WDID review

What is $\sin(90^o)$?

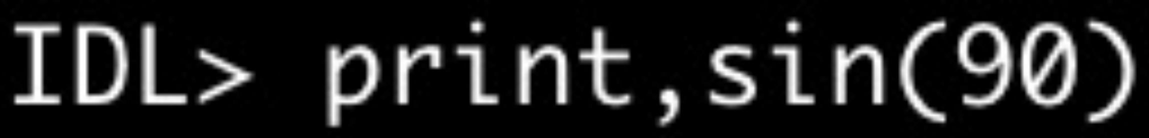
A) 0

B) 1

C) -1

D) $\pi / 2$

E) None of the above / I don’t know

Evaluate: 

```
IDL> print,sin(90)
```

A) 0

B) 1

C) -1

D) π / 2

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

Evaluate: `IDL> print,-4+24./6`

A) 3.3333

B) 0.0000

C) 3

D) 4.6666

E) None of the above / I don’t know

Evaluate:

```
IDL> print,(-4+24.)/6
```

A) 3.3333

B) 0.0000

C) 3

D) 4.6666

E) None of the above / I don’t know

Will this work? `IDL> print,"I have "+6+" frogs"`

A) Yes

B) No

C) I don’t know

Which of these will *not* work?

A)
```
IDL> print,"I have ",6," frogs"
```

B)
```
IDL> print,"I have "+string(6)+" frogs"
```

C)
```
IDL> print,"I have "+'6'+" frogs"
```

D)
```
IDL> print,"I have "+"6"+" frogs"
```

E) None of the above / I don’t know

Which of these could be `randomu(5,2)`?

A)
```
0.521414     -0.945489     0.683157      1.22728    0.0753817
```

B)
```
0.521414     -0.945489
```

C)
```
0.172861      0.680409     0.917078     0.917510     0.766779
```

D)
```
0.172861      0.680409
```

E) None of the above / I don’t know

Saving and Recovering Data

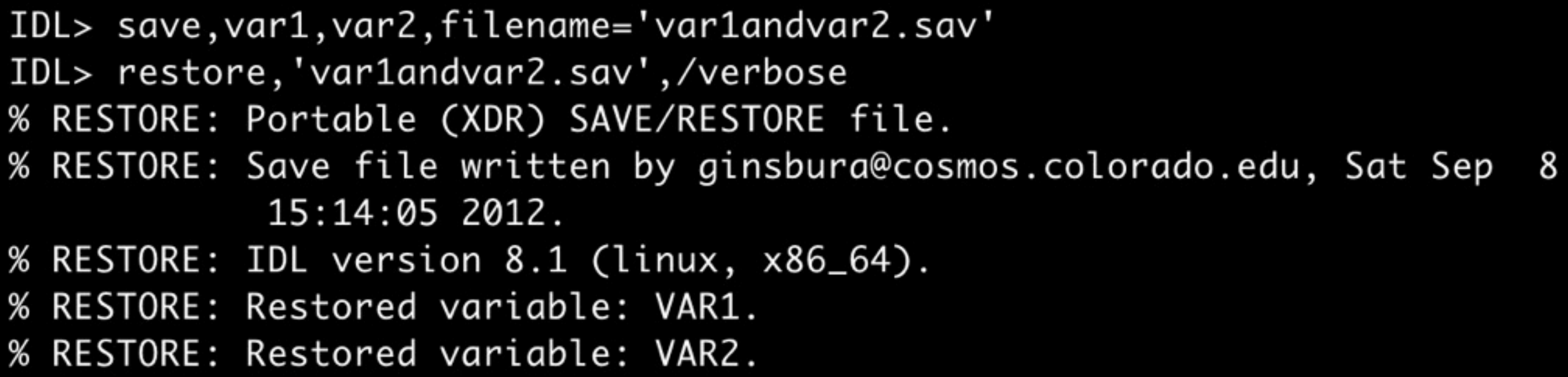
- IDL has convenience procedures '`save`' and '`restore`'
- You can save any combination of variables to a save file
  - This is unique to IDL and very convenient
  - Python can be used to read, but not write, IDL save files

Save

- `save,var1,var2,filename='var1andvar2.sav'`
- If you leave filename unspecified, defaults to `idlsave.dat`
  - This is bad: `.dat` files frequently refer to ASCII text files, while savefiles are binary
- I prefer `.sav` as a suffix

Restore

- To get your data back in IDL, in the same variable names you used to save them, use:



```
IDL> save,var1,var2,filename='var1andvar2.sav'
IDL> restore,'var1andvar2.sav',/verbose
% RESTORE: Portable (XDR) SAVE/RESTORE file.
% RESTORE: Save file written by ginsbura@cosmos.colorado.edu, Sat Sep  8
           15:14:05 2012.
% RESTORE: IDL version 8.1 (linux, x86_64).
% RESTORE: Restored variable: VAR1.
% RESTORE: Restored variable: VAR2.
```

Clean Slate

- `.reset_session` will delete all variables and unload all compiled programs
- Will be covered more in later chapters

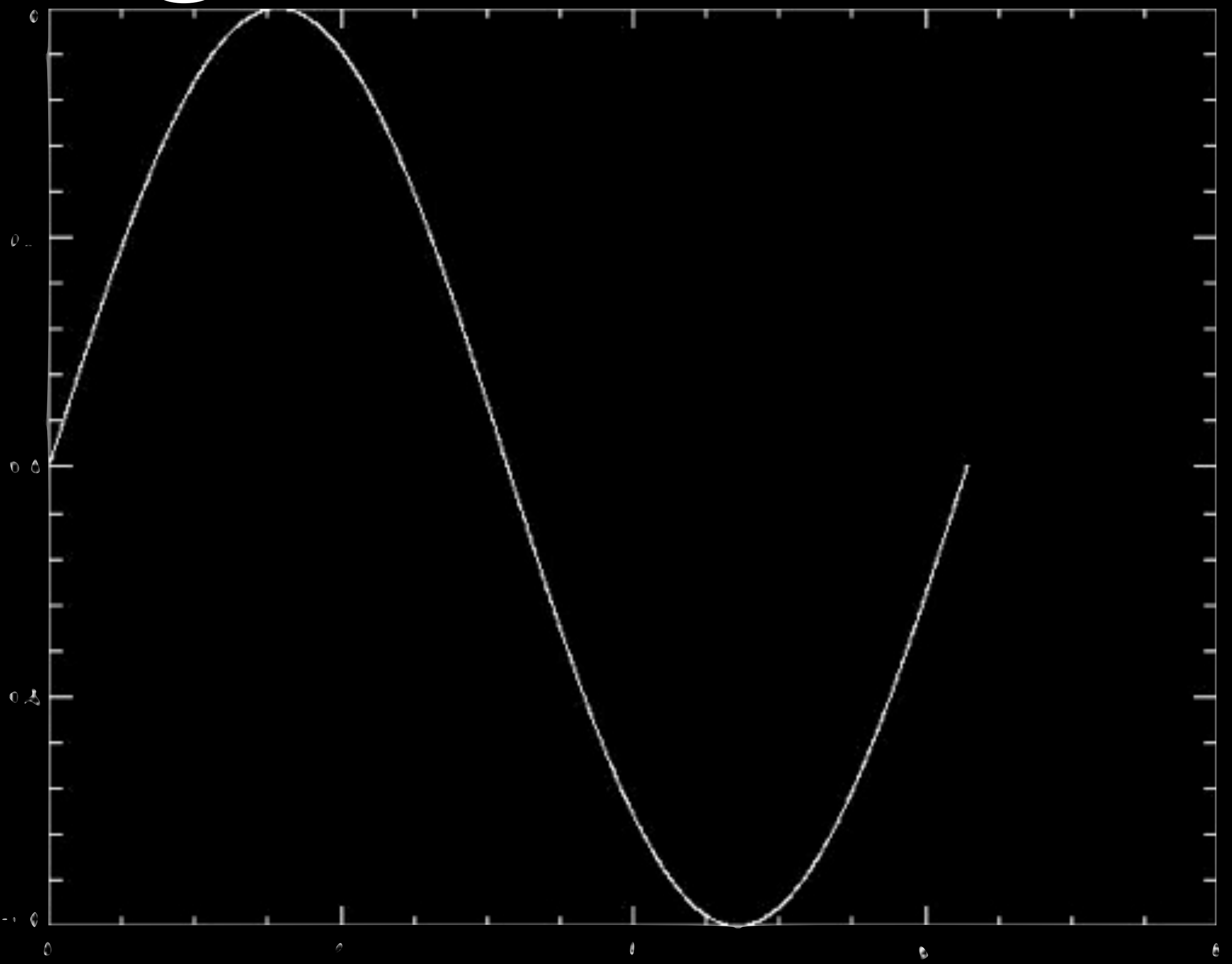
Saving Images

- There are procedures to write various image formats:
- `write_bmp, write_jpeg, write_pict, write_png, write_tiff`

Reading Images

- Functions:
  - `read_bmp, read_tiff, read_png`
- Procedures:
  - `read_jpeg, read_pict, read_png`

Saving & Reading Images



```
IDL> x=findgen(2000)/999.*!pi
IDL> plot,x,sin(x)
IDL> img=tvrd()
IDL> help,img
IMG             BYTE      = Array[640, 501]
IDL> write_jpeg,'test.jpg',img
% Loaded DLM: JPEG.
```

File I/O in python

- Brief summary:
  - `file=open(‘filename.txt’,’r’)`
  - `stuff = file.readlines()`
  - `file.close()`
- OR:

  -
    ```
    with open(‘file.txt’,’r’) as file:
            stuff = file.readlines()
    ```

Convenience functions / libraries

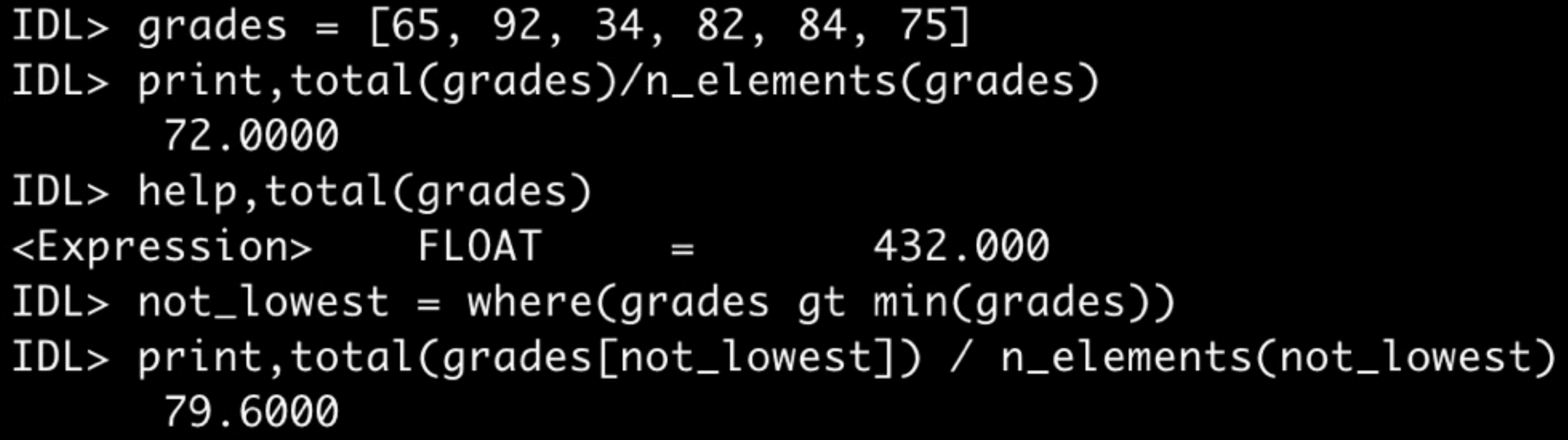
- Python (really, numpy) has nicer built-in file reading than IDL
    - `data = np.loadtxt(‘file.txt’)`
- If you need to read an IDL save, FITS, HDF, NCDF or other format, though, best to install other packages

The `where` function cont’d

- We’ll cover some common uses & pitfalls

Use Examples

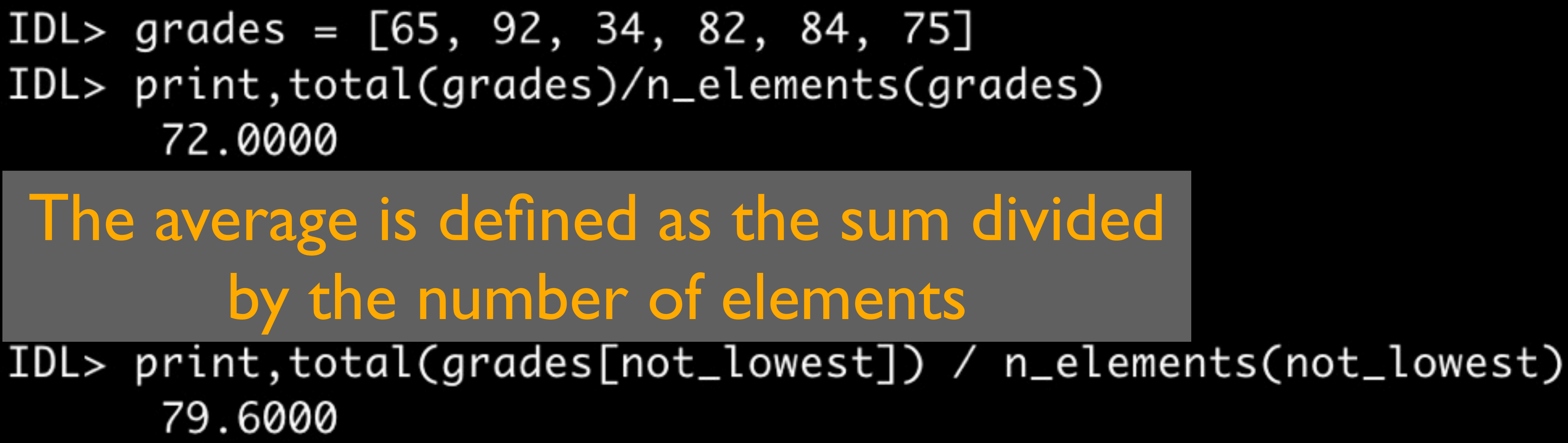
- Example 1: You have a list of grades, and want to compute the average after dropping the lowest of them



```
IDL> grades = [65, 92, 34, 82, 84, 75]
IDL> print,total(grades)/n_elements(grades)
      72.0000
IDL> help,total(grades)
<Expression>    FLOAT     =       432.000
IDL> not_lowest = where(grades gt min(grades))
IDL> print,total(grades[not_lowest]) / n_elements(not_lowest)
      79.6000
```

Use Examples

- Example 1: You have a list of grades, and want to compute the average after dropping the lowest of them



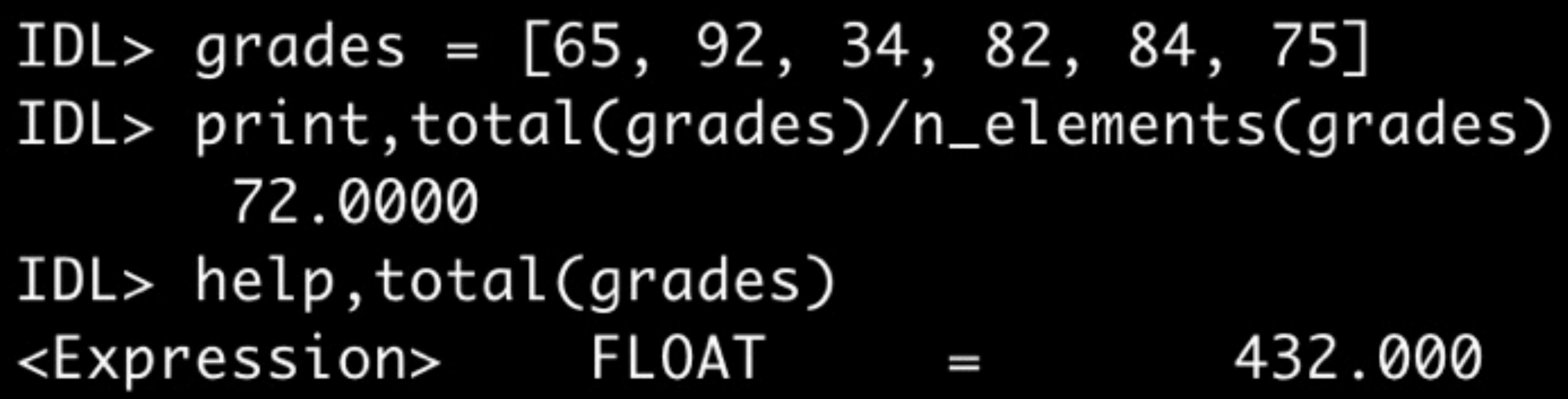
```
IDL> grades = [65, 92, 34, 82, 84, 75]
IDL> print,total(grades)/n_elements(grades)
      72.0000
```

The average is defined as the sum divided by the number of elements

```
IDL> print,total(grades[not_lowest]) / n_elements(not_lowest)
      79.6000
```

Use Examples

- Example 1: You have a list of grades, and want to compute the average after dropping the lowest of them

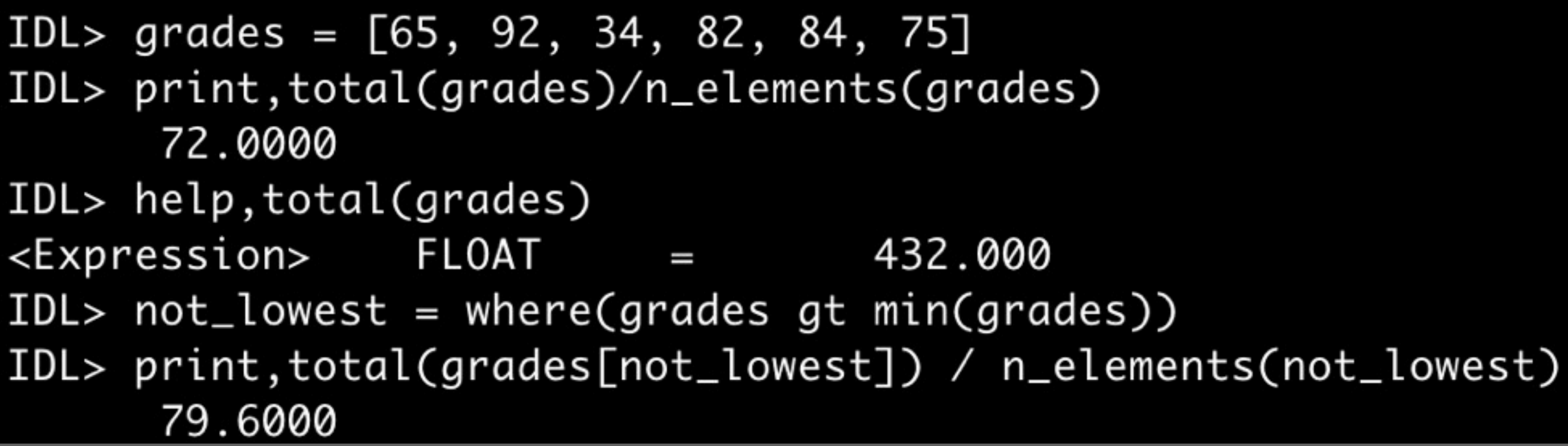


```
IDL> grades = [65, 92, 34, 82, 84, 75]
IDL> print,total(grades)/n_elements(grades)
      72.0000
IDL> help,total(grades)
<Expression>    FLOAT     =       432.000
```

`total` "casts" the integer array `grades` to a float array

Use Examples

- Example 1: You have a list of grades, and want to compute the average after dropping the lowest of them



```
IDL> grades = [65, 92, 34, 82, 84, 75]
IDL> print,total(grades)/n_elements(grades)
      72.0000
IDL> help,total(grades)
<Expression>    FLOAT     =       432.000
IDL> not_lowest = where(grades gt min(grades))
IDL> print,total(grades[not_lowest]) / n_elements(not_lowest)
      79.6000
```

Use some tricks to pick all but the least element.
`min(grades)` returns the *value* of the lowest element

Review: `where`

What is the return from `where([0,0,0,0])`?

A) 0

B) [1,2,3,4]

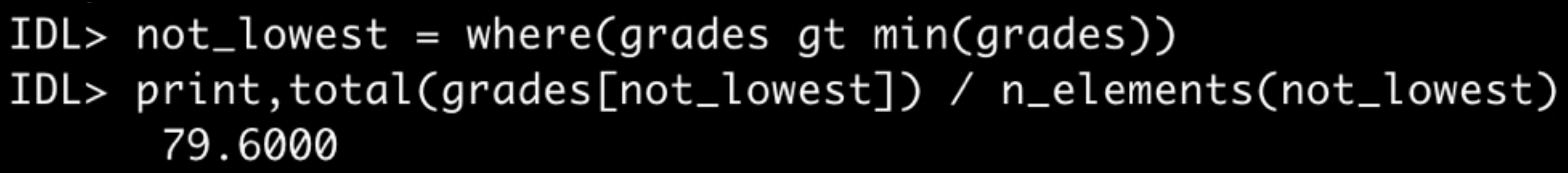
C) -1

D) 4

E) None of the above / I don’t know

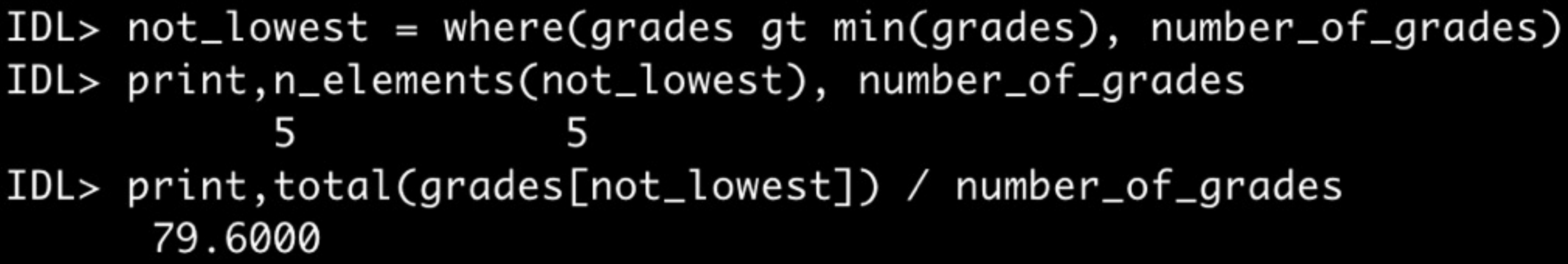
Use Examples

- Example 1: You have a list of grades, and want to compute the average after dropping the lowest of them



```
IDL> not_lowest = where(grades gt min(grades))
IDL> print,total(grades[not_lowest]) / n_elements(not_lowest)
      79.6000
```

There's a better way to do this!



```
IDL> not_lowest = where(grades gt min(grades), number_of_grades)
IDL> print,n_elements(not_lowest), number_of_grades
           5           5
IDL> print,total(grades[not_lowest]) / number_of_grades
      79.6000
```

`where(array, nmatch)`

- Can check to make sure `nmatch>0`
- Then there's no risk of indexing just the last element

Evaluate:

```
IDL> a = indgen(5)
IDL> print,a[where(a lt 0)]
```

A)
```
% Attempt to subscript A with <INT      (      -1)> is out
    of range.
```

B) 4

C) 0

D) -1

E) None of the above / I don’t know

Evaluate:

```
IDL> a = indgen(5)
IDL> b = where(a lt 0, nb)
IDL> c = where(a le 0, nc)
```

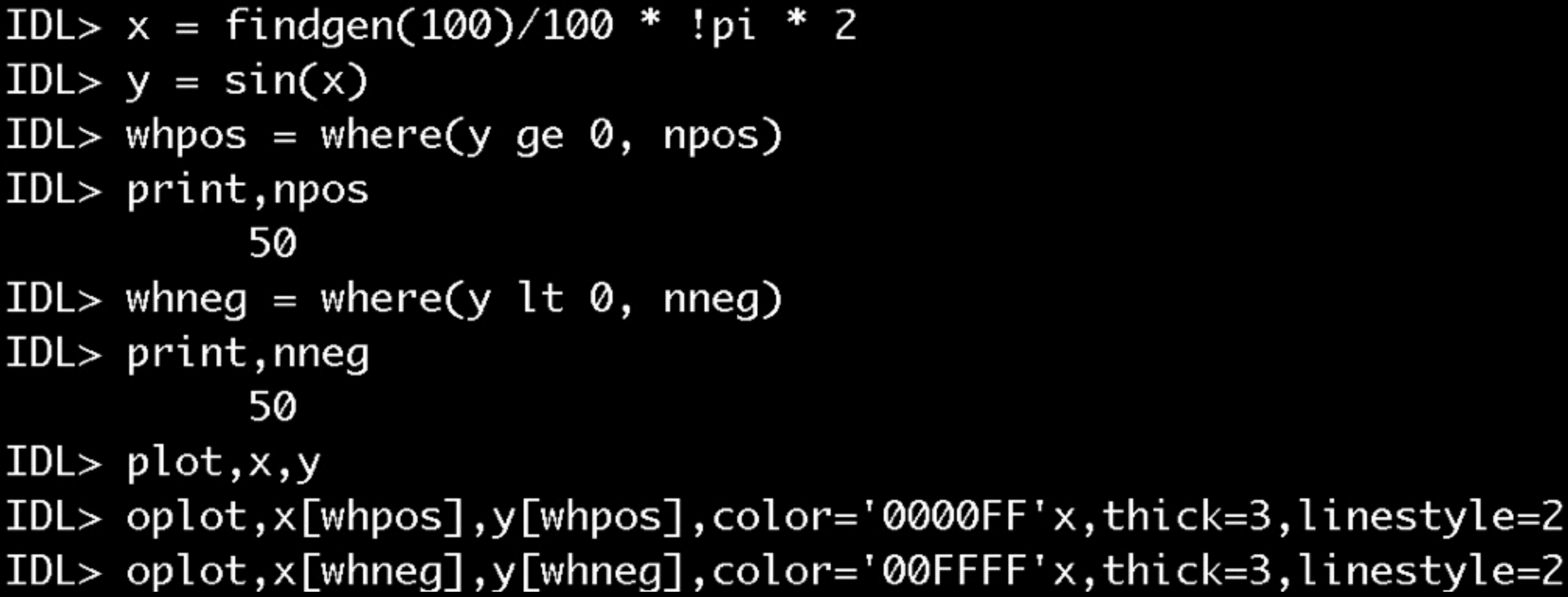
What are the values of `b`, `nb`, `c`, and `nc`?

	b	nb	c	nc
A)	0	1	0	0
B)	0	0	-1	1
C)	-1	1	-1	0
D)	-1	0	0	1

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

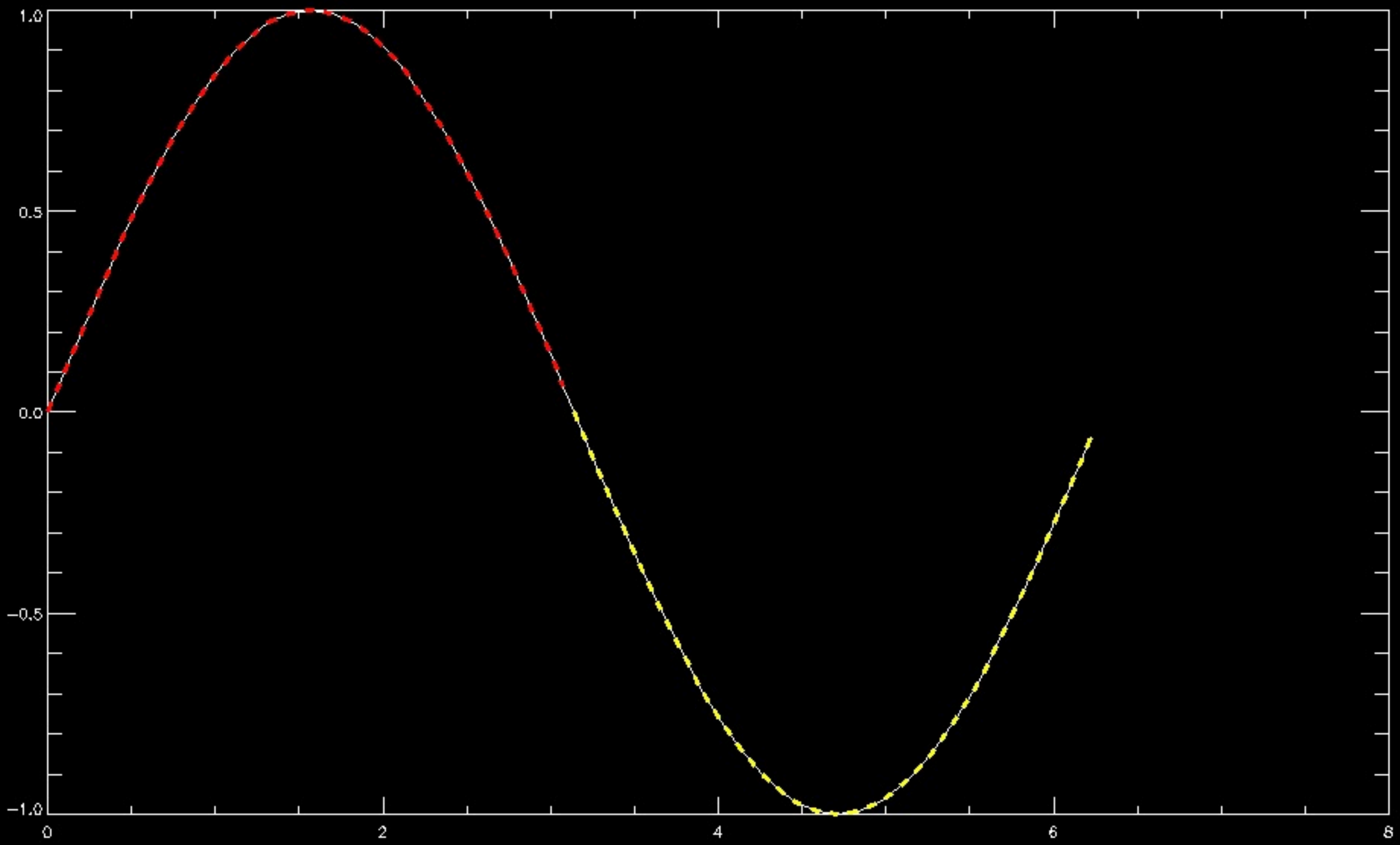
Example 2

- Plot only the positive part of the sine curve, then only the negative



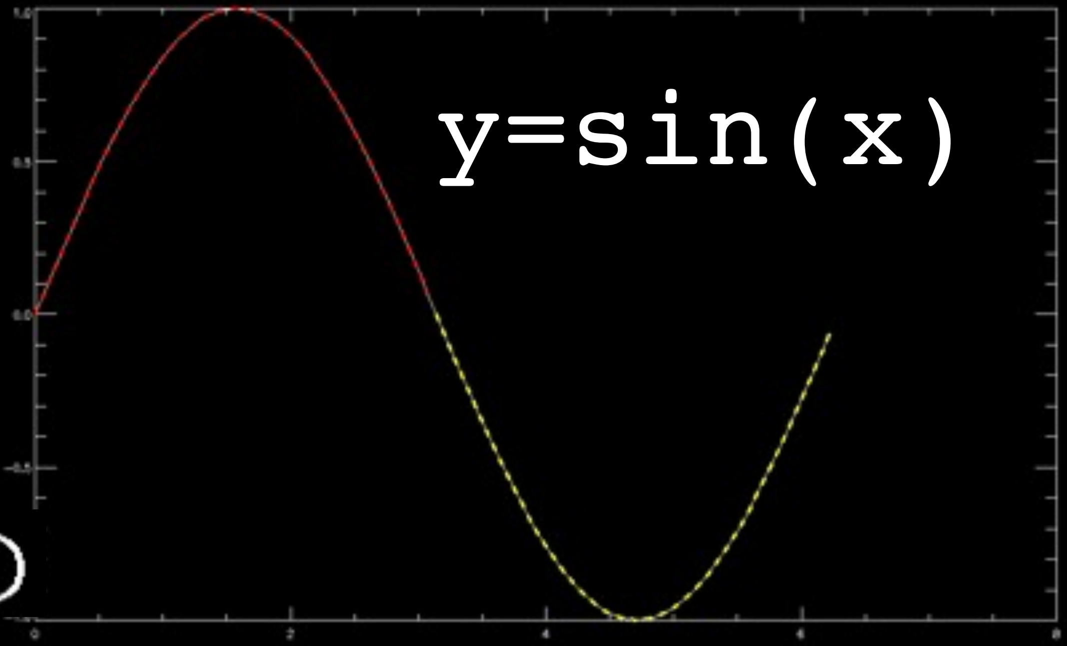
```
IDL> x = findgen(100)/100 * !pi * 2
IDL> y = sin(x)
IDL> whpos = where(y ge 0, npos)
IDL> print,npos
          50
IDL> whneg = where(y lt 0, nneg)
IDL> print,nneg
          50
IDL> plot,x,y
IDL> oplot,x[whpos],y[whpos],color='0000FF'x,thick=3,linestyle=2
IDL> oplot,x[whneg],y[whneg],color='00FFFF'x,thick=3,linestyle=2
```

Example 2



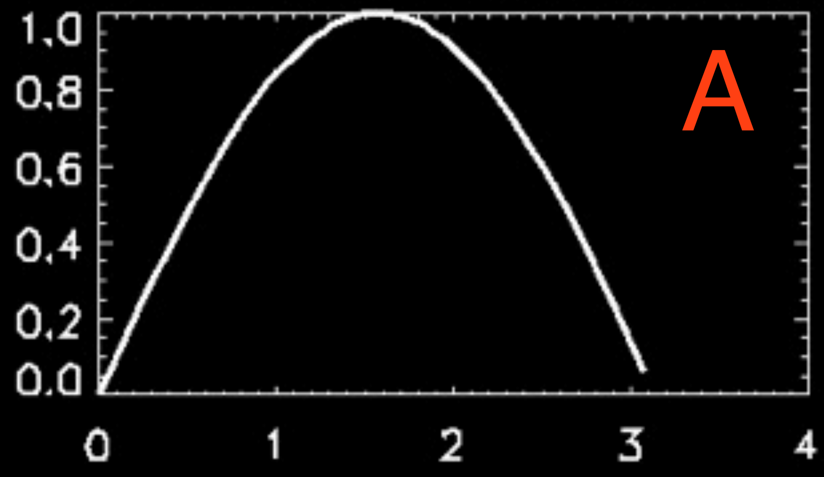
Array Truncation

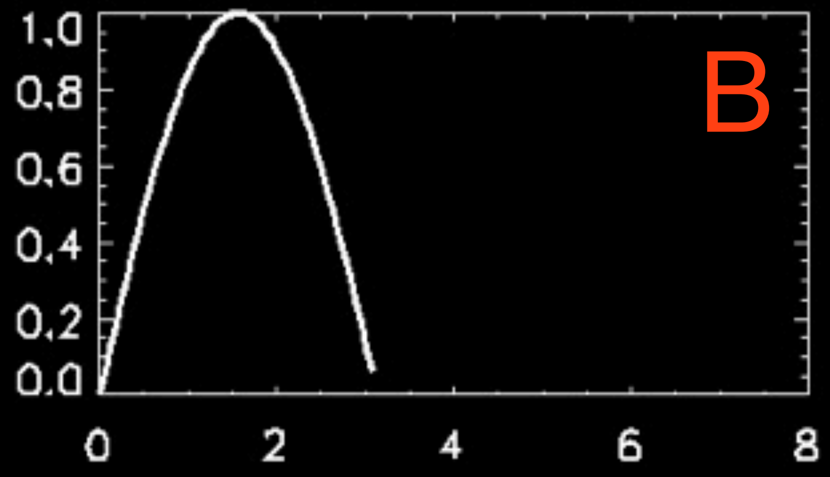
- What happens if you try to plot arrays of different length?
- x = [0,1,2,3,4,5]
- y = [0,1,4,9,16,25,49,64,81,100]
- It just truncates the longer array: it will plot,[0,1,2,3,4,5],[0,1,4,9,16,25]



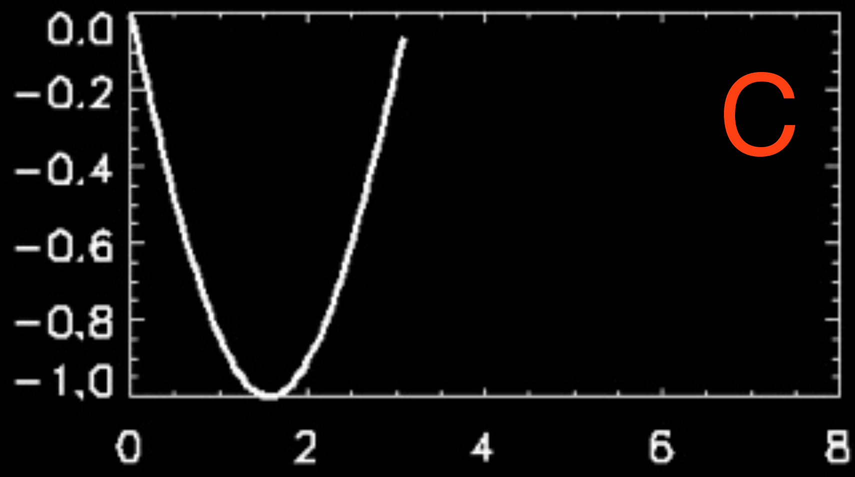
```
IDL> whneg = where(y lt 0, nneg)
```

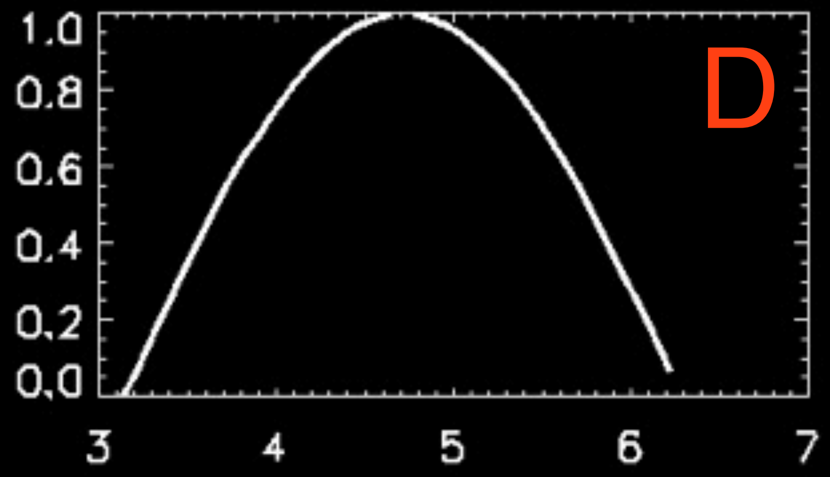
What would we see if we did: `plot,x[whneg],y`





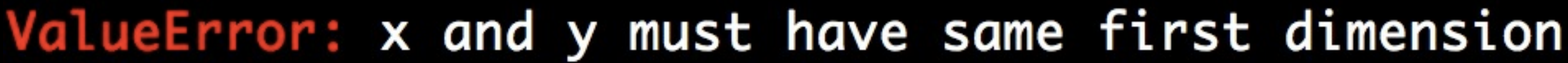
E: Something Else





IDL v Python on zipping

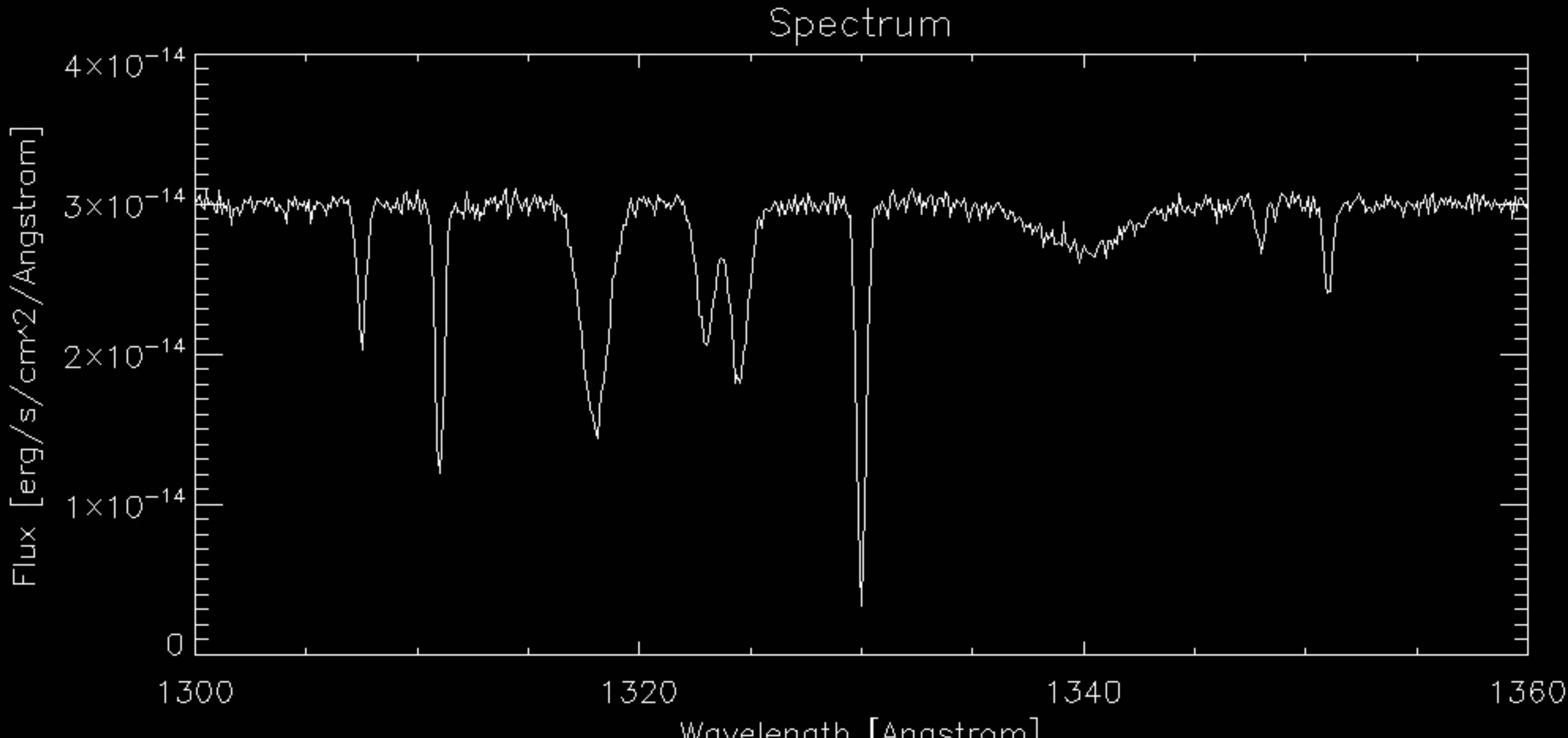
- plot([0,1,2,3],[0,1,4,9,16]) will not work in python!



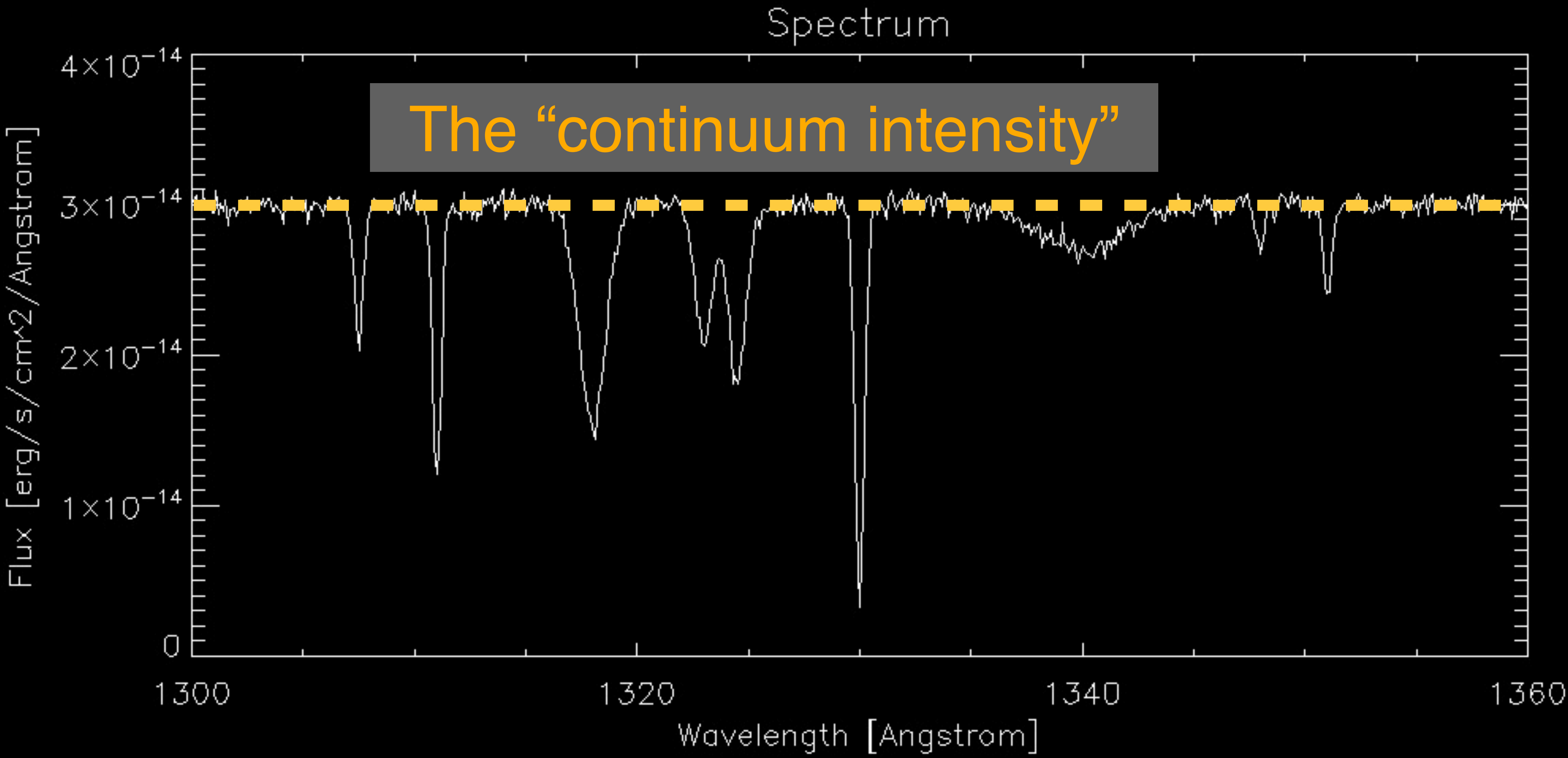
```
ValueError: x and y must have same first dimension
```

- This is nice because it tells you when you've screwed up

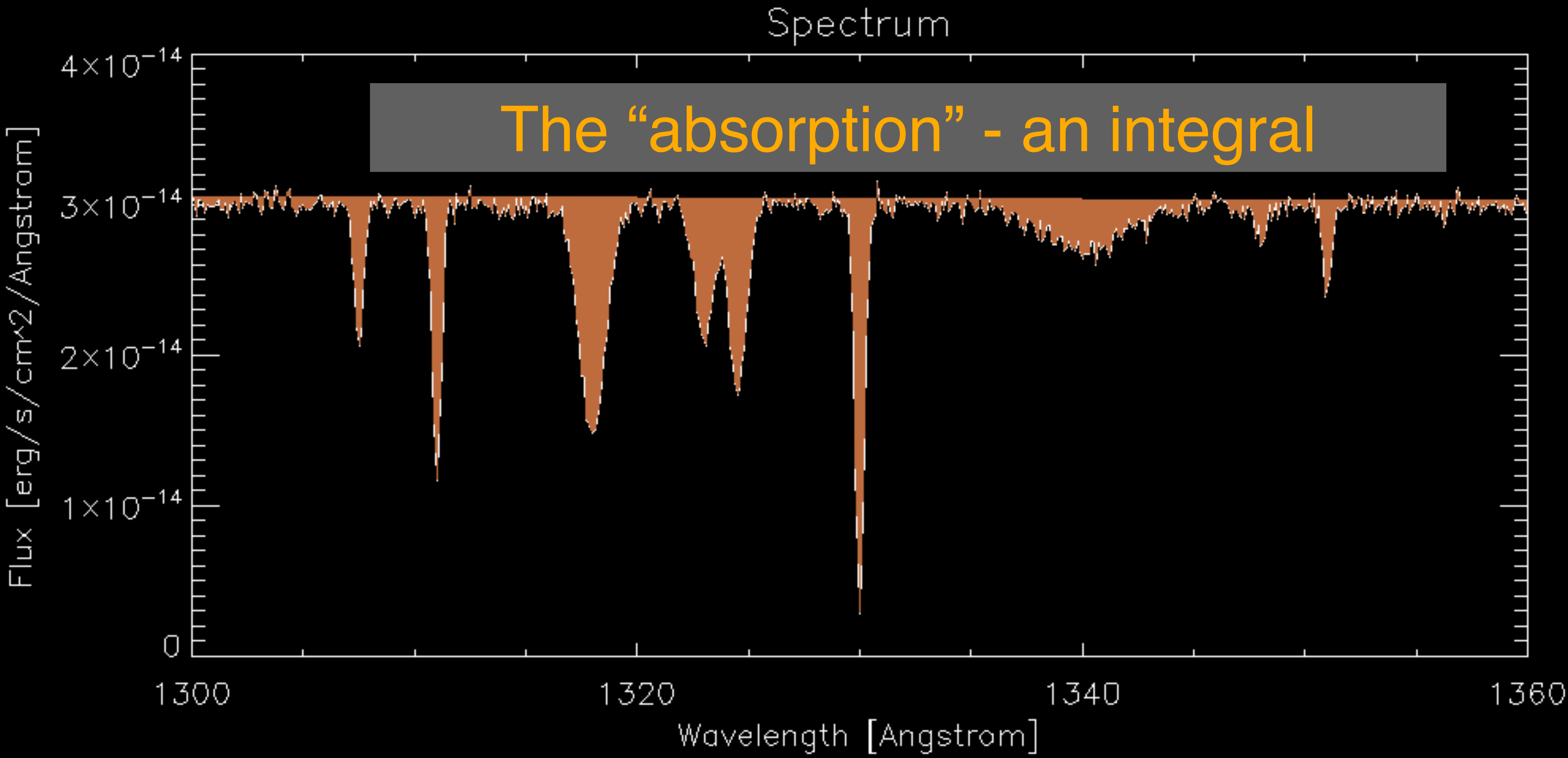
Example 3: Astronomical Spectroscopy



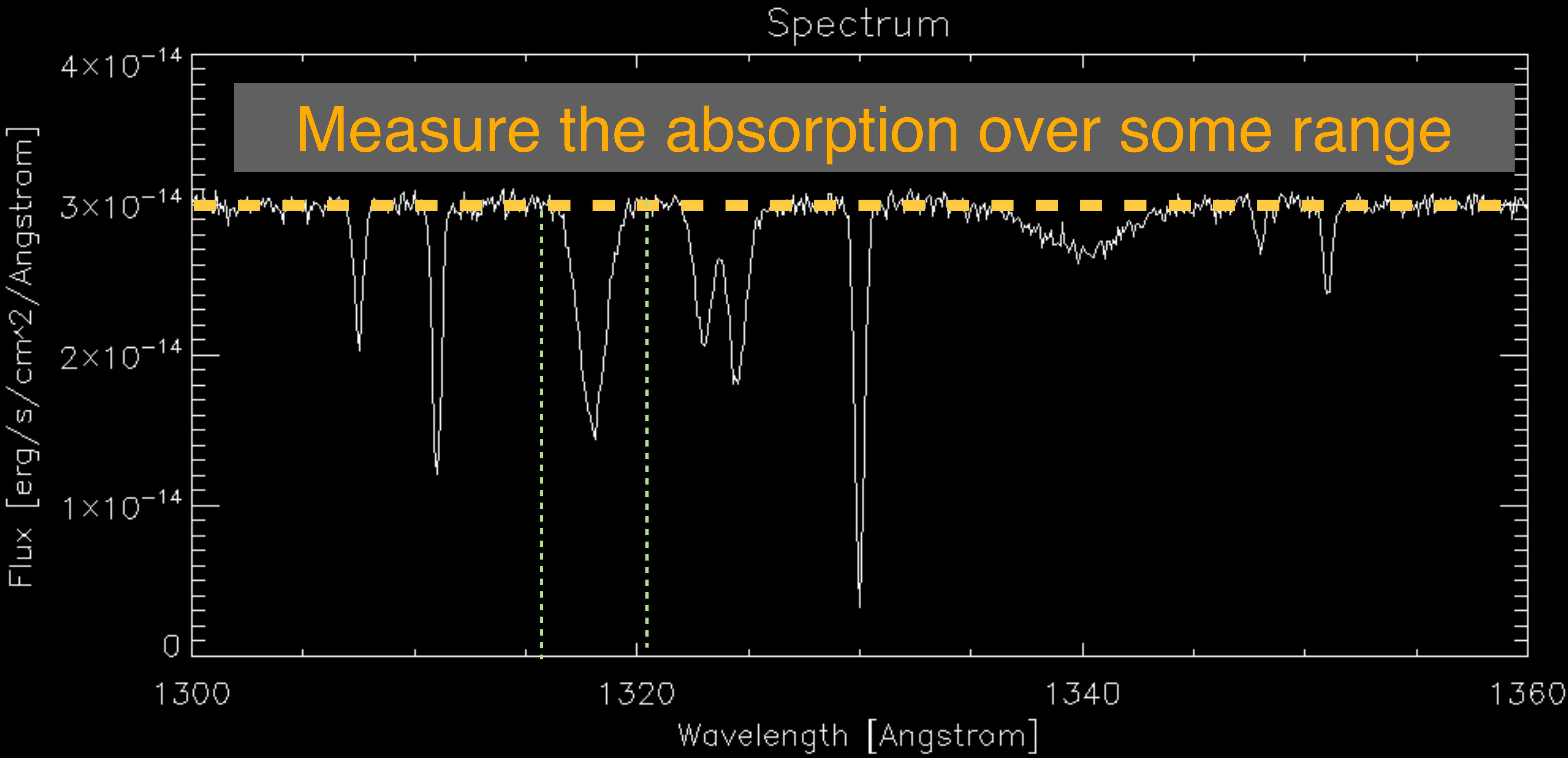
Example 3: Astronomical Spectroscopy



Example 3: Astronomical Spectroscopy



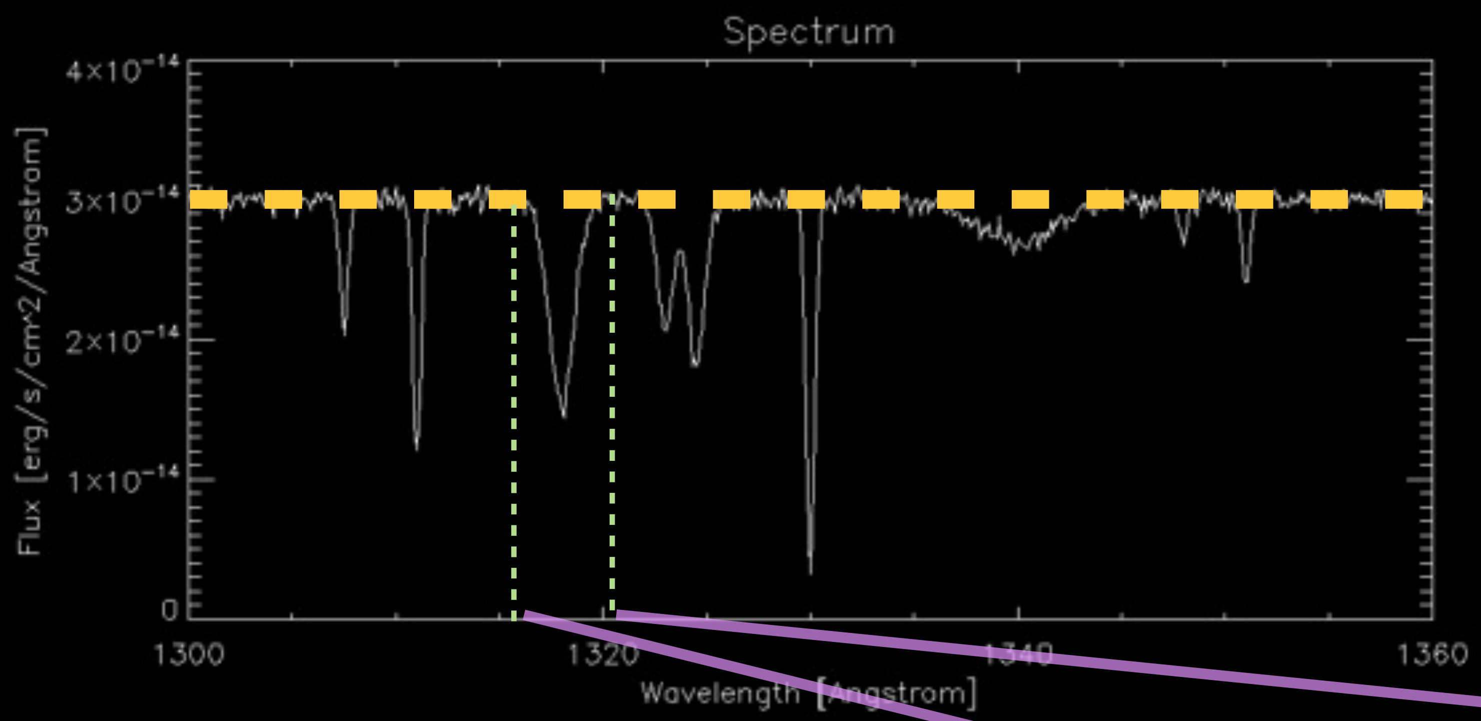
Example 3: Astronomical Spectroscopy



Example 3

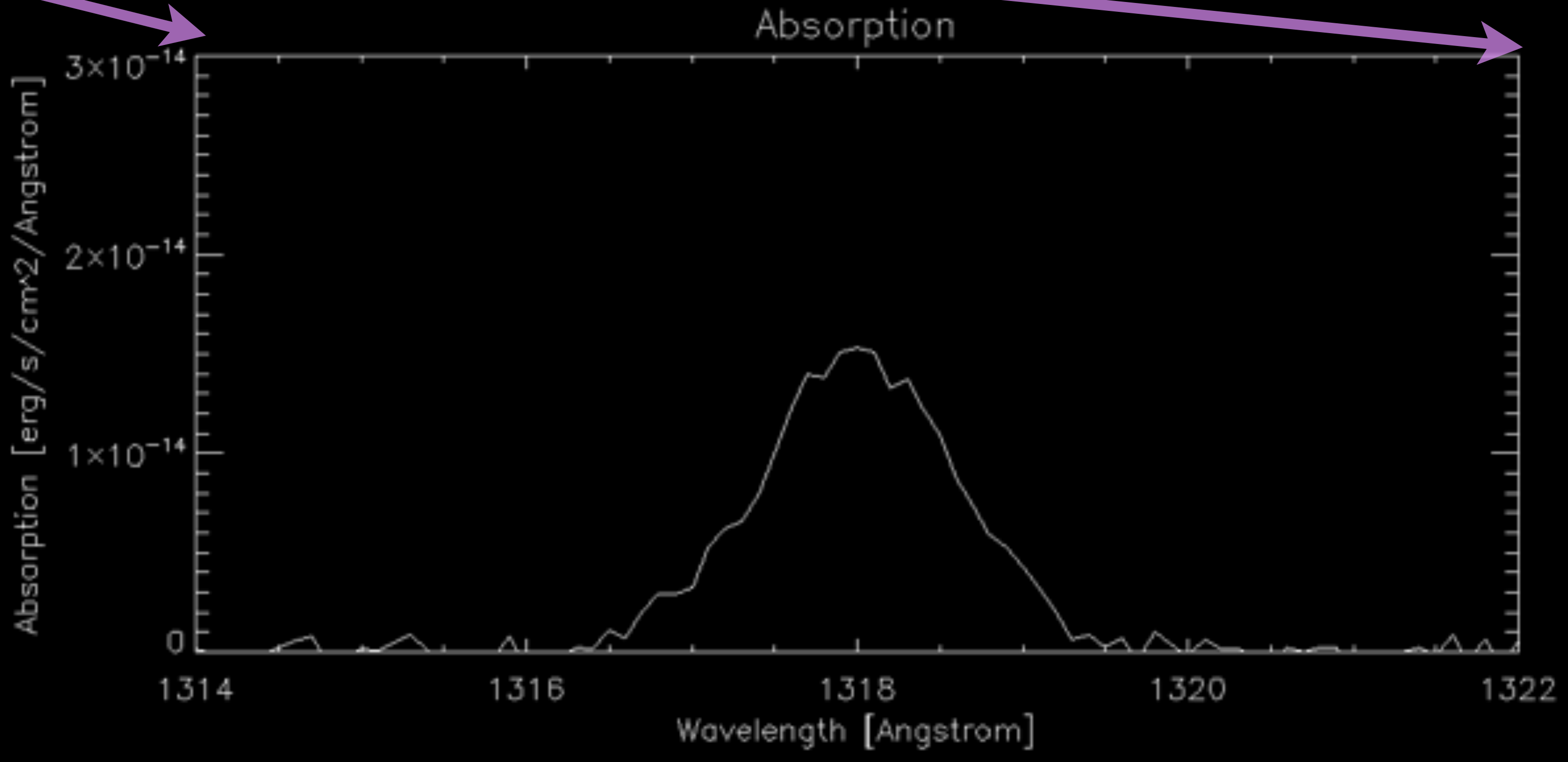
- `flux = [some array]`
- `lambda = [some array]`
- `continuum = 3e-14` (scalar constant)
- `absorption = continuum - flux`

Example 3: Zoomed

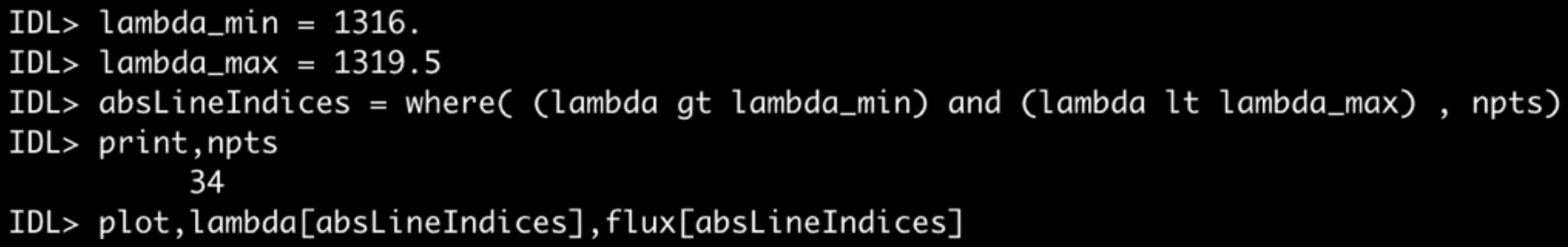


Absorption is how much star light is removed by a particular atom (so it's positive)

This is our goal:



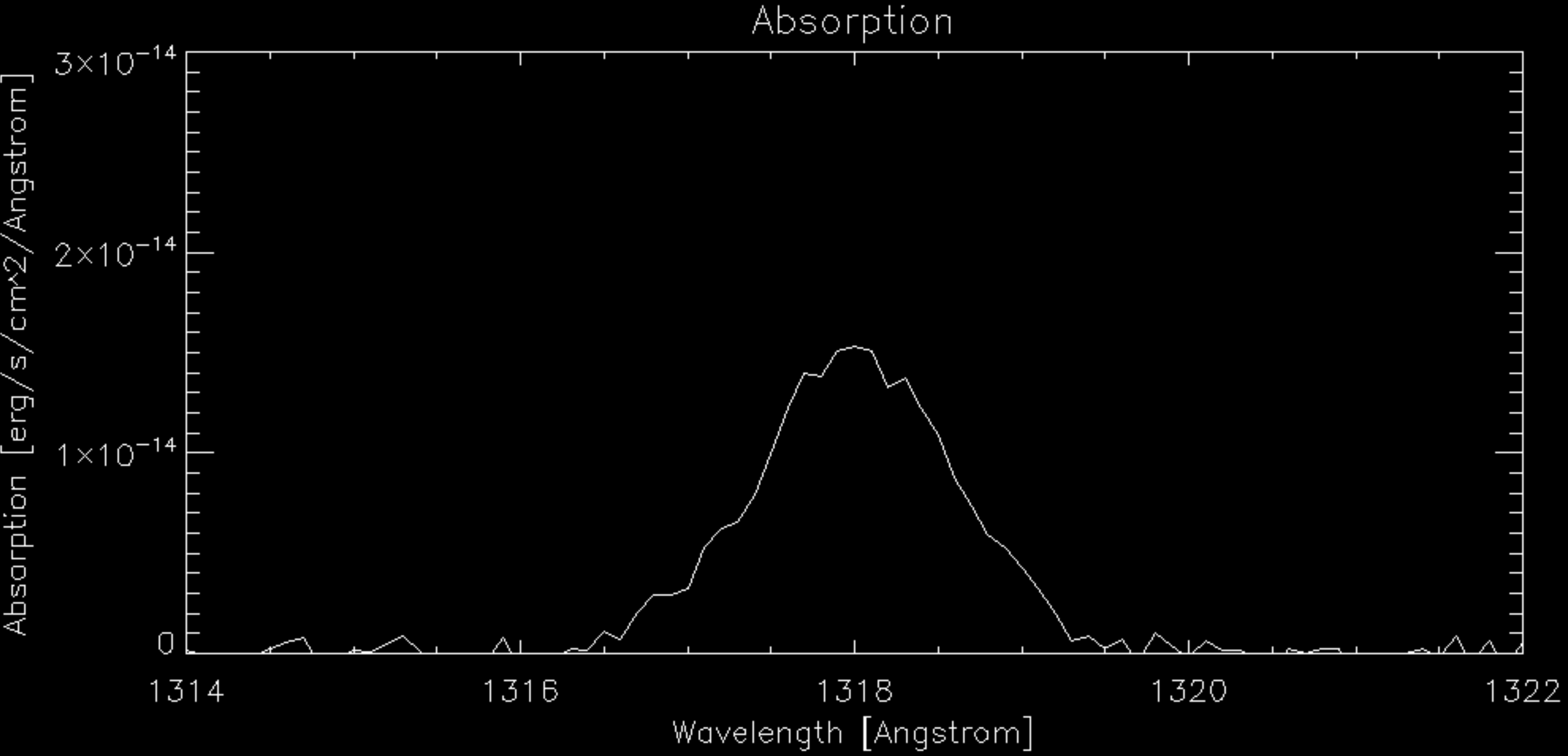
Example 3: Code



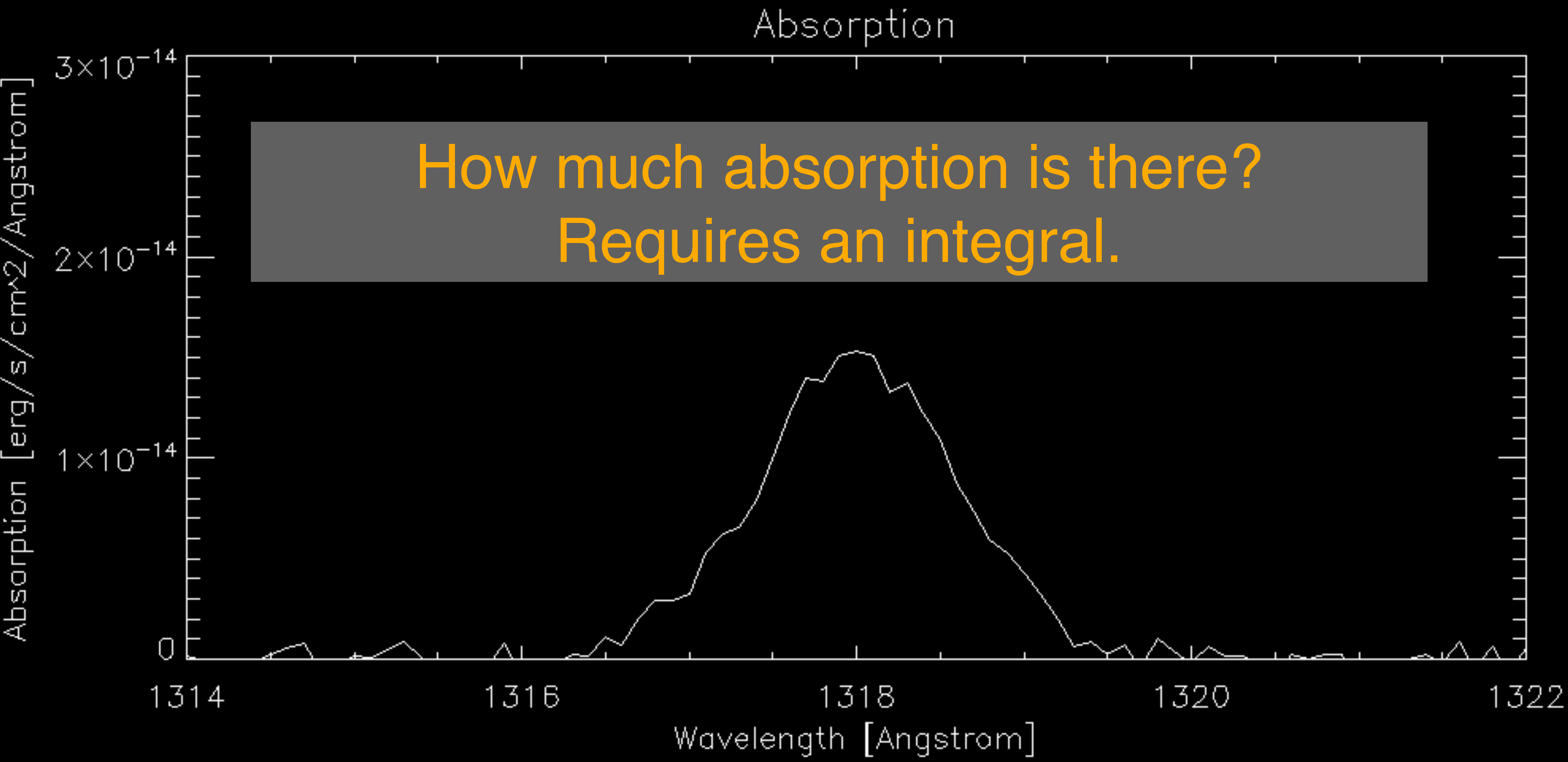
```
IDL> lambda_min = 1316.
IDL> lambda_max = 1319.5
IDL> absLineIndices = where( (lambda gt lambda_min) and (lambda lt lambda_max) , npts)
IDL> print,npts
          34
IDL> plot,lambda[absLineIndices],flux[absLineIndices]
```

- `npts` to check to make sure there's a match
- index BOTH x and y arrays

Example 3

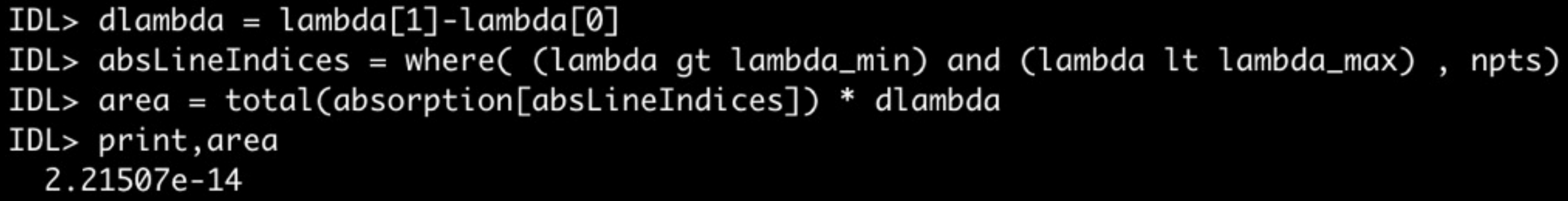

Absorption
3×10^{-14}
2×10^{-14}
1×10^{-14}
0
Absorption [erg/s/cm^2/Angstrom]
1314
1316
1318
1320
1322
Wavelength [Angstrom]

Example 3



How much absorption is there?
Requires an integral.

Example 3

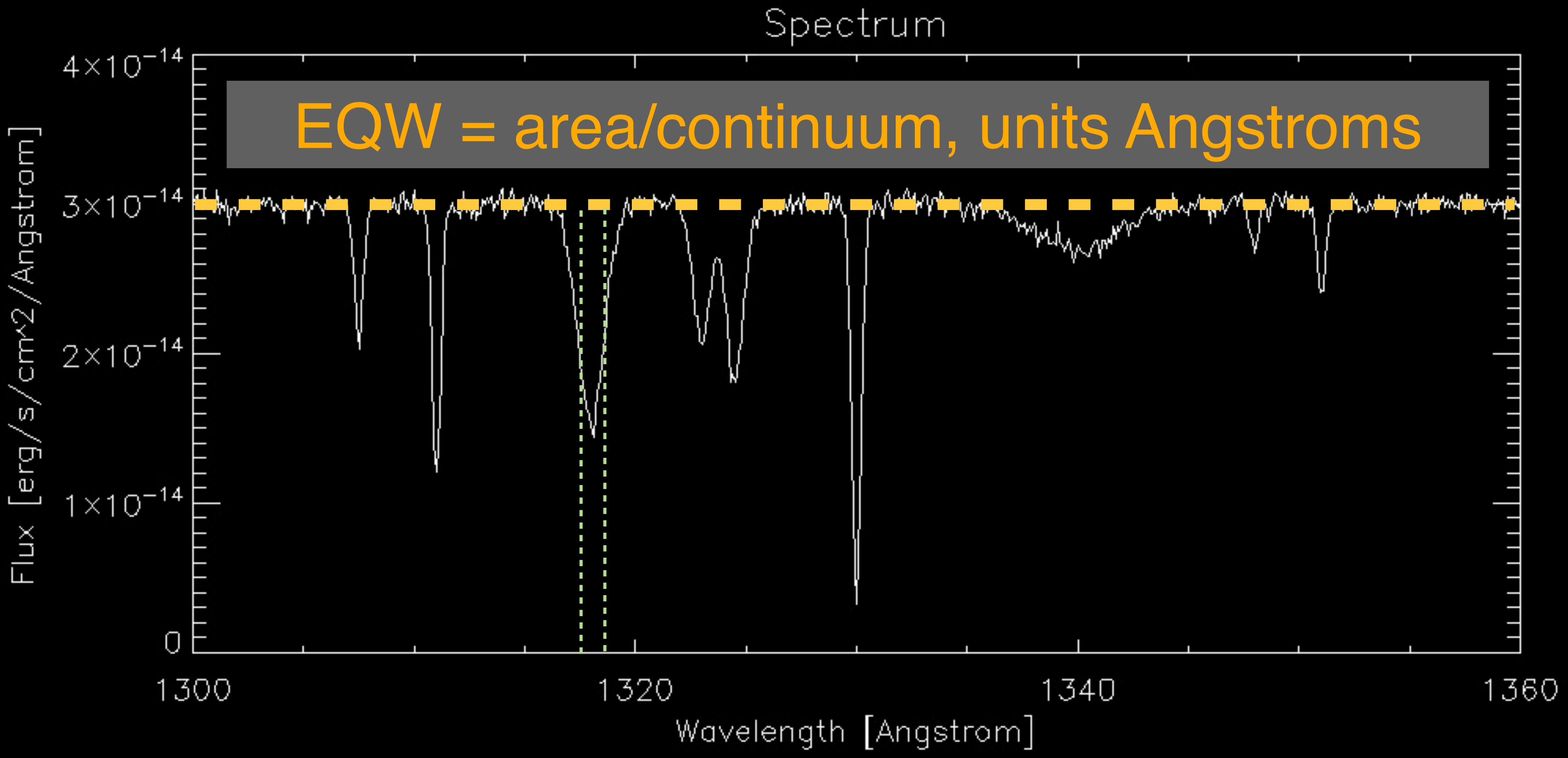


```
IDL> dlambda = lambda[1]-lambda[0]
IDL> absLineIndices = where( (lambda gt lambda_min) and (lambda lt lambda_max) , npts)
IDL> area = total(absorption[absLineIndices]) * dlambda
IDL> print,area
  2.21507e-14
```

- Integral $= \int f(x)dx \approx \sum f(x_i)\Delta x$

Example 3: EqW

Sometimes astronomers like to measure the “Equivalent Width” of a line



EQW = area/continuum, units Angstroms