LISTS

Outline

- List Basics
- Creating and Accessing Lists
- List Details
- Length of a List

Zombie Apocalypse

```
Level: 0
. . ! . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . * . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . #
Direction? s
You walked south
Zombie went east
```

How do I keep track of location of the person and the zombie?

```
personX = 0
personY = 0

zombieX = 0
zombieY = 0
```

How do I detect when the person gets eaten?

```
if personX == zombieX and personY == zombieY:
    print("Zombie got your braaaains!")
    gameOver = True
```

Extreme Zombie Apocalypse

```
Level: 0
. . ! . . . . . . .
. . . . *. . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . .  *  . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . #
Direction? s
You walked south
Zombie went east
```

What if we need to keep track of two zombies?

```
personX = 0
personY = 0

zombieX1 = 0
zombieY1 = 0

zombieX2 = 0
zombieY2 = 0

...

if (personX == zombieX1 and personY == zombieY1) or
    (personX == zombieX2 and personY == zombieY2):
    print("Zombie got your braaaains!")
    gameOver = True
```

Super Extreme Zombie Apocalypse

```
Level: 0
. . ! . . . * . . .
. . . . * . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . * . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . #
Direction? s
You walked south
Zombie went east
```

What if we need to keep track of three zombies?

```
personX = 0
personY = 0

zombieX1 = 0
zombieY1 = 0

zombieX2 = 0
zombieY2 = 0

zombieX3 = 0
zombieY3 = 0

...

if (personX == zombieX1 and personY == zombieY1) or
   (personX == zombieX2 and personY == zombieY2) or
   (personX == zombieX3 and personY == zombieY3):
    print("Zombie got your braaaains!")
    gameOver = True
```

Zombie Apocalypse: The Rising

```
You walked south
Zombie went west
Level: 5
. * . * .
. . . * .
! . * . .
* . . . .
. . * . #
Direction?
```

What if we want to add one zombie every time the player advances a level?

No good way to do this with simple variables!

Lists to the Rescue!

- We've already seen lists:

```
import sys
```

```
% python CostCalc.py bananas 12 0.21
To buy 12 bananas you will need $2.52
```

identifier	meaning	value	type
`sys.argv[0]`	The name of your program	`CostCalc.py`	`string`
`sys.argv[1]`	1st thing on command line after Python program name	`"bananas"`	`string`
`sys.argv[2]`	2nd thing on command line	`"12"`	`string`
`sys.argv[3]`	3rd thing on command line	`"0.21"`	`string`
`len(sys.argv)`	# of things on command line	`4`	`int`

Lists: Storing Many Things

- Lists: store many variables
- Goal: Ten variables
  - e.g. To hold the values 0-9

```
a0 = 0
a1 = 1
a2 = 2
a3 = 3
a4 = 4
a5 = 5
a6 = 6
a7 = 7
a8 = 8
a9 = 9
```

Lists: Accessing Elements

- Lists: we can use a variable as the index!
  - Makes code shorter, cleaner, less buggy

```
N = 10                              # number of items
a = []                              # empty list

for i in range(0,N):                # initialize list elements
   a.append(i)                      # to be 0 - 9
```

1st element of array is a[0]. We count from zero in computer science!

Lists: Easy to Extend

- Lists: can hold lots and lots of data
  - Same code, but now holds 100,000 integers:

```
N = 100000                          # size of list
a = []                               # declare list

for i in range(0,N):                 # initialize list elements
   a.append(i)                       # to be 0 - 99999
```

More About List Indices

- Index of first list element is 0
- Last valid Index is `len(listName) - 1`
- List indices must be within bounds to be valid
  - When program tries to access outside bounds, IndexError occurs
- To access an element use
  - The name of the list
  - An index number enclosed in braces
  - In Python, special index can be used to get the last item
    - `listName[-1]`

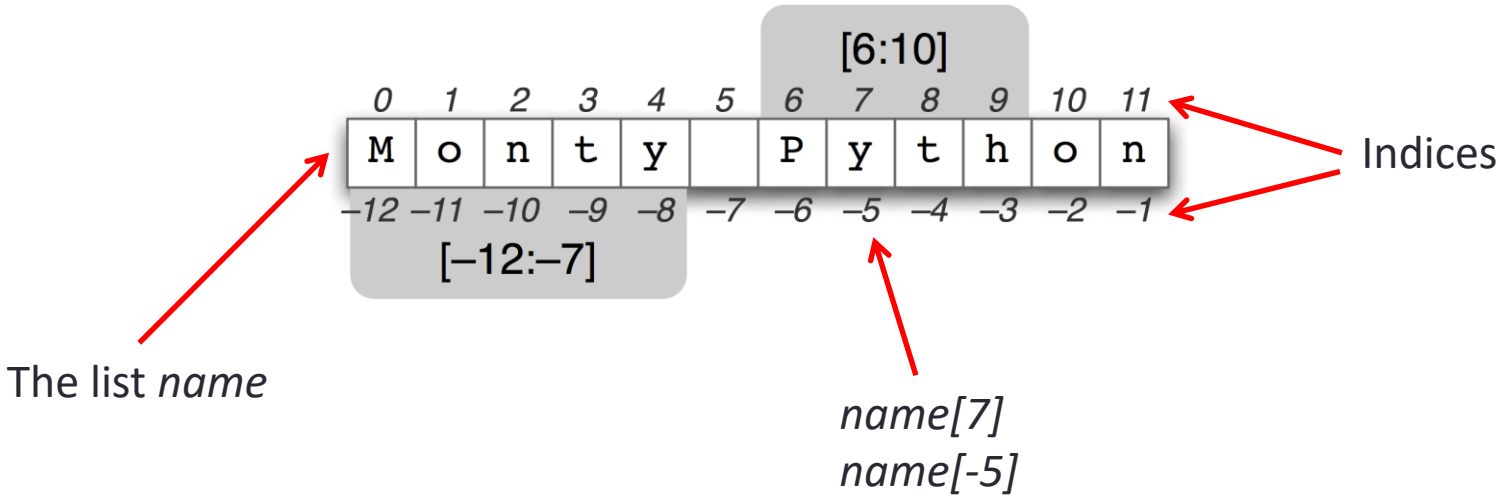
Gotcha – Don't Exceed List Bounds

- The code below fails if the user enters a number like 4.
- Should use input validation to catch this.

```
count = [0,0,0,0]

print("Enter ten numbers between 0 and 3.")
for i in range(0, 10):
    num = int(input())
    count[num] += 1
for i in range (0, len(count)):
    print("You entered " + str(count[i]) + " " + str(i) + "'s")
```

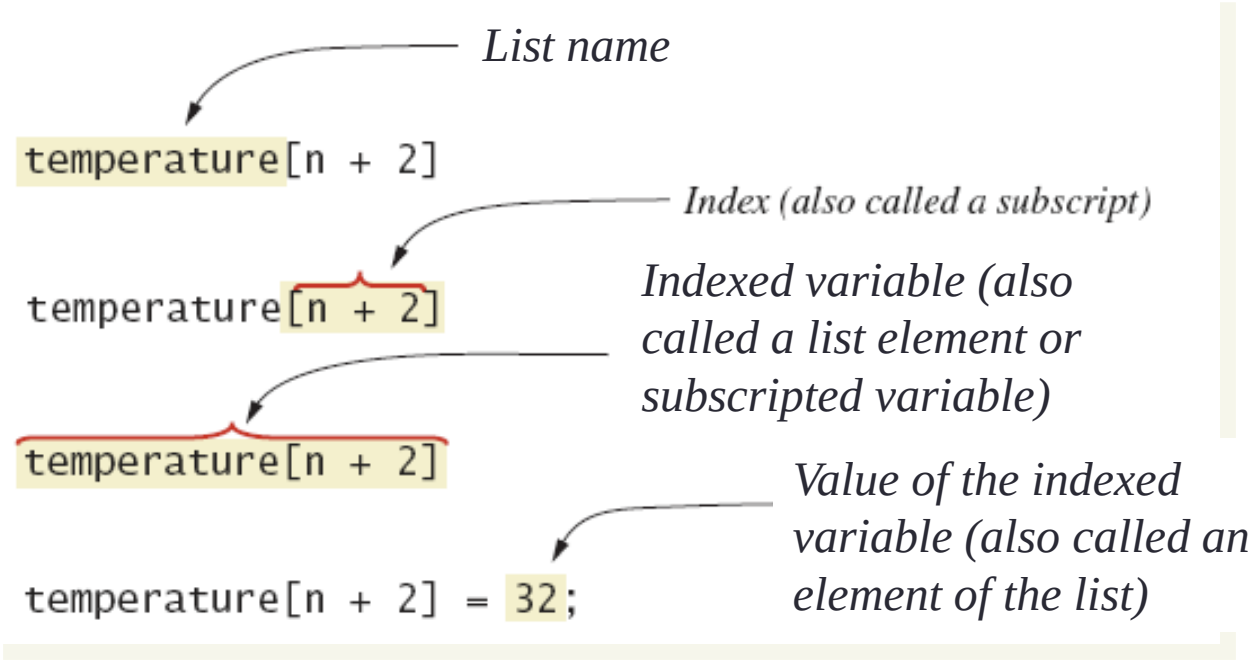
Creating and Accessing Lists

- A common way to visualize a list
  - `name = ['M', 'o', 'n', 't', 'y', ' ', 'P', 'y', 't', 'h', 'o', 'n']`



List Details

- List terminology



Initializing Lists

- Possible to initialize when created

```
reading = [3.3, 15.8, 9.7]
```

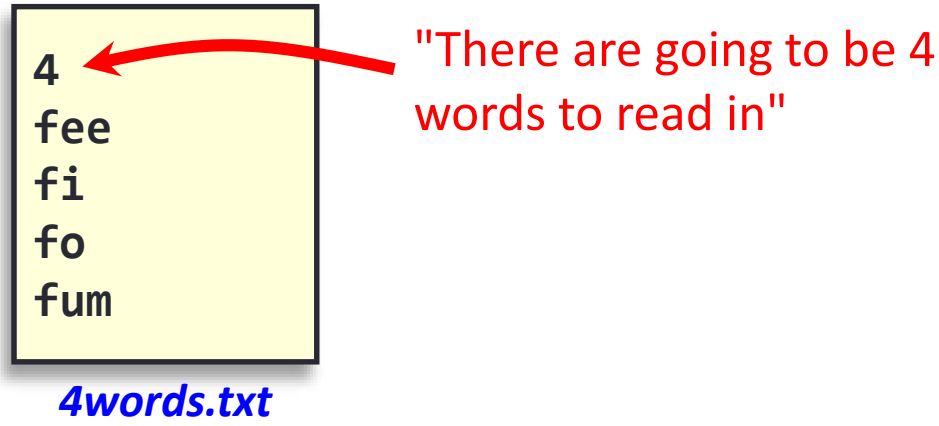
- Also may use normal assignment statements
  - One at a time, if the list is already defined:

```
reading[1] = 4.5
```

  - In a loop, with an empty list, use append:

```
count = []
for i in range(0, 100):
    count.append(0)
```

Lists: Loading Data from File



- Read words into list
- Print out words in reverse order

```
% python Backwards.py 4words.txt
fum fo fi fee
```

Lists: Loading Data from File

```
% python Backwards.py 4words.txt
```

```
import sys

fileName = sys.argv[1]
file = open(fileName, 'r')
num = int(file.readline())
words = []

for i in range (0, num):
    words.append(file.readline().strip())
file.close()

for i in range(num-1, -1, -1):
    print(words[i])
```

```
4
fox
brown
quick
the
```

Super Extreme Zombie Apocalypse

What if we need to keep track of three zombies?

```
Level: 0
. . ! . . . . . . .
. . . . . . . . . *
. . . . . . . . . .
. . . . . . . . . .
. . . . * . . * . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . .
. . . . . . . . . #
Direction? s
You walked south
Zombie went east
```

```
import random

personX = 0
personY = 0
NUM_ZOMBIES = 3  # constant defining # of zombies

zombieX = []     # create x-pos array
zombieY = []     # create y-pos array
for i in range(0, NUM_ZOMBIES):
    zombieX.append(0)
    zombieY.append(0)

# Set random initial location for each zombie (they can overlap)
for i in range (0, NUM_ZOMBIES):
   zombieX[i] = random.randint(0,width)  # set i-th zombie's x-pos
   zombieY[i] = random.randint(0,height) # set i-th zombie's y-pos


...

i = 0
while i < len(zombieX) and not(gameOver):
    if personX == zombieX[i]) and personY == zombieY[i]:
        print("Zombie got your braaaains!")
        gameOver = True
    i += 1
```

Super Mega Extreme Zombie Apocalypse

What if we need to keep track of thirty zombies?

```
Level: 0
* . ! * . . . * . .
. * . . * * . * . *
. . * . . . . . * *
* . * . . . * . . .
. . . . * . . * * *
. . * . * . . . . .
. * . . . . * . * .
. . . . . * . . . .
. . * . . . * . * .
. . . . * . . . . #
Direction? s
You walked south
Zombie went east
```

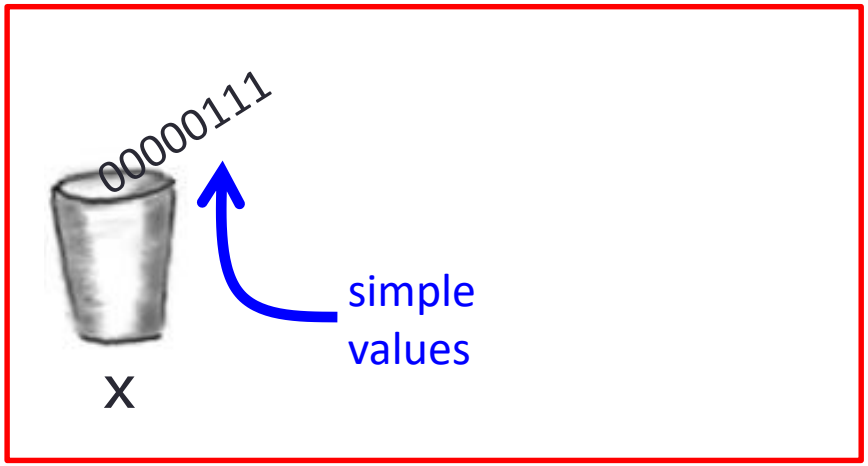
```
import random

personX = 0
personY = 0
NUM_ZOMBIES = 30 # constant defining # of zombies

zombieX = []     # create x-pos array
zombieY = []     # create y-pos array
for i in range(0, NUM_ZOMBIES):
    zombieX.append(0)
    zombieY.append(0)

# Set random initial location for each zombie (they can overlap)
for i in range (0, NUM_ZOMBIES):
   zombieX[i] = random.randint(0,width)  # set i-th zombie's x-pos
   zombieY[i] = random.randint(0,height) # set i-th zombie's y-pos



...

i = 0
while i < len(zombieX) and not(gameOver):
    if personX == zombieX[i]) and personY == zombieY[i]:
        print("Zombie got your braaaains!")
        gameOver = True
    i += 1
```

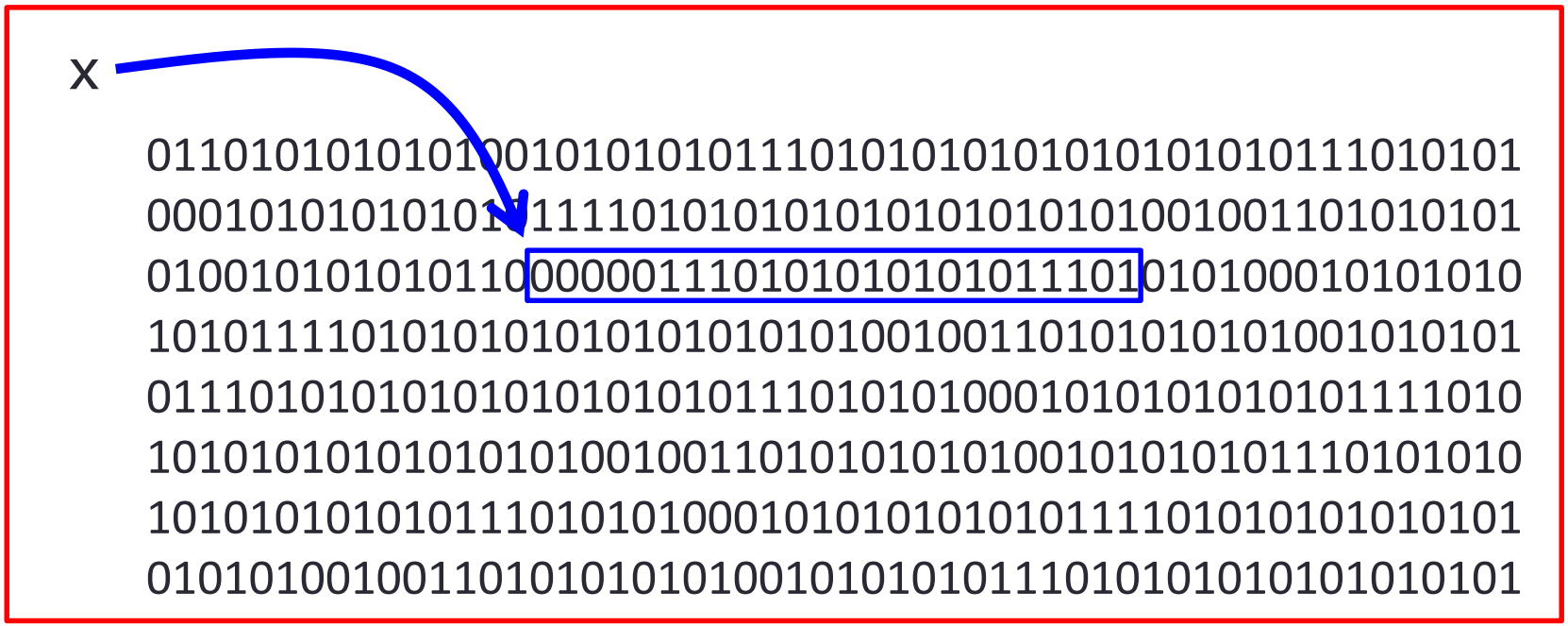
List Assignment and Equality

- Variable names refer to where the value is stored
  - Assignment and equality operators can behave (misbehave)
- Variable for the list object contains memory address of the object
  - Assignment operator `=` copies this address
  - Equality operator `==` tests whether two arrays contain the same elements
- To assign the contents of one array to a new separate array, you need to do slicing
  - We'll talk about this next week

Creating and Initializing a Variable

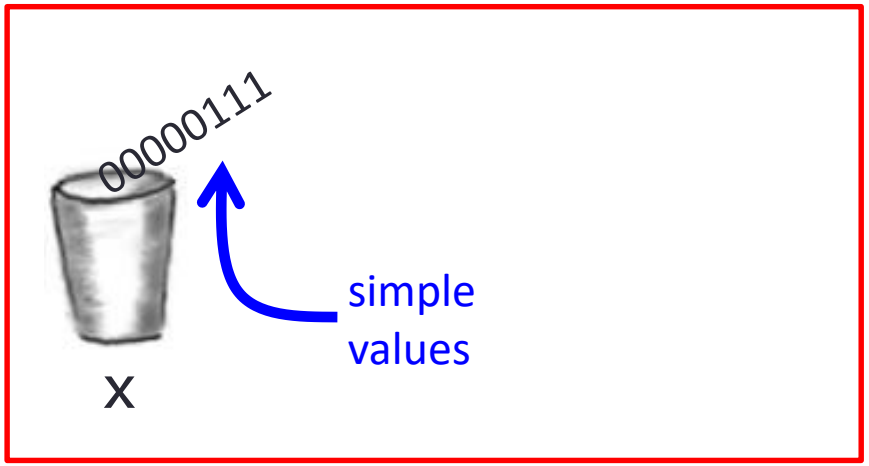
```
x = [7, 85, 93]
```

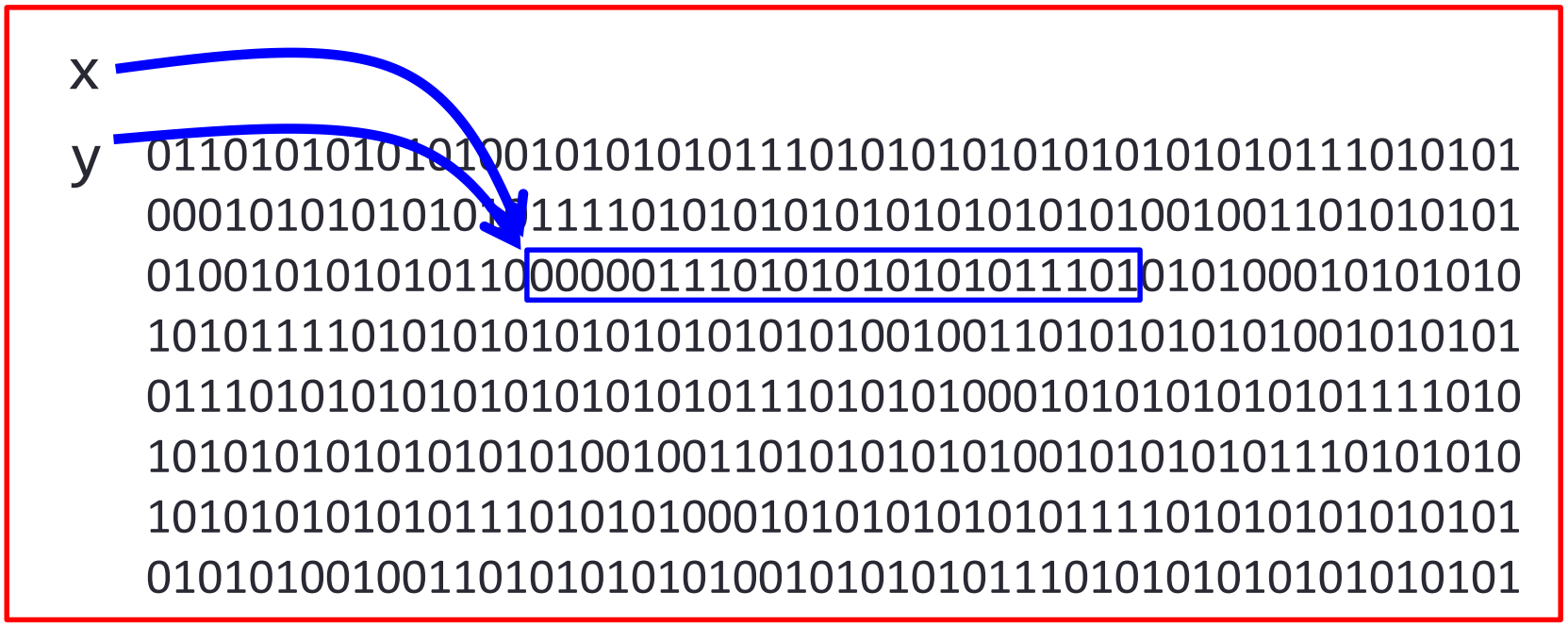




Creating and Initializing a Variable

```
x = [7, 85, 93]
y = x
```





Summary

- List Basics
- Creating and Accessing Lists
- List Details
- Length of a List

Your Turn

- Write a program that creates a list of 10 items. Use a ***for*** construct to assign the values of 10 through 19 to the elements of the list. Print out the value of each element of the list as you assign them.

- Name your program List.py and submit it to the Activity01 dropbox on Moodle. 1 Extra Credit (EC) point for turning something in, 2 EC points for turning in something that is correct.