

Objects & Strings: Operators & Methods.

Week 5 | Lecture 1 (5.1)

While waiting, open the Jupyter Notebook for today's lecture

Upcoming

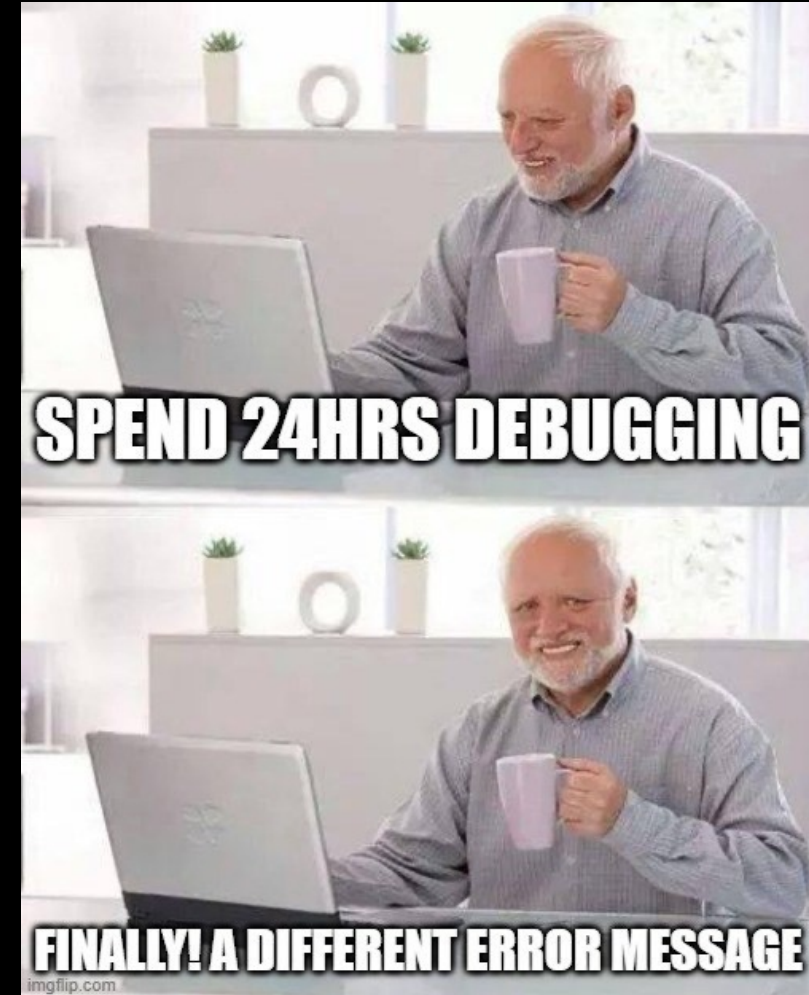
- NO lab due this Friday.
- Lab 3 Released Thursday 6:00 pm.
- Reflection 5 Released Friday 6:00 pm.
- Tutorial (in-person AND online) running all week.
- Practical sessions (in-person AND online) running ONLY Friday this week. if nothing else, write `#cleancode`

This Week's Content

- **Lecture 5.1**
 - **Objects & Strings: Operators and Methods**
- **Lecture 5.2**
 - Strings: Conversions, Indexing, Slicing, and Immutability
- **Lecture 5.3**
 - Introduction to Object-Oriented Programming and the File Object

Error Reduction vs Debugging

- It is pretty much impossible to write code without errors.
 - Error Reduction: techniques we can use to reduce the number and severity of errors.
 - Write Readable Code
 - Comment comment comment!
 - Test test test!
 - Debugging: techniques for identifying and correcting errors
 - How to use a debugger video coming in the How Does That Work? (HDTW) series on Quercus.



Do 'software'/coding engineers need to worry about **Designing for Safety?**

Therac-25: A Cancer Radiation Therapy Machine

- “Improvement” on Therac-20
 - Previous versions used hardware guards
 - Therac-25 relied only on software
- Meaningless error messages
 - Many misleading/incorrect errors
 - No clarification or mention of severity
 - **MALFUNCTION 54**
- Nothing checked if the strength of radiation was appropriate
 - How might you fix this?



PATIENT NAME: John		BEAM TYPE: E	ENERGY (KeV):	10
TREATMENT MODE: FIX				
	UNIT RATE/MINUTE	ACTUAL	PREScribed	
	MONITOR UNITS	0.000000	0.000000	
	TIME (MIN)	200.000000	200.000000	
		0.270000	0.270000	
GANTRY ROTATION (DEG)	0.000000	0.000000		VERIFIED
COLLIMATOR ROTATION (DEG)	359.200000	359.200000		VERIFIED
COLLIMATOR X (CM)	14.200000	14.200000		VERIFIED
COLLIMATOR Y (CM)	27.200000	27.200000		VERIFIED
WEDGE NUMBER	1.000000	1.000000		VERIFIED
ACCESSORY NUMBER	0.000000	0.000000		VERIFIED
DATE: 2012-04-16	SYSTEM: BEAM READY	OP.MODE: TREAT	AUTO	
TIME: 11:48:58	TREAT: TREAT PAUSE	X-RAY	173777	
OPR ID: 033-tfs3p	REASON: OPERATOR	COMMAND: █		

Therac-25: An Engineering Disaster

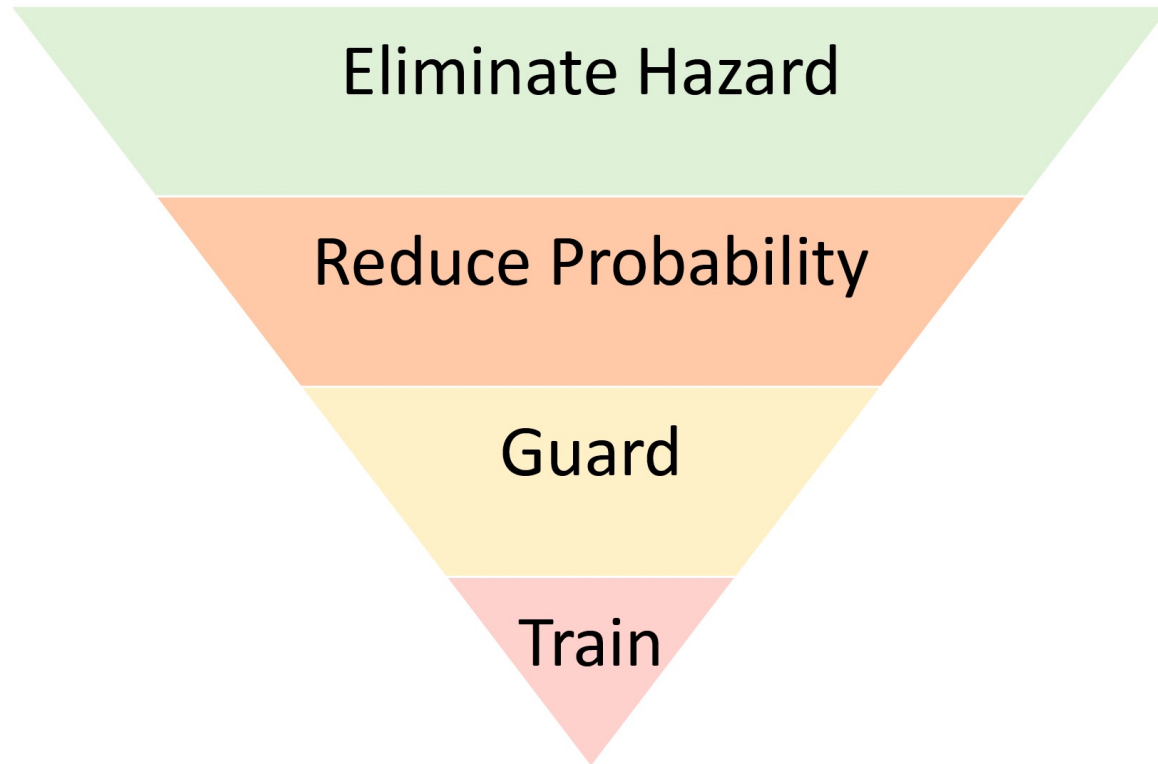
- Technicians began ignoring the misleading error messages
 - "the boy who cried wolf"
- Machine delivered radiation hundreds of times greater than normal
- 4 patients died and others severely injured...

```
if radiation_level < max_safe_level:  
    give_radiation(radiation_level)
```



When Designing for Safety: Training/warnings/labels are the last resort!

Hierarchy of Design for Safety



Remember:

No matter how bad today's midterm went...

You are safe.

&

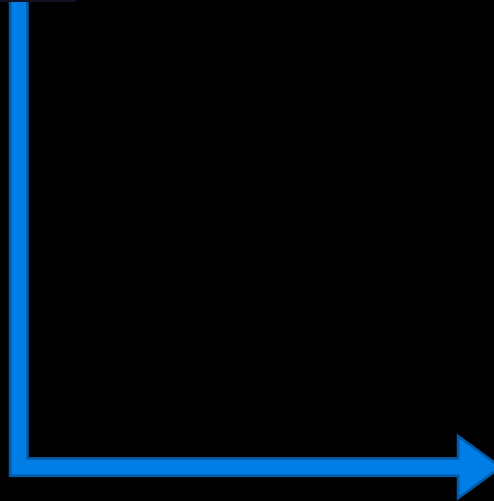
You didn't harm anyone with incorrect code.

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)
```

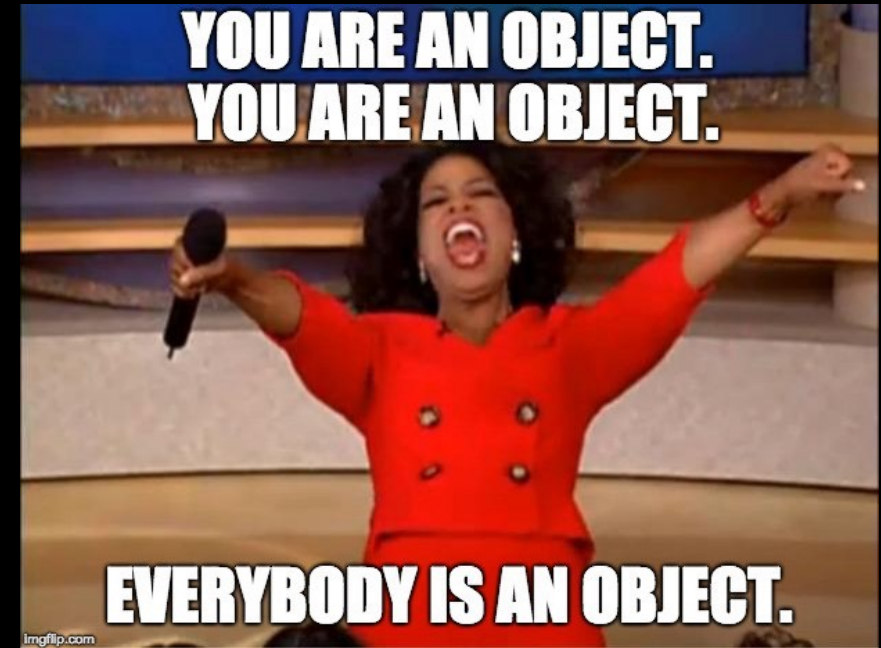
```
True
```

```
>>> isinstance(4, object)
```

```
True
```

```
>>> isinstance('Hello', object)
```

```
True
```



Everything is an Object!

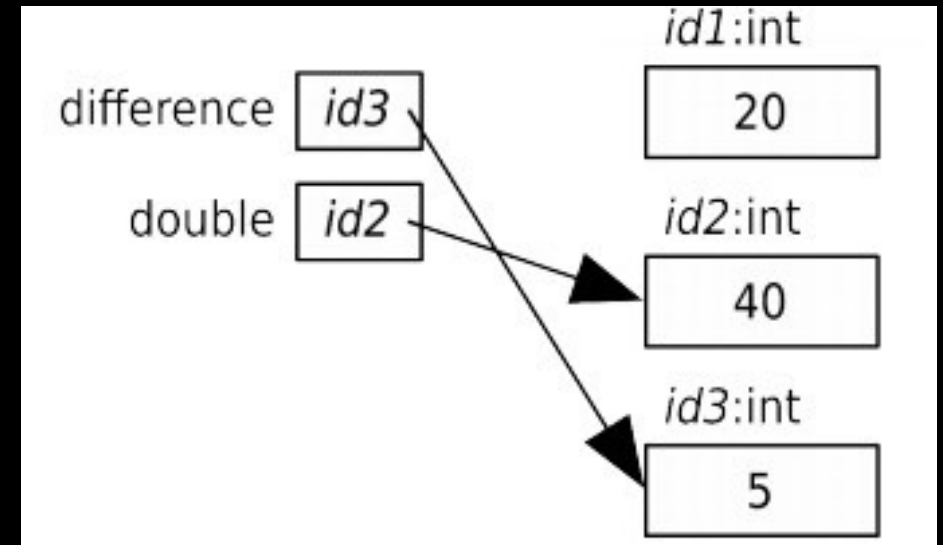
- Remember the `id` (or identity) function
 - It returns each object's location in memory

```
>>> 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



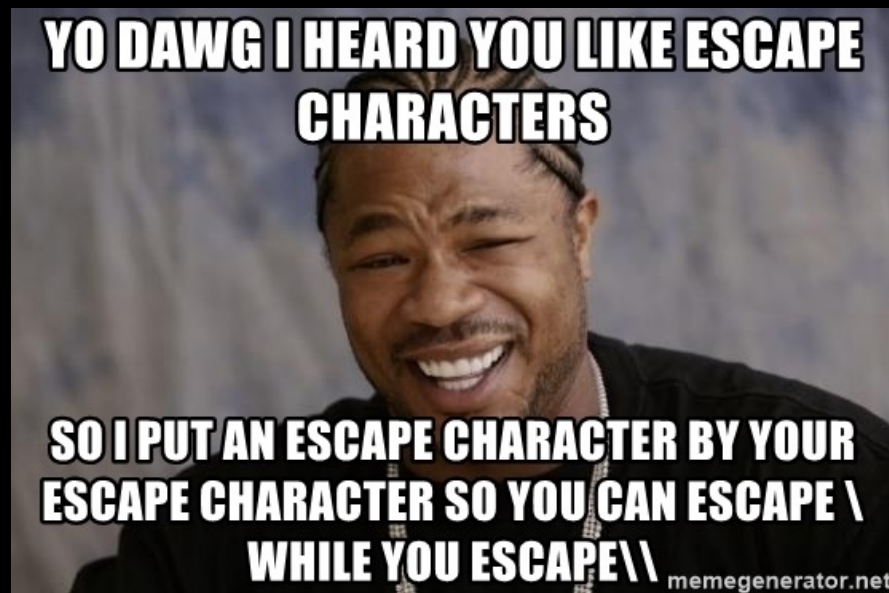
Escape Sequences

- 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
<code>\n</code>	newline (ASCII - line feed)	<code>"How\nare\nyou?"</code>	How are you?
<code>\t</code>	tab (ASCII - horizontal tab)	<code>'3\t4\t5'</code>	3 4 5
<code>\\</code>	backslash (<code>\</code>)	<code>'\\'</code>	<code>\</code>
<code>\'</code>	single quote (<code>'</code>)	<code>'don\'t'</code>	don't
<code>\"</code>	double quote (<code>"</code>)	<code>"He says, \"Hi\"."</code>	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'
```

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

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

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

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

```
>>> 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
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)

Objects & Strings: Operators & Methods.

Week 5 | Lecture 1 (5.1)

if nothing else, write `#cleancode`