Introduction to Python 😊
Weakened typed language → string

INTRODUCTION TO PYTHON

PROGRAMMING BASICS

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

SIMPLE DEFINITION
BASIC INSTRUCTIONS AS BUILDING BLOCKS

- If this condition is true, then do that.
  - Do this, then do that.
  - If this condition is true, perform this action.
  - Otherwise do that action.

- For
  - Do this action that # of times

- While
  - Keep doing that until this condition is true.
What is Python

- It is a programming language
- It is interpreted

Runs on Linux, MacOS, and Windows

Interpreted: reads Python code, performs instructions
1. It is all about breaking down big problems into very small solvable solutions → detailed steps

BABY STEPS TO PROGRAMMING
Python Programming Basics

Python 2 vs Python 3

- not backwards compatible
- some libraries won't work

New to Python
Learn 3 & then learn differences up 2.x.

Interactive Shell:
Python:
IDLE
Interactive Development Environment

Ready for you!

>>> print("Hello world")
Hello world

Great for Learning Basic Python Instructions
Programming Basics w/ Python

- Simple math
  - "evaluated" result
  - a single value
  - `2 + 2` is a 'value'
  - `4` is an 'operator'
  - this is an 'expression'

checked
Aha moment

YAY!

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

evaluated to itself!

this is also an 'expression'
<table>
<thead>
<tr>
<th>Math Operators</th>
<th>evaluated 1st: L to R</th>
<th>Example</th>
<th>Evaluates to</th>
</tr>
</thead>
<tbody>
<tr>
<td>** exponent **</td>
<td>2<strong>3</strong></td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>% modulus/remainder</td>
<td>22%8</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>// integer division (floored quotient)</td>
<td>22//8</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>/ division</td>
<td>22/8</td>
<td>2.75</td>
<td></td>
</tr>
<tr>
<td>* multiplication</td>
<td>3*5</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>- subtraction</td>
<td>5-2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>+ addition</td>
<td>2+2</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

**FIZZ BUZZ**

- 2.8 → 2
- 3.1 → 3

IF X%3 == 0
$1238$ is divisible by $3$ → $1378$
$1235$ is divisible by $5$ → Buzz
$385$ → $33$ Buzz
DO IT:

How would these be evaluated?

☐ 2 + (3 * 6)
☐ (2 + 3) * 6
☐ 23 // 7 * 3
☐ 23 % 7
☐ (5 - 1) * ((7 + 1) / (3 - 1))
Look at this in detail

Evaluating an expression reduces it to a single expression
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>'abc', 'cogs3', 'hi there'</td>
</tr>
</tbody>
</table>
Operators do different things depending on the data types!

"ABCDEFG" = "ABC" + "DEF"

1 + 5 = 6

String concatenation

Replication
5 + "Hello"  
+ → with numbers is "addition"  "mary" +  
+ → with strings is "concatenation"  "mary" + " " + "polly"  
2 + 2 → 4  
operator is addition  
numeric datatypes  
mary" + "polly"  
⇒ MaryPolly  
operator is string concatenator  
string datatypes  
"mary" + "polly"
REPLICATION...

```
>>> 'Polly' * 5
PollyPollyPollyPollyPolly
```

proof >>> 5 * 'polly'

Note - must be an integer (cannot be a float data type!)
STORING VALUES INTO VARIABLES!

It's the thing to do!!

hello_class

RULES
- only one word
- letters, numbers, underscore
- cannot begin with a number
the value 42 is stored in the variable SPAM

the variable SPAM now has the integer value 42 in it
TRY IT...

AM I VALID...

- balance
- currentBalance
- 4account
- _spam
- SPAM

- account4
- total_sum
- 'my_var'
- cogs3
- theBest1
I KNOW

VARIABLES ARE:

* case sensitive

CONVENTION ← Python variables:

* start w/ lower case

BE CONSISTENT!

good habit

* use camel case

use Python convention: not_camel_case
FUNCTIONS

\[ \text{print()} \]
\[ \text{display the string value inside the parentheses} \]
\[ \text{print(\text{'Hello CO6S3'})} \]
\[ \text{print(8963)} \]
\[ \text{this is the string value being passed to the function.} \]

\[ \text{argument} \]
\[ \text{a value that is passed to a function call is called an argument.} \]

\[ \text{Python will call this func & print the argument to the screen.} \]
\[ \text{In this case it will print nothing} \]
\[ \text{+ you get a blank line.} \]
`input()`  
this function waits for the user to type some text on the keyboard and press `ENTER`.  

This function always returns a string data type.