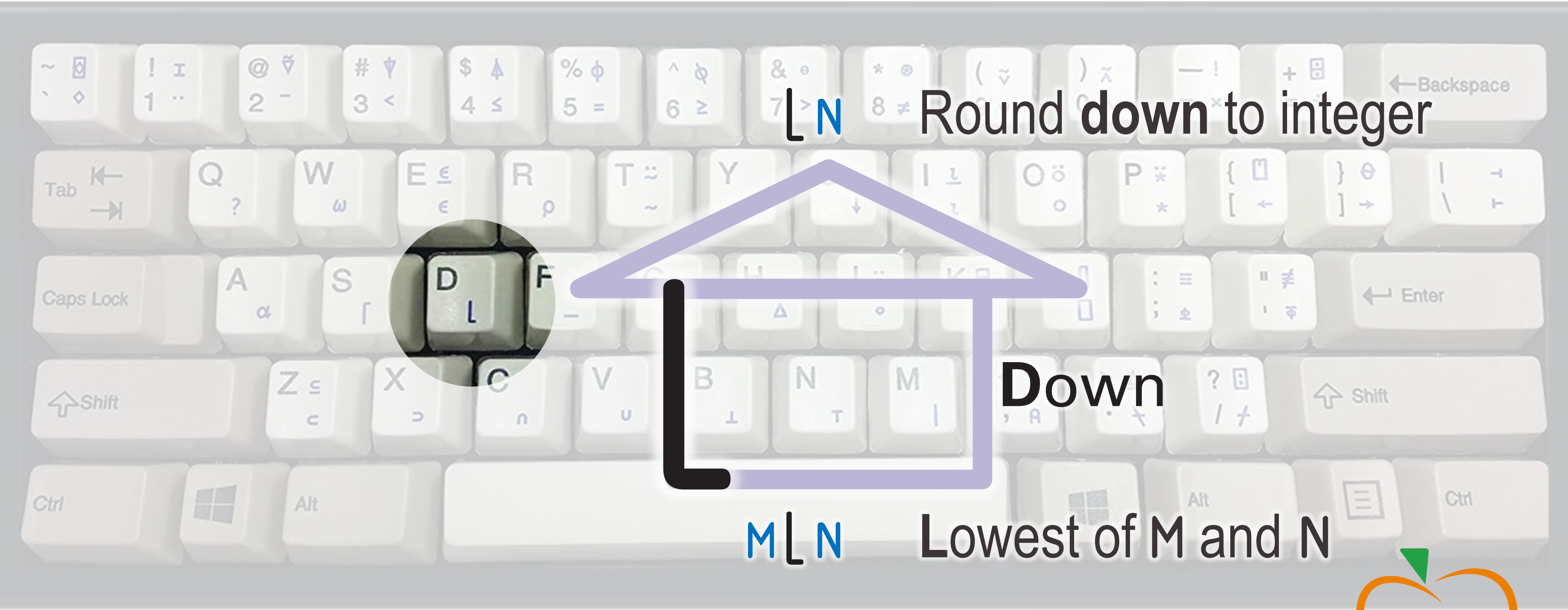
Lots to remember



Lots to remember



Lots to remember



⌊N Round **down** to integer

M⌊N Lowest of M and N

How to remember monad/dyads

Related Concepts

⌈	up: ceiling/max
⌊	down: floor/min

How to remember monad/dyads

Tilde

Related Concepts

⌈ up: ceiling/max

⌊ down: floor/min

~ not: logical/set

~~stuff~~

"not"

"but not"

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

```
   4 1 2 1 5 ~ 5 1
→   4 2
```

How to remember monad/dyads

"shape of"

```
    ⍴ 3 1 2 7
4
```

Related Concepts

⌈ up: ceiling/max

⌊ down: floor/min

~ not: logical/set

⍴ shape: query/change

"reshapes"

```
      2 2 ⍴ 3 1 2 7
3 1
2 7
```

Reshape

How to remember monad/dyads

Default Left Arg.

Related Concepts

⌈	up: ceiling/max
⌊	down: floor/min
~	not: logical/set
⍴	shape: query/change

How to remember monad/dyads

Default Left Arg.

- negate/subtract

```
    3-10
→   ¯7

    -7
→   ¯7
```

How to remember monad/dyads

Default Left Arg.

- negate/subtract

```
      3-10
→     ¯7

      -7
→     ¯7
```

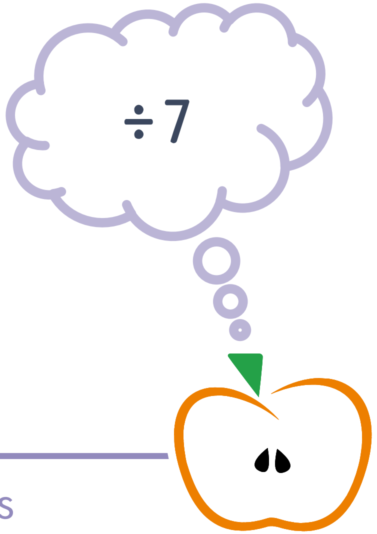
How to remember monad/dyads

Default Left Arg.

- -   negate/subtract

```
      3-10
→     ¯7

      0-7
→     ¯7
```



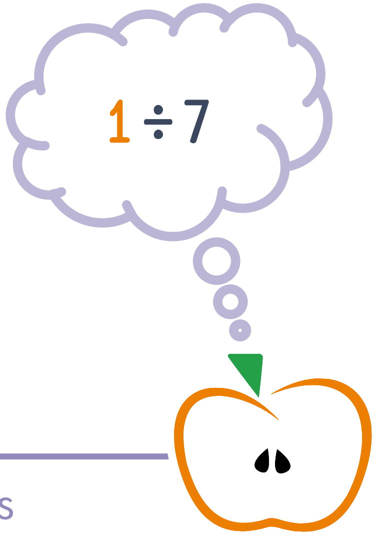
How to remember monad/dyads

<u>Default Left Arg.</u>

- negate/subtract

```
      3-10
→     ¯7

      0-7
→     ¯7
```

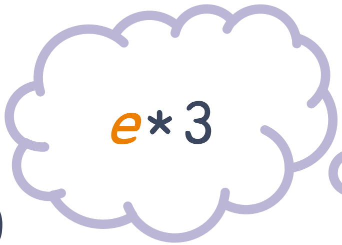


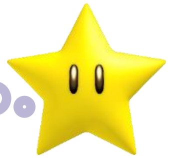
How to remember monad/dyads

Default Left Arg.

- - negate/subtract
- ⍟ (natural) log

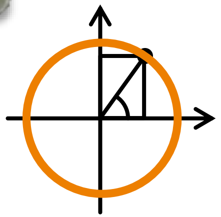
```
      10⍟1000
→    3
      ⍟1000
→    6.907755279
      e⍟¯1
→    ¯3.141592654
```

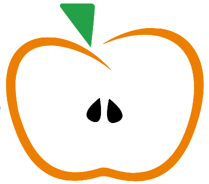




* Power

○ Circular





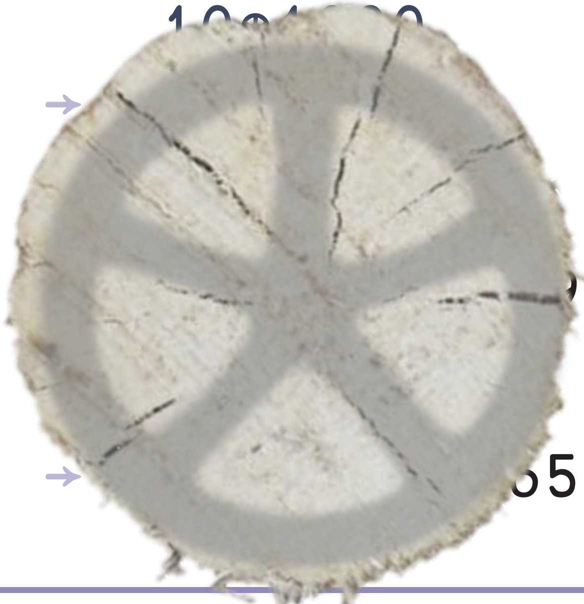
How to remember monad/dyads

<u>Default Left Arg.</u>

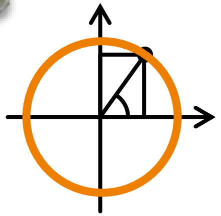
- `-` negate/subtract
- `⍟` (natural) log

logarithm

```
10⍟1000
→
→          654
```



`⍟` Power Circular



How to remember monad/dyads

Default Left Arg.

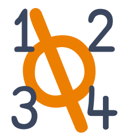
- `-` negate/subtract
- `⍟` (natural) log
- `⍉` transpose

```
1 2
3 4
```

How to remember monad/dyads

<u>Default Left Arg.</u>

-	negate/subtract
⍟	(natural) log
	transpose



How to remember monad/dyads

Default Left Arg.

- `-` negate/subtract
- `⍟` (natural) log
- `⍉` transpose

```
1 2
3 4

5 6
7 8
```

How to remember monad/dyads

Default Left Arg.

- `-` negate/subtract
- `⍟` (natural) log
- `⍉` transpose

```
1 2
3 4
```

```
5 6
7 8
```

How to remember monad/dyads

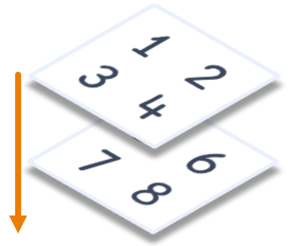
Default Left Arg.

- `-` negate/subtract
- `⍟` (natural) log
- `⍉` transpose

How to remember monad/dyads

Default Left Arg.

- `-` negate/subtract
- `⍟` (natural) log
- `1⍉` transpose



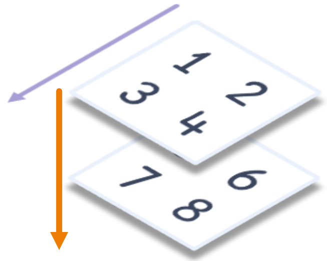
How to remember monad/dyads

<u>Default Left Arg.</u>

- negate/subtract

⍟ (natural) log

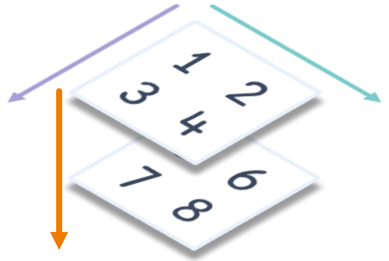
1 2⍉ transpose



How to remember monad/dyads

Default Left Arg.

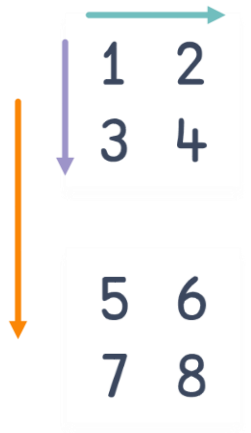
- `-` negate/subtract
- `⍟` (natural) log
- `1 2 3⍉` transpose



How to remember monad/dyads

<u>Default Left Arg.</u>

- `-` negate/subtract
- `⍟` (natural) log
- `1 2 3⍉` transpose



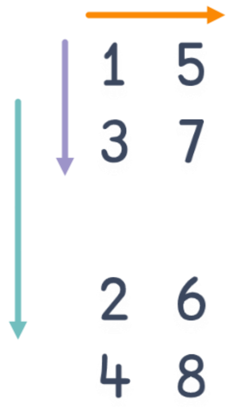
How to remember monad/dyads

Default Left Arg.

- negate/subtract

⍟ (natural) log

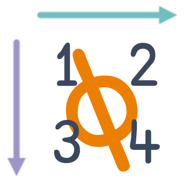
3 2 1⍉ transpose



How to remember glyphs

Default Left Arg.

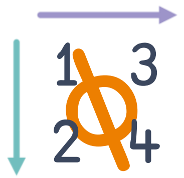
- - negate/subtract
- ⍟ (natural) log
- ⍉ transpose



How to remember glyphs

Default Left Arg.

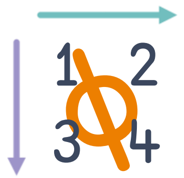
- `-` negate/subtract
- `⍟` (natural) log
- `⍉` transpose



How to remember glyphs

Default Left Arg.

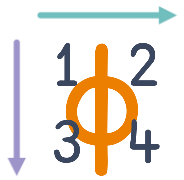
- − negate/subtract
- ⍟ (natural) log
- ⍉ transpose



How to remember glyphs

Default Left Arg.

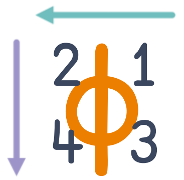
- negate/subtract
⍟ (natural) log
⍉ transpose



How to remember glyphs

Default Left Arg.

- `-` negate/subtract
- `⍟` (natural) log
- `⍉` transpose

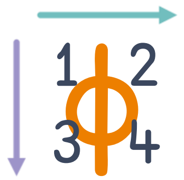


“reverse”

How to remember glyphs

Default Left Arg.

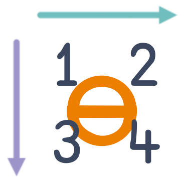
- - negate/subtract
- ⍟ (natural) log
- ⍉ transpose



How to remember glyphs

Default Left Arg.

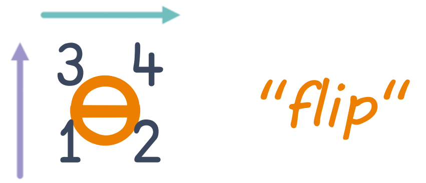
-	negate/subtract
⍟	(natural) log
⍉	transpose



How to remember glyphs

Default Left Arg.

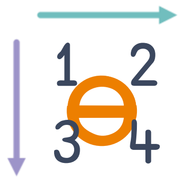
- `-` negate/subtract
- `⍟` (natural) log
- `⍉` transpose



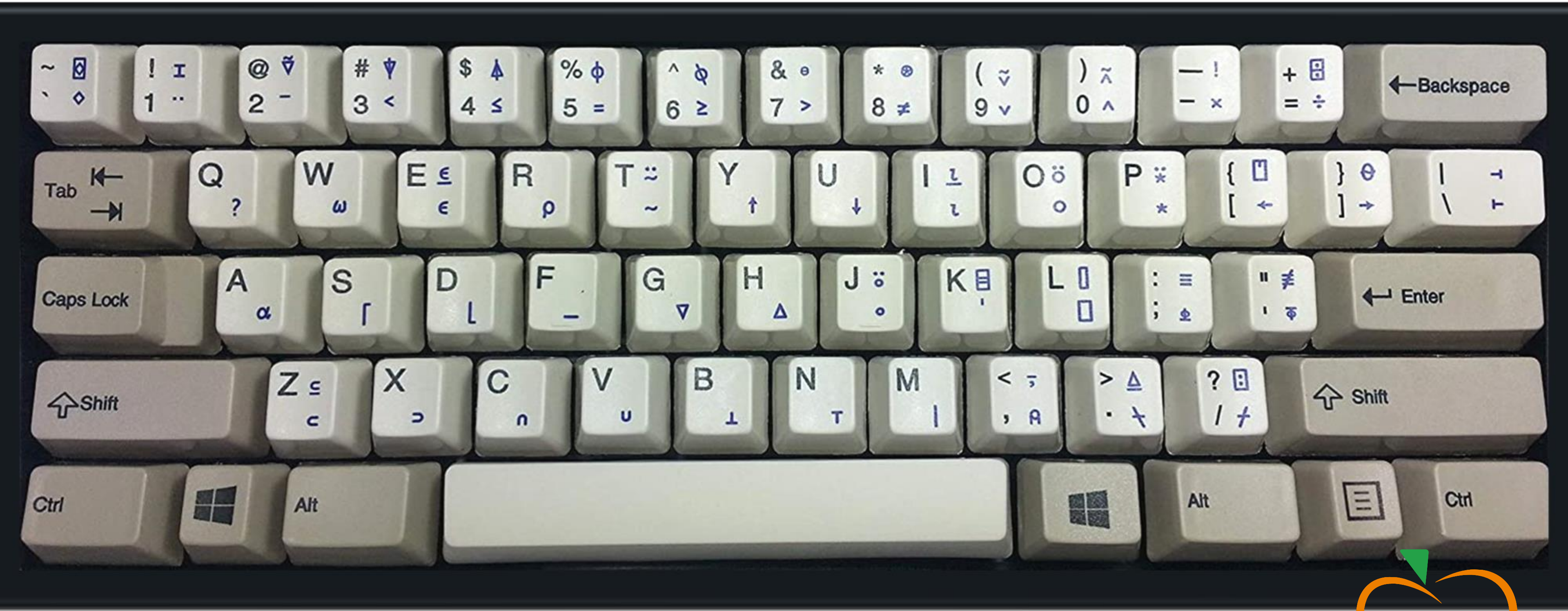
How to remember glyphs

Default Left Arg.

- - negate/subtract
- ⍟ (natural) log
- ⍉ transpose

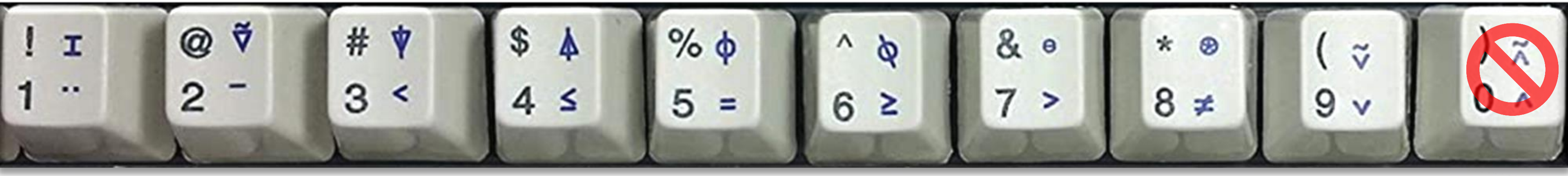


How to remember keyboarding



How to remember keyboarding

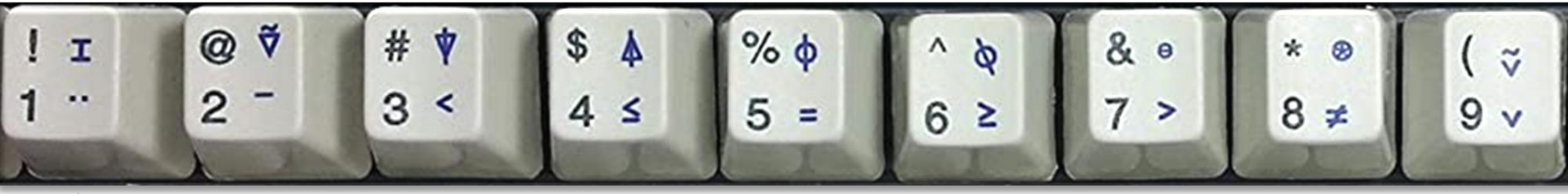
Default Left Arg.



⍉ transpose

How to remember keyboarding

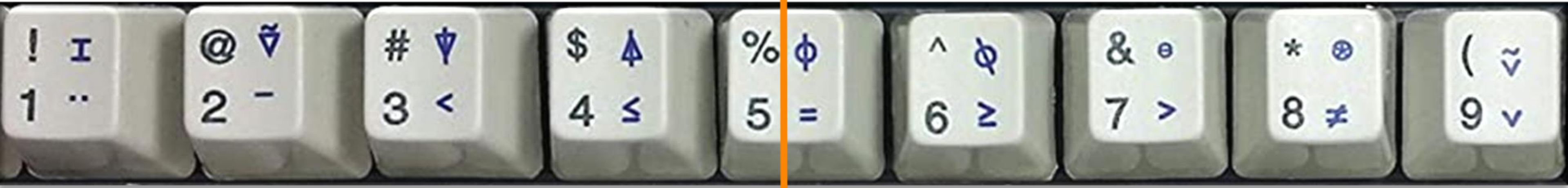
Default Left Arg.



⍉ transpose

How to remember keyboarding

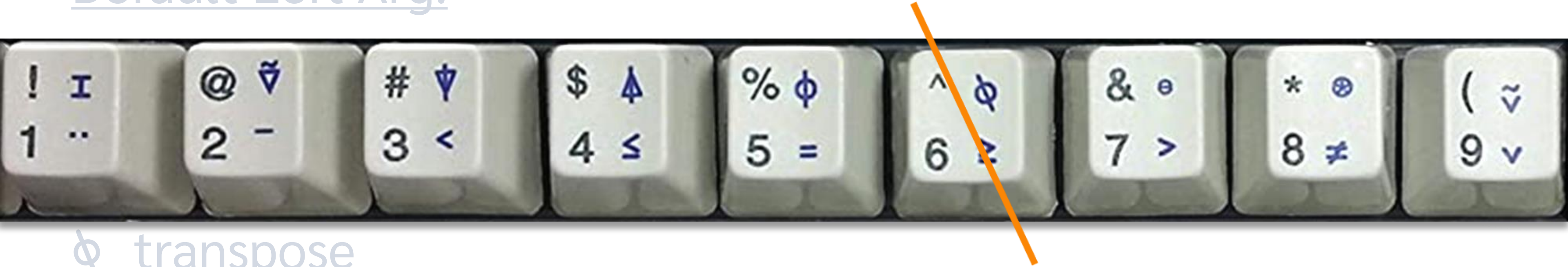
Default Left Arg.



⍉ transpose

How to remember keyboarding

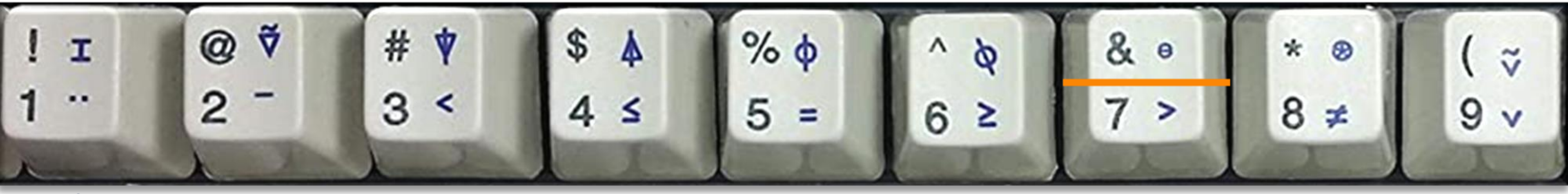
Default Left Arg.



⍉ transpose

How to remember keyboarding

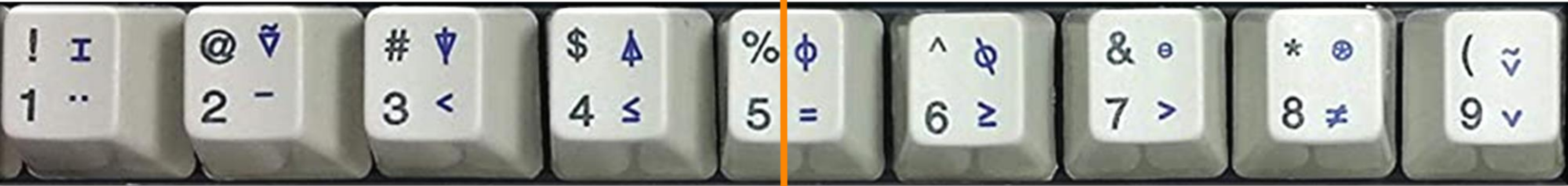
Default Left Arg.



⍉ transpose

How to remember keyboarding

Default Left Arg.

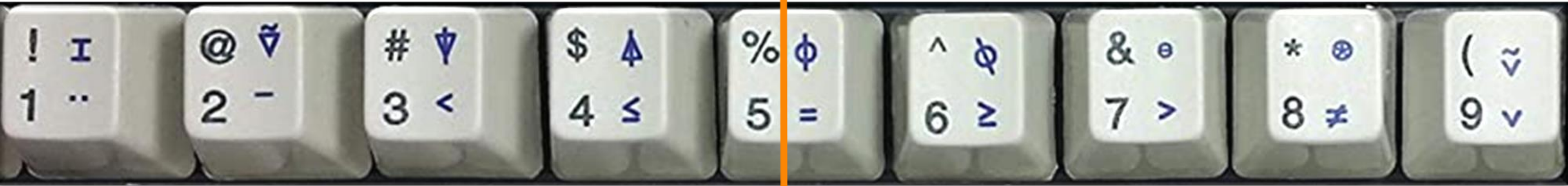


⍉ transpose

=

How to remember keyboarding

Default Left Arg.

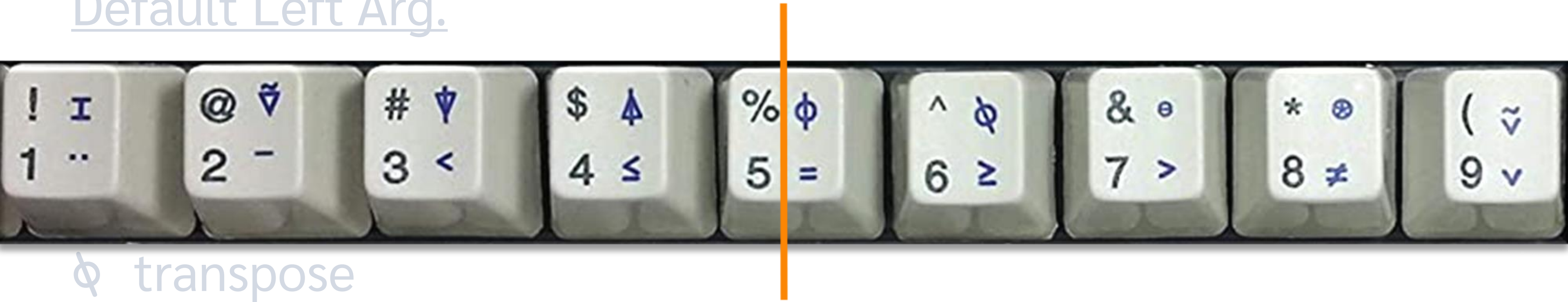


⍉ transpose

≤ = ≥

How to remember keyboarding

Default Left Arg.

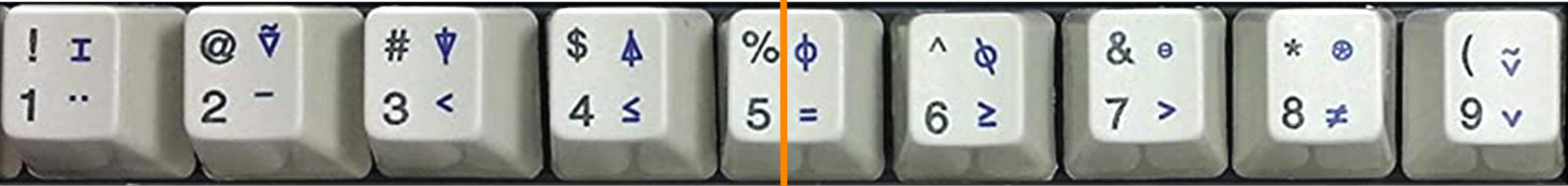


⍉ transpose

< ≤ = ≥ >

How to remember keyboarding

Default Left Arg.

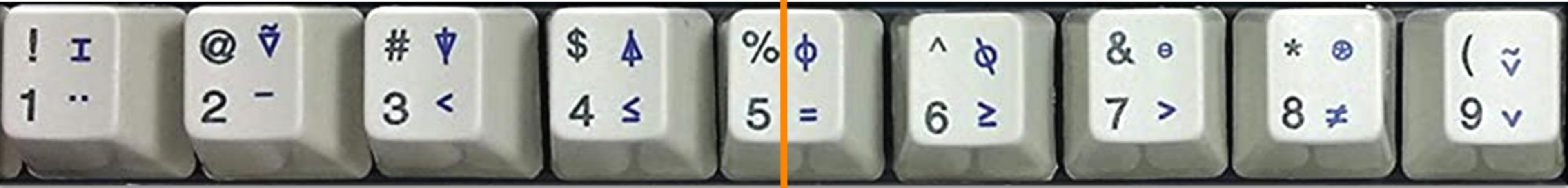


⍉ transpose

¯ < ≤ = ≥ >

How to remember keyboarding

Default Left Arg.



⍉ transpose

¯ < ≤ = ≥ > ≠

How to remember monad/dyads

Default Left Arg.

-	negate/subtract
⍟	(natural) log
⍉	transpose

How to remember monad/dyads

Both Fit Glyph

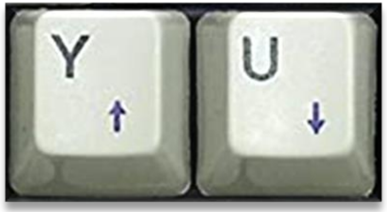
↑	mix/take
↓	split/drop

How to remember monad/dyads

```
      2↑3 1 4 1 5
3 1
```

```
      2↓3 1 4 1 5
4 1 5
```

take=
Yank!



drop=
send to
Under-
world!

<u>Both Fit Glyph</u>

↑ mix/take

↓ split/drop

How to remember monad/dyads

```
      ↑(3 1)(2 7)
3 1
2 7
```

Both Fit Glyph

↑ mix/take

↓ split/drop

How to remember monad/dyads

```
↑(3 1) (2 7)
```

```
3 1
2 7
```

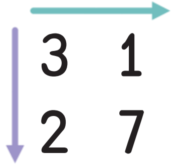
Both Fit Glyph

↑ mix/take

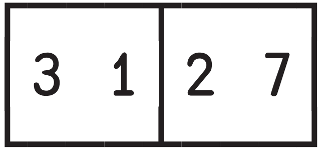
↓ split/drop

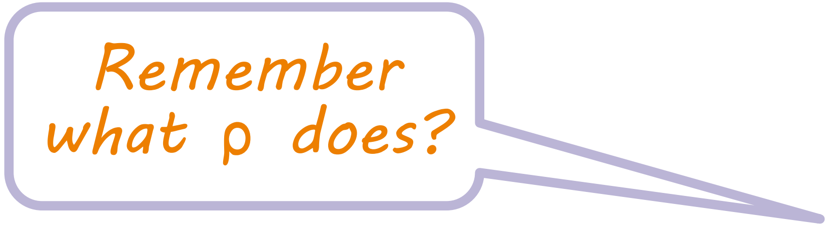
How to remember monad/dyads

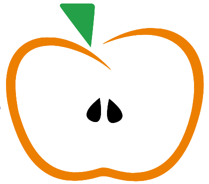
```
↑(3 1) (2 7)
```



```
↓2 2⍴3 1 2 7
```







__Both Fit Glyph__

↑ mix/take

↓ split/drop

How to remember monad/dyads

```
        ↑(3 1) (2 7)
3 1
2 7
```

Both Fit Glyph

↑ mix/take

↓ split/drop

```
        ↓2 2⍴3 1 2 7
┌───┬───┐
│3 1│2 7│
└───┴───┘
```

How to remember monad/dyads

```
↑(3 1) (2 7)
```

```
3 1
2 7
```

Both Fit Glyph

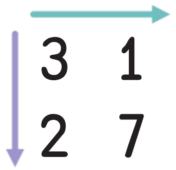
↑	mix/take
↓	split/drop

```
↓2 2⍴3 1 2 7
```

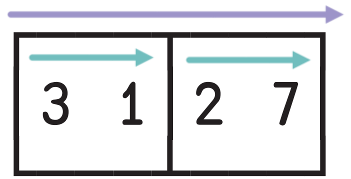
```
┌───┬───┐
│3 1│2 7│
└───┴───┘
```

How to remember monad/dyads

```
↑(3 1) (2 7)
```



```
↓2 2⍴3 1 2 7
```



Both Fit Glyph

- ↑ mix/take
- ↓ split/drop

How to remember monad/dyads

Both Fit Glyph

↑ mix/take

↓ split/drop

∊ **e**nlist/**e**lement

How to remember monad/dyads

```
   ∊(3 1) (2 7)
→      3 1 2 7
```

Both Fit Glyph

↑ mix/take

↓ split/drop

∊ **enlist**/element

How to remember monad/dyads

```
      ∊(3 1) (2 7)
→     3 1 2 7

      1 2 3 4 5 ∊ 3 1 2 7
→     1 1 1 0 0
```

Both Fit Glyph

↑ mix/take

↓ split/drop

∊ enlist/**element**

How to remember monad/dyads

```
      ∊(3 1) (2 7)
→        3 1 2 7

      1 2 3 4 5 ∊ 3 1 2 7
→     1 1 1 0 0
```

<u>Both Fit Glyph</u>

↑ mix/take

↓ split/drop

∊ **e**nlist/**e**lement

How to remember monad/dyads

Both Fit Glyph

↑	mix/take
↓	split/drop
∊	**e**nlist/**e**lement

How to remember

Default Left Arg.	Related Concepts	Both Fit Glyph
- negate/subtract	⌈ up: ceiling/max	↑ mix/take
⍟ (natural) log	⌊ down: floor/min	↓ split/drop
⍉ transpose	~ not: logical/set	∊ **e**nlist/**e**lement
	⍴ shape: query/change	

What you remember from today

-Y X-Y ⌈Y X⌈Y ⌊Y X⌊Y X∩Y ⍴Y X⍴Y ↑Y X↑Y

⍟Y X⍟Y *Y X*Y ¯N ~Y X~Y ∊Y X∊Y ↓Y X↓Y

⍉Y X⍉Y ⌽Y ⊖Y X<Y X≤Y X=Y X≥Y X>Y X≠Y

More at apl.wiki/mnemonics