Lecture 12

Topics to be covered:

Chapter 8:

- Section 8.5 Boolean operators and expressions
- Section 8.6 Membership and identity operators
- Section 8.7 Order of evaluation
- Section 8.8 Code blocks and indentation
- Section 8.9 Conditional expressions
- Section 8.10 Additional practice: Tweet decoder

Nested if-else statements

Nested `if-else` statements

A branch's statements can include any valid statements, including another `if-else` statement, which are known as *nested* `if-else` *statements*.

```
if grade >= 90:
   if grade < 93:
      print(“that’s an A-”)
   elif grade >= 97:
      print(“that’s an A+”)
   else:
      print(“that’s an A”)
else:
   print(“not an A grade”)
```

Nested if-else statements

Nested `if-else` statements

A branch's statements can include any valid statements, including another `if-else` statement, which are known as *nested* `if-else` *statements*.

if grade = 78

```
if grade >= 90:
   if grade < 93:
      print("that's an A-")
   elif grade >= 97:
      print("that's an A+")
   else:
      print("that's an A")
else:
   print("not an A grade")
```

Nested if-else statements

Nested `if-else` statements

A branch's statements can include <u>any valid statements</u>, including another `if-else` statement, which are known as *nested* `if-else` *statements*.

if grade = 78

```
if grade >= 90:
   if grade < 93:
      print("that's an A-")
   elif grade >= 97:
      print("that's an A+")
   else:
      print("that's an A")
else:
   print("not an A grade")
```

Nested if-else statements

Nested `if-else` statements

A branch's statements can include <u>any valid statements</u>, including another `if-else` statement, which are known as *nested* `if-else` *statements*.

if grade = 78

```
if grade >= 90:
   if grade < 93:
      print("that's an A-")
   elif grade >= 97:
      print("that's an A+")
   else:
      print("that's an A")
else:
   print("not an A grade")
```

Nested if-else statements

Nested `if-else` statements

A branch's statements can include <u>any valid statements</u>, including another `if-else` statement, which are known as *nested* `if-else` *statements*.

if grade = 78

```
if grade >= 90:
   if grade < 93:
      print("that's an A-")
   elif grade >= 97:
      print("that's an A+")
   else:
      print("that's an A")
else:
   print("not an A grade")
```

not an A grade

Nested if-else statements

Nested `if-else` statements

A branch's statements can include <u>any valid statements</u>, including another `if-else` statement, which are known as *nested* `if-else` *statements*.

if grade = 95

```
if grade >= 90:
   if grade < 93:
      print("that's an A-")
   elif grade >= 97:
      print("that's an A+")
   else:
      print("that's an A")
else:
   print("not an A grade")
```

Nested if-else statements

Nested `if-else` statements

A branch's statements can include <u>any valid statements</u>, including another `if-else` statement, which are known as *nested* `if-else` *statements*.

if grade = 95

```
if grade >= 90:
   if grade < 93:
      print("that's an A-")
   elif grade >= 97:
      print("that's an A+")
   else:
      print("that's an A")
else:
   print("not an A grade")
```

Nested if-else statements

Nested `if-else` statements

A branch's statements can include any valid statements, including another `if-else` statement, which are known as *nested* `if-else` *statements*.

`if grade = 95`

```
if grade >= 90:
    if grade < 93:
        print("that's an A-")
    elif grade >= 97:
        print("that's an A+")
    else:
        print("that's an A")
else:
    print("not an A grade")
```

Nested if-else statements

Nested `if-else` statements

A branch's statements can include any valid statements, including another `if-else` statement, which are known as *nested* `if-else` *statements*.

if grade = 95

```
if grade >= 90:
   if grade < 93:
      print("that's an A-")
   elif grade >= 97:
      print("that's an A+")
   else:
      print("that's an A")
else:
   print("not an A grade")
```

Nested if-else statements

Nested `if-else` statements

A branch's statements can include any valid statements, including another `if-else` statement, which are known as *nested* `if-else` *statements*.

if grade = 95

```
if grade >= 90:
   if grade < 93:
      print("that's an A-")
   elif grade >= 97:
      print("that's an A+")
→  else:
      print("that's an A")
else:
   print("not an A grade")
```

Nested if-else statements

Nested `if-else` statements

A branch's statements can include any valid statements, including another `if-else` statement, which are known as *nested* `if-else` *statements*.

if grade = 95

```
if grade >= 90:
   if grade < 93:
      print("that's an A-")
   elif grade >= 97:
      print("that's an A+")
   else:
   →  print("that's an A")
else:
   print("not an A grade")
```

that's an A

Multiple `if` statements

Multiple `if` statements

Consider the following code fragment:

```
if num >= 10:
   print("A")
if num >= 0:
   print("B")
if num < 0:
   print("C")
if num < -10:
   print("D")
```

What would the program output if `num = 12`?

Multiple `if` statements

Multiple `if` statements

Consider the following code fragment:

```
if num >= 10:
   print("A")
if num >= 0:
   print("B")
if num < 0:
   print("C")
if num < -10:
   print("D")
```

What would the program output if `num = 12`?

A
B

Multiple `if` statements

Multiple `if` statements

Consider the following code fragment:

```
if num >= 10:
   print("A")
if num >= 0:
   print("B")
if num < 0:
   print("C")
if num < -10:
   print("D")
```

What would the program output if `num = 1`?

Multiple `if` statements

Multiple `if` statements

Consider the following code fragment:

```
if num >= 10:
   print("A")
if num >= 0:
   print("B")
if num < 0:
   print("C")
if num < -10:
   print("D")
```

What would the program output if `num = 1`?

B

Multiple `if` statements

Multiple `if` statements

Consider the following code fragment:

```
if num >= 10:
   print("A")
if num >= 0:
   print("B")
if num < 0:
   print("C")
if num < -10:
   print("D")
```

What would the program output if `num = -1`?

Multiple `if` statements

Multiple `if` statements

Consider the following code fragment:

```
if num >= 10:
   print("A")
if num >= 0:
   print("B")
if num < 0:
   print("C")
if num < -10:
   print("D")
```

What would the program output if `num = -1`? C

Multiple `if` statements

Multiple `if` statements

Consider the following code fragment:

```
if num >= 10:
   print("A")
if num >= 0:
   print("B")
if num < 0:
   print("C")
if num < -10:
   print("D")
```

What would the program output if `num = -12`?

Multiple `if` statements

Multiple `if` statements

Consider the following code fragment:

```
if num >= 10:
   print("A")
if num >= 0:
   print("B")
if num < 0:
   print("C")
if num < -10:
   print("D")
```

What would the program output if `num = -12`?

C
D

In-class Activity

In-class Activity

See exercises 1-2

Boolean operators and expressions

Booleans and Boolean operators

A Boolean refers to a value that is either True or False.
These two are constants in Python.

- we can assign a Boolean value by specifying True or False,

```
x = True
```

- an expression can evaluate to a Boolean value

```
y > 10
```

Boolean operators and expressions

`and` operator

The Boolean expression `a and b` is True if and only if both `a` and `b` are True.

Boolean operators and expressions

`and` operator

The Boolean expression `a and b` is `True` if and only if both `a` and `b` are `True`.

a	b	a and b
True	True	True
True	False	False
False	True	False
False	False	False

Boolean operators and expressions

`and` operator

The Boolean expression `a and b` is `True` if and only if both `a` and `b` are `True`.

a	b	a and b
True	True	True
True	False	False
False	True	False
False	False	False

Examples: assume that `a = 8` and `b = 3`, then the Boolean value of

1) `( a > 10 ) and ( b < 5 )` is `False`
2) `( a != 10 ) and ( b > 1 )` is `True`

Boolean operators and expressions

`or` operator

The Boolean expression `a or b` is False if and only if both `a` and `b` are False.

Boolean operators and expressions

`or` operator

The Boolean expression `a or b` is False if and only if both `a` and `b` are False.

a	b	a or b
True	True	True
True	False	True
False	True	True
False	False	False

Boolean operators and expressions

`or` operator

The Boolean expression `a or b` is False if and only if both `a` and `b` are False.

a	b	a or b
True	True	True
True	False	True
False	True	True
False	False	False

Examples: assume `a = 8` and `b = 3`, then the Boolean value of

1) `( a > 10 ) or ( b < 5 )` is True
2) `( a == 10 ) or ( b < 1 )` is False

Boolean operators and expressions

`not` operator

The Boolean expression `not a` is False when `a` is True, and is True when `a` is False.

Boolean operators and expressions

`not` operator

The Boolean expression `not a` is `False` when `a` is `True`, and is `True` when `a` is `False`.

a	not a
True	False
False	True

Boolean operators and expressions

`not` operator

The Boolean expression `not a` is False when `a` is True, and is True when `a` is False.

a	not a
True	False
False	True

Examples: assume `a = 8` and `b = 3`, then the Boolean value of

1) `not ( a > 10 )` is True
2) `not ( a * b > 20 )` is False

Boolean operators and expressions

Booleans and Boolean operators

Consider the following code fragment:

```
if letter == 'a' or letter == 'b':
  print("Help!")
elif letter == 'c' or letter == 'd':
  print("We are in trouble!")
else:
  print("We are good!")
```

Boolean operators and expressions

Booleans and Boolean operators

Consider the following code fragment:

```
if letter == 'a' or letter == 'b':
  print("Help!")
elif letter == 'c' or letter == 'd':
  print("We are in trouble!")
else:
  print("We are good!")
```

if `letter = 'a'`, then we will get:

Boolean operators and expressions

Booleans and Boolean operators

Consider the following code fragment:

```
if letter == 'a' or letter == 'b':
  print("Help!")
elif letter == 'c' or letter == 'd':
  print("We are in trouble!")
else:
  print("We are good!")
```

if `letter = 'a'`, then we will get:

```
Help!
```

Boolean operators and expressions

Booleans and Boolean operators

Consider the following code fragment:

```
if letter == 'a' or letter == 'b':
  print("Help!")
elif letter == 'c' or letter == 'd':
  print("We are in trouble!")
else:
  print("We are good!")
```

if `letter = 'c'`, then we will get:

Boolean operators and expressions

Booleans and Boolean operators

Consider the following code fragment:

```
if letter == 'a' or letter == 'b':
  print("Help!")
elif letter == 'c' or letter == 'd':
  print("We are in trouble!")
else:
  print("We are good!")
```

if `letter = 'c'`, then we will get:

```
We are in trouble!
```

Order of evaluation

Precedence rules

The order in which operators are evaluated in an expression is known as precedence of operators.

operator	description	Example
()	parentheses are evaluated first	(2+5*3) – (5/6+2*4)
+ - * / % // **	arithmetic operations next (in their order)	10-2**5 >= 10%7
< <= > >= == !=	then comparisons and membership operators	a > 9 and b in [1,2,3]
not	negation operator next	not (a > 9) or b == 2
and	conjunction (and) next	a >9 or a < 0 and b > 1
or	disjunction (or) last	a >9 or a < 0 and b > 1

Membership and identity operators

Membership operators: `in`/`not in`

Quite often we need to check is a value can be or cannot be found within a container, such as a list or dictionary.

`in` and `not in` operators, known as *membership operators*, can help us!

Order of evaluation

Precedence rules

Example: Let's evaluate the Boolean expression below for `g = 12`, `b = True`, and `a = 17`

```
g >= 90 or b and a > 100
```

Order of evaluation

Precedence rules

Example: Let's evaluate the Boolean expression below for `g = 12`, `b = True`, and `a = 17`

```
g >= 90 or b and a > 100
```

```
(g >= 90) or (b and a > 100)
```

Order of evaluation

Precedence rules

Example: Let's evaluate the Boolean expression below for `g = 12, b = True, and a = 17`

```
g >= 90 or b and a > 100
```

```
(g >= 90) or (b and a > 100)

    F      or  ( T and F)
```

Order of evaluation

Precedence rules

Example: Let's evaluate the Boolean expression below for `g = 12, b = True, and a = 17`

```
g >= 90 or b and a > 100
```

```
(g >= 90) or (b and a > 100)

    F      or  ( T and F)

     F     or      F

           F
```

In-class activity

In-class work:

Do exercises 3-5

Membership and identity operators

Membership operators: `in`/`not in`

Quite often we need to check is a value can be or cannot be found within a container, such as a list or dictionary.

`in` and `not in` operators, known as *membership operators*, can help us!

Example:

```
num = int(input("Enter an integer:"))
myContainer = [1,2,3,4,5,6,7]

if num in myContainer:
  print("Found it! It is in myContainer!")

else: print("Nope. It is not in myContainer.")
```

Membership and identity operators

Membership operators: `in`/`not in`

Example:

```
name = int(input("Enter a name:"))
MyNamesContainer = {
"Maria" : 23,
"Anna" : 19,
"Jack" : 5,
"Alex" : 12,
"John" : 18}

if name in myNamesContainer:
  print("Found it! It is corresponds to",
     MyNamesContainer[name])

else: print("No such name in the container.")
```

Membership and identity operators

Membership operators: `in`/`not in`

Example:

```
name = int(input("Enter a name:"))
MyNamesContainer = {
"Maria" : 23,
"Anna" : 19,
"Jack" : 5,
"Alex" : 12,
"John" : 18}

if name in myNamesContainer:
  print("Found it! It is corresponds to",
     MyNamesContainer[name])

else: print("No such name in the container.")
```

Note that the keys are matched, not the values!

Membership and identity operators

Identity operators: `is/is not`

Sometimes we want to determine whether two variables are the same object.

`is` and `is not` operators, known as *identity operators*, can help us out!

Identity operators return True only if the operands reference the same object (they do not compare object's values).

Membership and identity operators

Identity operators: `is/is not`

Example:

```
myContainer = [1,2,3,4,5,6,7]
otherContainer = [9,8,7,6,5,4,3,2,1]

a = myContainer
b = otherContainer
a = b

if a is myContainer:
  print("a is myContainter!")

elif a is otheContainer:
  print("a is otherContainter!")
```

Membership and identity operators

Identity operators: `is/is not`

Example:

```
myContainer = [1,2,3,4,5,6,7]
otherContainer = [9,8,7,6,5,4,3,2,1]

a = myContainer
b = otherContainer
a = b

if a is myContainer:
   print("a is myContainter!")

elif a is otheContainer:
    print("a is otherContainter!")

else: print("I have no idea that is a!")
```

In-class activity

In-class work:

See exercises 6-7