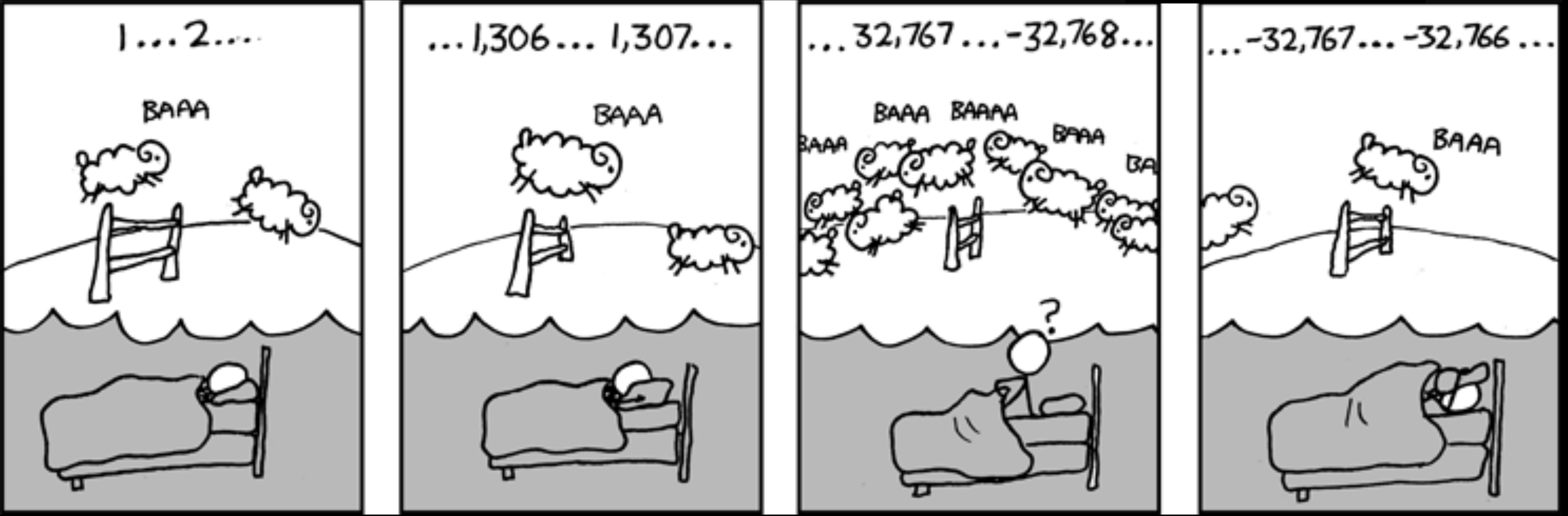
Reading Assignment

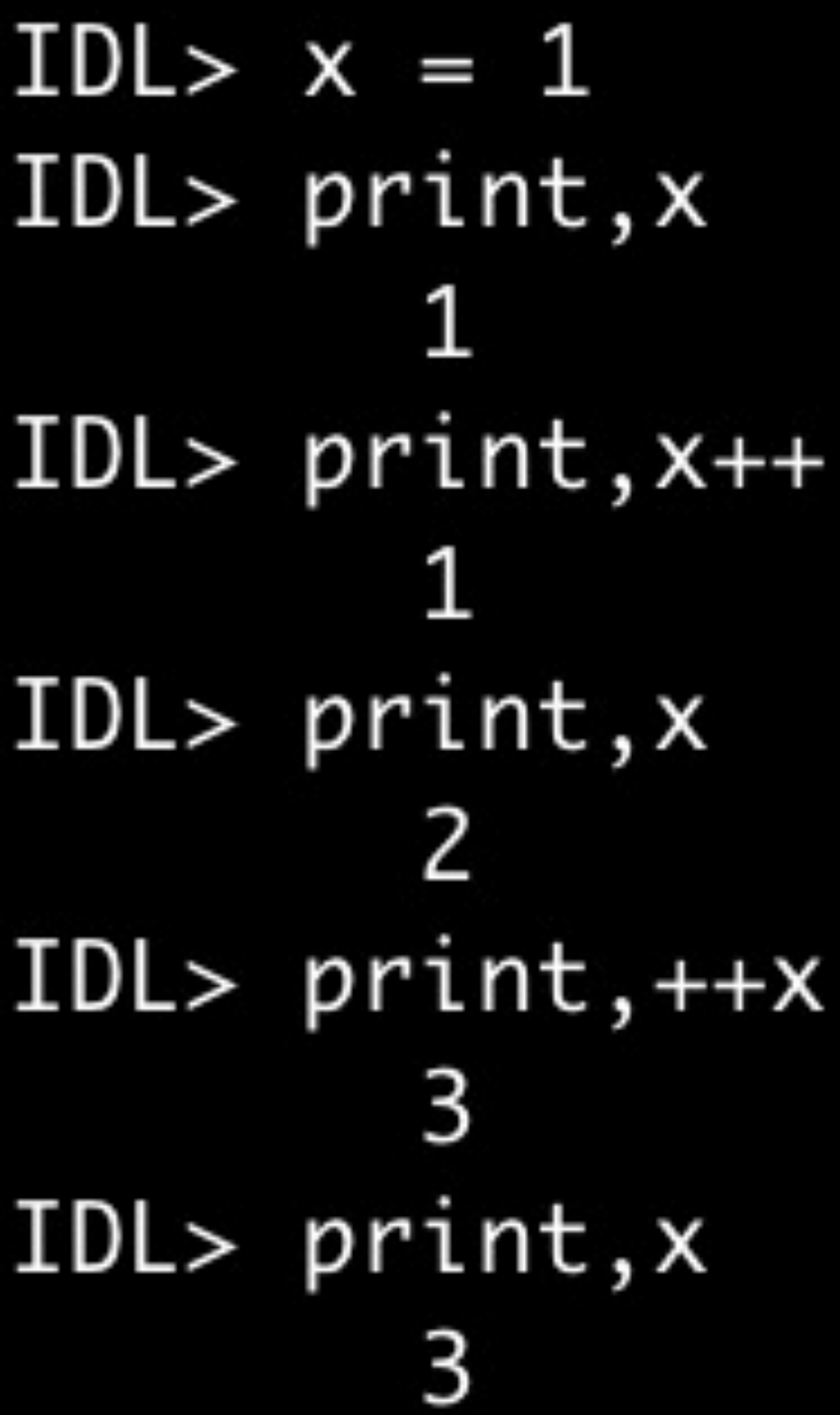
- Chapter 2 - should be review
    - Assignment 0 was due
    - Assignment 1 is due Thursday

Two's Complement and Overflows

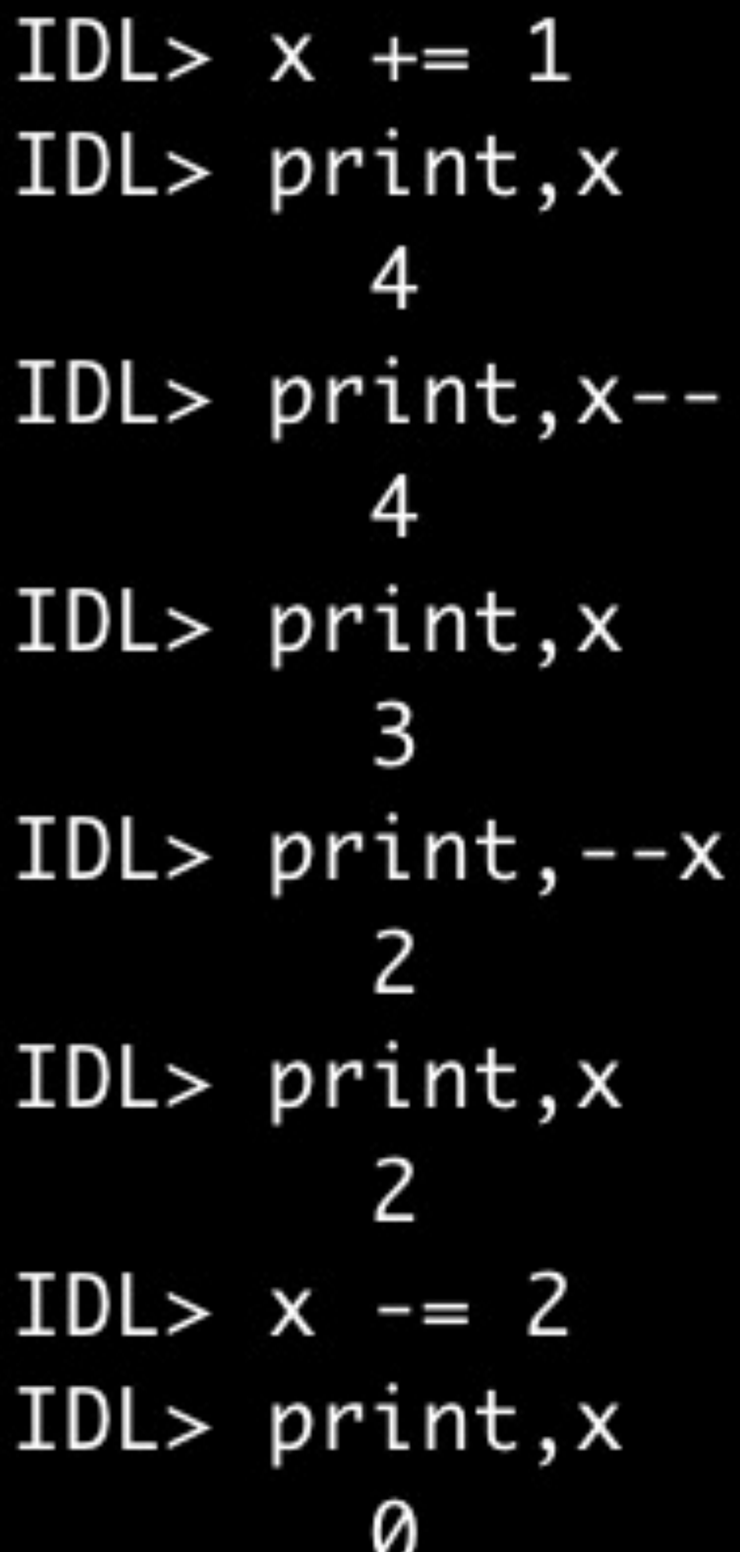
```
IDL> x=2^15-1
IDL> print,x++
   32767
IDL> print,x++
  -32768
IDL> print,x++
  -32767
IDL> print,x++
  -32766
```



++ / --



```
IDL> x = 1
IDL> print,x
       1
IDL> print,x++
       1
IDL> print,x
       2
IDL> print,++x
       3
IDL> print,x
       3
```

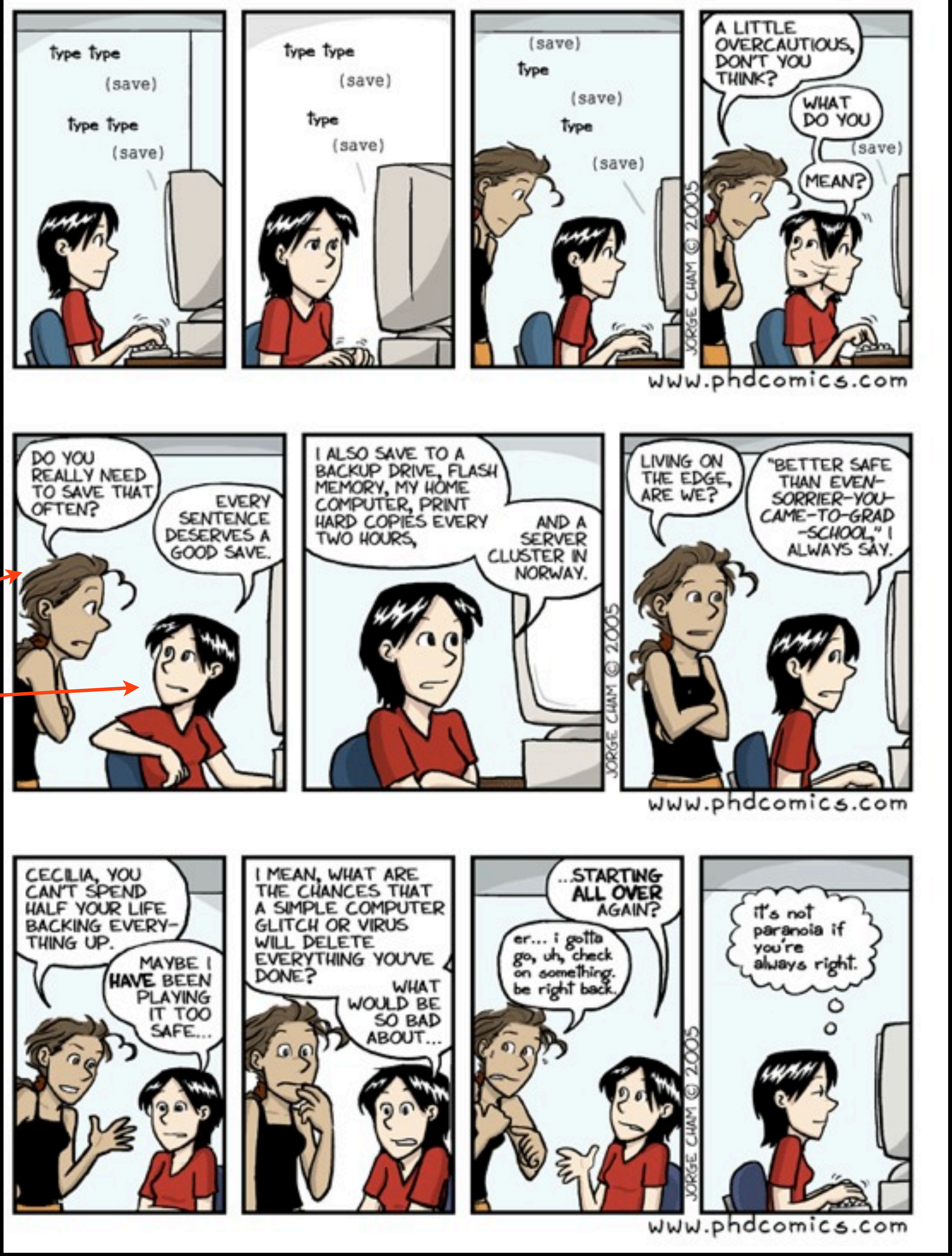


```
IDL> x += 1
IDL> print,x
       4
IDL> print,x--
       4
IDL> print,x
       3
IDL> print,--x
       2
IDL> print,x
       2
IDL> x -= 2
IDL> print,x
       0
```

Shorthand for `x = x + 1`

Good practice: Save Often

Grad Students


type type
(save)
type type
(save)
type type
(save)
type
(save)
(save)
type
(save)
type
(save)
A LITTLE OVERCAUTIOUS, DON'T YOU THINK?
WHAT DO YOU
(save)
MEAN?
www.phdcomics.com
DO YOU REALLY NEED TO SAVE THAT OFTEN?
EVERY SENTENCE DESERVES A GOOD SAVE.
I ALSO SAVE TO A BACKUP DRIVE, FLASH MEMORY, MY HOME COMPUTER, PRINT HARD COPIES EVERY TWO HOURS,
AND A SERVER CLUSTER IN NORWAY.
LIVING ON THE EDGE, ARE WE?
"BETTER SAFE THAN EVEN-SORRIER-YOU-CAME-TO-GRAD-SCHOOL," I ALWAYS SAY.
www.phdcomics.com
CECILIA, YOU CAN'T SPEND HALF YOUR LIFE BACKING EVERYTHING UP.
MAYBE I HAVE BEEN PLAYING IT TOO SAFE...
I MEAN, WHAT ARE THE CHANCES THAT A SIMPLE COMPUTER GLITCH OR VIRUS WILL DELETE EVERYTHING YOU'VE DONE?
WHAT WOULD BE SO BAD ABOUT...
...STARTING ALL OVER AGAIN?
er... i gotta go, uh, check on something. be right back.
it's not paranoia if you're always right.
www.phdcomics.com
JORGE CHAM © 2005

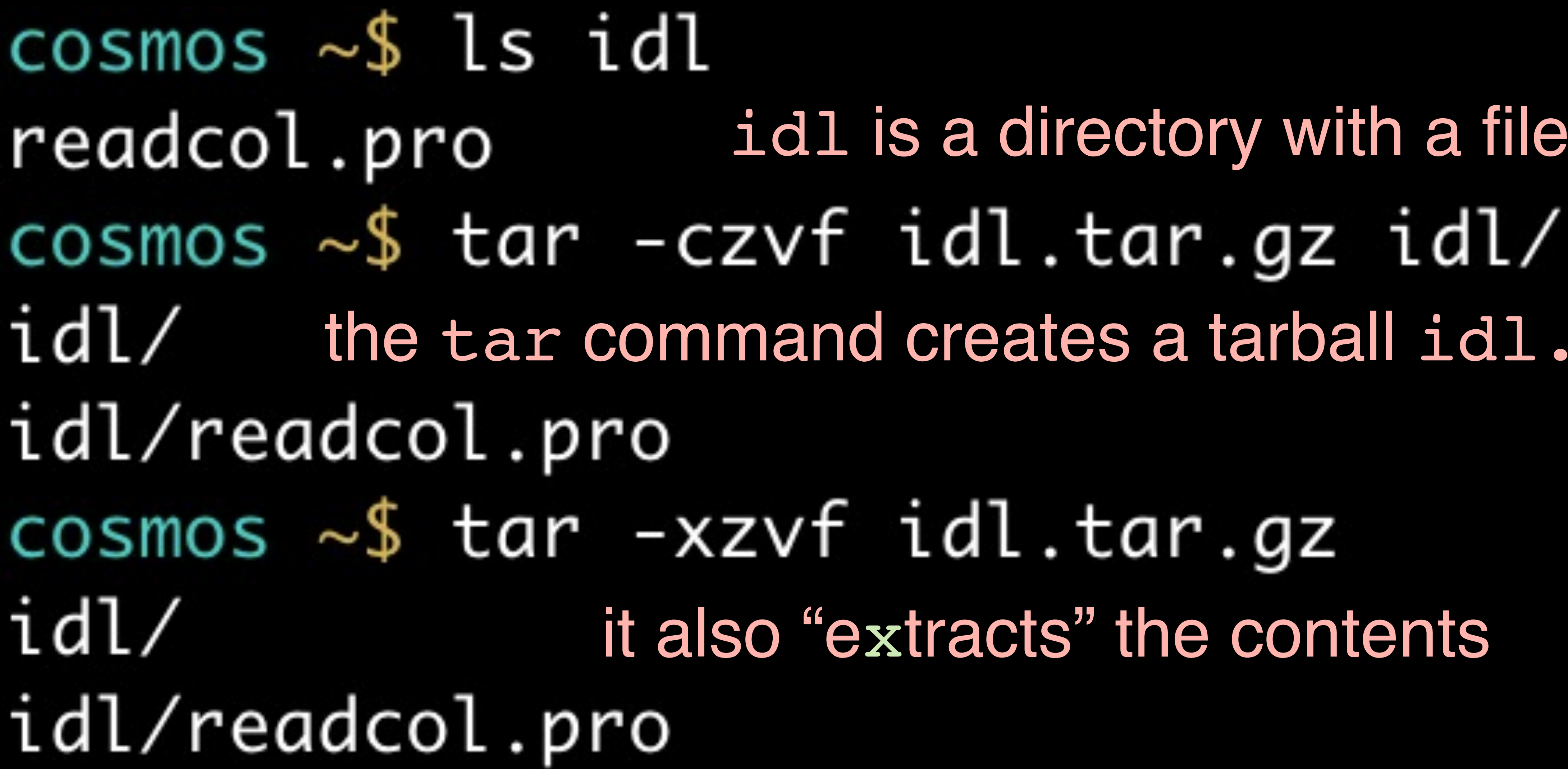
Backups Continued

- Last class we made a ‘backup’ directory
- Over the semester, we will go progressively deeper into ‘data preservation’ - i.e., making sure you don’t lose files!

tarballs

- What do you do if you want to send someone lots of files?
  - Most modern mail clients let you attach a bunch, but... tarballs are sometimes better
- Files with the suffix `.tar` or `.tar.gz` or `.tgz` are “tarballs”

Making and Opening



```
cosmos ~$ ls idl
readcol.pro
```

`idl` is a directory with a file in it

```
cosmos ~$ tar -czvf idl.tar.gz idl/
idl/
idl/readcol.pro
```

the `tar` command creates a tarball `idl.tar.gz`

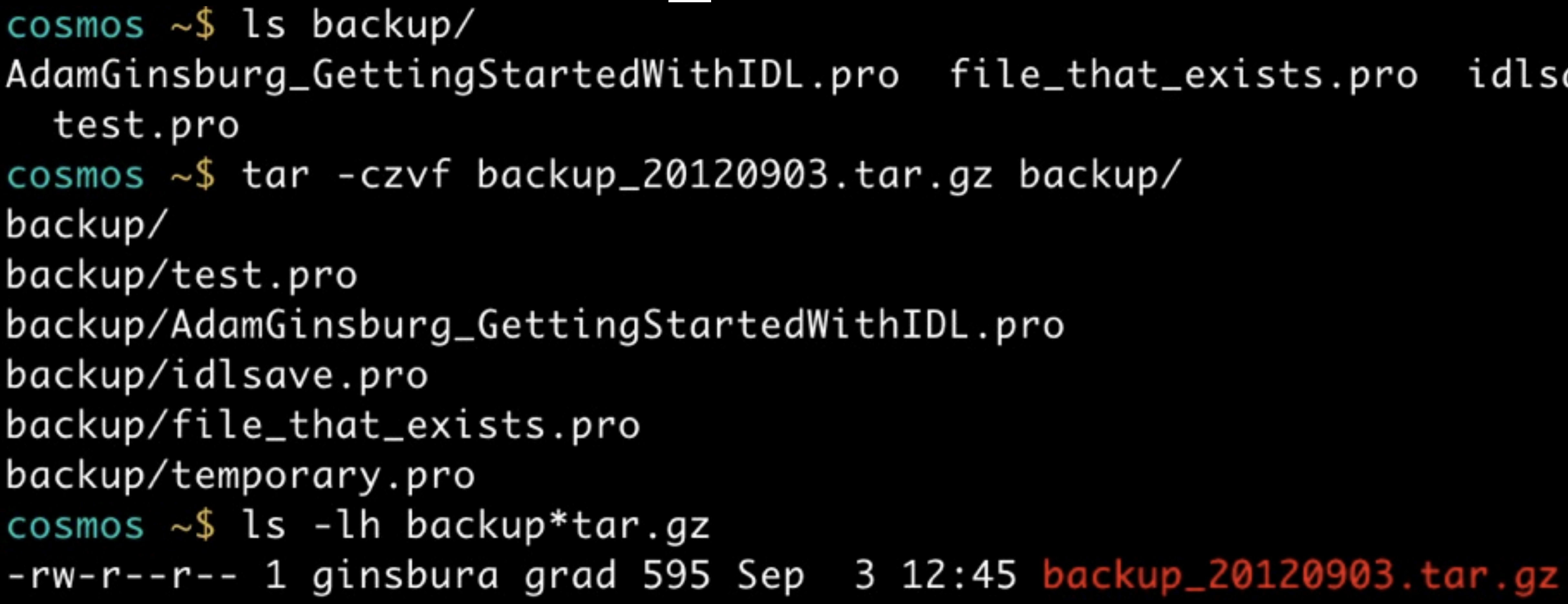
```
cosmos ~$ tar -xzvf idl.tar.gz
idl/
idl/readcol.pro
```

it also “extracts” the contents

Useful!

- You can make daily backups!

```
tar -czvf backup_20120903.tar.gz backups/
```



```
cosmos ~$ ls backup/
AdamGinsburg_GettingStartedWithIDL.pro  file_that_exists.pro  idlsave.pro
  test.pro
cosmos ~$ tar -czvf backup_20120903.tar.gz backup/
backup/
backup/test.pro
backup/AdamGinsburg_GettingStartedWithIDL.pro
backup/idlsave.pro
backup/file_that_exists.pro
backup/temporary.pro
cosmos ~$ ls -lh backup*tar.gz
-rw-r--r-- 1 ginsbura grad 595 Sep  3 12:45 backup_20120903.tar.gz
```

Copying data to cosmos

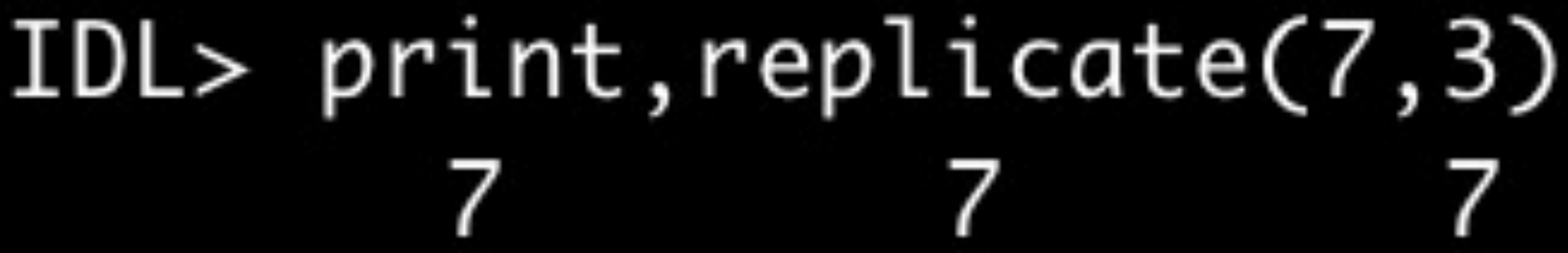
- To copy data to cosmos from your own machine, use scp
- `scp [localfile] yourname@cosmos.colorado.edu:[remotefile]`
  - `[remotefile]` can be a full path, i.e. something that starts with /home or ~/
- `scp test.pro ginsbura@cosmos.colorado.edu:~/test.pro`

ARRAYS

arrrr

Creating Arrays

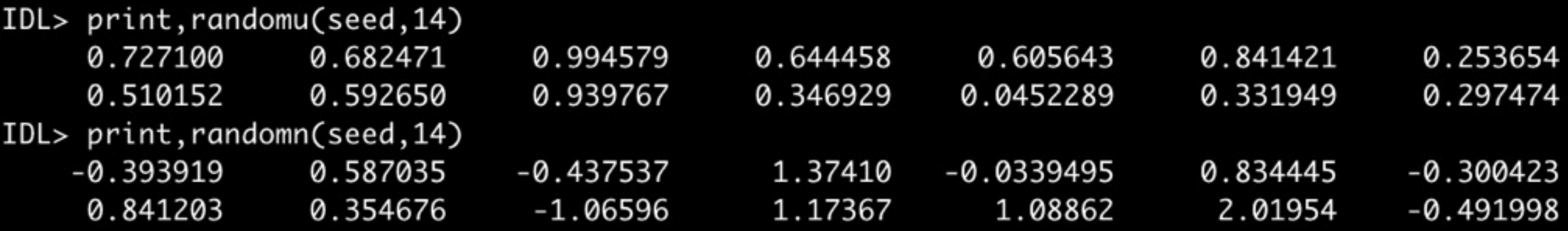
- `indgen` and its cousins `findgen`, `dindgen`, `sindgen`
  - `0,1,2,3,4....`
- `intarr, fltarr, dblarr, strarr`
  - `0,0,0,0,0....`
- `replicate(m,n)`
  - `m,m,m,m,m....`



```
IDL> print,replicate(7,3)
       7       7       7
```

Random Arrays

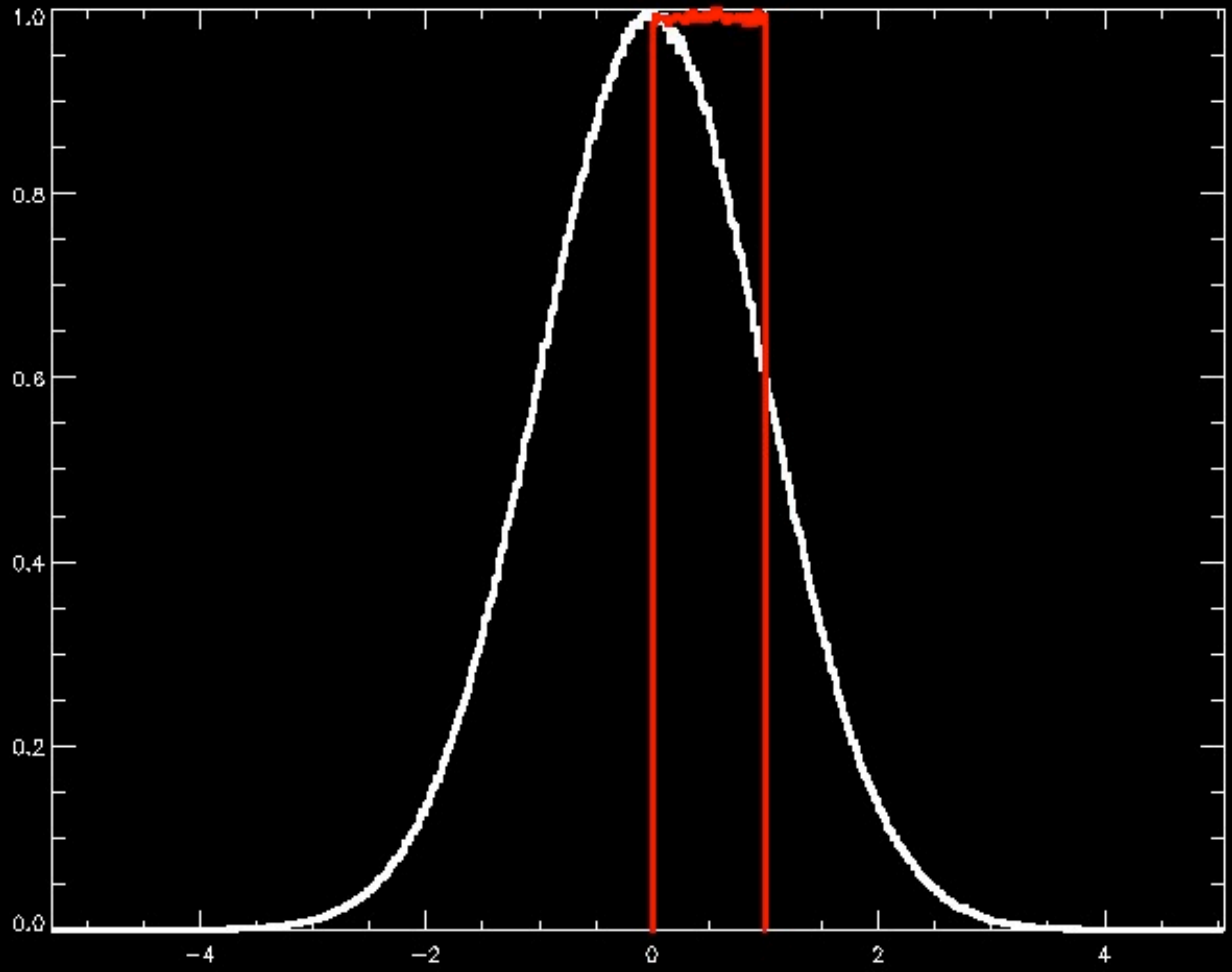
- `randomu`: Random numbers drawn from a uniform distribution
  - All numbers are between 0 and 1
- `randomn`: random normal (gaussian)
  - mean (average) = 0, standard deviation=1



```
IDL> print,randomu(seed,14)
     0.727100     0.682471     0.994579     0.644458     0.605643     0.841421     0.253654
     0.510152     0.592650     0.939767     0.346929    0.0452289     0.331949     0.297474
IDL> print,randomn(seed,14)
    -0.393919     0.587035    -0.437537      1.37410   -0.0339495     0.834445    -0.300423
     0.841203     0.354676     -1.06596      1.17367      1.08862      2.01954    -0.491998
```

Normal and Uniform Distributions

```
IDL> plothist,randomn(seed,10000000),/halfbin,bin=0.01,peak=1
IDL> plothist,randomu(seed,10000000),/halfbin,bin=0.01,/overplot,color=cgcolor('red'),peak=1
```



Array Generation

Which function generated this array?

```
-0.145539 -1.26853 0.911027 1.27301 -0.742994
```

A) `findgen`

B) `randomu`

C) `randomn`

D) `fltarr`

E) None of the above / I don't know

Array Generation

Which function generated *this* array?

```
5.00000 5.00000 5.00000 5.00000 5.00000
```

A) `findgen`

B) `randomu`

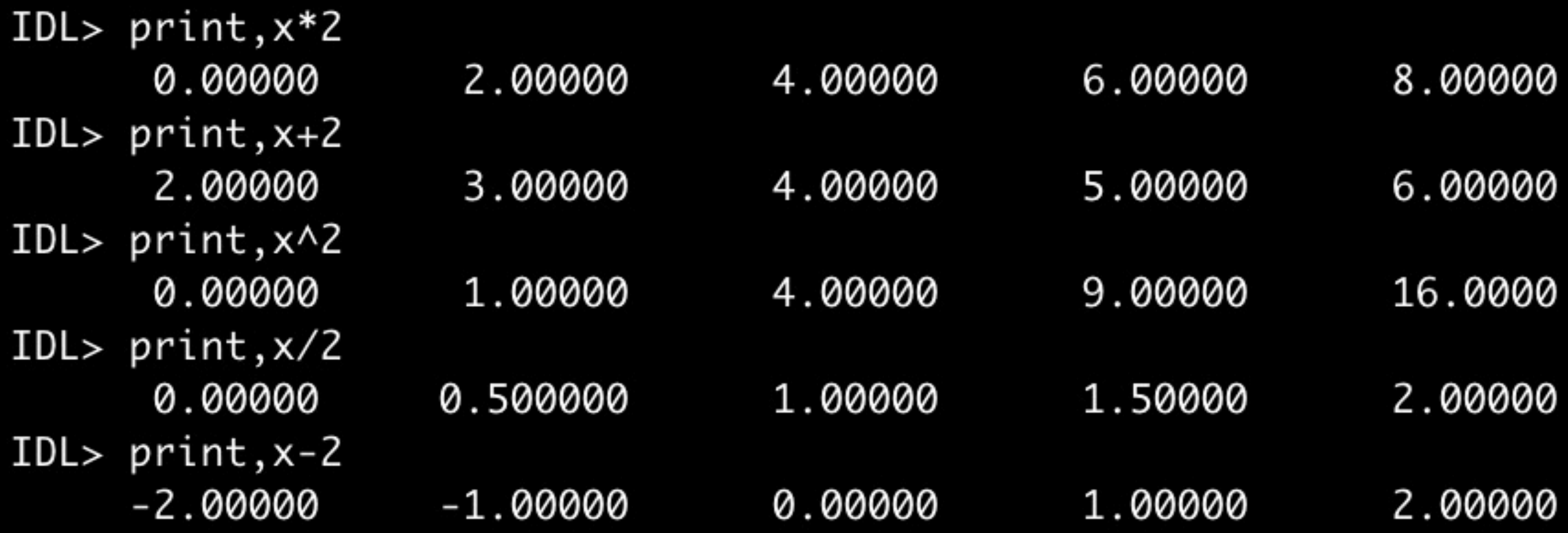
C) `randomn`

D) `fltarr`

E) None of the above / I don’t know

Elementwise Operations

- With scalars:
  - (scalars refer to non-arrays, i.e. single numbers)



```
IDL> print,x*2
      0.00000       2.00000       4.00000       6.00000       8.00000
IDL> print,x+2
      2.00000       3.00000       4.00000       5.00000       6.00000
IDL> print,x^2
      0.00000       1.00000       4.00000       9.00000       16.0000
IDL> print,x/2
      0.00000      0.500000       1.00000       1.50000       2.00000
IDL> print,x-2
     -2.00000      -1.00000       0.00000       1.00000       2.00000
```

Note the promotion: 2 is an int, but all outputs are float

With other arrays

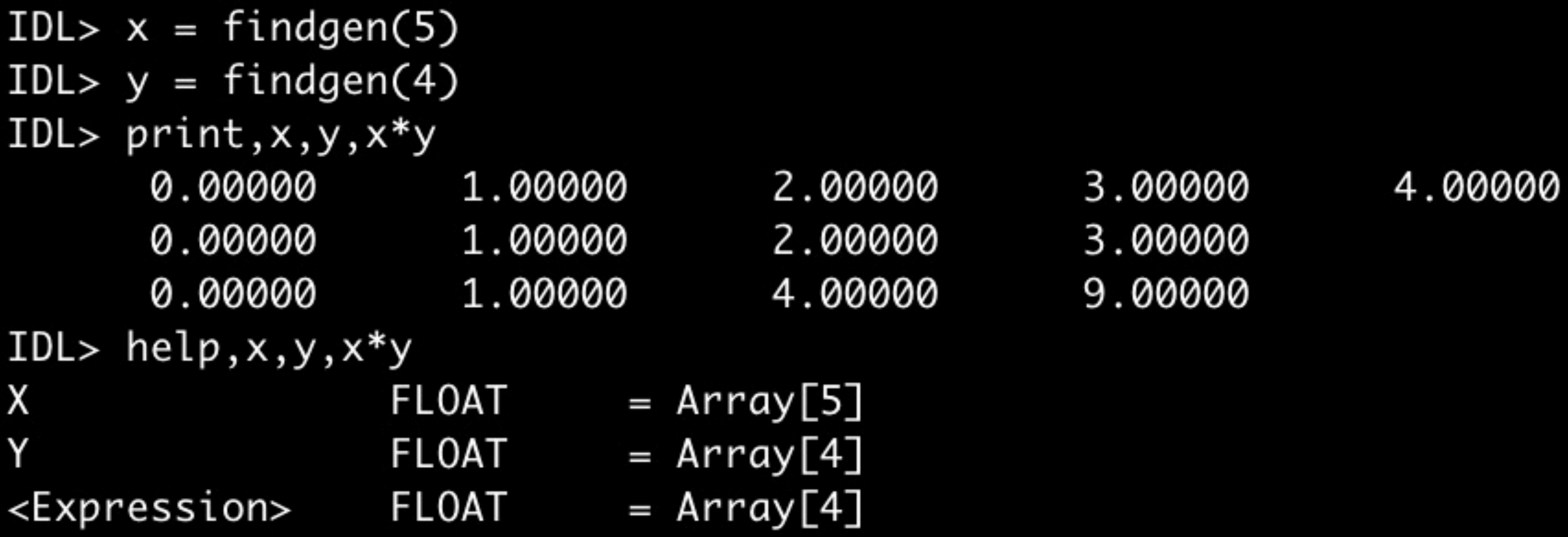
```
IDL> x = findgen(5)
IDL> y = reverse(findgen(5))
IDL> print,x,y
 x    0.00000      1.00000      2.00000      3.00000      4.00000
 y    4.00000      3.00000      2.00000      1.00000      0.00000
IDL> print,x,y,x+y
      0.00000      1.00000      2.00000      3.00000      4.00000
    + 4.00000      3.00000      2.00000      1.00000      0.00000
      4.00000      4.00000      4.00000      4.00000      4.00000
IDL> print,x,y,x/y
      0.00000      1.00000      2.00000      3.00000      4.00000
    ÷ 4.00000      3.00000      2.00000      1.00000      0.00000
      0.00000     0.333333      1.00000      3.00000          Inf
% Program caused arithmetic error: Floating divide by 0
IDL> print,x,y,x^y
      0.00000      1.00000      2.00000      3.00000      4.00000
    ^ 4.00000      3.00000      2.00000      1.00000      0.00000
      0.00000      1.00000      4.00000      3.00000      1.00000
IDL> print,x,y,x*y
      0.00000      1.00000      2.00000      3.00000      4.00000
   x  4.00000      3.00000      2.00000      1.00000      0.00000
      0.00000      3.00000      4.00000      3.00000      0.00000
```

With other arrays

```
IDL> x = findgen(5)
IDL> y = reverse(findgen(5))
IDL> print,x,y
 x     0.00000      1.00000      2.00000      3.00000      4.00000
 y     4.00000      3.00000      2.00000      1.00000      0.00000
IDL> print,x,y,x+y
       0.00000      1.00000      2.00000      3.00000      4.00000
     + 4.00000      3.00000      2.00000      1.00000      0.00000
       4.00000      4.00000      4.00000      4.00000      4.00000
IDL> print,x,y,x/y
       0.00000      1.00000      2.00000      3.00000      4.00000
     ÷ 4.00000      3.00000      2.00000      1.00000      0.00000
       0.00000     0.333333      1.00000      3.00000          Inf
% Program caused arithmetic error: Floating divide by 0
IDL> print,x,y,x^y
       0.00000      1.00000      2.00000      3.00000      4.00000
     ^ 4.00000      3.00000      2.00000      1.00000      0.00000
       0.00000      1.00000      4.00000      3.00000      1.00000
IDL> print,x,y,x*y
       0.00000      1.00000      2.00000      3.00000      4.00000
     x 4.00000      3.00000      2.00000      1.00000      0.00000
       0.00000      3.00000      4.00000      3.00000      0.00000
```

Multiplication with Other Arrays

- This is something that IDL does “wrong”
  - (really, it’s just “bad behavior” but it is evil)



```
IDL> x = findgen(5)
IDL> y = findgen(4)
IDL> print,x,y,x*y
      0.00000      1.00000      2.00000      3.00000      4.00000
      0.00000      1.00000      2.00000      3.00000
      0.00000      1.00000      4.00000      9.00000
IDL> help,x,y,x*y
X               FLOAT     = Array[5]
Y               FLOAT     = Array[4]
<Expression>    FLOAT     = Array[4]
```

Multiplication with Other Arrays

IDL (bad behavior)

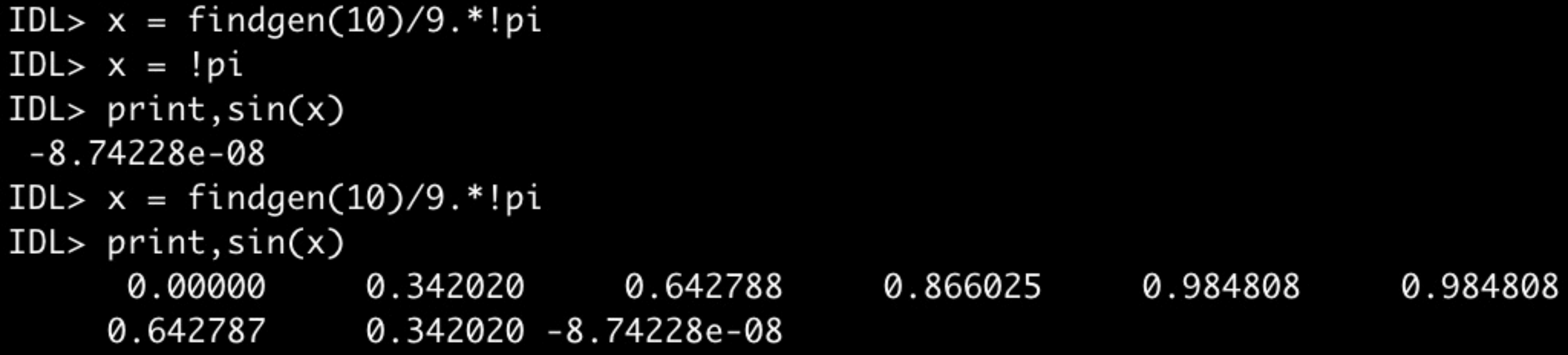
```
IDL> x = findgen(5)
IDL> y = findgen(4)
IDL> print,x,y,x*y
      0.00000      1.00000      2.00000      3.00000      4.00000
      0.00000      1.00000      2.00000      3.00000
      0.00000      1.00000      4.00000      9.00000
IDL> help,x,y,x*y
X               FLOAT     = Array[5]
Y               FLOAT     = Array[4]
<Expression>    FLOAT     = Array[4]
```

Python (good behavior)

```
>>> x = arange(5)
>>> y = arange(4)
>>> print x*y
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: operands could not be broadcast together with shapes (5) (4)
```

Functions on Arrays

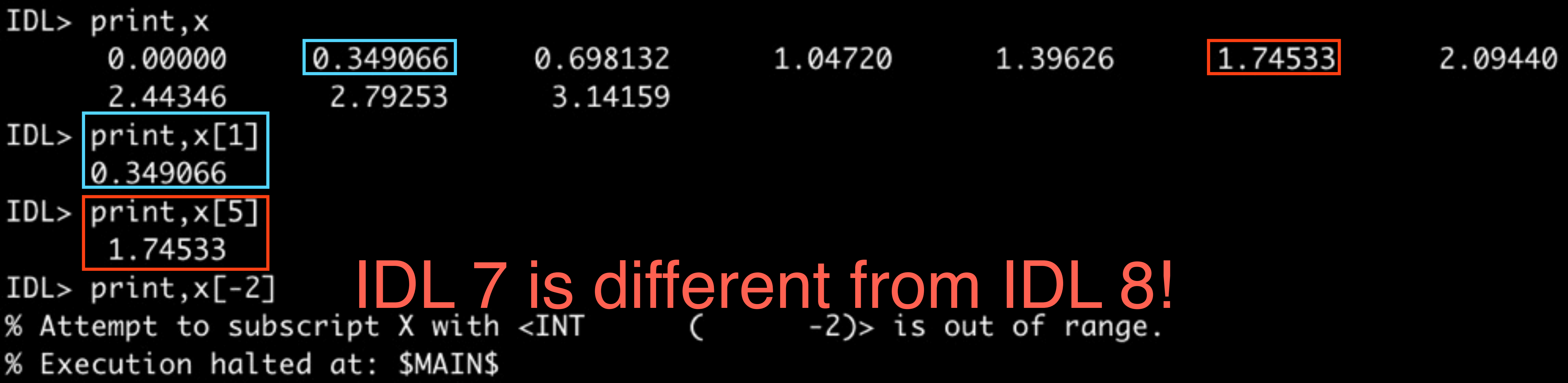
- Same as addition/subtraction/etc: applied elementwise



```
IDL> x = findgen(10)/9.*!pi
IDL> x = !pi
IDL> print,sin(x)
 -8.74228e-08
IDL> x = findgen(10)/9.*!pi
IDL> print,sin(x)
      0.00000     0.342020     0.642788     0.866025     0.984808     0.984808
     0.642787     0.342020 -8.74228e-08
```

Indexing Arrays

- How do you get part of the array? Or just one element?

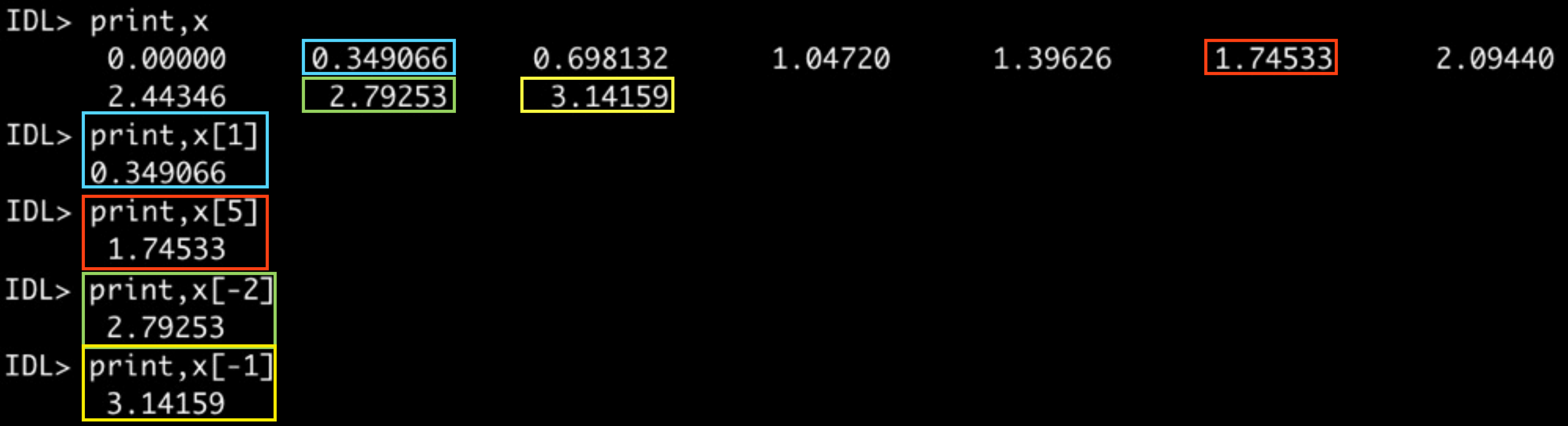


```
IDL> print,x
      0.00000     0.349066     0.698132      1.04720      1.39626      1.74533      2.09440
      2.44346      2.79253      3.14159
IDL> print,x[1]
0.349066
IDL> print,x[5]
      1.74533
IDL> print,x[-2]
% Attempt to subscript X with <INT      (      -2)> is out of range.
% Execution halted at: $MAIN$
```

IDL 7 is different from IDL 8!

Indexing Arrays

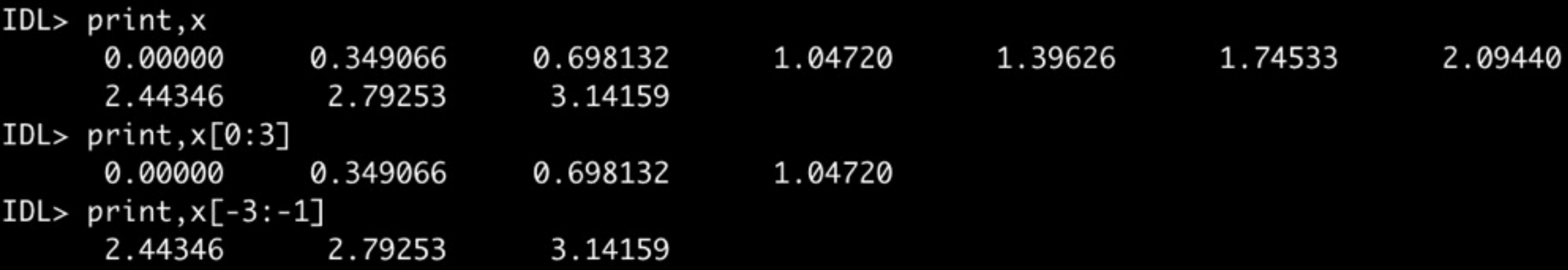
- How do you get part of the array? Or just one element?



```
IDL> print,x
      0.00000     0.349066     0.698132      1.04720      1.39626      1.74533      2.09440
      2.44346      2.79253      3.14159
IDL> print,x[1]
     0.349066
IDL> print,x[5]
      1.74533
IDL> print,x[-2]
      2.79253
IDL> print,x[-1]
      3.14159
```

Indexing Arrays

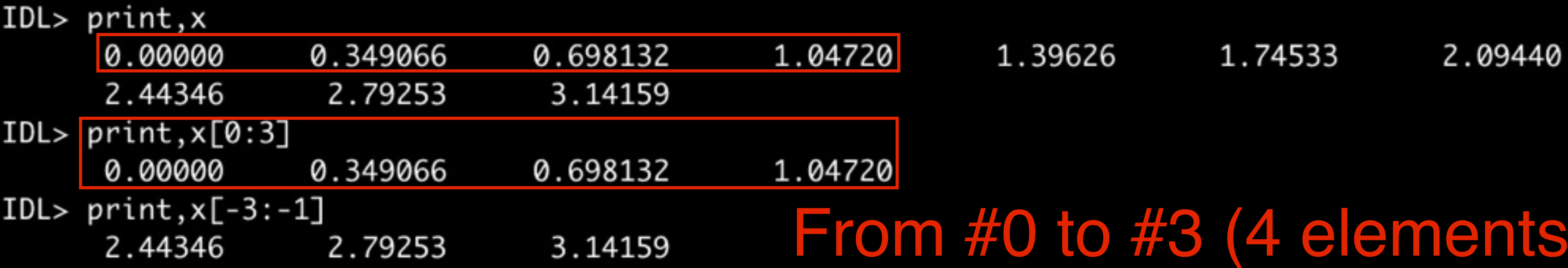
- “Subarrays”
  - `array[start:finish]`
  - there are `(finish-start)+1` elements in the subarray



```
IDL> print,x
      0.00000      0.349066      0.698132       1.04720       1.39626       1.74533       2.09440
      2.44346       2.79253       3.14159
IDL> print,x[0:3]
      0.00000      0.349066      0.698132       1.04720
IDL> print,x[-3:-1]
      2.44346       2.79253       3.14159
```

Indexing Arrays

- “Subarrays”
  - `array[start:finish]`
  - there are `(finish-start)+1` elements in the subarray

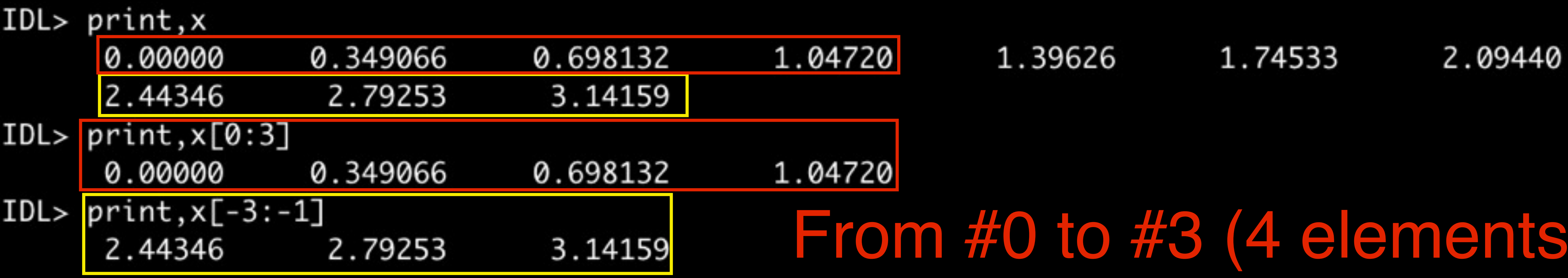


```
IDL> print,x
      0.00000      0.349066      0.698132       1.04720       1.39626       1.74533       2.09440
      2.44346       2.79253       3.14159
IDL> print,x[0:3]
      0.00000      0.349066      0.698132       1.04720
IDL> print,x[-3:-1]
      2.44346       2.79253       3.14159
```

From #0 to #3 (4 elements)

Indexing Arrays

- “Subarrays”
  - `array[start:finish]`
  - there are `(finish-start)+1` elements in the subarray



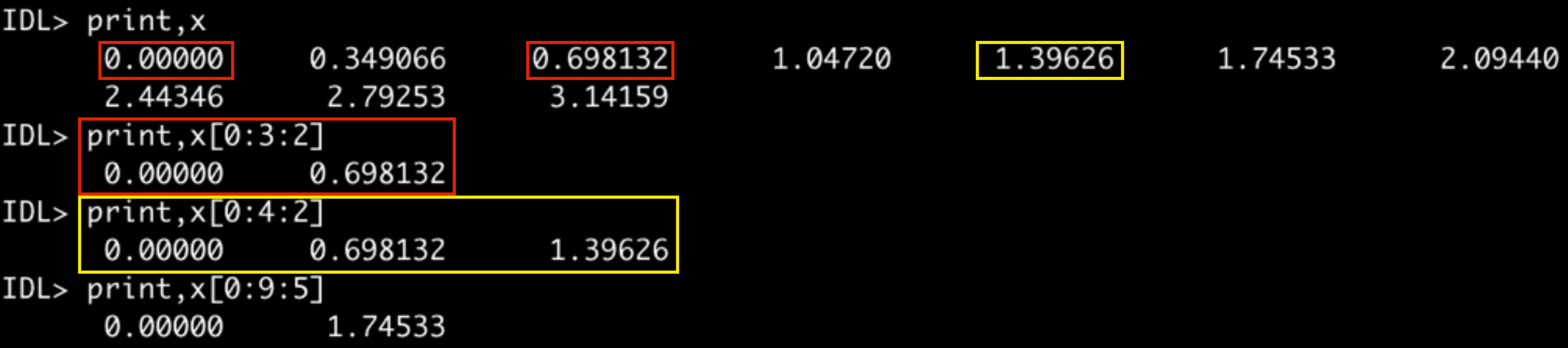
```
IDL> print,x
     0.00000      0.349066      0.698132       1.04720       1.39626       1.74533       2.09440
     2.44346       2.79253       3.14159
IDL> print,x[0:3]
     0.00000      0.349066      0.698132       1.04720
IDL> print,x[-3:-1]
     2.44346       2.79253       3.14159
```

From #0 to #3 (4 elements)

From third-to-last to last

Indexing Arrays

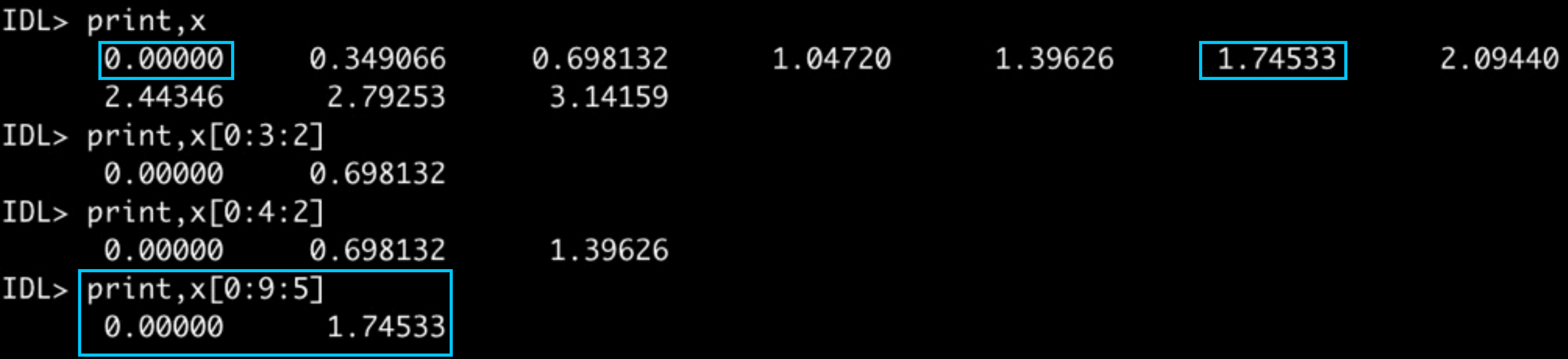
- “Striding”
  - `array[start:finish:step]`



```
IDL> print,x
       0.00000      0.349066      0.698132       1.04720       1.39626       1.74533       2.09440
       2.44346       2.79253       3.14159
IDL> print,x[0:3:2]
       0.00000      0.698132
IDL> print,x[0:4:2]
       0.00000      0.698132       1.39626
IDL> print,x[0:9:5]
       0.00000       1.74533
```

Indexing Arrays

- “Striding”
  - `array[start:finish:step]`



```
IDL> print,x
      0.00000     0.349066     0.698132      1.04720      1.39626      1.74533      2.09440
      2.44346      2.79253      3.14159
IDL> print,x[0:3:2]
      0.00000     0.698132
IDL> print,x[0:4:2]
      0.00000     0.698132      1.39626
IDL> print,x[0:9:5]
      0.00000      1.74533
```

Striding

Evaluate:

```
IDL> x=findgen(15)
IDL> print,x[0:12:3]
```

A)
```
0.00000      3.00000      6.00000      9.00000
```

B)
```
0.00000      3.00000      6.00000      9.00000      12.0000
```

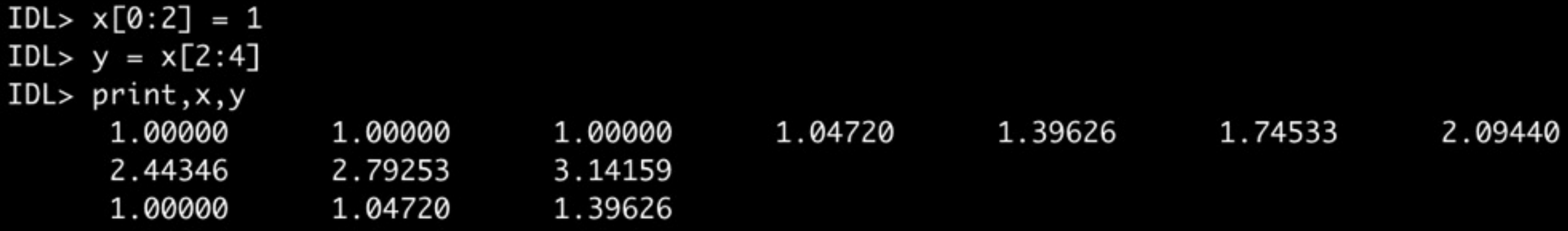
C)
```
1.00000      4.00000      7.00000      10.0000
```

D)
```
1.00000      4.00000      7.00000      10.0000      13.0000
```

E) None of the above / I don’t know

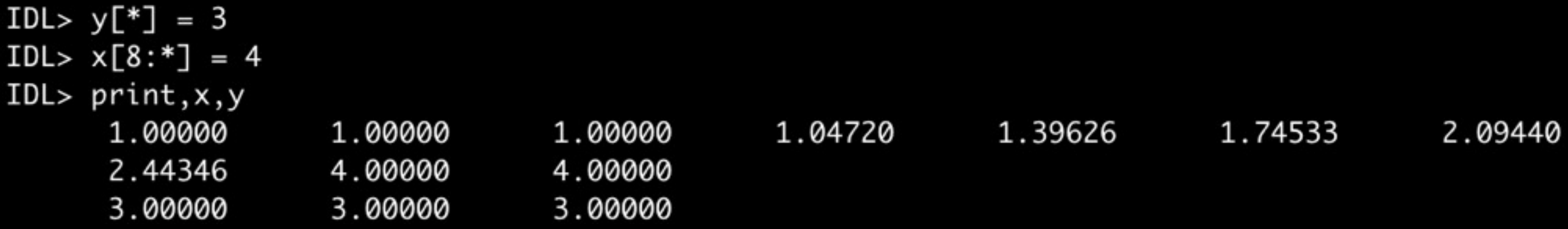
Indexing Arrays

- Can be used on either side of an = sign



```
IDL> x[0:2] = 1
IDL> y = x[2:4]
IDL> print,x,y
      1.00000       1.00000       1.00000       1.04720       1.39626       1.74533       2.09440
      2.44346       2.79253       3.14159
      1.00000       1.04720       1.39626
```

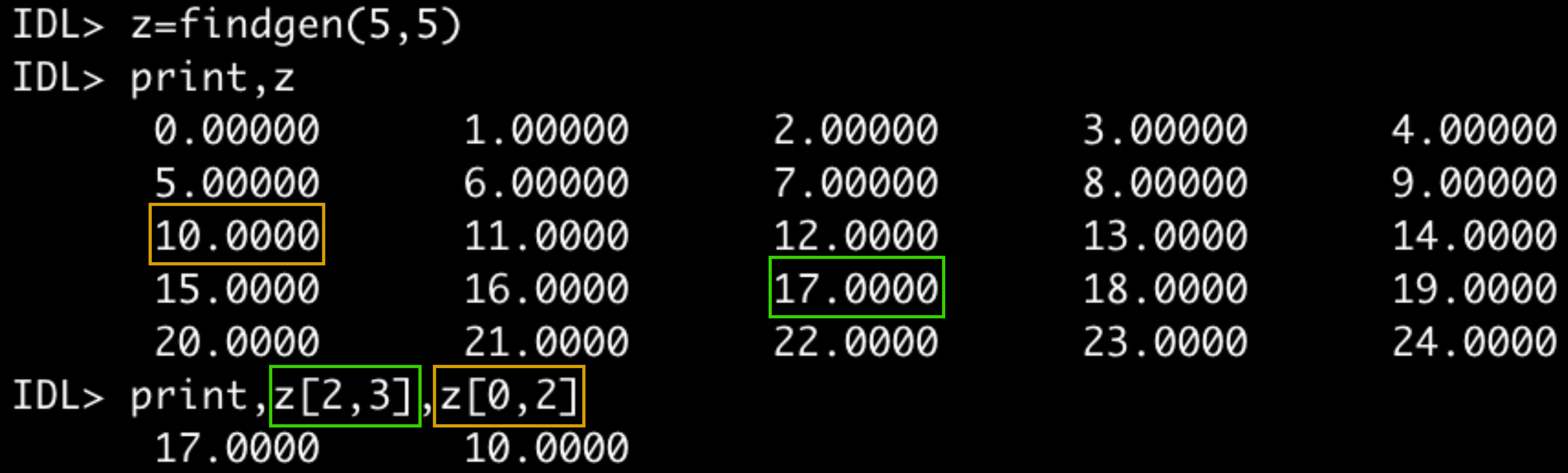
* can be used as a wildcard:



```
IDL> y[*] = 3
IDL> x[8:*] = 4
IDL> print,x,y
      1.00000       1.00000       1.00000       1.04720       1.39626       1.74533       2.09440
      2.44346       4.00000       4.00000
      3.00000       3.00000       3.00000
```

Arrays

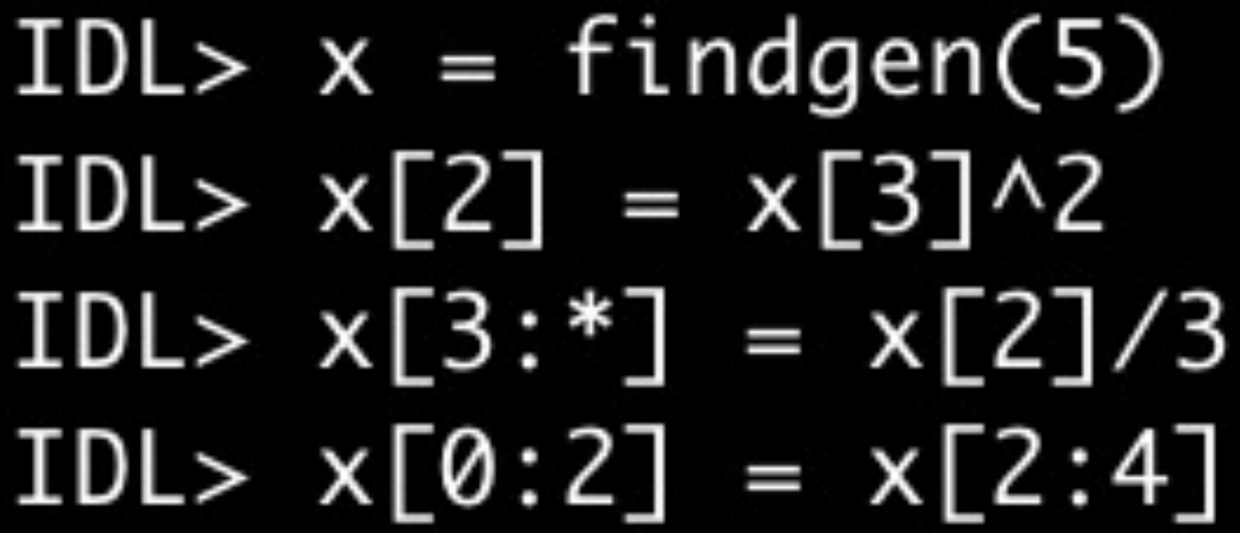
- There are also two-dimensional arrays



```
IDL> z=findgen(5,5)
IDL> print,z
      0.00000      1.00000      2.00000      3.00000      4.00000
      5.00000      6.00000      7.00000      8.00000      9.00000
      10.0000      11.0000      12.0000      13.0000      14.0000
      15.0000      16.0000      17.0000      18.0000      19.0000
      20.0000      21.0000      22.0000      23.0000      24.0000
IDL> print,z[2,3],z[0,2]
      17.0000      10.0000
```

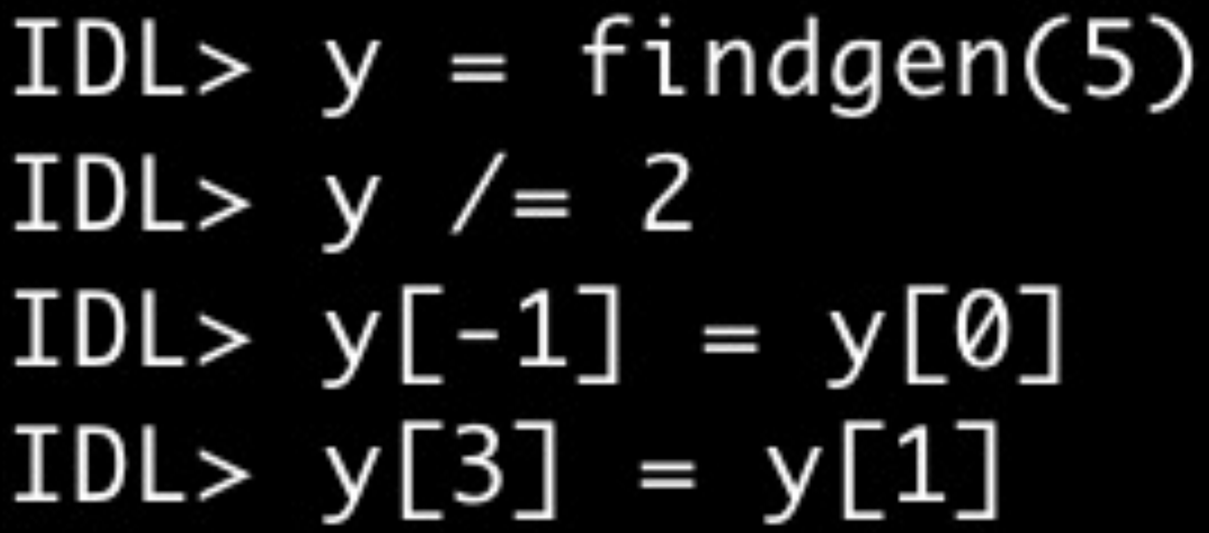
That was a lot...

- What is x?



```
IDL> x = findgen(5)
IDL> x[2] = x[3]^2
IDL> x[3:*] = x[2]/3
IDL> x[0:2] = x[2:4]
```

- What is y?



```
IDL> y = findgen(5)
IDL> y /= 2
IDL> y[-1] = y[0]
IDL> y[3] = y[1]
```

What is x+y? x*y?

Do these on your own, then after ~1 minute, you can discuss with a neighbor.

```
IDL> x = findgen(5)
IDL> x[2] = x[3]^2
IDL> x[3:*] = x[2]/3
IDL> x[0:2] = x[2:4]
```

A)
```
9.00000      3.00000      3.00000      3.00000      3.00000
```

B)
```
3.00000      3.00000      3.00000      3.00000      3.00000
```

C)
```
4.00000      1.33333      1.33333      1.33333      1.33333
```

D)
```
0.00000      1.00000      2.00000      3.00000      4.00000
```

E) I don’t know / None of the above

```
IDL> y = findgen(5)
IDL> y /= 2
IDL> y[-1] = y[0]
IDL> y[3] = y[1]
```

A)
```
0.00000      0.500000      0.00000      0.00000      2.00000
```

B)
```
0        0        0        0        2
```

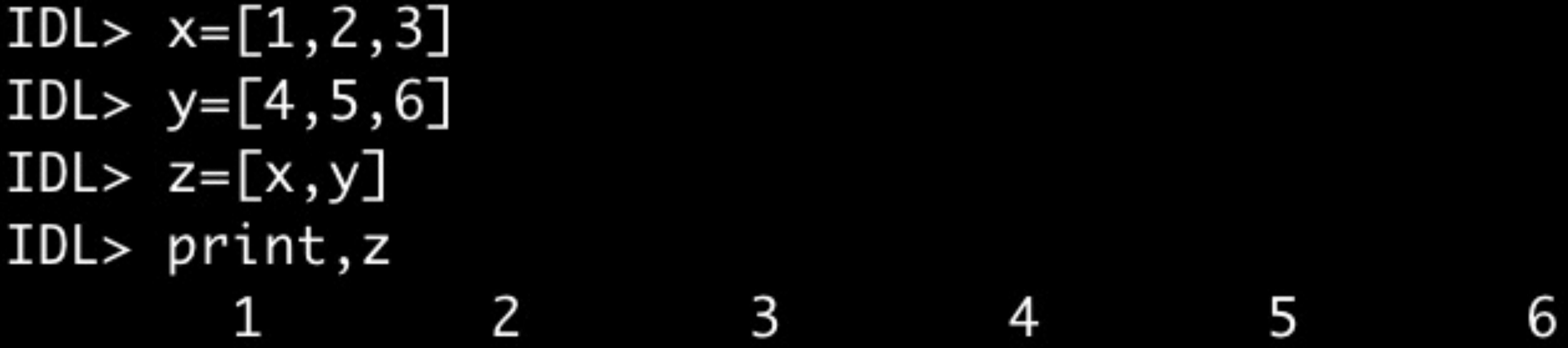
C)
```
0.00000      0.500000      1.00000      0.500000      0.00000
```

D)
```
0        0        1        0        0
```

E) I don’t know / None of the above

Array Concatenation

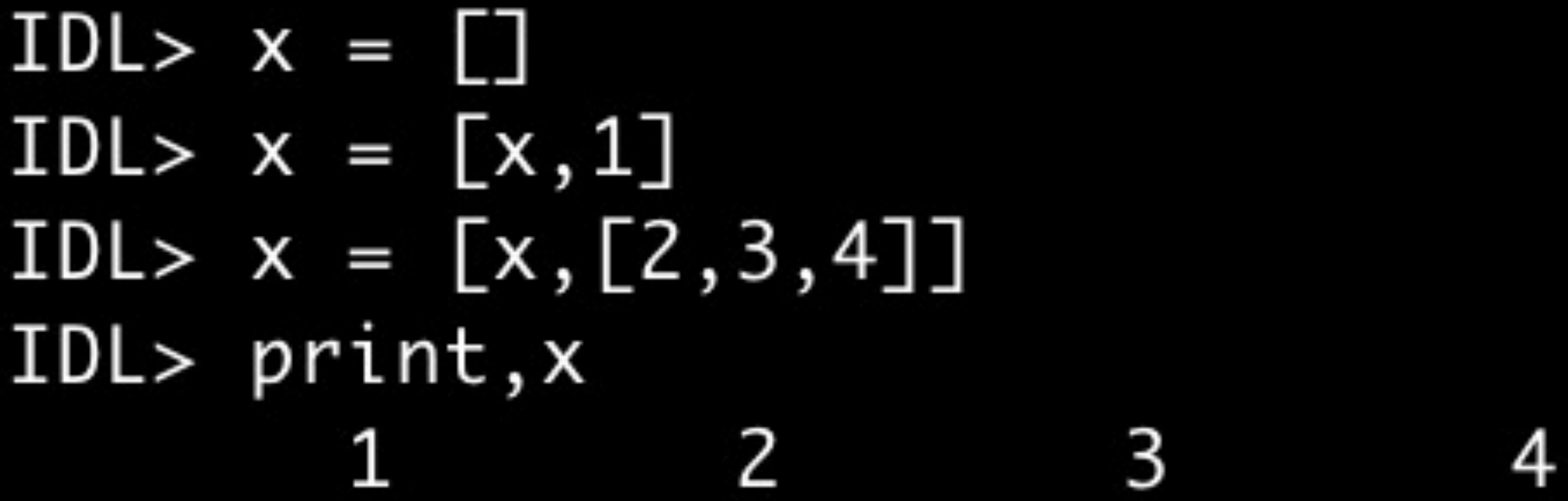
- Stick two arrays together end-to-end



```
IDL> x=[1,2,3]
IDL> y=[4,5,6]
IDL> z=[x,y]
IDL> print,z
       1       2       3       4       5       6
```

Array Concatenation

- Can start with empty arrays



```
IDL> x = []
IDL> x = [x,1]
IDL> x = [x,[2,3,4]]
IDL> print,x
       1       2       3       4
```

Evaluate:

```
IDL> x=[1,2,3]
IDL> y=[3.,4.,5.]
IDL> z=[x,y]
```

A) `4.00000      6.00000      8.00000`

B) `1          2          3          3          4          5`

C) `1.00000      2.00000      3.00000      3.00000      4.00000      5.00000`

D) `3.00000      8.00000      15.0000`

E) I don’t know / None of the above

Array-specific Functions

- `n_elements`: How many elements are in the array?
- `size`: Details about the size, shape, and type of an array [actually not array-specific]
- `total`: Sum of the array.
- `mean, stddev`: average and standard deviation

Array-specific Functions

- `min`, `max`, `reverse`, `shift`, `reform`, `rebin`... etc.

Array Generation

How would you make this array using `findgen`?

`[0.0, 0.2, 0.4, 0.6, 0.8, 1.0]`

A) `findgen(6) / 5.`

B) `findgen(5) / 5.`

C) `findgen(4) / 5.`

D) `findgen(4) / 4.`

E) None of the above / I don’t know

Array Generation

How would you make this array using `findgen`?

`[0.0, 0.2, 0.4, 0.6, 0.8, 1.0]`

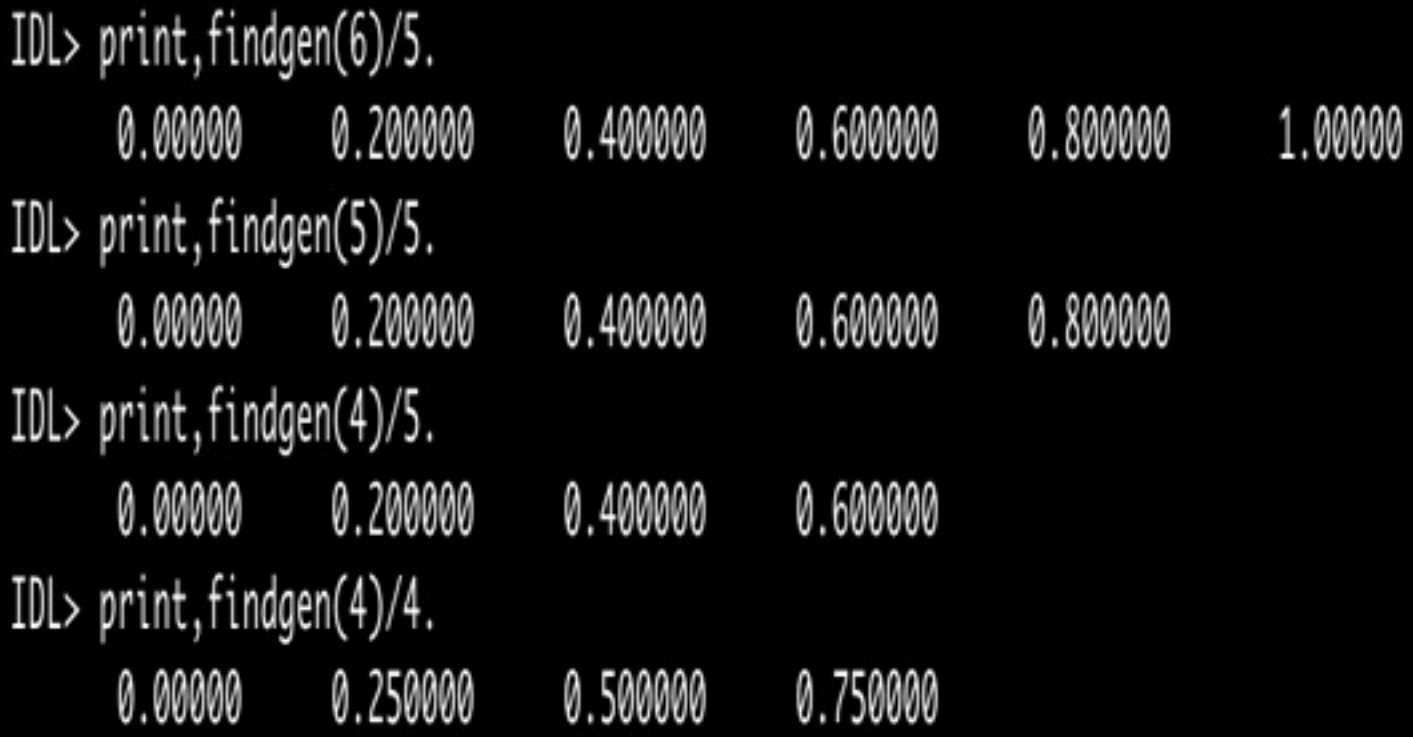
A) `findgen(6) / 5.`

B) `findgen(5) / 5.`

C) `findgen(4) / 5.`

D) `findgen(4) / 4.`

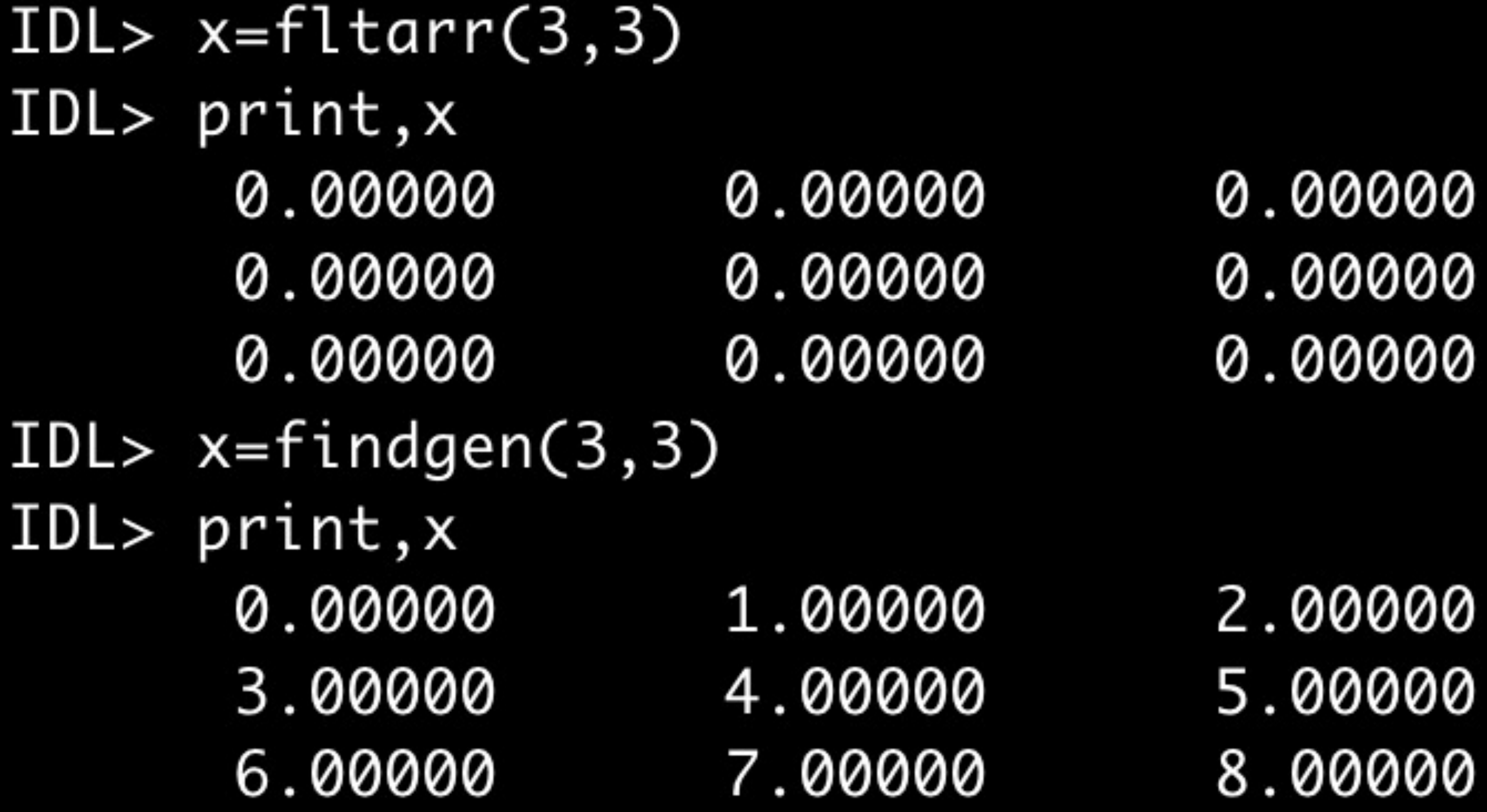
E) None of the above / I don’t know



```
IDL> print,findgen(6)/5.
      0.00000     0.200000     0.400000     0.600000     0.800000      1.00000
IDL> print,findgen(5)/5.
      0.00000     0.200000     0.400000     0.600000     0.800000
IDL> print,findgen(4)/5.
      0.00000     0.200000     0.400000     0.600000
IDL> print,findgen(4)/4.
      0.00000     0.250000     0.500000     0.750000
```

Two-dimensional arrays

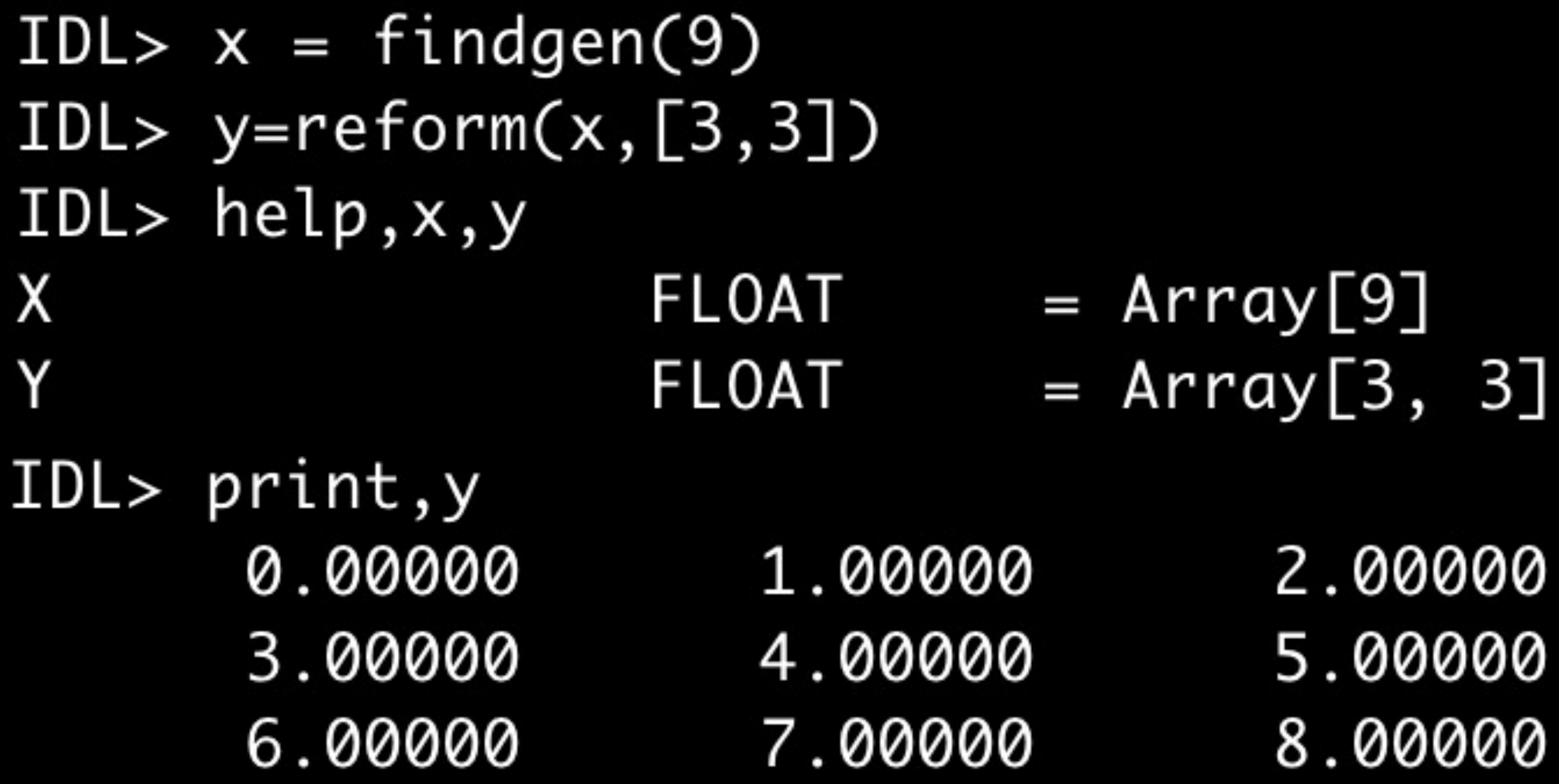
- IDL can handle up to 8 dimensions, but we rarely use more than 3 or 4
- The usual commands work as you might expect:



```
IDL> x=fltarr(3,3)
IDL> print,x
      0.00000      0.00000      0.00000
      0.00000      0.00000      0.00000
      0.00000      0.00000      0.00000
IDL> x=findgen(3,3)
IDL> print,x
      0.00000      1.00000      2.00000
      3.00000      4.00000      5.00000
      6.00000      7.00000      8.00000
```

Two-dimensional arrays

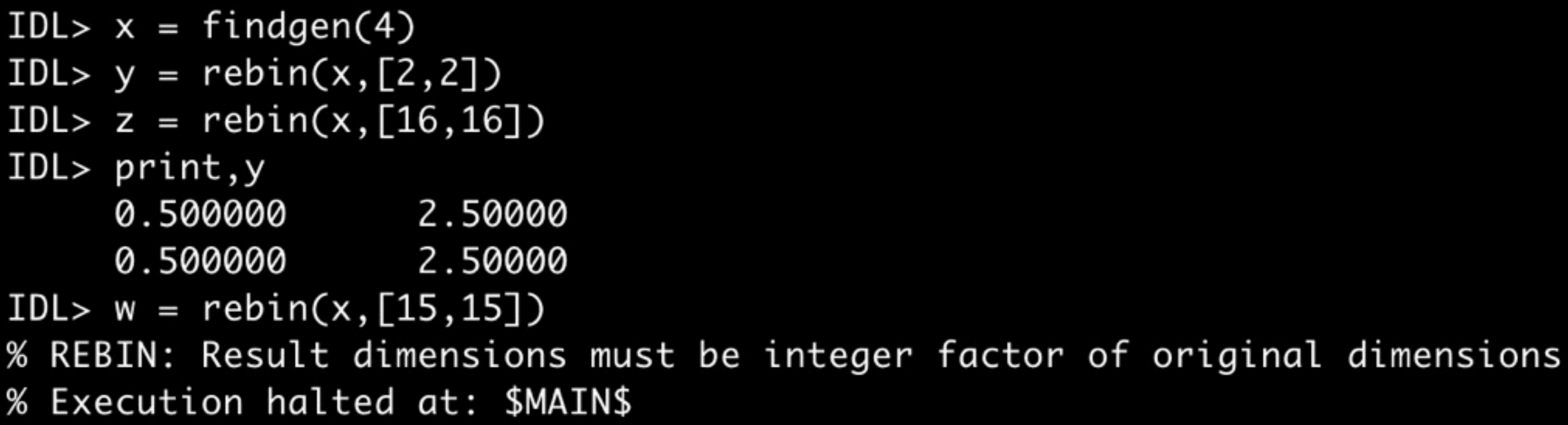
- `reform`: change the shape of an array



```
IDL> x = findgen(9)
IDL> y=reform(x,[3,3])
IDL> help,x,y
X               FLOAT     = Array[9]
Y               FLOAT     = Array[3, 3]
IDL> print,y
      0.00000      1.00000      2.00000
      3.00000      4.00000      5.00000
      6.00000      7.00000      8.00000
```

Two-dimensional arrays

- `rebin`: change the size of an array



```
IDL> x = findgen(4)
IDL> y = rebin(x,[2,2])
IDL> z = rebin(x,[16,16])
IDL> print,y
     0.500000      2.50000
     0.500000      2.50000
IDL> w = rebin(x,[15,15])
% REBIN: Result dimensions must be integer factor of original dimensions
% Execution halted at: $MAIN$
```