Introduction to Python

COGS3
Introduction to Computing
Mary ET Boyle, Ph. D.
INTRODUCTION TO PYTHON PROGRAMMING BASICS

"PROGRAMMING is the art of entering instructions for the computer to perform."

BASIC INSTRUCTIONS AS BUILDING BLOCKS

→ [] DO THIS, THEN DO THAT
→ [] IF THIS CONDITION IS TRUE, PERFORM THIS ACTION, OTHERWISE DO THAT ACTION.
→ [] DO THIS ACTION THAT # OF TIMES
→ [] KEEP DOING THAT UNTIL THIS CONDITION IS TRUE.
What is Python

- It is a programming language
- It is interpreted

It runs on Linux and Windows

Python code => performs instructions

The interpreter
1. It is all about breaking down big problems into very small solvable solutions → detailed steps
2. PROGRAMMING is a creative activity like building a castle out of legos.
PYTHON PROGRAMMING BASICS

the future

python 2 vs python 3

- not backwards compatible
- some libraries won't work
- Mac's still use 2.7

new to Python
learn 3
& then learn differences w/ 2.x.
IDLE is Python's Integrated Development and Learning Environment.

IDLE has the following features:

- coded in 100% pure Python, using the Tkinter GUI toolkit
- cross-platform: works mostly the same on Windows, Unix, and Mac OS X
- Python shell window (interactive interpreter) with colorizing of code input, output, and error messages
- multi-window text editor with multiple undo, Python colorizing, smart indent, call tips, auto completion, and other features
- search within any window, replace within editor windows, and search through multiple files (grep)
- debugger with persistent breakpoints, stepping, and viewing of global and local namespaces
- configuration, browsers, and other dialogs

```python
>>> print('Hello world')
Hello world
```
Python: IDLE
Interactive Development Environment

>>> print('Hello world')
Hello world

You can execute Python instructions one at a time & you get the immediate results!

GREAT FOR LEARNING BASIC PYTHON INSTRUCTIONS
Programming Basics w/ Python

- Simple math

```
>>> 2 + 2
4
```

- "evaluated" result
- a single value

- This is an 'operator'

- This is an 'expression'

- 'value'

- 'value'
Aha moment 😊

YAY!

you can use expressions anywhere in python that you could also use a value.
Note:  
4  
>>> 4  

This is also an 'expression'

evaluated to itself!
<table>
<thead>
<tr>
<th>Operator</th>
<th>Operation</th>
<th>Example</th>
<th>Evaluates to</th>
</tr>
</thead>
<tbody>
<tr>
<td>**</td>
<td>Exponent</td>
<td>2**3</td>
<td>8</td>
</tr>
<tr>
<td>%</td>
<td>Modulus/Remainder</td>
<td>22%8</td>
<td>6</td>
</tr>
<tr>
<td>//</td>
<td>Integer Division (Floored Quotient)</td>
<td>22//8</td>
<td>2</td>
</tr>
<tr>
<td>/</td>
<td>Division</td>
<td>22/8</td>
<td>2.75</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication</td>
<td>3*5</td>
<td>15</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction</td>
<td>5-2</td>
<td>3</td>
</tr>
<tr>
<td>+</td>
<td>Addition</td>
<td>2+2</td>
<td>4</td>
</tr>
</tbody>
</table>
How would these be evaluated?

- $2 + 3 \times 6$
- $(2 + 3) \times 6$
- $23 \div 7 \times 3$
- $23 \% 7$
- $(5 - 1) \times \left( \frac{7 + 1}{3 - 1} \right)$
How would these be evaluated?

- $2 + 3 \times 6$
- $(2 + 3) \times 6$
- $23 \div 7 \times 3$
- $23 \% 7$
- $(5 - 1) \times ((7 + 1) / (3 - 1))$

### Math Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
<th>Evaluates to</th>
</tr>
</thead>
<tbody>
<tr>
<td>**</td>
<td>Exponent</td>
<td>$2**3$</td>
<td>$8$</td>
</tr>
<tr>
<td>%</td>
<td>Modulus/remainder</td>
<td>$22 % 8$</td>
<td>$6$</td>
</tr>
<tr>
<td>//</td>
<td>Integer division (floored quotient)</td>
<td>$22 // 8$</td>
<td>$2$</td>
</tr>
<tr>
<td>/</td>
<td>Division</td>
<td>$22 / 8$</td>
<td>$2.75$</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication</td>
<td>$3 \times 5$</td>
<td>$15$</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction</td>
<td>$5 - 2$</td>
<td>$3$</td>
</tr>
<tr>
<td>+</td>
<td>Addition</td>
<td>$2 + 2$</td>
<td>$4$</td>
</tr>
</tbody>
</table>

Order of operations is important in determining the result of the calculations.
Look at this in detail

\[
(5-1) \times \left( \frac{7+1}{3-1} \right)
\]

\[
\downarrow
\]

\[
4 \times \left( \frac{7+1}{3-1} \right)
\]

\[
\downarrow
\]

\[
4 \times \left( \frac{8}{2} \right)
\]

\[
\downarrow
\]

\[
4 \times 4.0
\]

\[
16.0
\]
3 DATA TYPES:
INTEGER, FLOATING-POINT & STRING

WOW
A DATA TYPE IS A CATEGORY FOR VALUES

EVERY VALUE BELONGS TO EXACTLY ONE DATA TYPE!!
<table>
<thead>
<tr>
<th>Data Type</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integers (int)</td>
<td>whole numbers: {-2, -1, 0, 2, 3, 5}</td>
</tr>
<tr>
<td>Floating-point (float)</td>
<td>{-1.25, -1.0, 0.5, 1.0}</td>
</tr>
<tr>
<td>Strings (str)</td>
<td><code>abc</code>, <code>'cogs3'</code>, <code>'hi there'</code></td>
</tr>
</tbody>
</table>

Note: the quotation marks are not part of the string value. Just mark the boundaries.
Context rules → data type dependent

Operators do different things depending on the data types!

- String concatenation
- Replication
+ \rightarrow \text{with numbers is "addition"}

+ \rightarrow \text{with strings is "concatenation"}

$2 + 2 \Rightarrow 4$

operator is addition

numeric datatypes

Mary" + "Polly"

⇒ MaryPolly

operator is string concatenator

string datatypes
Q: WHAT WOULD HAPPEN HERE: 2 + 'mary'
REPLICATION...

- string
- string replication operator
- integer
- note - must be an integer (cannot be a float data type!)
STORING VALUES INTO VARIABLES!

It's the thing to do!!

RULES
- only one word
- letters, numbers, underscore
- cannot begin with a number

NAME THE VARIABLE

ONE VALUE STORED
the variable SPAM now has the integer value 42 in it.

located somewhere

the value 42 is stored in the variable SPAM
TRY IT...

1. >>> spam = 40
   >>> nothing returned.

2. >>> spam = 40
   evaluated

3. >>> eggs = 2
   use it in expressions

4. >>> spam + eggs
   42

5. >>> spam + eggs + spam
   82

6. >>> spam = spam + 2
   42

variable is initialized 1st time it is created

“overwriting a variable”

assigned a new value
Try it...

Am I valid...

- balance
- current-balance
- currentBalance
- 4account
- _spam
- SPAM
- account4
- total_$um
- 'my_var'
- cogs3
- theBest1

Rules:
- only one word
- letters, numbers, underscore
- cannot begin with a number
14000 ← VARIABLES ARE:

- case sensitive

CONVENTION ← PYTHON VARIABLES:

- START W/ LOWER CASE
- good habit: use Camel Case

BE CONSISTENT!
FUNCTIONS

\( \text{print()} \) \( \rightarrow \) \{string, integer, float\}

\text{print}() \quad \text{displays the string value inside the parenthesis}

\text{print}('Hello CO653')

\text{yes!}

\text{python calls the print() function}

\text{this is the string value being passed to the function.}

\text{argument}

\text{a value that is passed to a function call is called an argument.}
print()  
Python will call this fcn & print the argument to the screen.

In this case it will print nothing + you get a blank line.
**Boolean Data Type**

- Only can have one of two values:
  - True
  - False

  - Notice the first letter is capitalized.

  - Boolean
    - Capitalized.
    - Named after mathematician George Boole.
<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>==</code></td>
<td>equal to</td>
</tr>
<tr>
<td><code>!=</code></td>
<td>not equal to</td>
</tr>
<tr>
<td><code>&lt;</code></td>
<td>less than</td>
</tr>
<tr>
<td><code>&gt;</code></td>
<td>greater than</td>
</tr>
<tr>
<td><code>&lt;=</code></td>
<td>less than or equal to</td>
</tr>
<tr>
<td><code>&gt;=</code></td>
<td>greater than or equal to</td>
</tr>
</tbody>
</table>

These operators work with values of any data type.
These operators only work with integer and floating point data type values.
WATCH OUT: assignment vs comparison

this is the assignment operator

this is the is equal to operator
Boolean Operators

(and, or, not)
	hey evaluate to a Boolean value namely \{True, False\}.

these operators are used to compare Boolean values
Binary Boolean operators

Called binary b/c they take two Boolean values (or expressions)

→ Truth Table → They show every possible result of a Boolean operator.