City University of New York (CUNY)

CUNY Academic Works

2019

Python if statements

Natalia Novak
Bronx Community College, City University of New York

How does access to this work benefit you? Let us know!

If-else

Topics to be covered:

- If-else branches (general)
- If-else statement
- Equality and relational operators
- More if-else

Additional topics:

- Boolean operators and expressions
- Membership and identity operators

If-else branches (general)

In many circumstances when we write a program we need the ability to check conditions and change the behavior of the program accordingly.

Selection statements or *conditional statements*, give us this ability.

If-else branches (general)

In many circumstances when we write a program we need the ability to check conditions and change the behavior of the program accordingly.

Selection statements or *conditional statements*, give us this ability.

Example: Let's look through the following code

```python
if my_class_average > 1:
  print("I passed the class! Hooray!")
else:
  print("Bummer! I will have to re-take this class!")
```

If-else branches (general)

Consider another code fragment:

```python
x = int(input("Enter an integer value:"))
y = int(input("Enter another integer value:"))

if x > y:
    a = x

if x < y:
    a = y
else:
    print("They are equal!")
```

If-else branches (general)

Consider another code fragment:

```
x = int(input("Enter an integer value:"))
y = int(input("Enter another integer value:"))

if x > y:
    a = x

if x < y:
    a = y
else:
    print("They are equal!")
```

conditions
(evaluated to a Boolean value: True or False)

If-else branches (general)

If we type the following commands in the Python shell, we will get the responses highlighted with blue

```
>>> 2==2
True
>>> 2<3
True
>>> 3>7
False
>>> 5>9 or 2<3
True
```

If-else statement

Let's write a program that will report the grade for the test, given a numeric score.

```python
test_score = float(input("Enter test score:"))
if test_score >= 90:
  print("This is an A grade!")
if 80 <= test_score < 90:
  print("This is a B grade!")
if 70 <= test_score < 80:
  print("This is a C grade!")
if 60 <= test_score < 70:
  print("This is a D grade!")
else: print("Unfortunately this is an F grade")
```

If-else statement

Let's write a program that will report the grade for the test, given a numeric score.

```python
test_score = float(input("Enter test score:"))
if test_score >= 90:
  print("This is an A grade!")
if 80 <= test_score < 90:
  print("This is a B grade!")
if 70 <= test_score < 80:
  print("This is a C grade!")
if 60 <= test_score < 70:
  print("This is a D grade!")
else: print("Unfortunately this is an F grade")
```

If-else statement

Let's write a program that will report the grade for the test, given a numeric score.

```python
test_score = float(input("Enter test score:"))
if test_score >= 90:
    print("This is an A grade!")
elif 80 <= test_score < 90:
    print("This is a B grade!")
elif 70 <= test_score < 80:
    print("This is a C grade!")
elif 60 <= test_score < 70:
    print("This is a D grade!")
else: print("Unfortunately this is an F grade")
```

only one branch will execute!

Equality and relational operators

Equality operators

An equality operator checks whether two operands' values are the same (**==**) or different (**!=**).
Note that equality is **==**, not just =.

Equality operators	Description	Example (assume x is 3)
==	a **==** b means a is equal to b	x == 3 is true x == 4 is false
!=	a **!=** b means a is not equal to b	x != 3 is false x != 4 is true

An expression evaluates to a *Boolean value*.
A Boolean is a type that has just two values: True or False.

Equality and relational operators

Relational operators

A relational operator checks how one <u>operand's</u> value relates to another, like being greater than.

Relational operators	Description	Example (assume x is 3)
<	a **<** b means a is less than b	x < 4 is true x < 3 is false
>	a **>** b means a is greater than b	x > 2 is true x > 3 is false
<=	a **<=** b means a is less than or equal to b	x <= 4 is true x <= 3 is true x <= 2 is false
>=	a **>=** b means a is greater than or equal to b	x >= 2 is true x >= 3 is true x >= 4 is false

Equality and relational operators

Operator chaining

Python supports *operator chaining*.

Example: a < b < c
determines whether **b** is greater-than **a** but less-than **c**.

Chaining performs comparisons left to right,
evaluating a < b first.
- If the result is true, then b < c is evaluated next.
- If the result of the first comparison a < b is false, then there is
- no need to continue evaluating the rest of the expression.

Equality and relational operators

In-class work: see the handout, problems 1-5

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*.

```python
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")
```

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 <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 <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")
```

that's an A

Multiple `if` statements

Consider the following code fragment:

```python
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

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

Consider the following code fragment:

```python
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

Consider the following code fragment:

```python
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

Consider the following code fragment:

```python
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

Consider the following code fragment:

```python
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

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

Consider the following code fragment:

```python
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

More if-else

In-class Activity

See exercises 6-7

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:

```python
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:

```python
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:

```python
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:

```python
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:

```python
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

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

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

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

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

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`

Boolean operators and expressions

In-class work

Exercises 8-10

Membership and identity operators

Membership operators: `in/not in`

Quite often we need to check is a <u>value can be</u> or <u>cannot be found</u> 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!")
```

Membership and identity operators

In-class work

Exercise 11

Code blocks and indentation

Consider the following code fragment:

```python
if a > 5:

    myString = input("Enter a word:")
    print(myString*a)


else:

    myNum = int(input("Enter an integer:"))
    print(myNum-a)

print("That's it!")
```

Code blocks and indentation

Consider the following code fragment:

```python
if a > 5:

    myString = input("Enter a word:")
    print(myString*a)


else:

    myNum = int(input("Enter an integer:"))
    print(myNum-a)

print("That's it!")
```

Code blocks and indentation

Consider the following code fragment:

```python
if a > 5:

    myString = input("Enter a word:")
    print(myString*a)



else:

    myNum = int(input("Enter an integer:"))
    print(myNum-a)

print("That's it!")
```

code blocks

Code blocks and indentation

Consider the following code fragment:

3-4 spaces
Tab: 3 spaces

```python
if a > 5:
    myString = input("Enter a word:")
    print(myString*a)

else:
    myNum = int(input("Enter an integer:"))
    print(myNum-a)

print("That's it!")
```

code blocks

Code blocks and indentation

Consider the following code fragment:

3-4 spaces
Tab: 3 spaces

```python
if a > 5:
    myString = input("Enter a word:")
    print(myString*a)

else:
    myNum = int(input("Enter an integer:"))
    print(myNum-a)

print("That's it!")
```

code blocks

Caution: be consistent!
Either use 4 spaces or a Tab (3 spaces)

Code blocks and indentation

Consider the following code fragment:

a = 3

```python
if a > 5:

    myString = input("Enter a word:")
    print(myString*a)


else:

    myNum = int(input("Enter an integer:"))
    print(myNum-a)

print("That's it!")
```

Code blocks and indentation

Consider the following code fragment:

$a = 3$

```python
if a > 5:

    myString = input("Enter a word:")
    print(myString*a)


else:

    myNum = int(input("Enter an integer:"))
    print(myNum-a)

print("That's it!")
```

Enter an integer: 10
7
That's it!

Code blocks and indentation

Consider the following code fragment:

```
                                    a = 6
if a > 5:

    myString = input("Enter a word:")
    print(myString*a)


else:

    myNum = int(input("Enter an integer:"))
    print(myNum-a)

print("That's it!")
```

Code blocks and indentation

Consider the following code fragment:

$a = 6$

```
if a > 5:

    myString = input("Enter a word:")
    print(myString*a)


else:

    myNum = int(input("Enter an integer:"))
    print(myNum-a)

print("That's it!")
```

Enter a word: my
mymymymymy
That's it!

Code blocks and indentation

Consider the following code fragment:

a = 4

```python
if a > 5:

myString = input("Enter a word:")
print(myString*a)


else:

myNum = int(input("Enter an integer:"))
print(myNum-a)

print("That's it!")
```

Code blocks and indentation

Consider the following code fragment:

a = 4

```python
if a > 5:

myString = input("Enter a word:")
print(myString*a)


else:

myNum = int(input("Enter an integer:"))
print(myNum-a)

print("That's it!")
```

Enter a word: ten
tentententen
Enter an integer: 20
16
That's it!

Code blocks and indentation

Consider the following code fragment:

`a = 4`

```python
if a > 5:

    myString = input("Enter a word:")
    print(myString*a)


else:

    myNum = int(input("Enter an integer:"))
    print(myNum-a)

print("That's it!")
```

DO NOT FORGET INDENTATION

Enter a word: ten
tentententen
Enter an integer: 20
16
That's it!

Code blocks and indentation

A conditional expression has the following form:

`<expr_t>` `if` `<condition>` `else` `<expr_when_f>`

Example:
```
print("A") if a < 10 else print("B")
```

Code blocks and indentation

A conditional expression has the following form:

```
<expr_t> if <condition> else <expr_when_f>
```

Example:
```
print("A") if a < 10 else print("B")
```

A conditional expression has three operands and thus is sometimes referred to as a *ternary operation*.

9.9 Conditional expressions

A conditional expression has the following form:

`<expr_t>` `if` `<condition>` `else` `<expr_when_f>`

Example:
> `x = 5` `if` `a < 10` `else` `x = 6`

Conditional expressions

In-class Activity

This OER material was produced as a result of the CS04ALL CUNY OER project.