Python Operators and Conditionals

February 8, 2021

Administrative Notes

Homework 1 is out

- Worth 50 points
- Due for our section next Monday, February 15
- Submission instructions are in the assignment
- Reminder: please submit early.
  - You can submit multiple times. Only the final submission is graded.
  - It always happens:
    - You don’t submit until you’re done
    - At 11:59 pm on the due date, you’re ready to submit
    - Your internet connection goes down; it comes back up at 12:03 am on Tuesday
    - The submission system has closed; you couldn’t submit on time; and it looks like a 0 is headed your way
    - DON’T LET THAT HAPPEN TO YOU!!! Submit a 95% complete version at 6 pm. When your Internet connection fails at 11:59 pm, you’re still getting 48 points out of 50!!

Operators

An ***operator*** is a special symbol that is used to carry out some operation on variables or values in Python

- Kind of a circular definition - an operator is used to carry out operations - but we'll clarify that

Types of operators you need to know about now:

- Arithmetic: +, -, *, /, //, %, ** - used to perform arithmetic on ints and floats
- Comparison: ==, !=, >=, >, <=, < - used to compare values
- Assignment: =, +=, -=, *=, /= - used to assign values to variables
- Logical: and, or, not - used to combine Boolean values into another Boolean value

There are other operators that will come up later, but this will get you started

Arithmetic operators

Most of these do the same things you're used to in math class

+	Addition; add two ints or floats. Adding two ints produces an int; anything else produces a float
-	Subtraction; subtract one int or float from another int or float. An int - an int is an int; anything else is a float
*	Multiplication; multiply two ints or floats. Multiplying two ints produces an int; anything else produces a float
/	Floating point division. Divide one int or float by another. Always produces a float
//	Integer division. Divide one int by another. Always produces an int - any "remainder" is truncated
%	Integer modulus. Divide one int by another. Throw away the quotient (the whole number part); this is the remainder.
**	Exponentiation. Raises one int or float to the power of an int or float

Comparison Operators

Again, mostly what you would expected

==	Is the value to the left the same as the value on the right? Are they numerically equal or are they identical strings? = is the assignment operator so you have to use == for comparisons
!=	Not equal
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to

Assignment Operators

Python lets you take some shortcuts in common situations

=	Assign the value; x = 5 (remember = vs. ==)
+=	Add the number on the right then assign x += 5 is the same as x = x+ 5
-=	Subtract the number on the right from the number on the left then assign x -= 5 is the same as x = x - 5
*=	Multiply the number on the right then assign x *= 5 is the same as x = x* 5
/=	Divide the number on the left by the number on the right then assign x /= 5 is the same as x = x/ 5

Logical Operators: and, or, not (also called “Boolean Operators”)

x	y	x and y
True	True	True
True	False	False
False	True	False
False	False	False

x	y	x or y
True	True	True
True	False	True
False	True	True
False	False	False

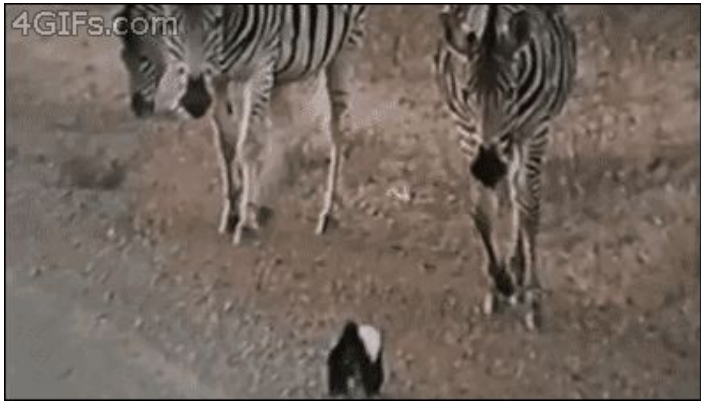
X	not X
True	False
False	True

Thanks, Dr. Johnson

Order of operations

`**`	Highest
`* / % //`	
`- +`	
`<= < > >= != ==`	
`not`	
`and`	
`or`	Lowest

Remember, like the noble and majestic honey badger, parentheses don't care about order of operations and will take precedence over everything.



Why operator precedence is important

What's the value of each:

6 * 5.0 + 2 - 5 / 5

(6 * (5.0 + 2) - 5) / 5

6 < 2 or 5 > 1

6 < (2 or 5) > 1 # let's talk about this one

Conditionals

Program control flow

Everything we've done to date is "sequential" program execution

- Execute one statement; then execute the next; and so on

But it's useful to be able to execute statements "sometimes" - only if something is True or something else is False

- This is called 'conditional' control flow

Conditional control flow

Start with an expression that equates to a Boolean value

- It evaluates to True or False
- It can be as simple or as complex as you want, as long as it evaluates in the end to True or False

Use the ‘if’ statement to control execution:

```
if  (boolean expression):
```

Some notes:

- That colon is critical; leave it out and your program doesn’t work correctly
- Python will evaluate the boolean expression and execute what comes after the colon ONLY IF that boolean expression evaluates to True

Three forms of conditionals:

1. One option: If a condition is true, do something. Otherwise, do nothing. “If” statement
2. Two options: If a condition is true, do something, Otherwise, do something else. “If….else…” statement
3. More than two options: If a condition is true, do something. Otherwise, check to see if another condition is true; do something else. Otherwise, keep checking conditions until we find one that’s true or we just give up. “If...elif...elif...else…” statement

The simplest conditional: the if statement

```
if [boolean]:
   # do something
```

Example:

```
if student == "awesome":
   print("You must be here for CMSC 201")
```

Now you know why we spent so much time talking about True, False, comparison and logical operators! They go after ‘if'!

Two cases: If and Else

Ask a student for her major. If she's a Physics major, print out "smart choice." Otherwise print out "there is still time"

```
major = input("please enter your current major")

if major == "Physics":
    print("smart choice")
else:
    print("there is still time")
```

Remember that input returns a string, so this comparison is valid

Colon after else, as well!

Indent the code that's part of the "else" block

Python and indentation

The boolean condition is everything between “if” and the colon : A colon terminates the condition. It can be as simple or as complex as you want

Indentation matters!! To python, white space - either tabs or spaces - indicates what’s in the code to be executed. You only have to indent one space, but I’m a believer that you should indent with tabs.

```
If x == 6:
    print (“X is 6”)
    print (“and we’re done”)
```

If x is really equal to 5, this doesn’t print out anything.
This still prints “and we’re done” because that’s not part of the conditional

```
if x == 6:
  print (“X is 6”)
print (“and we’re done”)
```

“if...else…”

- There must always be at least one line of code under the “if” statement
- You don’t have to have an “else” part, but if you do have an “else” there must be at least one line of code under it.

Three or more cases: elif

Input returns a string. This turns it into an integer

Ask a student her age. If it's less than 18, tell her she's a minor and faces some restrictions. If it's between 18 and 21, tell her she's an adult but still has some limitations. If she's over 21, tell her she's legally an adult.

This is 'pseudocode'

```
age = int(input("Please enter your current age in years."))
if age < 18:
        print("Sorry but you are a minor")
        print("there are a lot of things you cannot do on your own")
elif age < 21:
        print("you are an adult but there are still some things you can't do")
else
        print("congratulations you are legally an adult")
```