APS106

UNIVERSITY OF TORONTO

Objects & Strings: Operators & Methods.

Week 3 | Lecture 1 (3.1.1)

While waiting for class to start:

Download and open the Jupyter Notebook (.ipynb) for Lecture 3.1.1

You may also use this lecture's JupyterHub link instead (although opening it locally is encouraged).

Upcoming:

- Reflection 3 released Friday @ 11 AM
- Lab 3 deadline this Friday @ 11 PM
- TUT (Tutorial) this week is **ONLINE**
- PRA (Lab) on Friday @ 2PM this week (ONLINE)

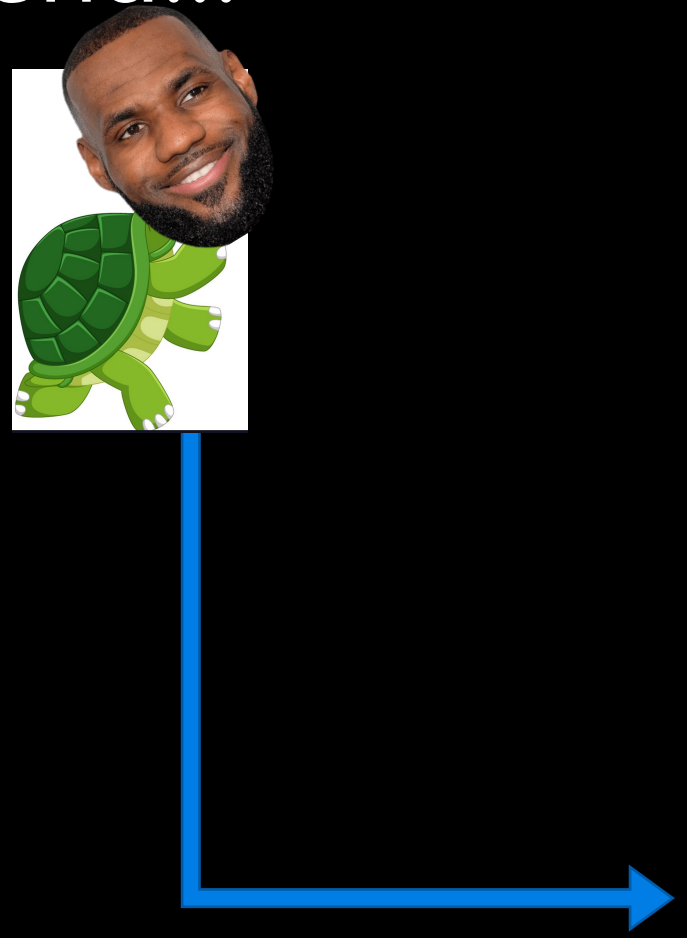
if nothing else, write #cleancode

Today's Content

- **Lecture 3.1.1**
  - **Objects & Strings: Operators and Methods**
- **Lecture 3.1.2**
  - Strings: Conversions, Indexing, Slicing, and Immutability

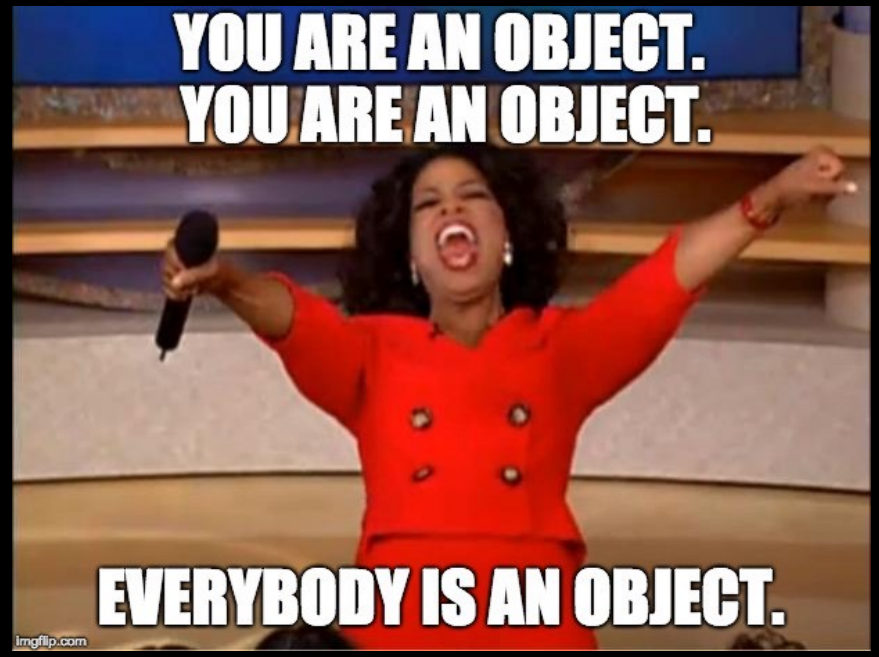
Let's revisit our Turtle friend...

```
import turtle
LeBron = turtle.Turtle()

LeBron.right(90)
LeBron.forward(200)
LeBron.left(90)
LeBron.forward(100)

turtle.done()
```



Everything is an Object!

- Python keeps track of every value, variable, function, etc. as an object
- There is a function that you can call to confirm:




```
>>> isinstance(4, int)
True
>>> isinstance("Hello", str)
True
```

```
>>> isinstance(4, object)
True
>>> isinstance('Hello", object)
True
```

Everything is an Object!

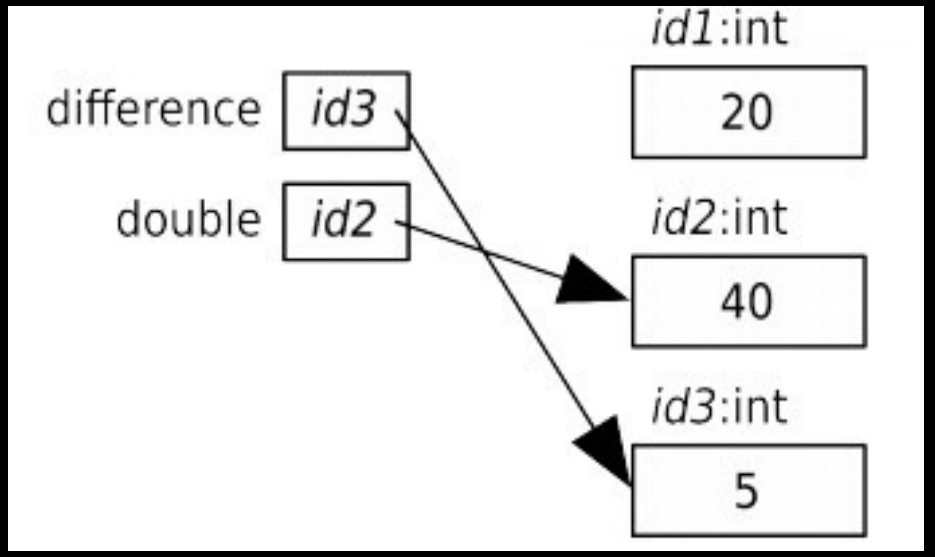
- Remember the id (or identity) function
  - It returns each object's location in memory

```
>>> id(4)
4482837280
>>> id(int)
4482678880
```

```
>>> id('a')
140432546199344
>>> id(str)
4482716800
```

Memory Visualization Example

```
>>> difference = 20
>>> double = 2 * difference
>>> double
40
>>> difference = 5
>>> double
40
```



Objects have Methods

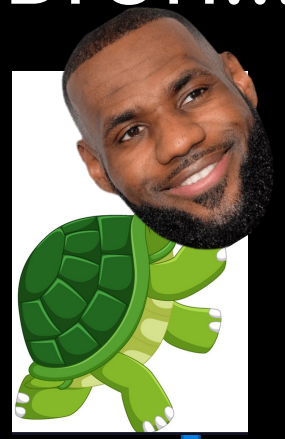
- Each Python object has certain functions that can only be applied to that object
  - These are called methods
- The general form for calling a method is:

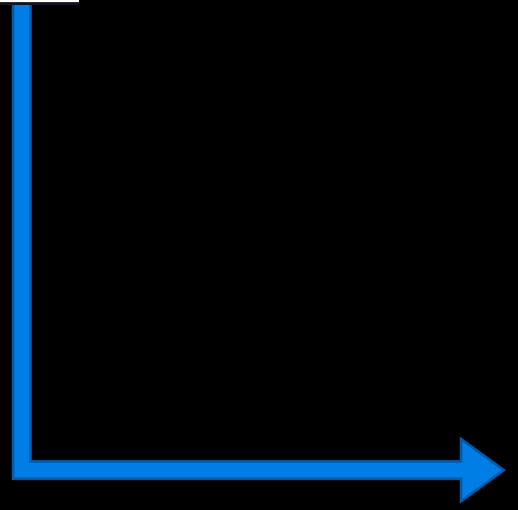
```
object_name.method_name(arguments)
```

- Since methods are applied to objects, we need to provide the object name with the "dot operator" (".") before the method name. Look familiar?

Let's revisit our friend LeBron...

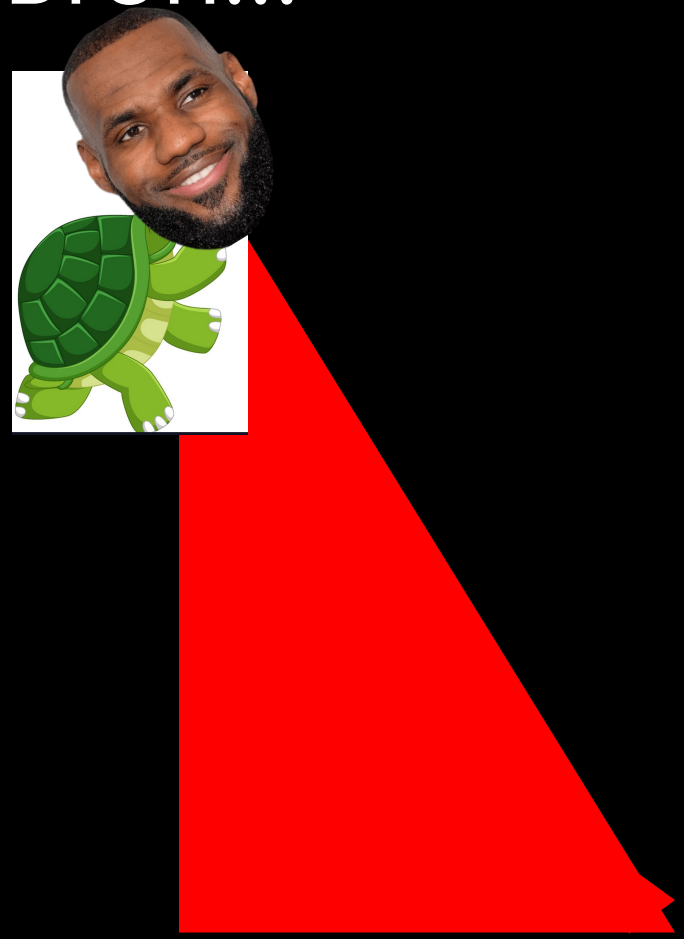
```
import turtle
LeBron = turtle.Turtle()

LeBron.right(90)
LeBron.forward(200)
LeBron.left(90)
LeBron.forward(100)

turtle.done()
```





Let's revisit our friend LeBron...

```
import turtle
LeBron = turtle.Turtle()

LeBron.begin_fill()
LeBron.color('red')
LeBron.right(90)
LeBron.forward(200)
LeBron.left(90)
LeBron.forward(100)
LeBron.end_fill()

turtle.done()
```



Let's Code!

- Let's take a look at how this works in Python!
  - id function to get object's memory address
  - isinstance function to determine object type
  - Turtle LeBron drawing shapes!

Open your notebook

Click Link:
1. Objects & Methods

RECAP: Input and Output

- Python has a built-in Input/Output functions:
  - print – for displaying text to the user
  - input – for reading text from the user

- These functions require a good understanding of strings and string formatting

Working with Strings

- The string (str) type was briefly introduced in previous weeks
- Let's take our string knowledge to the next level!
  - escape sequences
  - str operations
  - type conversion
  - str indexing and slicing
  - str methods

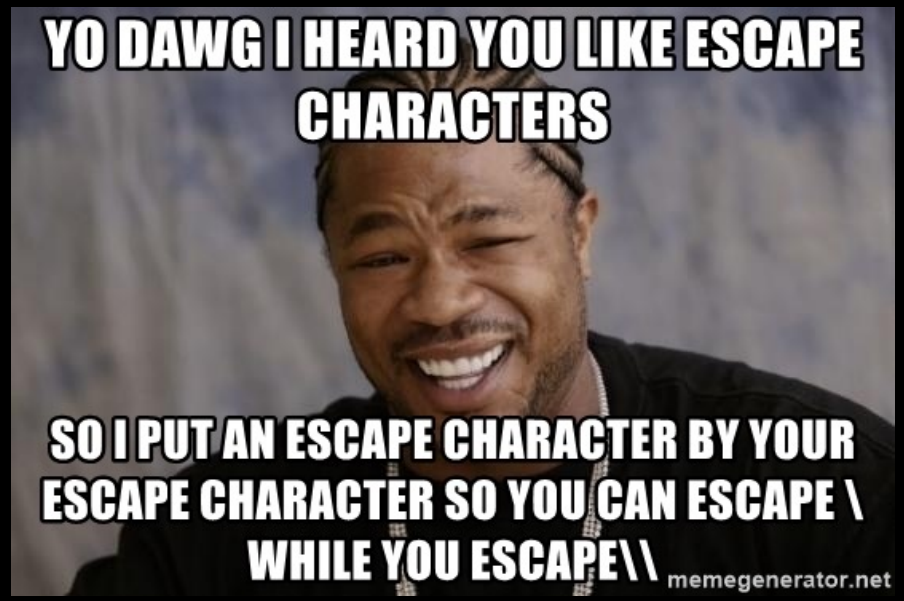
Escape Sequences

- Special character called an escape character: \ (backslash)
- When used in a string, the character following the escape character is treated differently from normal.

Escape sequence	Name	Example	Output
\n	newline (ASCII - line feed)	`"How\nare\nyou?"`	How are you?
\t	tab (ASCII - horizontal tab)	`'3\t4\t5'`	3 4 5
****	backslash (\)	`'\\'`	\
\'	single quote (')	`'don\'t'`	don't
\"	double quote (")	`"He says, \"Hi\"."`	He says, "hi".

Let's Code!

- Let's take a look at how this works in Python!
  - Working with the print function
  - Escape sequences!
    - New lines
    - Tabs
    - Quotes



Open your notebook

Click Link:
2. Escape Sequences

String Operators

- There are certain mathematical operators that can be applied on strings
  - The * and + obey standard precedence rules (i.e. * before +)
  - All other mathematical operators and operands result in a TypeError

Expression	Name	Example	Output
str1 + str2	concatenate str1 and str1	print('ab' + 'c')	abc
str1 * int1	concatenate int1 copies of str1	print('a' * 5)	aaaaa
int1 * str1	concatenate int1 copies of str1	print(4 * 'bc')	bcbcbcbc

Let's Code!

- Let's take a look at how this works in Python!
  - Concatenation
    - + operator
    - * operator

Open your notebook

Click Link:
3. String Operators

Working with Strings

- The string (str) type was briefly introduced in previous weeks
- Let's take our string knowledge to the next level!
  - escape sequences ✓
  - str operations ✓
  - type conversion
  - str indexing and slicing
  - str methods

Type Conversion

- The built-in function `str` takes any value and returns a string representation of that value
- Like our built-in functions `int` and `float` that can take a string and attempt to return a number representation of the string

```
>>> str(4)
'4'

>>> str(4482678880)
'4482678880'
```

```
>>> int('12345')
12345

>>> int(-99.9)
-99
```

```
>>> float('-43.2')
-43.2

>>> float('432')
432.0
```

Let's Code!

- Let's take a look at how this works in Python!
  - Type conversion
    - int/float to string
    - string to int/float

Open your notebook

Click Link:
4. Type Conversions

Breakout Coding Session!

- Ask the user how many times they would like to see the string "knock knock knock… Penny" repeated. Then, print it!
- Can you customize the name?

Open Python
(PyCharm or Jupyter)

Work with your table (or those around you) to solve this problem!

Consider this…

- Ask the user how many times they would like to see the string "knock knock knock… Penny" repeated, and print it!
- Can you customize the name?

Hints for getting started:

- Ask the user for a number of times (think: input function)
  - Remember input function returns a string…
- Repeated string (think: concatenation, * operator might be useful)
- Make the output readable (think: escape characters)

APS106

UNIVERSITY OF TORONTO

Objects & Strings: Operators & Methods.

Week 3 | Lecture 1 (3.1.1)

if nothing else, write #cleancode