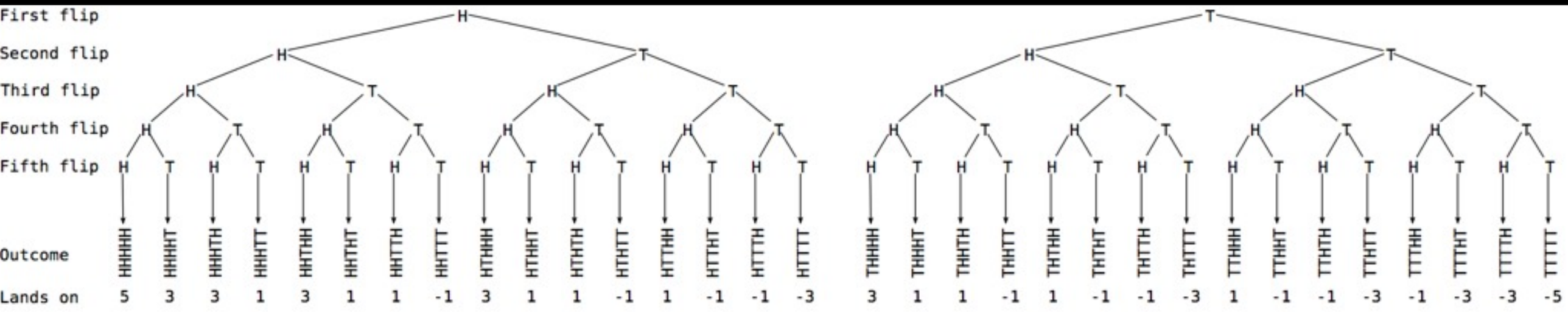
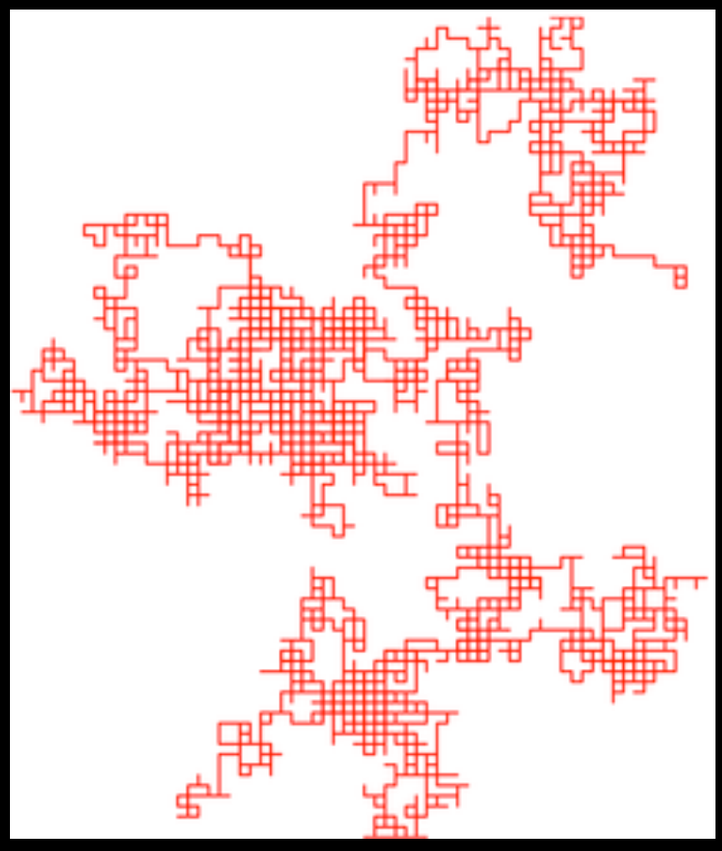
Case Study: Random Walk

- A “random walk” is taking sequential coin flips to decide which direction to move



Random Walk

- Take N steps of some size, where each step is random
- This results in brownian motion (motion of, e.g., air molecules)



Bottom-up

- Start with the simplest component:
  - Take one *random* step
- If motion is restricted to be in one dimension (easiest case), that means there must be equal likelihood to go *forward* and *backward*

What does this do?

```
step = randomu(seed,1) gt 0.5 ? 1 : -1
```

A) Makes `step` a random variable greater than 0.5

B) Makes `step` +1 50% of the time and -1 50% of the time

C) Makes `step` an array of length `seed` with values +1 and -1 randomly distributed

D) Crashes

E) None of the above

2D steps

- On a 2 dimensional grid, can take a step in any direction
  - step size is always the same
  - all directions are equally likely

Which is best for determining a random direction?

A) `randomn(seed) * 360`

B) `randomu(seed) * 360`

C) Neither

```
; random step
; Take a step in a random direction

step_direction = randomu(seed) * !pi * 2
step_x = cos(step_direction)
step_y = sin(step_direction)
```

Should random step be...

A) A procedure

B) A function

C) A program

D) A script

E) None of the above

Random Step pro

```
; random step
; Take a step in a random direction
; INPUTS:
;   seed : random seed
; OUTPUTS:
;   step_x, step_y: step length in x, y direction

pro random_step,seed,step_x,step_y
    step_angle = randomu(seed) * !pi * 2
    step_x = cos(step_angle)
    step_y = sin(step_angle)
end ; random_step
```

Random step...

- It works, I think....
- How do I test it?
- What aspects could we test?

```
IDL> random_step,seed,dx,dy
IDL> print,dx,dy
     -0.423450     -0.905919
IDL> random_step,seed,dx,dy
IDL> print,dx,dy
      0.867316     -0.497759
IDL> random_step,seed,dx,dy
IDL> print,dx,dy
      0.868664     -0.495402
IDL> random_step,seed,dx,dy
IDL> print,dx,dy
      0.105232     -0.994448
```

Which is a useful test of `random_step`?
(you just called `random_step,seed,dx,dy`
so `dx` and `dy` should be random variables with some other properties....)

```
A)print,abs((dx^2+dy^2)^0.5 - 1.0) lt 1e-7

B)print,(dx^2+dy^2)^0.5 eq 1.0

C)print,abs(dx + dy) - 1.0 lt 1e-7

D)print,dx^2 lt dy^2
```

E) None of these

```
In [35]: r = np.random.random(10)
In [36]: angle = r*2*pi
In [37]: x,y=cos(angle),sin(angle)
In [38]: x**2+y**2
Out[38]: array([ 1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.])
In [39]: np.set_printoptions(precision=24)
In [40]: x**2+y**2
Out[40]:
array([ 1.                      ,  1.                      ,
        1.                      ,  1.                      ,
        1.                      ,  1.                      ,
        1.0000000000000000222044605,  1.                      ,
        1.                      ,  1.                      ])
In [41]: np.arcsin(x**2+y**2)
Out[41]:
array([ 1.570796326794896557998982,  1.570796326794896557998982,
        1.570796326794896557998982,  1.570796326794896557998982,
        1.570796326794896557998982,  1.570796326794896557998982,
                               nan,  1.570796326794896557998982,
        1.570796326794896557998982,  1.570796326794896557998982])
```

RuntimeWarning: invalid value encountered in arcsin

Next Step ...insteption...

- We made a random step, but we want to random walk
- How many steps should we take?
  - Leave that decision to the user
- How should we take those steps?

Random Walk

You have a `random_step` procedure. What should you use to make a `random_walk` procedure now?
(how many steps are we taking/when do we stop walking?)

A) `while` loop

B) `repeat ... until` loop

C) `foreach` loop

D) `for` loop

E) 

Random Walk

```
; random walk
; Take N steps in independent random directions
; Start somewhere, report where you end up
; INPUTS:
;   xpos,ypos : Starting X,Y
;   nsteps : number of steps
;   seed : starting random seed
;   stepsize : size of steps (defaults to 1)
pro random_walk,xpos,ypos,nsteps

    for stepcount=0,nsteps-1 do begin
        random_step,seed,dx,dy
        xpos += dx
        ypos += dy
    endfor ; steps
end ; random_walk
```

Other improvements?

- What are we really interested in?
  - Each step? The total distance? The path taken? Maybe all of these!
- Need more features!
  - referred to as a “feature request” on the intertubes

Expanding Random Walk

- Make `xpos,ypos` arrays instead of scalars
  - can keep track of each step
- Plot the random walk!
  - Obviously. It will look cool.
- Determine the total distance traveled (at each step)

Building Up Code

- Tutorial today will be about adding features and code development
- You should have “Diagnostic Code” (i.e. `print` statements) dispersed throughout
- If you don’t want it to print, “comment it out” (start the line with a `;`)

Chapter 14

- Use Chapter 14 as a reference
- Dewey goes through the whole development process - many of the same ideas we went over in lecture, but with concrete examples throughout
- There are LOTS OF MISTAKES in programming, and Ch 14 shows some

Ch 14 vs Lecture

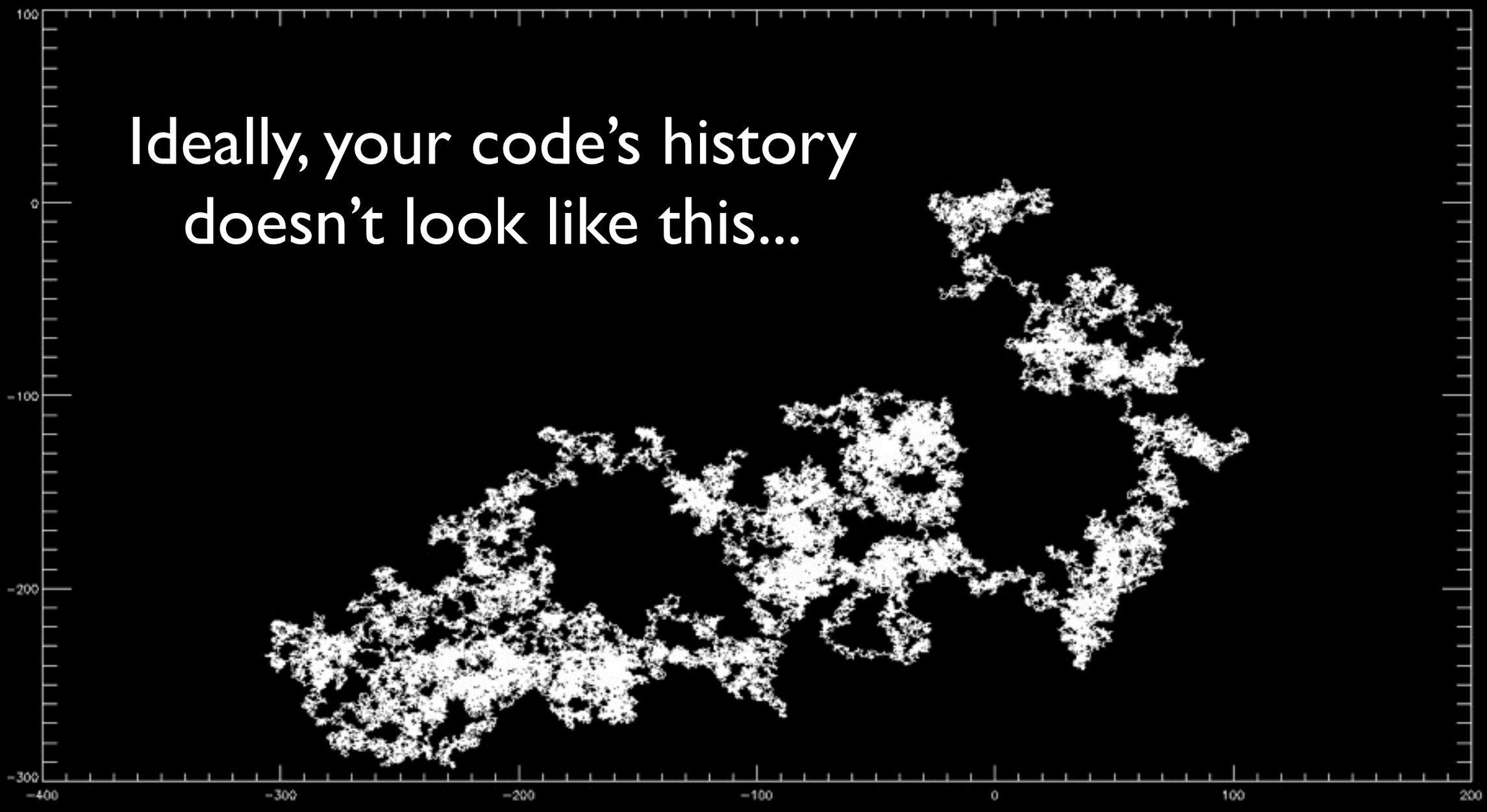
- Chapter 14 implements the same general idea, but with different approaches
  - I coded things in a different order
  - I chose different variable manipulations
- You can choose either way!

“Refactoring”

- If you want to change the code you’re handed to look more like Dewey’s, go ahead
- When you change code so that the code looks different but it does the same thing, that’s called “refactoring”
  - usually, you do it to “clean up” messy code or remove duplicate code

Brownian Motion Again

Ideally, your code's history doesn't look like this...



Tutorial 15

- For tutorial 15, we will do Chapter 14
  - With a twist: You will do a "`git commit`" every 10 minutes
    - In this case, it doesn't matter if your code works at any given step, just keep committing (which means, SAVING too!)
      - Good practice: Save often
      - Will show your "work flow"

Random Walk Goal

- How does the distance from the starting point change *on average*?
- The book shows you how to do this: Average the distances of many random walks

Code Golf

- Bad technique, but useful thought exercise:
- Officially, “Code Golf” is trying to accomplish a task using the fewest characters possible.
  - It’s actually kind of dumb.
- BUT, it can be useful to try to accomplish a task in the fewest *commands* possible

“Good” code golf

- Terse code is often easier to parse than verbose code
- “Brevity is the soul of wit.”
- But really, we’re interested in a different brevity - *faster* code is better most of the time
  - But, it is NEVER worth sacrificing “correctness” for speed

Code Optimization

- As a rule, you don't optimize code unless you have to (i.e., unless your code is slow)
- But, I brought up optimization and code golf because the functions in Chapter 14 can be accomplished in probably fewer than 10 lines of code. Kudos* if you can figure out how!

*I am not actually offering food, just props... for now

SURVEY: How long did Assignment 5 exercises & WDIDs take?

A) <~ 1 hour

B) ~ 2 hours

C) ~ 3 hours

D) > 3 hours

E) I didn’t do the exercises & WDIDs / can’t remember

SURVEY: How long did Assignment 5 homework take?

A) <~ 1 hour

B) ~ 2 hours

C) ~ 3 hours

D) > 3 hours

E) I didn’t do / haven’t finished the homework (but if you know how long it will take you, answer one of the others)

SURVEY: How long did Assignment 6 exercises & WDIDs take?

A) <~ 1 hour

B) ~ 2 hours

C) ~ 3 hours

D) > 3 hours

E) I didn’t do the exercises & WDIDs

Which of these is a correct program definition?

```
A)pro mypro(argument,keyword=keyword)

B)pro mypro | argument | keyword=keyword

C)pro mypro, argument, keyword=keyword

D)pro mypro, argument | keyword=keyword
```

E) None of the above

Which of these statements makes a ‘default’ keyword?

```
; parrot
; says what you say if you say something
; otherwise, says "squawk"
pro parrot,what_I_say=what_I_say
    if X then begin
        parrot_says = what_I_say
    endif else begin
        parrot_says = "squawk"
    endelse
    print,parrot_says
end
```

(what do you replace X with?)

A) X = keyword_set(what_I_say)
B) X = ~keyword_set(what_I_say)
C) X = n_elements(what_I_say)
D) X = n_elements(what_I_say) gt 0
E) None of the above

```
x = [1,2,3]
y = [4,5,6]

x = x[0] + y
```

How many elements will `x` have?

A) 0

B) 1

C) 2

D) 3

E) None of the above

```
x = [1,2,3]
y = [4,5,6]

x = x[0:0] + y
help,x
```

How many elements will `x` have?

A) 0

B) 1

C) 2

D) 3

E) None of the above

```
sqvar = 2L
for ii=0,4 do begin
    sqvar = sqvar * sqvar
endfor
```

What will `sqvar` be?

A) 2^3

B) 2^4

C) 2^2^2^2^2^2

D) One of these, but I'm not sure which

E) None of these, and I'm sure

```
sqvar = 2L
for ii=0,4 do begin
    sqvar = sqvar * sqvar
    ;1 I want to track sqvar...
endfor
;2 Should I do it here?
```

How could you “watch” what sqvar does?
(i.e., find out what’s going on in this code?)

A) Use a `plot` statement at line 1

B) Use a `plot` statement at line 2

C) Use a `print` statement at line 1

D) Use a `print` statement at line 2

E) None of the above

```
sqvar = 2L
for ii=0,4 do begin
    sqvar = sqvar * sqvar
    ;1 I want to track sqvar...
endfor
;2 Should I do it here?
```

What do you need to do to make use of a `plot` statement at line 2?

A) Fill an array with the values of `sqvar`

B) Make `sqvar += sqvar*sqvar` instead

C) Nothing, it's fine as it is

D) Come up with a set of X values to complement the Y values

E) None of the above

Where should these lines be placed?

```
choice = ""
read,"Do you want to print 1-5?",choice
```

```
; ask the user if they want to print 1-5
pro loop_print
    ; A
    if choice eq "yes" or choice eq "y" then begin
        ; B
        for ii=1,5 do begin
            ; C
            print,ii
            ; D
        endfor
    endif
    ; E
end
```

Which of these lines contains an error?

```
; fill an array with random elements
x = fltarr(5)                          ; A
for ii=0,n_elements(x)-1 do begin      ; B
    x[ii] = randomu(seed)              ; C
endfor                                 ; D
```

E) None of the above

Which of these lines contains an error?

```
; add a random number to each element
; of an array

x = fltarr(5)                          ; A
for ii=0,n_elements(x)-1 do begin      ; B
    x[ii] = x + randomu(seed)          ; C
endfor                                 ; D
```

E) None of the above

```
; add a random number to each element
; of an array
x = fltarr(5)                              ; A
for ii=0,n_elements(x)-1 do begin          ; B
    x[ii] = x + randomu(seed)              ; C
endfor                                     ; D
```

What is the error?

A) the x on the left side should not be indexed; it should be an array

B) the x on the right side should be indexed with `ii`

C) `randomu` should have another argument

D) should use `randomn` instead of `randomu`

E) none of the above

What could this error message mean?

```
% Attempt to call undefined procedure/function:
```

A) You tried to call a function as a procedure

B) You tried to compile a program but there was a syntax error

C) You tried to define a procedure outside after the end of the program

D) You didn't define a variable in the local namespace

E) None of the above

What could this error message mean?

```
% Variable is undefined: A.
```

A) You tried to call a function as a procedure

B) You tried to compile a program but there was a syntax error

C) You tried to define a procedure outside after the end of the program

D) You didn’t define a variable in the local namespace

E) None of the above

What could this error message mean?

```
% Procedure header must appear first and only once:
```

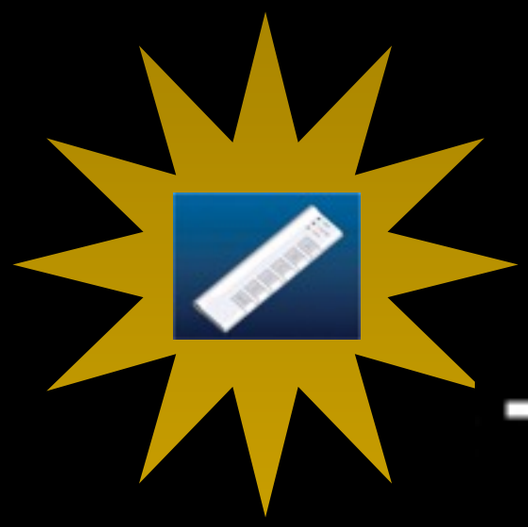
A) You tried to call a function as a procedure

B) You tried to compile a program but there was a syntax error

C) You tried to define a procedure outside after the end of the program

D) You didn't define a variable in the local namespace

E) None of the above



What's wrong with this statement?

```
IDL> x=print(5)
```

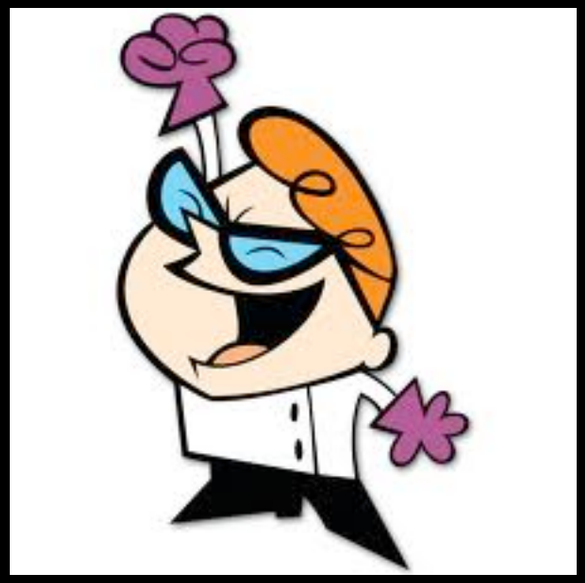
A) Didn't declare `x` first

B) `print` is a procedure, not a function

C) the `print` function only takes strings

D) should be `eq` instead of =

E) It's WAY TOO BIG



To the laboratory!

Time to work on Tutorial 14 &
15
Use the book for Tutorial 15!