Preliminaries: Announcements

Homework #1

• For the programming task, I am asking you for the second line of output to “Greet the user and ask to supply the first integer”. This can be done many ways.

```
>> What is your name? Filip
>> Hi Filip, what is the first integer? 7
```

This behavior can also be achieved using the concatenation operator (not yet discussed)

Hint: Refer to the lecture notes and use a combination of sep and end as optional arguments to print.

Homework #2

• Will be posted to the course website on Thursday.
The most-recent “take home” exercise

Q: What is the binary equivalent of the year of your birth?
From Last Time

Q: What is the output of the program shown in the box?

A. 1
B. 2
C. 3
D. 6
E. 9
F. 312
Q: What is the output of the program shown in the box?

A. 1  
B. 2  
C. 3  
D. 6  
E. 9  
F. 312

Last time I mentioned that operators have a precedence level:

- Parentheses
- Exponentiation
- Multiplication and Division
- Addition and Subtraction

I also mentioned that multiple operators of the same level are evaluated left-to-right (also called left associativity).

Thus in python you’d expect ...
From Last Time

Q: What is the output of the program shown in the box?

A. 1
B. 2
C. 3
D. 6
E. 9
F. 312
From Last Time

Q: What is the output of the program shown in the box?

3 ** 1 ** 2

A. 1
B. 2
C. 3
D. 6
E. 9
F. 312

However the exponentiation operator has right associativity, thus the calculation is ...

3 ** 2
3
9
Q: What is the output of the program shown in the box?

A. 1  
B. 2  
C. 3  
D. 6  
E. 9  
F. 312

However the exponentiation operator has right associativity, thus the calculation is ...

To prevent confusion, consider using parentheses to group multiple ** operators.
From Last Time

Q: What is the binary equivalent of the base-10 number 513?
From Last Time

Q: What is the binary equivalent of the base-10 number 513?

0 1 0 0 0 1 0 1

Q: What do each of these 0s and 1s represent?
From Last Time

Q: What is the binary equivalent of the base-10 number 513?

0 1 0 0 0 1 0 1

2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0

Q: What are each of these powers of 2?
From Last Time

Q: What is the binary equivalent of the base-10 number 513?

```
0 1 0 0 0 1 0 1
```

$2^7$ $2^6$ $2^5$ $2^4$ $2^3$ $2^2$ $2^1$ $2^0$

128 64 32 16 8 4 2 1

Q: Will 11111111 “add” up to 513?
From Last Time

Q: What is the binary equivalent of the base-10 number 513?

\[
\begin{array}{cccccccccccc}
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
\end{array}
\]

\[
\begin{array}{cccccccccccc}
2^9 & 2^8 & 2^7 & 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\
? & ? & 128 & 64 & 32 & 16 & 8 & 4 & 2 & 1 \\
\end{array}
\]
From Last Time

Q: What is the binary equivalent of the base-10 number 513?

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^9$</td>
<td>512</td>
<td>256</td>
<td>128</td>
<td>64</td>
<td>32</td>
<td>16</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
From Last Time

Q: What is the binary equivalent of the base-10 number 513?

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2^9</td>
<td>2^8</td>
<td>2^7</td>
<td>2^6</td>
<td>2^5</td>
<td>2^4</td>
<td>2^3</td>
<td>2^2</td>
<td>2^1</td>
<td>2^0</td>
</tr>
</tbody>
</table>

Q: 512 is just one less than 513, so what do we do?
From Last Time

Q: What is the binary equivalent of the base-10 number 513?

\[
\begin{array}{ccccccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
2^9 & 2^8 & 2^7 & 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\
512 & 256 & 128 & 64 & 32 & 16 & 8 & 4 & 2 & 1 \\
\end{array}
\]

Q: And if I had asked you to provide the binary equivalent of 514?

1000000001
Today

Boolean Values
Conditionals
Booleans

We are at the point of the course where I’ve taught you WHAT different types of data a computer program (python in our case) can use, and HOW that data is stored …

but computer programs do much more than just “store” information …

Today we’ll begin discussing what a computer program can DO with data
Booleans

Q: What is a “bool”?  

A. Slang for drool  
B. The word “cool” with the “c” replaced with a “b”  
C. A bowling bowl
Booleans

Q: What is a “bool”? 

A. Slang for drool  
B. The word “cool” with the “c” replaced with a “b”  
C. A bowling bowl

Correct answer: B.
Booleans

Q: What is a “bool”?
A. Slang for drool
B. The word “cool” with the “c” replaced with a “b”
C. A bowling bowl

Q: What is a Boolean value?
A. The cost of a lean ghoul
B. The basic data type that is the foundation of Boolean algebra
C. An eight letter phrase
Booleans

Q: What is a “bool”?
A. Slang for drool
B. The word “cool” with the “c” replaced with a “b”
C. A bowling bowl

Q: What is a Boolean value?
A. The cost of a lean ghoul
B. The basic data type that is the foundation of Boolean algebra
C. An eight letter phrase
Booleans

Q: What is a “bool”?  
- A. Slang for drool
- B. The word “cool” with the “c” replaced with a “b”  
- C. A bowling bowl  

Q: What is a Boolean value?  
- A. The cost of a lean ghoul
- B. The basic data type that is the foundation of Boolean algebra
- C. An eight letter phrase  

Q: Who are such types named after?  
- A. Bill Gates
- B. Copernicus
- C. George Boole
Booleans

**Q: What is a “bool”?**

- A. Slang for drool
- B. The word “cool” with the “c” replaced with a “b”
- C. A bowling bowl

**Q: What is a Boolean value?**

- A. The cost of a lean ghoul
- B. The basic data type that is the foundation of **Boolean algebra**
- C. An eight letter phrase

**Q: Who are such types named after?**

- A. Bill Gates
- B. Copernicus
- C. George Boole

The branch of mathematics that is concerned with the values **true** and **false**, and which uses the operations **AND**, **OR** and **NOT**.
Boolean algebra allows us to reason abstractly about the truthfulness of complex statements ... real-world simple examples easily illustrate how Boolean algebra is used.

Assume that today ...

- cloudy
- no rain

```markdown
<table>
<thead>
<tr>
<th>cloudy</th>
<th>no rain</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="cloudy.png" alt="Cloudy Sky" /></td>
<td><img src="no_rain.png" alt="No Rain" /></td>
</tr>
</tbody>
</table>
```
Boolean algebra allows us to reason abstractly about the truthfulness of complex statements ... real-world simple examples easily illustrate how Boolean algebra is used

Assume that today ...

But tomorrow ...

<table>
<thead>
<tr>
<th>cloudy</th>
<th>no rain</th>
</tr>
</thead>
<tbody>
<tr>
<td>cloudy</td>
<td>rain</td>
</tr>
</tbody>
</table>
Boolean algebra allows us to reason abstractly about the truthfulness of complex statements... real-world simple examples easily illustrate how Boolean algebra is used.

Assume that today...

But tomorrow...

Is it cloudy and rainy?
Is it cloudy or rainy?

Is it cloudy and rainy?
Is it cloudy or rainy?
Boolean algebra allows us to reason abstractly about the truthfulness of complex statements ... real-world simple examples easily illustrate how Boolean algebra is used.

Assume that today ...

<table>
<thead>
<tr>
<th>cloudy</th>
<th>no rain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is it cloudy and rainy?</td>
<td>No</td>
</tr>
<tr>
<td>Is it cloudy or rainy?</td>
<td>Yes</td>
</tr>
</tbody>
</table>

But tomorrow ...

<table>
<thead>
<tr>
<th>cloudy</th>
<th>rain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Is it cloudy and rainy?</td>
<td>Yes</td>
</tr>
<tr>
<td>Is it cloudy or rainy?</td>
<td>Yes</td>
</tr>
</tbody>
</table>

But computers don’t reason about photos ... in order for a computer to evaluate such expressions, we need to create a variable that can store “yes” and “no” statements.
Booleans and Conditional

```python
isCloudy = True or isCloudy = False
isRaining = True or isRaining = False
```
Booleans and Conditional

```python
isCloudy = True or isCloudy = False
isRaining = True or isRaining = False
```

Good variable names

- Descriptive
- Camel capitalization
- A good idea to begin variables that hold True/False values with `is`
Booleans and Conditional

\[
isCloudy = \text{True or } isCloudy = \text{False} \\
isRaining = \text{True or } isRaining = \text{False}
\]

A Boolean value

- Is either True or False
- It is NOT a String, not an integer, float, etc. It IS of type bool (short for Boolean)
- Some textbooks will refer to True as 1, and False to 0 (this is from the electrical engineering perspective)
- In python, **capitalization** is important; true is not the same as True, and false is not the same as False
Booleans and Conditional

```python
isCloudy = True or isCloudy = False
isRaining = True or isRaining = False
```

A Boolean value

- Is either True or False
- It is NOT a String, not an integer, float, etc. It IS of type bool (short for Boolean)
- Some textbooks will refer to True as 1, and False to 0 (this is from the electrical engineering perspective)
- **In python, capitalization is important;** `true` is not the same as `True`, and `false` is not the same as `False`

Python is case sensitive!
On the fly question

Q: Which of the following statements is/are **syntactically correct** in python? Select all correct choices

A. `isYoung = true`
B. `isElderly = "False"`
C. `isYoung = 1`
D. `isElderly = "-44"`
E. `isYoung = (4 + 3)`
F. `isElderly = ((4 // 3) ** 9)`
G. `isYoung = TRUE`
H. `isYoung = True`
On the fly question

Q: Which of the following statements is/are **syntactically correct** in python? Select all correct choices

A. isYoung = true
B. isElderly = “False”
C. isYoung = 1
D. isElderly = “-44”
E. isYoung = (4 + 3)
F. isElderly = ((4 // 3) ** 9
G. isYoung = TRUE
H. isYoung = True

Remember that if something has incorrect syntax, it means it is not following the “rules” of python grammar ...

Be sure you understand why B-E and H are syntactically correct, and why the others are NOT
On the fly question

Q: Which of the following statements is/are **syntactically correct** in python? Select all correct choices

A. `isYoung = true`
B. `isElderly = "False"`
C. `isYoung = 1`
D. `isElderly = "-44"`
E. `isYoung = (4 + 3)`
F. `isElderly = ((4 // 3) ** 9`
G. `isYoung = TRUE`
H. `isYoung = True`

Remember that if something has incorrect syntax, it means it is not following the “rules” of python grammar ...

Each of these are syntactically correct because ...
Q: Which of the following statements is/are syntactically correct in python? Select all correct choices

A Boolean variable is True or False (case sensitive)

- A. `isYoung = true`
- B. `isElderly = "False"`
- C. `isYoung = 1`
- D. `isElderly = "-44"`
- E. `isYoung = (4 + 3)`
- F. `isElderly = ((4 // 3) ** 9`
- G. `isYoung = TRUE`
- H. `isYoung = True`

Mismatched parentheses

Remember that if something has incorrect syntax, it means it is not following the “rules” of python grammar ...

The others are syntactically incorrect because ...

A Boolean variable is True or False (case sensitive)
**Conditional**

\[
isCloudy = \text{True or } isCloudy = \text{False} \\
isRaining = \text{True or } isRaining = \text{False}
\]

Today
- cloudy
- no rain

Tomorrow
- cloudy
- rain

Referring back to our example, let’s write variables to represent the weather conditions today and tomorrow.
isCloudy = True or isCloudy = False
isRaining = True or isRaining = False

Today
isCloudy = True
isRaining = False

Tomorrow
cloudy
rain

Referring back to our example, let’s write variables to represent the weather conditions today and tomorrow.
isCloudy = True or isCloudy = False
isRaining = True or isRaining = False

Today
isCloudy = True
isRaining = False

Tomorrow
isCloudy = True
isRaining = True

Referring back to our example, let’s write variables to represent the weather conditions today and tomorrow.

Okay. Now we can “store” data that refers to whether a simple statement is true or false ...
isCloudy = True or isCloudy = False
isRaining = True or isRaining = False

Today
isCloudy = True
isRaining = False

Tomorrow
isCloudy = True
isRaining = True

But we want to reason about more complex statements

Referring back to our example, let’s write variables to represent the weather conditions today and tomorrow
There are three logical operators that we use to help us reason about complex statements such as

- is it cloudy and raining
- is it cloudy or raining
- is it cloudy and not raining
There are three logical operators that we use to help us reason about complex statements such as

- is it cloudy and raining
- is it cloudy or raining
- is it cloudy and not raining

Q: What is an operator, and how is it different from an operand?
There are three logical operators that we use to help us reason about complex statements such as:

- is it cloudy and raining
- is it cloudy or raining
- is it cloudy and not raining

The logical operators are **and**, **or**, and **not**

Q: Are the logical operators unary, binary, or ternary operators?
There are three logical operators that we use to help us reason about complex statements such as

- is it cloudy and raining
- is it cloudy or raining
- is it cloudy and not raining

The logical operators are **and**, **or**, and **not**

**and** and **or** are binary, because each of them requires two operands, but **not** is unary

These operators have very specific behaviors
Conditional

There are three logical operators that we use to help us reason about complex statements such as

- is it cloudy and raining
- is it cloudy or raining
- is it cloudy and not raining

The logical operators are and, or, and not

and and or are binary, because each of them requires two operands, but not is unary

These operators have very specific behaviors

And
- Under what conditions does the and operator return True
- Under what conditions does the and operator return False

Or
- Under what conditions does the or operator return True
- Under what conditions does the or operator return False

Not
- Under what conditions the not operator return True
- Under what conditions does the not operator return False
There are three logical operators that we use to help us reason about complex statements such as

- **is it cloudy and raining**
- **is it cloudy or raining**
- **is it cloudy and not raining**

The logical operators are **and**, **or**, and **not**.

**And**

“outputs” true if BOTH the left and right operands are true
“outputs” false if ONLY ONE or NEITHER of the operands is true

**Or**

“outputs” true if EITHER (or both) the left and right operands are true
“outputs” false BOTH the left and operands are true

**Not**

“outputs” true if the operand is False
“outputs” false if the operand is True

Truth table explanation on the board
Conditional

A few “math” examples to illustrate the point ... but first we need an operator to make the statement that one thing is equal to another ... only THEN can we determine whether the statement is true of false
Conditional

A few “math” examples to illustrate the point ... but first we need an operator to make the statement that one thing is equal to another ... only THEN can we determine whether the statement is true or false

If we want Python to tell us if 5 is equal to 9 can we type the following:

```python
print (5 = 9)
```
A few “math” examples to illustrate the point ... but first we need an operator to make the statement that one thing is equal to another ... only THEN can we determine whether the statement is true of false.

If we want Python to tell us if 5 is equal to 9 can we type the following:

```
print (5 = 9)
```

No, because `=` is the assignment operator, and this statement says assign the number 9 to the number 5 ... that doesn’t make sense ... technically, it is a syntax error because you are trying to assign a value (9) to a literal.
A few “math” examples to illustrate the point ... but first we need an operator to make the statement that one thing is equal to another ... only THEN can we determine whether the statement is true or false.

If we want Python to tell us if 5 is equal to 9 can we type the following:

```
print (5 = 9)
```

If we want Python to tell us if 5 is equal to 9, we type the following:

```
print (5 == 9)
```
A few “math” examples to illustrate the point ... but first we need an operator to make the statement that one thing is equal to another ... only THEN can we determine whether the statement is true or false.

If we want Python to tell us if 5 is equal to 9, we type the following:

```
print (5 == 9)
```

The `==` sign is the equality operator.

Q: Is it a unary, binary, or ternary operator?
A few “math” examples to illustrate the point … but first we need an operator to make the statement that one thing is equal to another … only THEN can we determine whether the statement is true or false.

If we want Python to tell us if 5 is equal to 9, can we type the following:

```
print (5 == 9)
```

Q: What does this statement evaluate to?

Q: What does this code output to the screen?
A few “math” examples to illustrate the point

Q: What do each of these output to the screen?

```python
print(3 == 5 and 4 < 7)
print(3 != 5 and 4 < 7)
print(3 == 5 or 4 < 7)
print(not False)
print(3 == 5 or 4 > 7)
print(not True)
```
A few “math” examples to illustrate the point

**Q: What do each of these output to the screen?**

```python
print(3 == 5 and 4 < 7)  
print(3 != 5 and 4 < 7)  
print(3 == 5 or 4 < 7)   
print(not False)         
print(3 == 5 or 4 > 7)   
print(not True)
```

*and* outputs True only if both operands evaluate to true.
A few “math” examples to illustrate the point

Q: What do each of these output to the screen?

```
print(3 == 5 and 4 < 7)  # False
print(3 != 5 and 4 < 7)  # True
print(3 == 5 or 4 < 7)   # True
print(not False)         # True
print(3 == 5 or 4 > 7)   # True
print(not True)          # False
```

and outputs True only if both operands evaluate to true
A few “math” examples to illustrate the point

Q: What do each of these output to the screen?

```python
print(3 == 5 and 4 < 7)
print(3 != 5 and 4 < 7)
print(3 == 5 or 4 < 7)
print(not False)
print(3 == 5 or 4 > 7)
print(not True)
```

This is how you write “not equal to”
A few “math” examples to illustrate the point

**Q: What do each of these output to the screen?**

```python
print(3 == 5 and 4 < 7)
p
print(3 != 5 and 4 < 7)
p
print(3 == 5 or 4 < 7)
p
print(not False)
p
print(3 == 5 or 4 > 7)
p
print(not True)
p
```

**and** outputs True only if both operands evaluate to true.
A few “math” examples to illustrate the point

Q: What do each of these output to the screen?

```
print(3 == 5 and 4 < 7)  # and outputs True only if both operands evaluate to true
print(3 != 5 and 4 < 7)
print(3 == 5 or 4 < 7)
print(not False)
print(3 == 5 or 4 > 7)
print(not True)
```
A few “math” examples to illustrate the point

Q: What do each of these output to the screen?

```python
print(3 == 5 and 4 < 7)
print(3 != 5 and 4 < 7)
print(3 == 5 or 4 < 7)
print(not False)
print(3 == 5 or 4 > 7)
print(not True)
```

or outputs True if AT LEAST one of the operands is true
Conditional

A few “math” examples to illustrate the point

Q: What do each of these output to the screen?

```
print(3 == 5 and 4 < 7)
print(3 != 5 and 4 < 7)
print(3 == 5 or 4 < 7) # or outputs True if AT LEAST one of the operands is true
print(not False)
print(3 == 5 or 4 > 7)
print(not True)
```
Conditional

A few “math” examples to illustrate the point

Q: What do each of these output to the screen?

```python
print(3 == 5 and 4 < 7)
p = print(3 != 5 and 4 < 7)
p = print(3 == 5 or 4 < 7)
p = print(3 == 5 or 4 > 7)
print(not False)
not outputs true if the single operand is False
print(not True)
```
A few “math” examples to illustrate the point

Q: What do each of these output to the screen?

```python
print(3 == 5 and 4 < 7)
print(3 != 5 and 4 < 7)
print(3 == 5 or 4 < 7)
print(3 == 5 or 4 > 7)
print(not False)
print(not True)
```

`not` outputs true if the single operand is False

True
Conditional

A few “math” examples to illustrate the point

Q: What do each of these output to the screen?

```python
print(3 == 5 and 4 < 7)
print(3 != 5 and 4 < 7)
print(3 == 5 or 4 < 7)
print(not False)
print(3 == 5 or 4 > 7)
print(not True)
```
Conditional

A few “math” examples to illustrate the point

Q: What do each of these output to the screen?

```python
print(3 == 5 and 4 < 7)
print(3 != 5 and 4 < 7)
print(3 == 5 or 4 < 7)
print(not False)
print(3 == 5 or 4 > 7)
print(not True)
```

or outputs True if AT LEAST one of the operands is True
Conditional

A few “math” examples to illustrate the point

Q: What do each of these output to the screen?

```python
print(3 == 5 and 4 < 7)
print(3 != 5 and 4 < 7)
print(3 == 5 or 4 < 7)
print(not False)
print(3 == 5 or 4 > 7)
print(not True)
```

or outputs True if AT LEAST one of the operands is True

False
A few “math” examples to illustrate the point

Q: What do each of these output to the screen?

```python
print(3 == 5 and 4 < 7)
p = 3 != 5 and 4 < 7
print(p)
p = 3 == 5 or 4 < 7
print(p)
p = not False
print(p)
p = 3 == 5 or 4 > 7
print(p)
p = not True
print(p)
```
A few “math” examples to illustrate the point

Q: What do each of these output to the screen?

```python
print(3 == 5 and 4 < 7)  # Output: False
print(3 != 5 and 4 < 7)  # Output: True
print(3 == 5 or 4 < 7)   # Output: True
print(not False)         # Output: True
print(3 == 5 or 4 > 7)   # Output: True
print(not True)          # Output: False
```
A few “math” examples to illustrate the point

**Q: What do each of these output to the screen?**

```python
print(3 == 5 and 4 < 7)  →  False
print(3 != 5 and 4 < 7)  →  True
print(3 == 5 or 4 < 7)   →  True
print(not False)         →  True
print(3 == 5 or 4 > 7)   →  False
print(not True)          →  True
```
When evaluating Boolean expressions, expressions inside of parentheses are evaluated first

Q: What does the following output?

```python
print ( 1 == 6 or ( 6 < ( 3 % 5))
```
When evaluating Boolean expressions, expressions inside of parentheses are evaluated first.

Q: What does the following output?

```python
print ( 1 == 6 or ( 6 < ( 3 % 5)))
```
When evaluating Boolean expressions, expressions inside of parentheses are evaluated first

Q: What does the following output?

```
print ( 1 == 6 or ( 6 < ( 3 % 5)))
```

```
print ( 1 == 6 or ( 6 < ( 3    )))
```
When evaluating Boolean expressions, expressions inside of parentheses are evaluated first

Q: What does the following output?

```python
print ( 1 == 6 or ( 6 < ( 3 % 5)))
print ( 1 == 6 or ( 6 < ( 3 )))
```
**Conditional**

When evaluating Boolean expressions, expressions inside of parentheses are evaluated first.

Q: What does the following output?

```python
print (1 == 6 or (6 < (3 % 5)))
print (1 == 6 or (6 < (3)))
print (1 == 6 or (False))
```
When evaluating Boolean expressions, expressions inside of parentheses are evaluated first

Q: What does the following output?

```
print ( 1 == 6 or ( 6 < ( 3 % 5)))
print ( 1 == 6 or ( 6 < ( 3      )))
print ( 1 == 6 or (     False   ))
```
When evaluating Boolean expressions, expressions inside of parentheses are evaluated first.

Q: What does the following output?

```
print ( 1 == 6 or ( 6 < ( 3 % 5)))
print ( 1 == 6 or ( 6 < ( 3 ) ))
print ( 1 == 6 or ( False ) )
print ( False )
```
When evaluating Boolean expressions, expressions inside of parentheses are evaluated first

Q: What does the following output?

```
print ( 1 == 6 or ( 6 < ( 3 % 5)))
print ( 1 == 6 or ( 6 < ( 3 )))
print ( 1 == 6 or (   False     ))
print (            False         )
```
Conditional

When evaluating Boolean expressions, expressions inside of parentheses are evaluated first

Q: What does the following output?

```python
print ( 1 == 6 or ( 6 < ( 3 % 5)))
print ( 1 == 6 or ( 6 < ( 3   )))
print ( 1 == 6 or (   False     ))
print (            False         )
```

Take home questions:

- How many checks does the or operator perform?
- How many checks does the and operator perform?
- How many checks does the not operator perform?
Conditional

Back to our original task ... writing programs that can reason about complex statements

Goal: You want to write a program that recommends what clothing and accessory items to wear
Back to our original task ... writing programs that can reason about complex statements

Goal: You want to write a program that recommends what clothing and accessory items to wear

Q: Is it cloudy and rainy? If yes, print “wear coat and use umbrella”
Q: Is it cloudy or rainy? If yes, print “bring coat and umbrella just in case”
Conditional

Back to our original task ... writing programs that can reason about complex statements

Goal: You want to write a program that recommends what clothing and accessory items to wear

Q: Is it cloudy and rainy? If yes, print “wear coat and use umbrella”
Q: Is it cloudy or rainy? If yes, print “bring coat and umbrella just in case”

Today
isCloudy = True
isRaining = False

Tomorrow
isCloudy = True
isRaining = True
Conditional

Back to our original task ... writing programs that can reason about complex statements

Goal: You want to write a program that recommends what clothing and accessory items to wear

Q: Is it cloudy and rainy? If yes, print “wear coat and use umbrella”
Q: Is it cloudy or rainy? If yes, print “bring coat and umbrella just in case”

```python
isCloudy = True
isRaining = False
if (isCloudy and isRaining) :
    print(“wear coat and use umbrella”)

isCloudy = True
isRaining = True
if (isCloudy or isRaining) :
    print(“bring coat and umbrella just in case”)```
Conditional

Back to our original task ... writing programs that can reason about complex statements

Goal: You want to write a program that recommends what clothing and accessory items to wear

Q: Is it cloudy and rainy? If yes, print “wear coat and use umbrella”
Q: Is it cloudy or rainy? If yes, print “bring coat and umbrella just in case”

```
isCloudy = True
isRaining = False

if (isCloudy and isRaining) :
    print("wear coat and use umbrella")
```

```
isCloudy = True
isRaining = True

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

Back to our original task ... writing programs that can reason about complex statements

Goal: You want to write a program that recommends what clothing and accessory items to wear

Q: Is it cloudy and rainy? If yes, print “wear coat and use umbrella”
Q: Is it cloudy or rainy? If yes, print “bring coat and umbrella just in case”

```python
isCloudy = True
isRaining = False
if (isCloudy and isRaining) :
    print("wear coat and use umbrella")
```

```python
isCloudy = True
isRaining = True
if (isCloudy or isRaining) :
    print("bring coat and umbrella just in case")
```
Back to our original task ... writing programs that can reason about complex statements

Goal: You want to write a program that recommends what clothing and accessory items to wear

Q: Is it cloudy and rainy? If yes, print “wear coat and use umbrella”
Q: Is it cloudy or rainy? If yes, print “bring coat and umbrella just in case”

```python
isCloudy = True
isRaining = False
if (isCloudy and isRaining) :
    print(“wear coat and use umbrella”)

isCloudy = True
isRaining = True
if (isCloudy or isRaining) :
    print(“bring coat and umbrella just in case”)
```
Back to our original task ... writing programs that can reason about complex statements

Goal: You want to write a program that recommends what clothing and accessory items to wear

Q: Is it cloudy and rainy? If yes, print “wear coat and use umbrella”
Q: Is it cloudy or rainy? If yes, print “bring coat and umbrella just in case”

```python
isCloudy = True
isRaining = False
if (isCloudy and isRaining) :
    print(“wear coat and use umbrella”)

isCloudy = True
isRaining = True
if (isCloudy or isRaining) :
    print(“bring coat and umbrella just in case”)
```
On-the-board exercise

Write python code that:

1. Prompts a user to input a number
2. Prompts a user to input a second number
3. Prompts the user to input the summation of the two numbers
4. Based on the user’s input and the “real” summation, outputs either of the two statements:

   'You are a math wiz' is a False Statement
   'You are a math wiz' is a True Statement

Hints
• This can be written using only 4 lines of code
• Do NOT use \texttt{if} (which hasn’t been explained yet)
Take home exercise ...

Write python code that:

1. Prompts a user to input a number
2. Prompts a user to input a second number
3. Prompts the user to input the summation of the two numbers
4. Based on the user’s input and the “real” summation, outputs either of the two statements:

   'You are a math wiz' is a False Statement
   'You are a math wiz' is a True Statement

Task : Write a program that accomplishes the above task but uses if statement(s) as needed
Up next

Much more on selection