

Comparison Examples

```
>>>
>>> x = 5
>>> x
5
>>> x == 5
True
>>> x < 5
False
>>>
```

True and False are reserved words in Python

Coming up: Comparison Operators

Comparison Operators

Operation	Meaning	Example	Evaluates To
<	strictly less than	4 < 10	True
<=	less than or equal	5 <= 5	True
>	strictly greater than	4 > 10	False
>=	greater than or equal	4 >= 10	False
==	equal	7 == 7	True
!=	not equal	8 != 5	True

From: <http://docs.python.org/library/stdtypes.html#comparisons>

Works with float, int, long and some with Strings also:
'dan' == 'bob' evaluates to False

Coming up: Compound Boolean Operators

Compound Boolean Operators

- **AND**: both conditions in the compound boolean operation must be True in order for the result to be True.
 - (condition_1) and (condition_2)
- **OR**: one of the two conditions in the compound boolean operation must be True in order for the result to be True.
 - (condition_1) or (condition_2)

Coming up: Truth Table (and)

Truth Table (and)

X	Y	X and Y
False	False	False
False	True	False
True	False	False
True	True	True

Coming up: Truth Table (and)

Truth Table (and)

X	Y	Z	X and Y and Z
False	False	False	False
False	False	True	False
False	True	False	False
False	True	True	False
True	False	False	False
True	False	True	False
True	True	False	False
True	True	True	True

Coming up: And examples

And examples

```
>>>
>>> x = 5
>>> y = 10
>>>
>>> (x == 5) and (y >= 5)
True
>>>
>>>
>>> (x > 5) and (y > 5)
False
>>>
```

Coming up: Truth Table (or)

Truth Table (or)

X	Y	X or Y
False	False	False
False	True	True
True	False	True
True	True	True

Coming up: Truth Table (or)

Truth Table (or)

X	Y	Z	X or Y or Z
False	False	False	False
False	False	True	True
False	True	False	True
False	True	True	True
True	False	False	True
True	False	True	True
True	True	False	True
True	True	True	True

Coming up: Truth Table (and / or)

Truth Table (and / or)

X	Y	Z	X and Y or Z
False	False	False	False
False	False	True	True
False	True	False	False
False	True	True	True
True	False	False	False
True	False	True	True
True	True	False	True
True	True	True	True

Coming up: Precedence


Precedence

- Use parenthesis to set precedence
 - False and False or True → True
 - False and (False or True) → False


Coming up: The Not operator

The Not operator

cool → True



not(cool) → True



NOT: produces the logical opposite of the logical expression
not(condition)

not(True) → False not(False) → True

Coming up: Truth Table (not)

Truth Table (not)

X	not(X)
False	True
True	False

```

>>>
>>>
>>>
>>>
>>>
>>> x = 5
>>> y = 10
>>> (x > 0) and (y < 0)
False
>>> not( (x > 0) and (y < 0) )
True
>>> |
    
```

Coming up: Example

Example

- Lets create an example for racquetball simulation
- Rules:
 - Either players wins at 15
 - If the score is 0-7, the shutout condition is imposed and the games ends

Coming up: Program Operation

Program Operation

- **Branching** : a program taking one path (or branch) of code instead of another, based on a **boolean condition**
- **Repetition** : a program repeating a block of code, some number of times, based on a **boolean condition**

Coming up: "if" statement

"if" statement

```
if <boolean condition>:
    <block>
```

Condition: logical expression that evaluates to True or False

Block: code to be executed if the condition evaluates to True

Note: Indentation is required to define the **block**

Coming up: "if else" statement

"if else" statement

```
if <boolean condition>:
    <Block_1>
else:
    <Block_2>
```

Condition: logical expression that evaluates to True or False

Block_1: code to be executed if the condition evaluates to True

Block_2: code to be executed if the condition evaluates to False

Coming up: "if elif else" statement

“if elif else” statement

```
if <condition_1>:
    <Block_1>

elif <condition_2>:
    <Block_2>

elif <condition_n>:
    <Block_n>

else:
    <Block_n+1>
```

Block_1: code to be executed if condition_1 evaluates to True

Block_2: code to be executed if the condition_2 evaluates to True

Block_n: code to be executed if the condition_n evaluates to True

Block_n+1: code to be executed if no conditions are True

Coming up: Examples

Examples

- See ifElseElifExamples.py
-
- Note: In a compound if statement only ONE BLOCK is ever executed! They are mutually exclusive.
- Else and elif blocks are optional
- If there is no Else block then it is possible that NO BLOCKs will be executed

Coming up: “while” loops

“while” loops (Indefinite Loop)

```
while <condition>:
    <block>
```

Condition: logical expression that evaluates to True or False

Block: code to be **repeated** as long as the condition evaluates to True

Note: condition must contain some sentinel value that is changed inside the loop. Otherwise, you’ll have an infinite loop!

Coming up: While Example

While Example

```
whileLoopExample.py - /Users/dfleck/Documents/gmuwebsite/classes/cs11
# While loop basic example
def main():
    sentinel_value = 5

    while (sentinel_value > 0):
        print sentinel_value
        sentinel_value -= 1 # What happens if I forget this?

main()
|
```

```
>>> ===== RESTART =====
5
4
3
2
1
>>> |
```

Coming up: While – Guess my number example

While – Guess my number example

- Lets write a guessing game.
- Pseudocode:
 - Loop until correct number is found
 - Guess a number
 - Ask user for higher/lower or correct
 - If higher set the lowest possible to the current guess
 - If lower set the highest possible to the current guess
 - new guess is halfway between highest and lowest

Coming up: Creating a CLI menu

Creating a CLI menu

- CLI = Command Line Input
- Typical pseudocode:
 - set sentinel value to dummy value
 - Start loop
 - print menu
 - get user's menu choice
 - do if statement (doing nothing if user wants to quit, otherwise doing what the user wants)

Coming up: CLI menu example

CLI menu example

- Lets modify the coolness calculator to have a menu!

Coming up: Continue Statement

Continue Statement

The `continue` statement

```
continue_stat := "continue"
```

`continue` may only occur syntactically nested in a `for` or `while` loop, but not nested in a function or class definition or `finally` clause within that loop. It continues with the next cycle of the nearest enclosing loop.

When `continue` passes control out of a `try` statement with a `finally` clause, that `finally` clause is executed before really starting the next loop cycle.

- Continue says skip the rest of the loop statements, and start the next iteration of the loop

Coming up: Continue Example

Continue Example

- See `continueBreakExamples.py`

Coming up: Break Statement

Break Statement

The break statement

```
break_stmt ::= "break"
```

`break` may only occur syntactically nested in a `for` or `while` loop, but not nested in a function or class definition within that loop.

It terminates the nearest enclosing loop, skipping the optional `else` clause if the loop has one.

If a `for` loop is terminated by `break`, the loop control target keeps its current value.

When `break` passes control out of a `try` statement with a `finally` clause, that `finally` clause is executed before really leaving the loop.

- Break says exit the loop right now

Coming up: Break and Continue: Style

Break and Continue: Style

- Note: even though they are included in most programming languages, using lots of breaks/continues frequently leads to code that is hard to understand/debug. (Sometimes called spaghetti code because it's all mixed up and hard to follow the flow of control.)
- Use Break and Continue sparingly!

Coming up: Terminology / Concepts

Terminology / Concepts

- Sequential Operation
- Branching Operation
- Repetition Operation
- Boolean Logic
- Indefinite Loop
- Boolean Operator
- Comparison Operator
- Control Structure
- Mutually Exclusive

Coming up: Lab Exercises

Lab Exercises

- Acceptable resources for lab exercises (*presented in order of precedence*):
 - Lecture/Lab material
 - Textbook (or other Python books) & LIB/LAN
 - Blackboard forums (no specific code)
 - Internet (documented)
 - Study Groups (no code exchange)
 - GTA (lab instructor) / Professor
- **Use of any other resources is a violation of the GMU Honor Code**

Coming up: Programming Projects

Programming Projects

- Acceptable resources for lab exercises (*presented in order of precedence*):
 - Lecture/Lab material
 - Textbook (or other Python books) & LIB/LAN
 - Blackboard forums (no specific code) (* added)
 - GTA (lab instructor) / Professor
- **Use of any other resources is a violation of the GMU Honor Code**

End of presentation