CSCI 141

Lecture 8:
Conditionals, continued:
nested and chained conditionals
Happenings

Tech Talk: Sea-Bird Scientific  Monday, October 14th 5:00-6:00 PM CF 110
  - Brian Daugherty, Manager of Software Engineering, presents on technical interviews and what a hiring manager looks for in applicants

Tech Talk: Facebook  Thursday, October 17th
  - 4:00-5:00 PM CF 115, Heidi Young: Women Leader in Tech, co-sponsored by the Association of Women in Computing and Society for Women Engineers
    - Presentation on being a female leader in the tech field, with Q&A
  - 5:00-6:00 PM CF 115, Facebook Tech Talk
    - Presentation on Facebook technology, careers, and opportunities, with Q&A
NEED HELP CHOOSING A MAJOR?

CHOOSING A MAJOR WORKSHOP

October 15
3 - 4:30pm, HH 233

November 13
3 - 4:30pm, HH 233

Complete interactive assessments
Talk with advisors and peers
Explore interests and career connections

Co-sponsored by the Academic Advising Center
and the Career Services Center

Academic Advising Center | wwu.edu/advising
Career Services Center | wwu.edu/careers

WWU is an equal opportunity institution.
For disability accommodation, please call (360) 650-3850.
One week advance notice appreciated.
NEED HELP DECLARING YOUR MAJOR?

DECLARING A MAJOR DROP-IN LAB

October 23
3 - 4:30pm, HH 233

November 14
3 - 4:30pm, HH 233

Learn about the application process
Receive hands-on help from advisors
Connect with department resources

Advice on the CS major?
Talk to Mary Hall
hallm22@wwu.edu
CF 459
Announcements
Announcements

• A2 is due Tuesday night
Goals

• Understand the behavior of the equality comparison operators (==, !=) on non-numeric types.

• Know how to use an if statement to conditionally execute a block of code.

• Know how to use an if/else statement to choose which of two code blocks to execute.

• Understand how conditional statements can be nested to make decisions among more than two possibilities.

• Know how to use chained conditionals (if/elif/else)
Last time: Equality Comparisons

• The operators == and != check whether two values are equal or not.

• Unlike some operators (e.g., //), the concept of equality has meaning for some non-numeric types:

  4 == 5
  "abc" == "bcd"
  "abc" == "abc"
  type(4) == type(5)
  5.0 == 5
The operators `==` and `!=` check whether two values are equal or not.

Unlike some operators (e.g., `//`), the concept of equality has meaning for some non-numeric types:

```
4 == 5          => False
"abc" == "bcd"
"abc" == "abc"
type(4) == type(5)
5.0 == 5
```
Last time: Equality Comparisons

• The operators == and != check whether two values are equal or not.

• Unlike some operators (e.g., //), the concept of equality has meaning for some non-numeric types:

  4 == 5 => False
  "abc" == "bcd" => False
  "abc" == "abc"
  type(4) == type(5)
  5.0 == 5
Last time:
Equality Comparisons

• The operators == and != check whether two values are equal or not.

• Unlike some operators (e.g., //), the concept of equality has meaning for some non-numeric types:

  4 == 5                      => False
  "abc" == "bcd"              => False
  "abc" == "abc"              => True
  type(4) == type(5)          => True
  5.0 == 5
The operators $==$ and $!=$ check whether two values are equal or not.

Unlike some operators (e.g., //), the concept of equality has meaning for some non-numeric types:

$4 == 5$ => False
$"abc" == "bcd"$ => False
$"abc" == "abc"$ => True
$type(4) == type(5)$ => True
$5.0 == 5$
Last time:
Equality Comparisons

• The operators == and != check whether two values are equal or not.

• Unlike some operators (e.g., //), the concept of equality has meaning for some non-numeric types:

  4 == 5   => False
  "abc" == "bcd"  => False
  "abc" == "abc"  => True
  type(4) == type(5) => True
  5.0 == 5  => True
Equality Comparisons

Lightning round!

True or False?
Equality Comparisons

Lightning round!

10 == 4 + 6

True or False?
Equality Comparisons

Lightning round!

10 == 4 + 6

True or False?
Equality Comparisons

Lightning round!

10 == 4 + 6

"abc" == "ab" + "c"

True or False?
Equality Comparisons

Lightning round!

\[ 10 == 4 + 6 \]

\[ "abc" == "ab" + "c" \]

True or False?
Equality Comparisons

Lightning round!

10 == 4 + 6

"abc" == "ab" + "c"

'abc' == "abc"

True or False?
Equality Comparisons

Lightning round!

\[ 10 \equiv 4 + 6 \]

"abc" \equiv "ab" + "c"

'abc' \equiv "abc"

True or False?
Equality Comparisons

Lightning round!

10 == 4 + 6

"abc" == "ab" + "c"

'abc' == "abc"

"Scott" == "scott"

True or False?
Equality Comparisons

Lightning round!

10 == 4 + 6
"abc" == "ab" + "c"
'abc' == "abc"
"Scott" == "scott"
(4+3 > 5) == (1.0 > 4)

True or False?
Equality Comparisons

Lightning round!

10 == 4 + 6

"abc" == "ab" + "c"

'abc' == "abc"

"Scott" == "scott"

(4+3 > 5) == (1.0 > 4)

True or False?
Equality Comparisons

Lightning round!

10 == 4 + 6
"abc" == "ab" + "c"
'abc' == "abc"
"Scott" == "scott"

(4+3 > 5) == (1.0 > 4)

int(5.6) != int(5.1)
Equality Comparisons

Lightning round!

10 == 4 + 6 => True
"abc" == "ab" + "c"
'abc' == "abc"
"Scott" == "scott"
(4+3 > 5) == (1.0 > 4)

int(5.6) != int(5.1)
Equality Comparisons

Lightning round!

10 == 4 + 6 => True

"abc" == "ab" + "c" => True

'abc' == "abc"

"Scott" == "scott"

(4+3 > 5) == (1.0 > 4)

int(5.6) != int(5.1)
Equality Comparisons

Lightning round!

10 == 4 + 6 => True

"abc" == "ab" + "c" => True

'abc' == "abc" => True

"Scott" == "scott"

(4+3 > 5) == (1.0 > 4)

int(5.6) != int(5.1)

True or False?
Equality Comparisons

Lightning round!

10 == 4 + 6 => True

"abc" == "ab" + "c" => True

'abc' == "abc" => True

"Scott" == "scott" => False

(4+3 > 5) == (1.0 > 4)

int(5.6) != int(5.1)
Equality Comparisons

Lightning round!

10 == 4 + 6 => True

"abc" == "ab" + "c" => True

'abc' == "abc" => True

"Scott" == "scott" => False

(4+3 > 5) == (1.0 > 4) => False

int(5.6) != int(5.1)
Equality Comparisons

Lightning round!

10 == 4 + 6 => True
"abc" == "ab" + "c" => True
'abc' == "abc" => True
"Scott" == "scott" => False
(4+3 > 5) == (1.0 > 4) => False
int(5.6) != int(5.1) => False
Last time: if statement

```python
if is_raining:
    print("You should wear a raincoat!")
```
if keyword

```python
if is_raining:
    print("You should wear a raincoat!")
```
Last time: *if* statement

```python
if is_raining:
    print("You should wear a raincoat!")
```
Last time: if statement

```python
if keyword a boolean expression (the condition) a colon:

if is_raining:
    print("You should wear a raincoat!")
```
Last time: **if statement**

**if keyword**

- **a boolean expression (the condition)**

  `if is_raining:
  print("You should wear a raincoat!")`

**a colon:**

**an indented code block:** one or more statements to be executed if the boolean expression evaluates to **True**
if is_raining:
    print("Wear a raincoat!")
else:
    print("Don’t wear a raincoat!")
Last time: if statement with an else clause

```python
if is_raining:
    print("Wear a raincoat!"")
else:
    print("Don’t wear a raincoat!"")
```

- **if** keyword
- A boolean expression (the condition)
- A colon:
- An indented code block to be executed if the condition evaluates to True
if is_raining:
    print("Wear a raincoat!")
else:
    print("Don’t wear a raincoat!")
if is_raining:
    print("Wear a raincoat!")
else:
    print("Don’t wear a raincoat!")
wwu_founded = 1893
if (wwu_founded // 1000) < 1:
    wwu_founded = "eighteen ninety three"
if type(wwu_founded) == type("some text"):
    print("WWU was founded in", wwu_founded)
else:
    print("Year founded: ", wwu_founded)

Demo: seeing how Thonny executes code using Debug mode.
QOTD

• Which of the following expressions could fill in the blank below to make the following program print oolong?

```python
if (True and (________ or not True)) and not False:
    print("green")
else:
    print("oolong")
```

True not True 3 == 4
False not False 17 % 2 == 1
QOTD

• Which of the following programs are equivalent to the reference program? In other words, which programs have exactly the same output regardless of the value of \( a \)?

Reference Program:

```python
if (a < 0) == True:
    print(0)
else:
    if a >= 0:
        print(a)
```
QOTD

• Which of the following programs are equivalent to the reference program? In other words, which programs have exactly the same output regardless of the value of \( a \)?

Reference Program:

```python
if (a < 0) == True:
    print(0)
else:
    if a >= 0:
        print(a)
```

Prints 0 if \( a \) is 0 or less
Prints \( a \) if \( a \) is positive
QOTD

• Which of the following programs are equivalent to the reference program? In other words, which programs have exactly the same output regardless of the value of \( a \)?

Reference Program:

```python
if (a < 0) == True:
    print(0)
else:
    if a >= 0:
        print(a)

Prints 0 if \( a \) is 0 or less
Prints \( a \) if \( a \) is positive
```

Program 1:

```python
if a < 0:
    print(0)
else:
    print(a)
```

Prints 0 if \( a \) is 0 or less
Prints \( a \) if \( a \) is positive
Which of the following programs are equivalent to the reference program? In other words, which programs have exactly the same output regardless of the value of \( a \)?

Reference Program:

```
if (a < 0) == True:
    print(0)
else:
    if a >= 0:
        print(a)
```

Program 2:

```
if (a < 0) == True:
    print(0)
print(a)
```

Prints 0 if \( a \) is 0 or less
Prints \( a \) if \( a \) is positive
• Which of the following programs are equivalent to the reference program? In other words, which programs have exactly the same output regardless of the value of a?

Reference Program:

```python
if (a < 0) == True:
    print(0)
else:
    if a >= 0:
        print(a)
```

Programs 3:

```python
if (a > 0) == True:
    print(a)
else:
    print(0)
```

Prints 0 if a is 0 or less
Prints a if a is positive
QOTD

• Which of the following programs are equivalent to the reference program? In other words, which programs have exactly the same output regardless of the value of a?

Reference Program:

```python
if (a < 0) == True:
    print(0)
else:
    if a >= 0:
        print(a)
```

Prints 0 if a is 0 or less
Prints a if a is positive

Program 4:

```python
if (a < 0) == True:
    print(0)
if a >= 0:
    print(a)
```
Demo:
Get is_raining from the user
Demo:

Get `is_raining` from the user

- Update `ifelse.py` to ask the user whether it’s raining, and set the `is_raining` bool accordingly.
Nested Conditionals

If/else lets you choose between two options.

What if there are more than two possibilities?
Nested Conditionals

If/else lets you choose between two options.

What if there are more than two possibilities?

```python
# assume x and y are numbers
if x < y:
    if y > x:
        print("x is less than y")
```
Nested Conditionals

If/else lets you choose between two options.

What if there are more than two possibilities?

# assume x and y are numbers
if x < y:
   print("x is less than y")
else:
Nested Conditionals

If/else lets you choose between two options.

What if there are more than two possibilities?

```python
# assume x and y are numbers
if x < y:
    print("x is less than y")
else:
    an indented code block
    containing one or more
    statements
```
Nested Conditionals

If/else lets you choose between two options.

What if there are more than two possibilities?

# assume x and y are numbers
if x < y:
    print("x is less than y")
else:
    if x > y:
        print("x is greater than y")
    else:
        print("x and y must be equal")
Nested Conditionals

If/else lets you choose between two options.

What if there are more than two possibilities?

```python
# assume x and y are numbers
if x < y:
    print("x is less than y")
else:
    if x > y:
        print("x is greater than y")
    else:
        print("x and y must be equal")
```

the inner if statement is the indented code block for the else clause of the outer if statement.
Nested Conditionals

If/else lets you choose between two options. What if there are more than two possibilities?

```python
# assume x and y are numbers
if x < y:
    print("x is less than y")
else:
    if x > y:
        print("x is greater than y")
    else:
        print("x and y must be equal")
```

*Note:* the conditions still have to be boolean expressions (i.e., they evaluate to True or False)

the inner if statement is the indented code block for the else clause of the outer if statement.
Nested Conditionals

How many comparison operators (\(<\), \(>\)) are evaluated by the following code?

```python
x = 4
y = 5
if x < y:
    print("x is less than y")
else:
    if x > y:
        print("x is greater than y")
    else:
        print("x and y must be equal")
```

A. 0  
B. 1  
C. 2  
D. 3
Demo

**Task:** Write a program to ask the user for their 141 section number and print out when their lab section happens.

```python
>>> %Run section_times.py
Enter your CSCI 141 section number: 20892
Your lab is on Tuesday from 10 - 12.
```
Chained Conditionals: Demo
Chained Conditionals: Demo

- sections.py: with chained if/else statements
- sections_elif.py: with if/elif/else
- sections_refactored.py: refactored to set variables then call print once
- sections_refactored.py: with feature to check for conflicts with lab
Chained Conditionals: Syntax

```python
if isRaining and not isWindy:
    print("Bring an umbrella!")
elif isRaining and isWindy:
    print("Wear a raincoat!")
else:
    print("No rain gear needed!")
```
if isRaining and not isWindy:
    print("Bring an umbrella!")
elif isRaining and isWindy:
    print("Wear a raincoat!")
else:
    print("No rain gear needed!")
Chained Conditionals: Syntax

`elif` keyword

```python
if isRaining and not isWindy:
    print("Bring an umbrella!"")
elif isRaining and isWindy:
    print("Wear a raincoat!")
else:
    print("No rain gear needed!")
```

An indented code block to be executed if:

- **none** of the above conditions was True
- **and** this `elif`’s condition is True

An indented code block to be executed if the **none** of the above conditions was true
Chained Conditionals: Syntax

**elif** keyword

```python
if isRaining and not isWindy:
    print("Bring an umbrella!"")
elif isRaining and isWindy:
    print("Wear a raincoat!")
else:
    print("No rain gear needed!")
```

- **elif** keyword
- An indented code block to be executed if:
  - **none** of the above conditions was True
  - **and** this **elif**’s condition is True

*This behaves exactly like nesting an if inside each else*

An indented code block to be executed if the **none** of the above conditions was true
Chained Conditionals: Syntax

An indented code block to be executed if:

- **none** of the above conditions was True
- **and** this elif’s condition is True

**elif** keyword

```python
if isRaining and not isWindy:
    print("Bring an umbrella!")
elif isRaining and isWindy:
    print("Wear a raincoat!")
else:
    print("No rain gear needed!")
```

An indented code block to be executed if the **none** of the above conditions was true

(this behaves exactly like nesting an if inside each else)

(the else clause is optional)
Exercise

• Write out pseudocode (the sequence of steps) to solve the challenge problem on A2:

  • Given three numbers, a, b, and c, print the median, using only techniques we’ve covered in class.
Next time: Repetition
Repetition
Repetition
Repetition
Repetition
Repetition
Repetition
Repetition