# CMSC201
# Computer Science I for Majors

## Lecture 05 – Comparison Operators and Boolean (Logical) Operators

Prof. Katherine Gibson

# Last Class We Covered

- Expressions
- Python's operators
  - Including mod and integer division
- The order of operations
- Different variables types
  - How to cast to a type
- Constants (and why using them is important)

# Any Questions from Last Time?

# Today's Objectives

- To learn a bit about **`main()`**
- To learn more of Python's operators
  - Comparison operators
  - Logical operators
- To practice using these new operators
- To become more familiar with using Boolean variables

# Quick Note about `main()`

# `main()`

- In Lab 2, we introduced the code

  `def main():`

  – as the first line of code in our file

- **`main()`** is an example of a **function**
- We can use functions to organize our code

# Functions

- We'll cover functions in more detail later

- For now, think of them as something similar to a variable
  - Variables hold data
  - Functions hold code

# Calling `main()`

- With variables, we use the variable name to access the data they store

- We must do the same with functions like **`main()`**, using the function name to execute the code they store

# Using `main()` for Your Code

- For our purposes, use `main()` with your code from now on:

declaring our `main()` function

```
def main():
    class = int(input("What class is this? ")
    print(class, "is awesome!")

main()
```

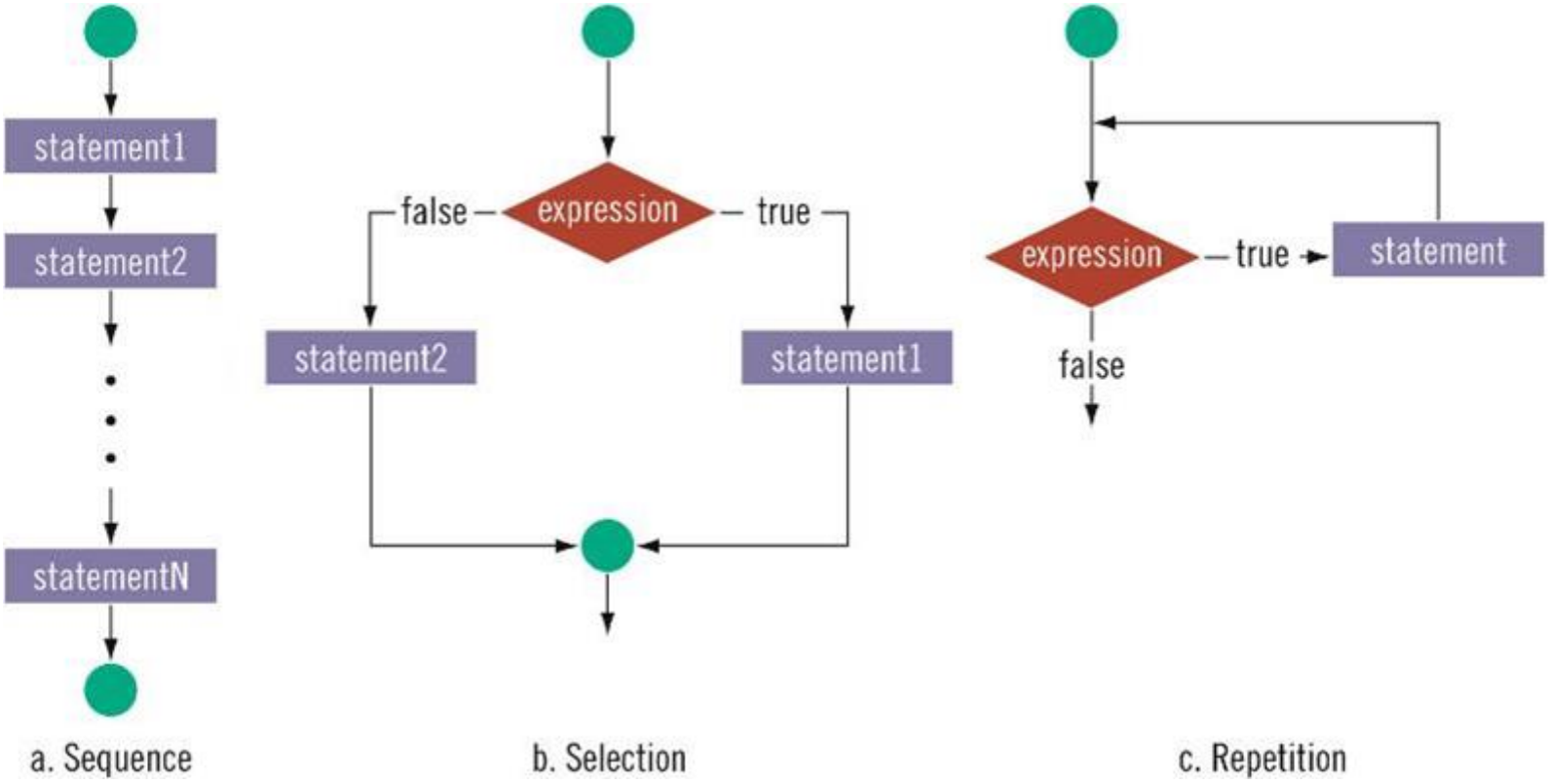
calling our `main()` function

# Review:
# Control Structures & Operators

# Control Structures

- What are the three control structures?
  - Sequential
  - Decision Making
    - Also known as “Selection”
  - Looping
    - Also known as “Repetition”
- We can also call a function

# Control Structures: Flowcharts



# Types of Operators in Python

- Arithmetic Operators ✓
- Comparison (Relational) Operators
- Assignment Operators ✓
- Logical Operators
- Bitwise Operators
- Membership Operators
- Identity Operators

focus of today's lecture

# Comparison Operators

# Vocabulary

- Comparison operators
- Relational operators
- Equality operators
  - Are all the same thing

- Include things like **>**, **>=**, **<**, **<=**, **==**, **!=**

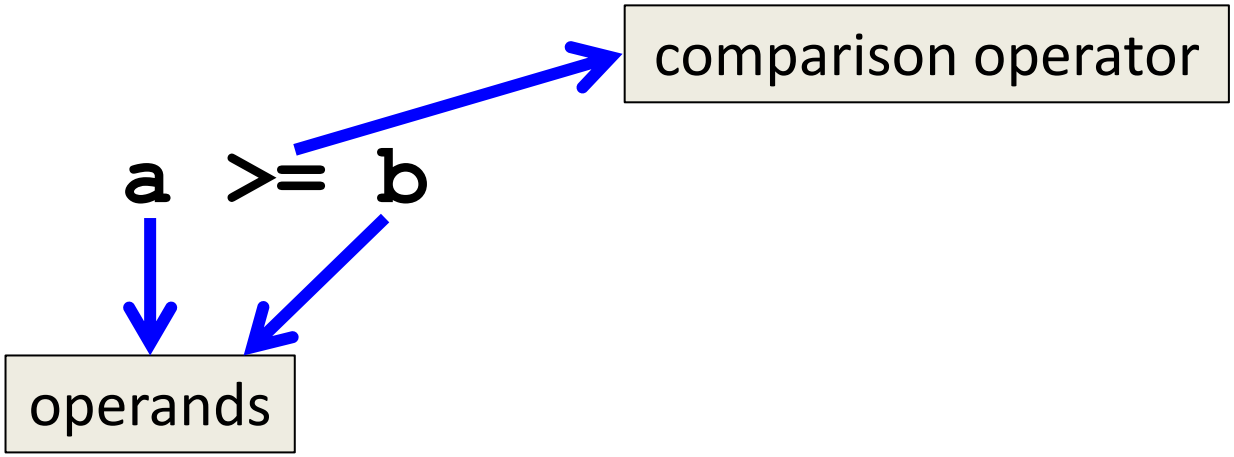
# Vocabulary

- Logical operators
- Boolean operators
  - Are the same thing

- Include **`and`**, **`or`**, and **`not`**

# Comparison Operators

- Always return a Boolean result
  - **`True`** or **`False`**
  - Indicates whether a relationship holds between their operands



# Comparison Examples

- What is the following comparison asking?

  `a >= b`

  – Is **`a`** greater than or equal to **`b`**?

  `a == b`

  – Is **`a`** equal to **`b`**?

# List of Operators

| Operator | Description |
|---|---|
| == | If the values of two operands are equal, then the condition becomes true. |
| != | If values of two operands are not equal, then condition becomes true. |
| <> | If values of two operands are not equal, then condition becomes true. |
| > | If the value of left operand is greater than the value of right operand, then condition becomes true. |
| < | If the value of left operand is less than the value of right operand, then condition becomes true. |
| >= | If the value of left operand is greater than or equal to the value of right operand, then condition becomes true. |
| <= | If the value of left operand is less than or equal to the value of right operand, then condition becomes true. |

**<>** is outdated use **!=** for "not equal to"

# List of Operators (Continued)

| Operation | Meaning |
|---|---|
| `<` | strictly less than |
| `<=` | less than or equal |
| `>` | strictly greater than |
| `>=` | greater than or equal |
| `==` | equal |
| `!=` | not equal |
| `is` | object identity |
| `is not` | negated object identity |

# Comparison Examples (Continued)

- What do these evaluate to if `a = 10` and `b = 20`?

  `a >= b`

  - Is `a` greater than or equal to `b`?
  - Is `10` greater than or equal to `20`?
  - **FALSE**

# Comparison Examples (Continued)

- What do these evaluate to if `a = 10` and `b = 20`?

  `a == b`

  - Is `a` equal to `b`?
  - Is `10` equal to `20`?
  - **FALSE**

# Comparison vs Assignment

- A common mistake is to use the assignment operator (=) in place of the relational (==)
  - This is a very common mistake to make!

What does **`a=b`** do? Sets **`a`** equal to **`b`**.

What does **`a==b`** do? Asks does **`a`** equal **`b`**?

This type of mistake will usually not trigger an error!

# Comparison Operator Examples

# Comparison Operators and Simple Data Types

- Examples:

| | |
|---|---|
| `8 < 15` | evaluates to **`True`** |
| `6 != 6` | evaluates to **`False`** |
| `2.5 > 5.8` | evaluates to **`False`** |
| `5.9 <= 7.5` | evaluates to **`True`** |

# "Value" of Boolean Variables

- When we discuss Boolean outputs, we think
  - **`True`** and **`False`**
- but we can also think of it in terms of
  - `1` and `0`

- **`True = 1`**
- **`False = 0`**

# Comparison Operation Examples

```
a = 10
b = 20
c = 30

bool1 = a == b
bool2 = c < b
bool3 = c != a

print(bool1, bool2, bool3)
```

**Prints:**

```
False False True
```

# More Comparison Operation Examples

```
a = 10
b = 20
c = 30

bool1 = int(a==a)
bool2 = a==a >= 10
bool3 = (a==a) + (b==b) + (c==c)

print(bool1, bool2, bool3)
```

**Prints:**

```
1 False 3
```

# Logical Operators

# Logical Operators

- There are three logical operators:
  - `and`
  - `or`
  - `not`
- They allow us to build more complex Boolean expressions
  - By combining simpler Boolean expressions

# Logical Operators – `and`

- Let's evaluate this expression

```
bool1 = a and b
```

| Value of `a` | Value of `b` | Value of `bool1` |
|---|---|---|
| | | |
| | | |
| | | |
| | | |

# Logical Operators – `and`

- Let's evaluate this expression

```
bool1 = a and b
```

| Value of `a` | Value of `b` | Value of `bool1` |
|---|---|---|
| `True` | `True` | |
| `True` | `False` | |
| `False` | `True` | |
| `False` | `False` | |

# Logical Operators – `and`

- Let's evaluate this expression

```
bool1 = a and b
```

| Value of `a` | Value of `b` | Value of `bool1` |
|---|---|---|
| `True` | `True` | `True` |
| `True` | `False` | `False` |
| `False` | `True` | `False` |
| `False` | `False` | `False` |

- For `a and b` to be `True`, both `a` and `b` must be true

# Logical Operators – `and`

- Two ways to write **`and`** expressions
  1. Explicitly use the keyword:

     ```
     3 > 2 and 2 > 1
     ```

  2. String them together, like in math:

     ```
     x > y > z
     ```

     – Evaluates to **`x > y and y > z`**

# Examples of **and**

```
a = 10
b = 20
c = 30

ex1 = a < b < c
ex2 = a < b and b < c
ex3 = a+b==c and b-10==a and c/3==a

print (ex1, ex2, ex3)
```

**Prints:**

```
True True True
```

# More Examples of `and`

```
a = 10
b = 20
c = 30

bool1 = a > b > c
bool2 = a == b > c
bool3 = a < b < c

print(bool1, bool2, bool3)
```

**Prints:**

```
False False True
```

# Logical Operators – `or`

- Let's evaluate this expression

```
bool1 = a or b
```

| Value of `a` | Value of `b` | Value of `bool1` |
|---|---|---|
| | | |
| | | |
| | | |
| | | |

# Logical Operators – `or`

- Let's evaluate this expression

```
bool1 = a or b
```

| Value of `a` | Value of `b` | Value of `bool1` |
|---|---|---|
| `True` | `True` | |
| `True` | `False` | |
| `False` | `True` | |
| `False` | `False` | |

# Logical Operators – `or`

- Let's evaluate this expression

```
bool1 = a or b
```

| Value of `a` | Value of `b` | Value of `bool1` |
|---|---|---|
| `True` | `True` | `True` |
| `True` | `False` | `True` |
| `False` | `True` | `True` |
| `False` | `False` | `False` |

- For `a or b` to be `True`, either `a` or `b` must be true

# Examples of `or`

```
a = 10
b = 20
c = 30

ex1 = a > b or c < b
ex2 = a + b <= c + 1 or b > c
ex3 = a == c or b + 10 <= a or c/3 == a

print (ex1, ex2, ex3)
```

**Prints:**

```
False True True
```

# Logical Operators – `not`

- Let's evaluate this expression

```
bool1 = not a
```

| Value of `a` | Value of `bool1` |
|---|---|
| `True` | `False` |
| `False` | `True` |

- `not a` returns the opposite Boolean value of a

# Complex Expressions

- We can put multiple operators together!

```
bool1 = a and (b or c)
```

- What does Python do first?
  - Computes **`(b or c)`**
  - Computes the **`and`** with **`a`** and the result

# Complex Expression Example

```
bool1 = a and (b or c)
```

| Value of a | Value of b | Value of c | Value of bool1 |
|---|---|---|---|
| True | True | True | True |
| True | True | False | True |
| True | False | True | True |
| True | False | False | False |
| False | True | True | False |
| False | True | False | False |
| False | False | True | False |
| False | False | False | False |

# “Short Circuit” Evaluation

# Short Circuit Evaluation

- "**`and`**" statements short circuit as soon as an expression evaluates to **`False`**

- "**`or`**" statements short circuit as soon as an expression evaluates to **`True`**

# Short Circuiting – `and`

- Notice that in the expression:

  ```
  bool1 = a and (b or c)
  ```

- If `a` is `False`
- The rest of the expression doesn't matter

- Python will realize this, and if `a` is false won't bother with the rest of the expression

# Short Circuiting – `or`

- Notice that in the expression:

  ```
  bool1 = a or (b or c)
  ```

- If `a` is `True`
- The rest of the expression doesn't matter

- Python will realize this, and if `a` is true won't bother with the rest of the expression

# More Practice

- Given:

  a = 4
  b = 5
  c = 6
  d = True
  e = False

```
bool1 = d and (a > b)
        False
bool2 = (not d) or (b != c)
        True
bool3 = (d and (not e)) or (a > b)
        True
bool4 = (a%b==2) and ((not d) or e)
        False
```

# More More Practice

- Given:

  a = 4
  b = 5
  c = 6
  d = True
  e = False

```
bool1 = (d + d) >= 2 and (not e)
        True
bool2 = (not e) and (6*d == 12/2)
        True
bool3 = (d or (e)) and (a > b)
        False
```

# Numbers and Booleans

- Python accepts anything that is non-zero as **`True`**
  - There are some exceptions, but we'll get into those later
- So technically you can use any integer as a Boolean expression

# Decision Making

- So, why do we care about comparison operators and logical operators so much?

Answer: Next Class

# Announcements

- Your Lab 3 is meeting normally this week!
  - Make sure you attend your correct section
- Homework 2 is out
  - Due by Tuesday (Sept 15th) at 8:59:59 PM
- Homeworks are on Blackboard
  - Weekly Agendas are also on Blackboard