Texas Global Introduction to Python

Aashish Gottipati

Lecture 3: Boolean Logic and Strings

Spring 2025

Agenda

- Boolean logic
- Strings
- Printing

Comparison operators

- I.e. comparison operators
- Return Boolean values

(i.e. True or False)

- Used extensively for conditional statements

Operator	Output
x == y	True if x and y have the same value
x != y	True if x and y don't have the same value
x < y	True if x is less than y
x > y	True if x is more than y
x <= y	True if x is less than or equal to y
x >= y	True if x is more than or equal to y

Comparison examples

```
x = 5        # assign 5 to the variable x
x == 5       # check if value of x is 5
```

```
True
```

Note that `==` *is not the same as* `=`

```
x > 7
```

```
False
```

Logical operators

- Allows us to extend the conditional logic
- Will become essential later on

Operation	Result
x or y	True if at least on is True
x and y	True only if both are True
not x	True only if x is False

a	not a
False	True
True	False

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

Truth-table definitions of bool operations

Combining both

```
x = 14
# check if x is within the range 10..20
```

True and True

```
True
```

Another example

```
x = 14
y = 42

not (( True ))
```

```
False
```

That wasn't very easy to read was it?
Is there a way we can make it more readable?

Another Example

```
x = 14
y = 42

xDivisible = ( x % 2 ) == 0 # check if x is a multiple of 2
yDivisible = ( y % 3 ) == 0 # check if y is a multiple of 3

not (xDivisible and yDivisible)
```

```
False
```

Strings

- Powerful and flexible in Python
- Can be added
- Can be multiplied
- Can be multiple lines

Strings

```
x = "Python"
y = "rocks"
x + " " + y
```

```
'Python rocks'
```

```
x = "This can be"
y = "repeated "
x + " " + y * 3
```

```
'This can be repeated repeated repeated '
```

Strings

```
x = "Edinburgh"
x = x.upper()

y = "University Of "
y = y.lower()

y + x
```

```
'university of EDINBURGH'
```

These are called methods and add extra functionality to the String.

Mixing up strings and numbers

Often we would need to mix up numbers and strings.
It is best to keep numbers as numbers (i.e. int or float)
and cast them to strings whenever we need them as a string.

```
x = 6
x = ( x * 5345 ) // 63
"The answer to Life, the Universe and Everything is " + str(x)
```

```
'The answer to Life, the Universe and Everything is 42'
```

Multiline strings

```
x = """To include
multiple lines
you have to do this"""
y ="or you can also\ninclude the special\ncharacter `\\n` between lines"
print(x)
print(y)
```

```
To include
multiple lines
you have to do this
or you can also
include the special
character `\n` between lines
```

Printing

- When writing scripts, your outcomes aren't printed on the terminal.
- Thus, you must print them yourself with the print() function.
- Beware to not mix up the different type of variables!

```
print("Python is powerful!")
```

```
Python is powerful!
```

```
x = "Python is powerful"
y = " and versatile!"
print(x + y)
```

```
Python is powerful and versatile!
```

Quick quiz

Do you see anything wrong with this block?

```
str1 = "which means it has even more than"
str2 = 76
str3 = "quirks"
print(str1 + str2 + str3)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call l
ast)
<ipython-input-2-3be15a6244a4> in <module>()
      2 str2 = 76
      3 str3 = " quirks"
----> 4 print(str1 + str2 + str3)

TypeError: must be str, not int
```

Another more generic way to fix it

```
str1 = "It has"
str2 = 76
str3 = "methods!"
print(str1, str2, str3)
```

```
It has 76 methods!
```

If we comma separate statements in a print function we can have different variables printing!

Commenting

- Useful when your code needs further explanation. Either for your future self and anybody else.
- Useful when you want to remove the code from execution but not permanently
- Comments in Python are done with #

- `print(totalCost)` is ambiguous and we can't exactly be sure what `totalCost` is.
- `print(totalCost)  # Prints the total cost for renovating the Main Library` is more informative

Python Cheat Sheets

- https://www.pythoncheatsheet.org/cheatsheet/basics
- https://quickref.me/python.html

Exercise: Movie Night Decision

Task: Write a program to decide if you should watch a movie

Given Variables:

- movie_length = 95 (minutes)
- is_weekend = True
- has_homework = True
- time_available = 120 (minutes)
- battery_level = 50 (%)

Write Expressions For:

- Can you watch the full movie?
  - Compare time_available vs movie_length
- Should you watch the movie?
  - True if: weekend OR no homework, AND enough time
- Is your device ready?
  - Battery needs 1% per 10 minutes of movie
- Make final decision combining all conditions

Output: Print out all your decisions

Sample Output

```
Can I watch the full movie? True
Should I watch based on schedule? True
Is my device ready? False
Final decision: False
```

Exercise: Fuel Efficiency Calculator

Task: Write a program to compute fuel efficiency

Given Variables:

- distance_traveled = 350 # kilometers
- fuel_used = 32 # liters
- fuel_price = 1.50 # price per liter
- tank_capacity = 45 # liters

Calculate:

- Kilometers per liter (km/L)
- Cost of the trip
- Maximum range with a full tank
- Remaining fuel in liters

Output: Print all results

Sample Output

```
Fuel efficiency: 10.94 km/L
Trip cost: $48.00
Max range: 492.19 km
Remaining fuel: 13.00 L
```