Preliminaries : Announcements

Homework #2 is now available

- 10 Questions
- A single programming task
- Start right away

9. Briefly define and explain the difference between unary selection, binary selection, chained conditionals, and nested conditionals.

10. What is the output of the following piece of python code (answer this question in the general sense; the values of a, b, y, and ζ are not provided for you):

   \[ a + b + y + ζ \]

II. Coding Task: 60 points

Assume that you are a programmer for the video game company Fungi. The text adventure game being prepared for launch involves a character meandering through the forest, during which he finds and picks up mushrooms. Your task is to write a program that simulates the game in which the character encounters

1. A shiitake mushroom
2. A portobella mushroom

The character encounters the mushrooms in one of the following situations:

- He asks the player if he wants to trade his mushrooms for rubies according to the table (the chef’s secret formula).
- He asks the player if he wants to make the exchange, if the player specifies “yes,” the game should output the final number of rubies that the player wins.

The table:

<table>
<thead>
<tr>
<th>Mushroom</th>
<th>Player Willing to Trade</th>
<th>Chef Willing to Trade</th>
<th>Rubies Offered by Chef</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shitake</td>
<td>Fewer than 10</td>
<td>Fewer than 5</td>
<td>Twice the number of Shitake offered for trade</td>
</tr>
<tr>
<td>Portobella</td>
<td>Fewer than 10</td>
<td>5 or more</td>
<td>Three times the number of Portobella offered for trade</td>
</tr>
<tr>
<td>Shitake + Portobella</td>
<td>Multiple of 24</td>
<td>20 or more</td>
<td>Four times the number of Portobella offered for trade</td>
</tr>
<tr>
<td>Portobella</td>
<td>Fewer than 20</td>
<td>Any</td>
<td>Five times the number of Portobella offered for trade</td>
</tr>
</tbody>
</table>

If the player specifies “yes,” the game should output the final number of rubies that the player wins.
Take home exercise ...

Write python code that:

1. Prompts a user to input how many cups of coffee have been consumed today
2. Prompts a user to input how many homework assignments are due tomorrow
3. If more than 10 cups of coffee have been consumed, then the program should output
   
   Enough caffeine to complete all assignments

4. If at least 5 cups of coffee have been consumed, then the program should indicate that 1/2 (integer division) of the assignments due will be completed
5. If 4 or fewer cups of coffee have been consumed, then the program should indicate that 1/3 (integer division) of the assignments due will be completed

```
Cups of coffee drank today? 7
Number of assignments due tomorrow? 5
Enough caffeine to complete only 2 assignments

Cups of coffee drank today? 3
Number of assignments due tomorrow? 43
Enough caffeine to complete only 14 assignments
```
From Last Time

code

```python
if (Boolean expression) :
    statement
```

```python
if (isCloudy or isRaining) :
    print("bring coat and umbrella just in case")
```

We learned how to write code to permit our programs to “make decisions”
The execution of the code for an unary selection statement (an if statement) proceeds down only ONE possible path, and NOT both.
From Last Time

The execution of the code for an unary selection statement (an if statement) proceeds down only ONE possible path, and NOT both.

If the Boolean expression evaluates to True,
From Last Time

The execution of the code for an unary selection statement (an if statement) proceeds down only ONE possible path, and NOT both

If the Boolean expression evaluates to True, then the statement(s) in the code block is/are executed
From Last Time

The execution of the code for an unary selection statement (an if statement) proceeds down only ONE possible path, and NOT both.

If the Boolean expression evaluates to False, then the statement(s) in the code block are NOT executed.
From Last Time

**Code**

```python
if (Boolean expression) :
    statement

if (isCloudy or isRaining) :
    print("bring coat and umbrella just in case")
```

**Flowchart**

```
start
\[\rightarrow\] condition
\[\rightarrow\] True
\[\rightarrow\] statement
\[\rightarrow\] end
\[\rightarrow\] False
```

**Be able to draw a flow diagram that specifies under what conditions which statements are executed and in WHICH order**

```python
if (Boolean expression) :
    statement_1
else :
    statement_2

if (4 == 3):
    print (3 != 4)
else:
    print ("not equal")
```
From Last Time

**code**

```python
if (Boolean expression) :
    statement

if (isCloudy or isRaining) :
    print("bring coat and umbrella just in case")
```

**flowchart**

```
start
condition

<table>
<thead>
<tr>
<th>True</th>
<th>statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td></td>
</tr>
</tbody>
</table>

end
```

```
if (Boolean expression) :
    statement_1
else :
    statement_2

if (4 == 3):
    print (3 != 4)
else:
    print ("not equal")
```

```
start
condition

<table>
<thead>
<tr>
<th>True</th>
<th>statement_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>statement_2</td>
</tr>
</tbody>
</table>

end
```
Today

“How” to program and debug
Finishing up Conditionals
How to “program”

1. Question: 40 points

Please keep your answers brief. A sentence or two usually suffices for free-response questions.

1. Define Boolean algebra
2. Explain why \((3^2)^2\) is evaluated differently than \((3^2)^2\)
3. Identify the error(s) in the following snippet of python code:
   ```python
   sumTurtles = int(input("How many turtles do you have?"))
   for x in range(sumTurtles):
       print("Wow that is a lot of turtles")
   ```
4. Explain the difference between initializing and updating a variable’s value.
5. What are the python operations for “greater than or equal to,” and “less than or equal to”?
6. Which of the following are python Boolean expressions? Select all correct choices.
   A. True
   B. False
   C. 3 > 2
   D. 12.74 > 16
   E. 14 > 9
7. Fix the following code so that it is syntactically correct and demonstrates if the variable sumTurtles is equal to 7, 8, or 32:
   ```python
   sumTurtles == 7 or 8 or 32
   ```

Answers to questions
How to “program”

Q: How should you complete the programming task?

Q: Is there a good versus a bad way to “code”?

Mine and the CS department’s goal is to teach you how to THINK and DO like a good programmer thinks and does ... learning python syntax is just a small portion of that process.

Therefore we want to teach you good practices in addition to how to write syntactically correct python code.
How to “program”

Q: How should you complete the programming task?

A. Sit down, write code, get up & get coffee, walk dog, return and sit down, code some more, take a nap, finish code and compile and test it

B. Write 50 lines of uncommented code, (attempt) to run it, (hopefully) fix bugs, pull hair out, not get any sleep, submit

C. Write pseudocode, write 50 lines of random code, attempt to run it, fix bugs, submit

D. Write pseudocode, use pseudocode to write the first 5 lines of code, run the code, fix bugs, write next 5 lines of code, run the code, fix bugs, take a break, write remainder of program, test program, submit, get sleep

Q: Is there a good versus a bad way to “code”?

Answers to questions
How to “program”

Q: How should you complete the programming task?

A. Sit down, write code, get up & get coffee, walk dog, return and sit down, code some more, take a nap, finish code and compile and test it

B. Write 50 lines of uncommented code, (attempt) to run it, (hopefully) fix bugs, pull hair out, not get any sleep, submit

C. Write pseudocode, write 50 lines of random code, attempt to run it, fix bugs, submit

D. Write pseudocode, use pseudocode to write the first 5 lines of code, run the code, fix bugs, write next 5 lines of code, run the code, fix bugs, take a break, write remainder of program, test program, submit, get sleep

Q: Is there a good versus a bad way to “code”?

Answers to questions
Q: How should you complete the programming task?

A. Sit down, write code, get up & get coffee, walk dog, return and sit down, code some more, take a nap, finish code and compile and test it
B. Write 50 lines of uncommented code, (attempt) to run it, (hopefully) fix bugs, pull hair out, not get any sleep, submit
C. Write pseudocode, write 50 lines of random code, attempt to run it, fix bugs, submit
D. Write pseudocode, use pseudocode to write the first 5 lines of code, run the code, fix bugs, write next 5 lines of code, run the code, fix bugs, take a break, write remainder of program, test program, submit, get sleep

Q: How should you “test” your python program?
How to “program”

Just like there are tried and proven “good practices” for how to write code well, there is a systematic way to test your program.

Formal testing methods are beyond the scope of this course (which you’ll learn about if you go on in computer science) ... but you can still use a systematic approach to test your program.

Q: How should you complete the programming task?

Q: Is there a good versus a bad way to “code”?

A. Sit down, write code, get up & get coffee, walk dog, return and sit down, code some more, take a nap, finish code and compile and test it

B. Write 50 lines of uncommented code, (attempt) to run it, (hopefully) fix bugs, pull hair out, not get any sleep, submit

C. Write pseudocode, write 50 lines of random code, attempt to run it, fix bugs, submit

D. Write pseudocode, use pseudocode to write the first 5 lines of code, run the code, fix bugs, write next 5 lines of code, run the code, fix bugs, take a break, write remainder of program, test program, submit, get sleep
How to “program”

<table>
<thead>
<tr>
<th>Shitake Found / Willing to Trade</th>
<th>Portaballa Found / Willing to Trade</th>
<th>Chef's Exchange Offer</th>
<th>Accept / Not Accept</th>
<th>Final Count of Player's Shitake/Portaballa/Rubies</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/5</td>
<td>30/22</td>
<td>66 rubies</td>
<td>Yes</td>
<td>5/8/66</td>
</tr>
<tr>
<td>100/0</td>
<td>40/5</td>
<td>15 rubies</td>
<td>Yes</td>
<td>100/35/15</td>
</tr>
<tr>
<td>10/10</td>
<td>5/6</td>
<td>Chef runs away</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>10/10</td>
<td>6/5</td>
<td>50 rubies</td>
<td>No</td>
<td>10/6/0</td>
</tr>
<tr>
<td>20/0</td>
<td>0/0</td>
<td>Unwilling to trade</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>13/12</td>
<td>9/8</td>
<td>8 rubies</td>
<td>Yes</td>
<td>1/1/8</td>
</tr>
</tbody>
</table>

From the HW2 handout ...

Q: How do you use these sample inputs/outputs to test your program?
How to “program”

<table>
<thead>
<tr>
<th>Shitake Found / Willing to Trade</th>
<th>Portabella Found / Willing to Trade</th>
<th>Chef’s Exchange Offer</th>
<th>Accept / Not Accept</th>
<th>Final Count of Player’s Shitake/Portabella/Rubies</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/5</td>
<td>30/22</td>
<td>66 rubies</td>
<td>Yes</td>
<td>5/8/66</td>
</tr>
<tr>
<td>100/0</td>
<td>40/5</td>
<td>15 rubies</td>
<td>Yes</td>
<td>100/35/15</td>
</tr>
<tr>
<td>10/10</td>
<td>5/6</td>
<td>Chef runs away</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>10/10</td>
<td>6/5</td>
<td>50 rubies</td>
<td>No</td>
<td>10/6/0</td>
</tr>
<tr>
<td>20/0</td>
<td>0/0</td>
<td>Unwilling to trade</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>13/12</td>
<td>9/8</td>
<td>8 rubies</td>
<td>Yes</td>
<td>1/1/8</td>
</tr>
</tbody>
</table>

**Input**

10, 30, 5, 22, yes

**Output**
### How to “program”

<table>
<thead>
<tr>
<th>Shitake Found / Willing to Trade</th>
<th>Portabella Found / Willing to Trade</th>
<th>Chef’s Exchange Offer</th>
<th>Accept / Not Accept</th>
<th>Final Count of Player’s Shitake/Portabella/Rubies</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/5</td>
<td>30/22</td>
<td>66 rubies</td>
<td>Yes</td>
<td>5/8/66</td>
</tr>
<tr>
<td>100/0</td>
<td>40/5</td>
<td>15 rubies</td>
<td>Yes</td>
<td>100/35/15</td>
</tr>
<tr>
<td>10/10</td>
<td>5/6</td>
<td>Chef runs away</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>10/10</td>
<td>6/5</td>
<td>50 rubies</td>
<td>No</td>
<td>10/6/0</td>
</tr>
<tr>
<td>20/0</td>
<td>0/0</td>
<td>Unwilling to trade</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>13/12</td>
<td>9/8</td>
<td>8 rubies</td>
<td>Yes</td>
<td>1/1/8</td>
</tr>
</tbody>
</table>

**Input**

10, 30, 5, 22, yes

**Output**

5/8, 66 rubies

[Image of computer output]
# How to “program”

<table>
<thead>
<tr>
<th>Shitake Found / Willing to Trade</th>
<th>Portabella Found / Willing to Trade</th>
<th>Chef’s Exchange Offer</th>
<th>Accept / Not Accept</th>
<th>Final Count of Player’s Shitake/Portabella/Rubies</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/5</td>
<td>30/22</td>
<td>66 rubies</td>
<td>Yes</td>
<td>5/8/66</td>
</tr>
<tr>
<td>100/0</td>
<td>40/5</td>
<td>15 rubies</td>
<td>Yes</td>
<td>100/35/15</td>
</tr>
<tr>
<td>10/10</td>
<td>5/6</td>
<td>Chef runs away</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>10/10</td>
<td>6/5</td>
<td>50 rubies</td>
<td>No</td>
<td>10/6/0</td>
</tr>
<tr>
<td>20/0</td>
<td>0/0</td>
<td>Unwilling to trade</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>13/12</td>
<td>9/8</td>
<td>8 rubies</td>
<td>Yes</td>
<td>1/1/8</td>
</tr>
</tbody>
</table>

**Input**

10, 30, 5, 22, yes  
100, 40, 0, 5, yes

**Output**

5/8, 66 rubies ✔  
100/35, 15 rubies ✔
How to “program”

<table>
<thead>
<tr>
<th>Shitake Found / Willing to Trade</th>
<th>Portabella Found / Willing to Trade</th>
<th>Chef’s Exchange Offer</th>
<th>Accept / Not Accept</th>
<th>Final Count of Player’s Shitake/Portabella/Rubies</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/5</td>
<td>30/22</td>
<td>66 rubies</td>
<td>Yes</td>
<td>5/8/66</td>
</tr>
<tr>
<td>100/0</td>
<td>40/5</td>
<td>15 rubies</td>
<td>Yes</td>
<td>100/35/15</td>
</tr>
<tr>
<td>10/10</td>
<td>5/6</td>
<td>Chef runs away</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>10/10</td>
<td>6/5</td>
<td>50 rubies</td>
<td>No</td>
<td>10/6/0</td>
</tr>
<tr>
<td>20/0</td>
<td>0/0</td>
<td>Unwilling to trade</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>13/12</td>
<td>9/8</td>
<td>8 rubies</td>
<td>Yes</td>
<td>1/1/8</td>
</tr>
</tbody>
</table>

Input
10, 30, 5, 22, yes
100, 40, 0, 5, yes
10, 5, 10, 6

Output
5/8, 66 rubies
100/35, 15 rubies
34/35, 4 rubies

Q: At this point, you should:

A. Fix the cause of the error, & continue testing by confirming that the last three combinations “work”
B. Fix the cause of the error and restart testing from the beginning
How to “program”

<table>
<thead>
<tr>
<th>Shitake Found / Willing to Trade</th>
<th>Portabella Found / Willing to Trade</th>
<th>Chef’s Exchange Offer</th>
<th>Accept / Not Accept</th>
<th>Final Count of Player’s Shitake/Portabella/Rubies</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/5</td>
<td>30/22</td>
<td>66 rubies</td>
<td>Yes</td>
<td>5/8/66</td>
</tr>
<tr>
<td>100/0</td>
<td>40/5</td>
<td>13 rubies</td>
<td>Yes</td>
<td>100/35/15</td>
</tr>
<tr>
<td>10/10</td>
<td>5/6</td>
<td>Chef runs away</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>10/10</td>
<td>6/5</td>
<td>50 rubies</td>
<td>No</td>
<td>10/6/0</td>
</tr>
<tr>
<td>20/0</td>
<td>0/0</td>
<td>Unwilling to trade</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>13/12</td>
<td>9/8</td>
<td>8 rubies</td>
<td>Yes</td>
<td>1/1/8</td>
</tr>
</tbody>
</table>

Input

10, 30, 5, 22, yes
100, 40, 0, 5, yes
10, 5, 10, 6

Output

5/8, 66 rubies ✔
100/35, 15 rubies ✔
34/35, 4 rubies ✗

Q: At this point, you should:

A. Fix the cause of the error, & continue testing by confirming that the last three combinations “work”

B. Fix the cause of the error and restart testing from the beginning

Q: Why?
How to “program”

A. Fix the cause of the error, & continue testing by confirming that the last three combinations “work”

✔️ B. Fix the cause of the error and restart testing from the beginning

Q: Why?

<table>
<thead>
<tr>
<th>Shitake Found / Willing to Trade</th>
<th>Portabella Found / Willing to Trade</th>
<th>Chef’s Exchange Offer</th>
<th>Accept / Not Accept</th>
<th>Final Count of Player’s Shitake/Portabella/Rubies</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/5</td>
<td>30/22</td>
<td>66 rubies</td>
<td>Yes</td>
<td>5/8/66</td>
</tr>
<tr>
<td>100/0</td>
<td>40/5</td>
<td>15 rubies</td>
<td>Yes</td>
<td>100/35/15</td>
</tr>
<tr>
<td>10/10</td>
<td>5/6</td>
<td>Chef runs away</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>10/10</td>
<td>6/5</td>
<td>50 rubies</td>
<td>No</td>
<td>10/6/0</td>
</tr>
<tr>
<td>20/0</td>
<td>0/0</td>
<td>Unwilling to trade</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>13/12</td>
<td>9/8</td>
<td>8 rubies</td>
<td>Yes</td>
<td>1/1/8</td>
</tr>
</tbody>
</table>

99 little bugs in the code.
99 little bugs in the code.
Take one down, patch it around.
127 little bugs in the code...
How to “program”

<table>
<thead>
<tr>
<th>Shitake Found / Willing to Trade</th>
<th>Portabella Found / Willing to Trade</th>
<th>Chef’s Exchange Offer</th>
<th>Accept / Not Accept</th>
<th>Final Count of Player’s Shitake/Portabella/Rubies</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/5</td>
<td>30/22</td>
<td>66 rubies</td>
<td>Yes</td>
<td>5/8/66</td>
</tr>
<tr>
<td>100/0</td>
<td>40/5</td>
<td>15 rubies</td>
<td>Yes</td>
<td>100/35/15</td>
</tr>
<tr>
<td>10/10</td>
<td>5/6</td>
<td>Chef runs away</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>10/10</td>
<td>6/5</td>
<td>50 rubies</td>
<td>No</td>
<td>10/6/0</td>
</tr>
<tr>
<td>20/0</td>
<td>0/0</td>
<td>Unwilling to trade</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>13/12</td>
<td>9/8</td>
<td>8 rubies</td>
<td>Yes</td>
<td>1/1/8</td>
</tr>
</tbody>
</table>

Just like in the “real” world (you cannot predict exactly how a user will interact with your program), when your homework is graded, different inputs will be tried.

Don’t worry too much about whether your program works correctly on ALL Possible input combinations, but be sure to test it a little to make sure it works reasonably well.
Goal: We want to check if two numbers, x and y, are the same, if \( x > y \), or \( y > x \)?

Write code that achieves that task.
Goal: We want to check if two numbers, x and y, are the same, if \( x > y \), or \( y > x \)?
Write code that achieves that task.

```python
if (numCows == 32):
    print("32 cows in the pasture")
else:
    if (numCows < 32):
        print("Fewer than 32 cows")
    else:
        print("More than 32 cows")
```

Task: Draw the flow diagram for the code in the box
Task: Draw the flow diagram for the code in the box

```
if (numCows == 32):
    print("32 cows in the pasture")
else:
    if (numCows < 32):
        print("Fewer than 32 cows")
    else:
        print("More than 32 cows")
```

Q: How many if conditionals?
Q: How many statements?

There are multiple approaches

1. Write down all of the statements and all of the conditionals, and then connect them with arrows
2. "build" the flow chart as you "execute" the program in your head
Conditional

Task: Draw the flow diagram for the code in the box

```
if (numCows == 32) :
    print("32 cows in the pasture")
else :
    if (numCows < 32) :
        print("Fewer than 32 cows")
    else :
        print ("More than 32 cows")
```

There are multiple approaches

1. Write down all of the statements and all of the conditionals, and then connect them with arrows
2. “build” the flow chart as you “execute” the program in your head
Conditional

Task: Draw the flow diagram for the code in the box

```python
if (numCows == 32):
    print("32 cows in the pasture")
else:
    if (numCows < 32):
        print("Fewer than 32 cows")
    else:
        print("More than 32 cows")
```

There are multiple approaches

1. Write down all of the statements and all of the conditionals, and then connect them with arrows
2. "build" the flow chart as you "execute" the program in your head
Conditional

Task: Draw the flow diagram for the code in the box

```python
if (numCows == 32):
    print("32 cows in the pasture")
else:
    if (numCows < 32):
        print("Fewer than 32 cows")
    else:
        print("More than 32 cows")
```

There are multiple approaches

1. Write down all of the statements and all of the conditionals, and then connect them with arrows
2. "build" the flow chart as you "execute" the program in your head
There are multiple approaches

1. Write down all of the statements and all of the conditionals, and then connect them with arrows
2. “build” the flow chart as you “execute” the program in your head

Q: A conditional evaluates to which possible values?
Conditional

Task: Draw the flow diagram for the code in the box

```python
if (numCows == 32):
    print("32 cows in the pasture")
else:
    if (numCows < 32):
        print("Fewer than 32 cows")
    else:
        print("More than 32 cows")
```

There are multiple approaches

1. Write down all of the statements and all of the conditionals, and then connect them with arrows
2. "build" the flow chart as you "execute" the program in your head
CSCI 141
Computer Programming I

Conditional

Task: Draw the flow diagram for the code in the box

```
if (numCows == 32):
    print("32 cows in the pasture")
else:
    if (numCows < 32):
        print("Fewer than 32 cows")
    else:
        print("More than 32 cows")
```

There are multiple approaches

1. Write down all of the statements and all of the conditionals, and then connect them with arrows
2. “build” the flow chart as you “execute” the program in your head
Conditional

Task: Draw the flow diagram for the code in the box

```python
if (numCows == 32):
    print("32 cows in the pasture")
else:
    if (numCows < 32):
        print("Fewer than 32 cows")
    else:
        print("More than 32 cows")
```

There are multiple approaches

1. Write down all of the statements and all of the conditionals, and then connect them with arrows

2. “build” the flow chart as you “execute” the program in your head

Q: A conditional evaluates to which possible values?
CSCI 141
Computer Programming I

Conditional

Task: Draw the flow diagram for the code in the box

```python
if (numCows == 32):
    print("32 cows in the pasture")
else:
    if (numCows < 32):
        print("Fewer than 32 cows")
    else:
        print("More than 32 cows")
```

There are multiple approaches

1. Write down all of the statements and all of the conditionals, and then connect them with arrows
2. "build" the flow chart as you "execute" the program in your head

start

numCows == 32

True

numCows < 32

print("32 cows...")

False

print("More ...")

end

print("Fewer ...")

False

True
Conditional

Task: Draw the flow diagram for the code in the box

```python
if (numCows == 32):
    print("32 cows in the pasture")
else:
    if (numCows < 32):
        print("Fewer than 32 cows")
    else:
        print ("More than 32 cows")
```

Things to notice

- Only ONE of the statements is executed when this piece of code is run
- All conditionals must have an outgoing **True** and an outgoing **False** arrow
- Designate the **start** and **end**
- You **cannot** have more conditionals or statements in your flow chart than there are in the code
There is yet another conditional, that the e-book calls a **chained** conditional
There is yet another conditional, that the e-book calls a **chained** conditional

```python
if (numCows == 32) :
    print("32 cows in the pasture")
elif (numCows < 32) :
    print("Fewer than 32 cows")
else :
    print("More than 32 cows")
```

Q: Which keyword do you see that you haven’t seen before?
Conditional

There is yet another conditional, that the e-book calls a **chained conditional**

```python
if (numCows == 32):
    print("32 cows in the pasture")
elif (numCows < 32):
    print("Fewer than 32 cows")
else:
    print("More than 32 cows")
```

Q: Which keyword do you see that you haven’t seen before?

**elif** is python syntax for “else if”. It is different than an **else** because there is a condition associated with it that is checked **only if** the FIRST **if** evaluates to False
There is yet another conditional, that the e-book calls a **chained** conditional

```python
if (numCows == 32) :
    print("32 cows in the pasture")
elif (numCows < 32) :
    print("Fewer than 32 cows")
else :
    print("More than 32 cows")
```

Q: How many checks (evaluations of Boolean expressions) is/are performed by the computer in this example?

A. 1  
B. 2  
C. 1 or more  
D. None of the above
There is yet another conditional, that the e-book calls a **chained** conditional

```python
if (numCows == 32):
    print("32 cows in the pasture")
elif (numCows < 32):
    print("Fewer than 32 cows")
else:
    print("More than 32 cows")
```

Q: How many checks (evaluations of Boolean expressions) is/are performed by the computer in this example?

A. **1**
B. **2**
C. **1 or more**
D. **None of the above**

If `numCows` is 32, then only **1** evaluation of a Boolean expression is performed.
Conditional

There is yet another conditional, that the e-book calls a **chained** conditional

```python
if (numCows == 32):
    print("32 cows in the pasture")
elif (numCows < 32):
    print("Fewer than 32 cows")
else:
    print("More than 32 cows")
```

Q: How many checks (evaluations of Boolean expressions) is/are performed by the computer in this example?

A. 1
B. 2
C. 1 or more
D. None of the above

If `numCows` is not 32, then 2 evaluations of a Boolean expression are performed.
There is yet another conditional, that the e-book calls a **chained** conditional

```python
if (numCows == 32) :
    print(“32 cows in the pasture”)
elif (numCows < 32) :
    print(“Fewer than 32 cows”)  
else :
    print(“More than 32 cows”)  
```

Q: How many checks (evaluations of Boolean expressions) is/are performed by the computer in this example?

A. 1  
B. 2  
C. 1 or more  
D. None of the above

Therefore it depends on how the Boolean expression in the first if is evaluated
There is yet another conditional, that the e-book calls a **chained conditional**

```python
if (numCows == 32) :
    print("32 cows in the pasture")
elif (numCows < 32) :
    print("Fewer than 32 cows")
else :
    print("More than 32 cows")
```

Q: How many print statements is/are executed when this program is run?

A. 1
B. 2
C. 1 or 2
D. 1 or 2 or 3
E. None of the above
There is yet another conditional, that the e-book calls a **chained** conditional.

```python
if (numCows == 32) :
    print("32 cows in the pasture")
elif (numCows < 32) :
    print("Fewer than 32 cows")
else :
    print("More than 32 cows")
```

Q: How many print statements is/are executed when this program is run?

A. 1  
B. 2  
C. 1 or 2  
D. 1 or 2 or 3  
E. None of the above  

If numCows is 32, then only 1 statement is executed.
Conditional

There is yet another conditional, that the e-book calls a **chained** conditional

```python
if (numCows == 32):
    print("32 cows in the pasture")
elif (numCows < 32):
    print("Fewer than 32 cows")
else:
    print("More than 32 cows")
```

Q: How many print statements is/are executed when this program is run?

A. 1  
B. 2  
C. 1 or 2  
D. 1 or 2 or 3  
E. None of the above

If `numCows < 32`, then only 1 statement is executed
Conditional

There is yet another conditional, that the e-book calls a **chained** conditional

```python
if (numCows == 32) :
    print("32 cows in the pasture")
elif (numCows < 32) :
    print("Fewer than 32 cows")
else :
    print("More than 32 cows")
```

Q: How many print statements is/are executed when this program is run?

A. 1  
B. 2  
C. 1 or 2  
D. 1 or 2 or 3  
E. None of the above

If numCows > 32, then only 1 statement is executed
Conditional

There is yet another conditional, that the e-book calls a **chained** conditional

```python
if (numCows == 32) :
    print(\"32 cows in the pasture\")
elif (numCows < 32) :
    print(\"Fewer than 32 cows\")
else :
    print(\"More than 32 cows\")
```

Q: How many print statements is/are executed when this program is run?

A. 1
B. 2
C. 1 or 2
D. 1 or 2 or 3
E. None of the above
There is yet another conditional, that the e-book calls a chained conditional

```python
if (numCows == 32) :
    print("32 cows in the pasture")
elif (numCows < 32) :
    print("Fewer than 32 cows")
else :
    print("More than 32 cows")
```

Task: Write the flow diagram for the code in the box

(on the board exercise)
Conditional

if (numCows == 32) :
    print("32 cows in the pasture")
elif (numCows < 32) :
    print("Fewer than 32 cows")
else :
    print("More than 32 cows")

if (numCows == 32) :
    print("32 cows in the pasture")
else :
    if (numCows < 32) :
        print("Fewer than 32 cows")
    else :
        print ("More than 32 cows")

How are nested ifs and chained conditionals related?

Q: Do the two pieces of code above output the same thing when ...

numCows = 32
numCows = 16
numCows = 70
How are nested ifs and chained conditionals related?

Q: Do the two pieces of code above output the same thing when ...

<table>
<thead>
<tr>
<th>numCows</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>32 cows in the pasture</td>
</tr>
<tr>
<td>16</td>
<td>Fewer than 32 cows</td>
</tr>
<tr>
<td>70</td>
<td>More than 32 cows</td>
</tr>
</tbody>
</table>

Q: If the two pieces of code output the same thing, why use one versus the other?
Conditional

```python
if (numCows == 32) :
    print("32 cows in the pasture")
elif (numCows < 32) :
    print("Fewer than 32 cows")
else :
    print("More than 32 cows")
```

Task: Draw the flow diagram for this code
if (numCows == 32) :
    print("32 cows in the pasture")
elif (numCows < 32) :
    print("Fewer than 32 cows")
else :
    print("More than 32 cows")

Task: Draw the flow diagram for this code
if (numCows == 32) :
    print("32 cows in the pasture")
elif (numCows < 32) :
    print("Fewer than 32 cows")
else :
    print("More than 32 cows")

Q: Is this flow diagram complete?
Conditional

A flow diagram specifies which selection and code statements are executed, and their possible order of execution.

No. The conditionals must have “exiting” True and False arrows.

if (numCows == 32) :
    print("32 cows in the pasture")
elif (numCows < 32) :
    print("Fewer than 32 cows")
else :
    print("More than 32 cows")

start

True

numCows == 32

print("32 cows...")

False

numCows < 32

print("Fewer than ...")

True

print("More ...")

False

end
**Chained Conditionals**

```python
if (numCows == 32):
    print("32 cows in the pasture")
elif (numCows < 32):
    print("Fewer than 32 cows")
else:
    print("More than 32 cows")
```

“rules” for `if`, `elif`, and `else` statements
Chained Conditionals

```
if (numCows == 32):
    print("32 cows in the pasture")
elif (numCows < 32):
    print("Fewer than 32 cows")
```

“rules” for if, elif, and else statements
• The else clause is optional

This is okay, but it means that depending on the value of the variable `numCows`, perhaps nothing will be printed.

What is the output of the program if `numCows = 600`?
Chained Conditionals

if (numCows == 32) :
    print(“32 cows in the pasture”)
elif (numCows == 31) :
    print(“31 cows”)
elif (numCows < 23) :
    print(“Fewer than 23 cows”)
elif (numCows > 2323) :
    print(“Moooo”)
elif (numCows == 7823) :
    print(“Too many cows”)

“rules” for if, elif, and else statements
• The else clause is optional
• You can have an unlimited number of elif statements

You should think of the elif statements being checked one-by-one, from the top to the bottom.

As soon as one of the elif Boolean expressions evaluates to True, then the code block for that elif is executed, and the if selection statement terminates.
Chained Conditionals

if (numCows == 32):
    print("32 cows in the pasture")
elif (numCows == 31):
    print("31 cows")
elif (numCows < 23):
    print("Fewer than 23 cows")
elif (numCows > 2323):
    print("Moooo")
elif (numCows == 7823):
    print("Too many cows")

“rules” for if, elif, and else statements
• The else clause is optional
• You can have an unlimited number of elif statements

You should think of the elif statements being checked one-by-one, from the top to the bottom

As soon as one of the elif Boolean expressions evaluates to True, then the code block for that elif is executed, and the if selection statement terminates

Q: What is printed when numCows = 7823?
Chained Conditionals

You should think of the `elif` statements being checked one-by-one, from the top to the bottom.

As soon as one of the `elif` Boolean expressions evaluates to True, then the code block for that `elif` is executed, and the `if` selection statement terminates.

Q: What is printed when `numCows = 7823`?

```
if (numCows == 32) :
    print(“32 cows in the pasture”)
elif (numCows == 31) :
    print(“31 cows”)  
elif (numCows < 23) :
    print(“Fewer than 23 cows”)  
elif (numCows > 2323) :
    print(“Moooo”)  
elif (numCows == 7823) :
    print(“Too many cows”)  
```

“rules” for `if`, `elif`, and `else` statements

• The `else` clause is optional
• You can have an unlimited number of `elif` statements

Although both of these `elifs` are True, the order of checking specifies that only the `print(“Moooo”)` statement is executed.
Chained Conditionals

```
if (numCows == 32) :
    print("32 cows in the pasture")
elif (numCows < 32) :
    print("Fewer than 32 cows")
else :
    print("More than 32 cows")
elif (numCows == 292834) :
    print("Too many moos")
```

“rules” for `if`, `elif`, and `else` statements

- The `else` clause is optional
- You can have an unlimited number of `elif` statements
- The `else` clause MUST be the last clause of an if-elif-elif, etc. chain of selection statements

Q: Is this syntactically correct?
Chained Conditionals

```python
if (numCows == 32) :
    print("32 cows in the pasture")
elif (numCows < 32) :
    print("Fewer than 32 cows")
else :
    print("More than 32 cows")
elif (numCows == 202834) :
    print("Too many moos")
```

“rules” for `if`, `elif`, and `else` statements

- The `else` clause is optional
- You can have an unlimited number of `elif` statements
- The `else` clause MUST be the last clause of an if-elif-elif, etc. chain of selection statements
How do the selections “relate” to each other?

Task: Be able to look at these pieces of code and “execute” them “mentally” in your head.
How do the selections “relate” to each other?

Q: Do these three pieces of code output the same thing?

A. Yes
B. No
C. Sometimes
D. I don’t know
How do the selections “relate” to each other?

Q: Do these three pieces of code output the same thing?

A. Yes
B. No
C. Sometimes
D. I don’t know

if (numTacos == 32) :
    print("32 tacos")
elif (numTacos < 32) :
    print("Too few tacos")
elif (numTacos == 34) :
    print("34 tacos")
eif (numTacos % 5 == 0) :
    print("multiple of 5 tacos!")
else :
    print("Too many tacos")
How do the selections “relate” to each other?

Q: Do these three pieces of code output the same thing?

A. Yes
B. No
C. Sometimes
D. I don’t know

Code B prints something twice because EACH if is checked

numTacos

32

Different outputs
How do the selections “relate” to each other?

Q: Do these three pieces of code output the same thing?

A. Yes
B. No
C. Sometimes
D. I don’t know

numTacos

25
How do the selections “relate” to each other?

Q: Do these three pieces of code output the same thing?

A. Yes
B. No
C. Sometimes
D. I don’t know

```python
if (numTacos == 32):
    print(“32 tacos”)
elif (numTacos < 32):
    print(“Too few tacos”)
elif (numTacos == 34):
    print(“34 tacos”)
elif (numTacos % 5 == 0):
    print(“multiple of 5 tacos!”)
else:
    print(“Too many tacos”)
```

```python
if (numTacos == 32):
    print(“32 tacos”)
else:
    if (numTacos < 32):
        print(“Too few tacos”)
    else:
        if (numTacos == 34):
            print(“34 tacos”)
        else:
            if (numTacos % 5 == 0):
                print(“multiple of 5 tacos!”)
            else:
                print(“Too many tacos”)
```

```python
if (numTacos == 32):
    print(“32 tacos”)
else:
    if (numTacos < 32):
        print(“Too few tacos”)
    elif (numTacos == 34):
        print(“34 tacos”)
    elif (numTacos % 5 == 0):
        print(“multiple of 5 tacos!”)
    else:
        print(“Too many tacos”)
```

numTacos
How do the selections “relate” to each other?

Q: Do these three pieces of code output the same thing?

A. Yes
B. No
C. Sometimes
D. I don’t know

Q: Is there a value for `numTacos` such that code A, B, and C output the same thing?
How do the selections “relate” to each other?

Q: Do these three pieces of code output the same thing?

A. Yes
B. No
C. Sometimes
D. I don’t know

A B C

numTacos

if (numTacos == 32) :
    print(“32 tacos”) 
elif (numTacos < 32) :
    print(“Too few tacos”) 
elif (numTacos == 34) :
    print(“34 tacos”) 
elif (numTacos % 5 == 0) :
    print(“multiple of 5 tacos!”) 
else :
    print(“Too many tacos”) 

if (numTacos == 32) :
    print(“32 tacos”) 
else :
    if (numTacos < 32) :
        print(“Too few tacos”) 
    else :
        if (numTacos == 34) :
            print(“34 tacos”) 
        else :
            if (numTacos % 5 == 0) :
                print(“multiple of 5 tacos!”) 
            else :
                print (”Too many tacos”) 

if (numTacos == 32) :
    print(“32 tacos”) 
if (numTacos < 32) :
    print(“Too few tacos”) 
if (numTacos == 34) :
    print(“34 tacos”) 
if (numTacos % 5 == 0) :
    print(“multiple of 5 tacos!”) 
else :
    print (“Too many tacos”) 

A B C

35

Same outputs
For loops

Want to major or minor in Computer Science?

- Orange flyer: Details about declaring CS pre-major
- Yellow flyer: Group advising session for pre-majors
- Blue flyer: Computer Science clubs